

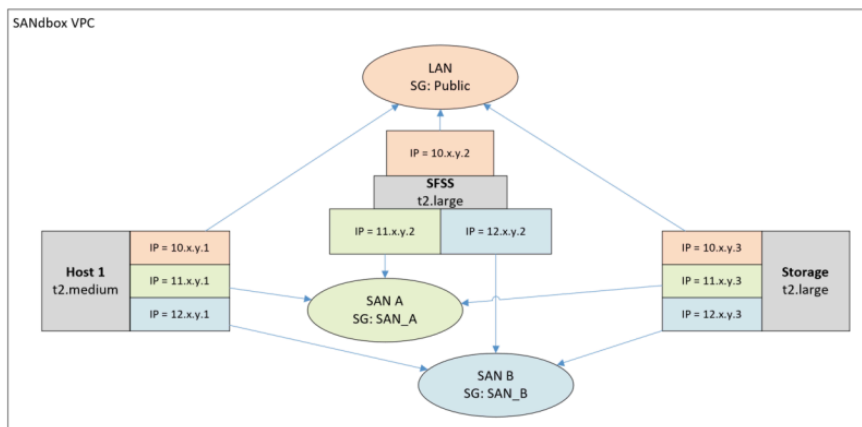

[Home](#) [Archives](#) [Profile](#) [Subscribe](#)
[Main](#)

09/30/2022

## Let's play in an AWS based SANdbox - Part 2 Setting up your AWS VPC and networks

In my previous post, I introduced the SANdbox Github repo and described the work that had been done to create an Virtual Lab that is capable of supporting NVMe/TCP traffic.

In this post I'll walk you through the steps to create your own SANdbox VPC in AWS that will eventually look similar to the following.



Before I get started, I need to acknowledge two sources of information that were invaluable to me as I was getting started with AWS:

1. The AWS documentation - I found [an article](#) that seemed to describe exactly what I wanted to do but I didn't have enough context/knowledge of AWS at the time to be able to follow it successfully.
2. This ["HOW TO Build a Home Lab in AWS for FREE"](#) video by Jon Good. It was incredibly helpful and I highly recommend reviewing it before continuing with the following steps.

## Create a Free AWS account

1. Click [here](#) to create a Free AWS account.


[Erik Smith](#)
[Follow](#)
[ARCHIVES](#)

SEARCH

 
[CATEGORIES](#)
[Subscribe to this blog's feed](#)

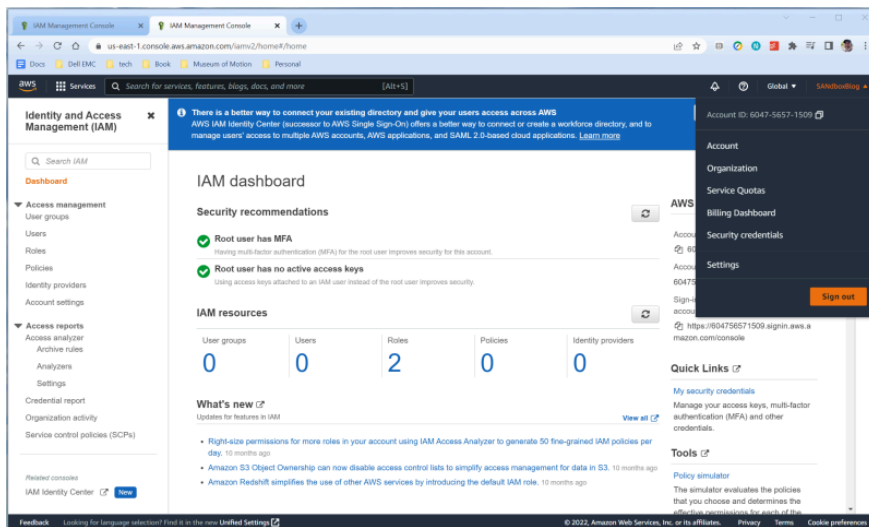
DISCLAIMER

This is not an official EMC blog.  
The opinions expressed here are my personal opinions.  
Content published here is not read or approved in advance by EMC and does not necessarily reflect the views and opinions of EMC nor does it constitute any official communication of EMC.

2. Provide an email address
3. Provide an AWS account name
4. Verify the email Address you provided by taking note of the Verification code they provide you an then entering it into the browser when prompted.
5. Create your Root user password.
6. Provide your contact information
7. Provide credit card information
8. Confirm your identity (I chose to use SMS)
9. Select a support plan (I chose Basic support - Free)
10. Click **Complete sign up**
11. Click **Go to the AWS console**
12. Enter or ensure that the correct email address is provided and click **Next**.
13. Enter the password associated with the email address provided and click **Next**.

## Secure your account with MFA

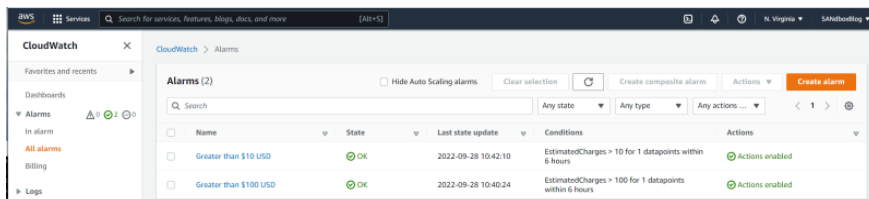
1. In AWS, under Console Home, click **IAM**
  - a. Note: If you ever lose track of a particular view or service, you can always click the Services icon at the top of the page or search for the service name you want to access and get to it that way. I do this all the time for accessing my “EC2” instances.
2. In the IAM dashboard, click **Add MFA**.
3. Under Your Security Credentials > Multi-factor authentication (MFA), click **activate MFA**.
4. Choose the type of MFA device to assign (I chose “Virtual MFA device”) and then click **Continue**
5. In the “Set Up virtual MFA device” dialog, you will be prompted to:
  - a. Install a compatible app on your mobile device or computer (I chose the App “Authenticator” because it was supported on IOS and had good reviews online)
  - b. Use the virtual MFA app and your device’s camera to scan the QR code shown within AWS.
  - c. Enter two consecutive tokens from the Authenticator APP, then click **OK**, and you should see “You have successfully assigned a virtual MFA. Click **Close**
  - d. Click on “Dashboard” (left side of the screen) and ensure there is a green check mark next to “Root user has MFA”.
6. Click on the user name at the top right of the screen (e.g., **SANdboxBlog**) and then click **Sign out**



## Setup a Billing Alert

Note: Jon Good recommended this and I cannot stress how good of an idea it is. You'd be surprised how quickly you can blow through \$10 once you start using non-free instances and Elastic IP Addresses. My only regret is I did not also create a \$100 alert :- ) ... do not make the same mistake I did! The steps for doing this are very straight-forward and can be found [here](#).

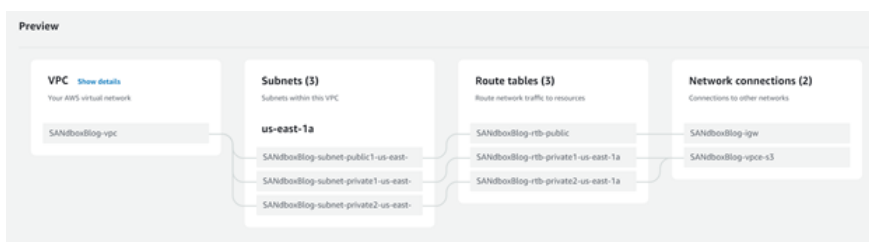
Here's my alarm configuration after setup.



## Create a Virtual Private Cloud (VPC)

The VPC is the top level “container” into which all of your Virtual Resources will be deployed.

1. To create a VPC, click on **Services** (at the top left of AWS console screen) and click on (or search for) **VPC**. Click **Create VPC**.
2. Under VPC settings, I chose **VPC and more**.
3. Under Name tag auto-generation, I provided a tag of **SANdbosxBlog**.
4. I decided to use a **single Availability Zone** as my lab configuration does not require HA.
5. I increased the number of private subnets from 1 to 2.
6. All other settings were left at defaults.
7. The VPC preview should look similar to the following
8. Click **Create VPC** and the VPC workflow will run. Click View VPC after the workflow completes.



## Create and associate Security Groups

Creating and managing security groups is a tedious but necessary task. It allows you to define the sources (e.g., IP Addresses) that can be used to access the resources inside of your VPC. I created three Security Groups, one for my LAN (public facing) network and two for my SANs (e.g., SAN A and SAN B). I've summarized the SGs I created below for ease of reference.

### LAN-Public

| Name  | IP ver | Type  | Protocol | Port Range | Source    |
|-------|--------|-------|----------|------------|-----------|
| SSH   | IPv4   | SSH   | TCP      | 22         | 0.0.0.0/0 |
| HTTPS | IPv4   | HTTPS | TCP      | 443        | 0.0.0.0/0 |

### SAN A and SAN B

| Name               | IP ver | Type            | Protocol | Port Range | Source    |
|--------------------|--------|-----------------|----------|------------|-----------|
| mDNS               | -      | Custom UDP      | UDP      | 5353       | 0.0.0.0/0 |
| NVMe/TCP Discovery | IPv4   | Custom TCP      | TCP      | 8009       | 0.0.0.0/0 |
| NVMe/TCP IO        | IPv4   | Custom TCP      | TCP      | 4420       | 0.0.0.0/0 |
| PING               | -      | All ICMP - IPv4 | ICMP     | ALL        | 0.0.0.0/0 |

## Create the “LAN-Public” Security Group

1. From “Services”, click on **EC2**. (Not really required but I like to start from the EC2 dashboard for everything)
2. On the left side of the console scroll down to “Network & Security” and click on **Security Groups** and then **Create Security Group**.
3. Add a Security Group Name (e.g., LAN-Public)
4. Add a Description (e.g., Allows SSH and HTTP access to instances)
5. **VERY IMPORTANT** Ensure the correct VPC is selected. The default VPC listed in the VPC field is probably not what you want. Click on the ‘X’ in the right side of the VPC field and ensure you have selected the correct VPC. In my case I selected the SANdbosxBlog-vpc. If you haven't already done this, take note of the last four digits of your VPCs instance ID. In my case it's “a245”, you will use this later on. Also, I highly recommend creating a diagram of your environment to keep track of all of the instance IDs. A completed example diagram will be provided at the end of this post.
6. Create Inbound rules - As described in the LAN-Public table above, we need to create rules that will allow for both SSH and HTTPS access to our instances.

### SSH

1. Under Inbound rules, click **Add Rule**

2. From the Type pulldown, select **SSH**. This will automatically populate the Protocol and Port Range fields for you.
3. From the Source pulldown select Anywhere-IPv4. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.

NOTE: Using Anywhere-IPv4 is NOT a best practice. Ideally, you would specify the IP Address of the system you will be using to access your VPC.

## HTTPS

1. Click **Add Rule**
2. From the Type pulldown, select **HTTPS**. This will automatically populate the Protocol and Port Range fields for you.
3. From the Source pulldown select Anywhere-IPv4. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.

Confirm that your security group looks similar to what is shown below and then click **Create Security Group**.

The screenshot shows the AWS 'Create security group' interface. Under 'Basic details', the name is 'LAN-Public', the description is 'Allows SSH and HTTP access to instances', and the VPC is 'vpc-00b463704786d245'. The 'Inbound rules' section contains two rules: an SSH rule (TCP, port 22) and an HTTPS rule (TCP, port 443), both with 'Anywhere IPv4' as the source. The 'Outbound rules' section contains one rule for 'All traffic' (All protocol, all ports) with a 'Custom' destination.

Once the SG was created I took note of the Instance number for the LAN-Public Security Group (i.e., 4872) and added this to my topology diagram.

## Create the “SAN A” Security Group

1. On the left side of the console click on **Security Groups** and then **Create Security Group**.
2. Add a Security Group Name (e.g., SAN A)
3. Add a Description (e.g., SAN A is a private network for NVMe/TCP Discovery and IO traffic.)
4. VERY IMPORTANT Ensure the correct VPC is selected. The default VPC listed in the VPC field is probably not what you want. In my case I selected the SANdbxBlog-vpc and it has an instance ID of “a245”.
5. Create Inbound rules - As described in the SAN A and SAN B table above, we need to create rules that will allow for mDNS, NVMe/TCP Discovery, NVMe/TCP IO and Ping to be used between the instances in our VPC.

## mDNS

1. Under Inbound rules, click **Add Rule**
2. From the Type pulldown, select **Custom UDP**. This will automatically populate the Protocol field for you.

3. In the Port range field enter **5353**
4. From the Source pulldown select **Anywhere-IPv4**. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.
5. Provide a description (e.g., **mDNS**)

### NVMe/TCP Discovery

1. Click **Add Rule**
2. From the Type pulldown, select **Custom TCP**. This will automatically populate the Protocol field for you.
3. In the Port range field enter **8009**
4. From the Source pulldown select **Anywhere-IPv4**. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.
5. Provide a description (e.g., **NVMe/TCP Discovery**)

### NVMe/TCP IO

1. Click **Add Rule**
2. From the Type pulldown, select **Custom TCP**. This will automatically populate the Protocol field for you.
3. In the Port range field enter **4420**
4. From the Source pulldown select **Anywhere-IPv4**. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.
5. Provide a description (e.g., **NVMe/TCP IO**)

**PING** (this is not technically necessary, but you will thank me later).

1. Click **Add Rule**
2. From the Type pulldown, select **All ICMP - IPv4**. This will automatically populate the Protocol and Port range fields for you.
3. From the Source pulldown select **Anywhere-IPv4**. The IP address of 0.0.0.0/0 will automatically be added to your list of sources.
4. Provide a description (e.g., **PING**)

Confirm that your security group looks similar to what is shown below and then click **Create Security Group**.

The screenshot shows the AWS Security Groups console for a security group named 'SAN A'. The 'Basic details' section shows the group name, description 'SAN A is a private network for NVMe/TCP Discovery and IO traffic.', and VPC 'vpc-009437547f8d435'. The 'Inbound rules' section lists four rules:

| Type            | Protocol | Port range | Source          | Description - optional |
|-----------------|----------|------------|-----------------|------------------------|
| Custom UDP      | UDP      | 5353       | Anywhere (IPv4) | mDNS                   |
| Custom TCP      | TCP      | 8009       | Anywhere (IPv4) | NVMe/TCP Discovery     |
| Custom TCP      | TCP      | 4420       | Anywhere (IPv4) | NVMe/TCP IO            |
| All ICMP - IPv4 | ICMP     | All        | Anywhere (IPv4) | PING                   |

The 'Outbound rules' section shows one rule:

| Type        | Protocol | Port range | Destination | Description - optional |
|-------------|----------|------------|-------------|------------------------|
| All traffic | All      | All        | Custom      |                        |

## Create the “SAN B” Security Group

Using the information from the SAN A and SAN B table above, repeat the steps outlined in “Create the SAN A Security Group” and create another Security

Group for SAN B.

OR... You can save a ton of time and do the following

1. Under **Actions**, at the top of the “SAN A Security Group” configuration screen, you can chose “Copy to a new security group”.

**VERY IMPORTANT** After providing a “Security group name” (e.g., SAN B) and a Description, you will still need to ensure the correct VPC is selected. The default VPC listed in the VPC field is probably not what you want. In my case I selected the SANdbosBlog-vpc and it has an instance ID of “a245”.

2. Click **Create Security Group**

## Create your first EC2 Instances

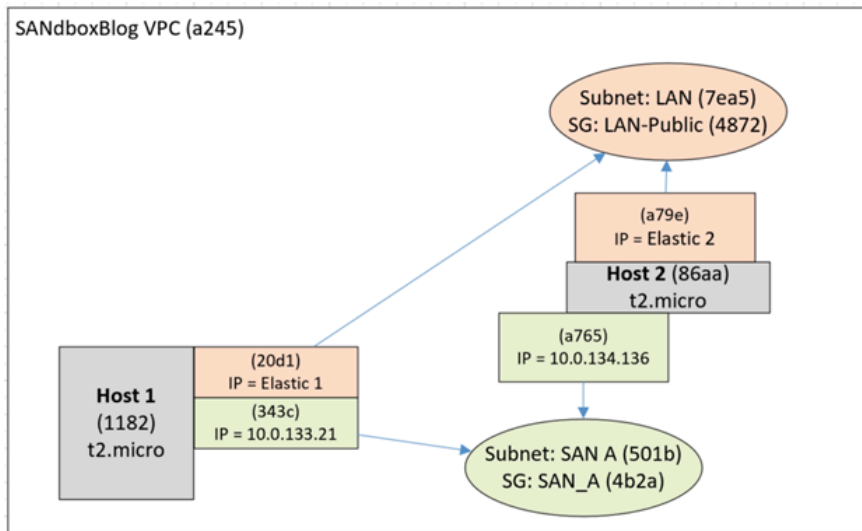
Amazon Elastic Compute Cloud (EC2) instances can be thought of as virtual servers. For the purposes of the SANdbos configuration, I’ve chosen to use EC2 instances that are running Ubuntu 20.04. When I was first getting started and needed to work my way through the process of setting up the environment and familiarizing myself with AWS in general, I chose EC2 instances that were available in the free tier. This means there is no charge up to a certain amount of usage. I will walk you through doing the same for two instances and will get us to the point where you can connect to each instance with SSH and ping across the network interfaces used for SAN A. We will not be setting up SAN B right now, just validating connectivity across one SAN is enough (for now).

1. From the “EC2 Dashboard”, click **Launch Instance**
2. Provide a “Name” for the instance (e.g., Host1)
3. Under “Application and OS Images (Amazon Machine Image)”, click on ubuntu and then select a particular version. I chose “Ubuntu Server 20.04 LTS (HVM), SSD Volume Type” because it was “Free Tier Eligible”, when we actually create the instances we will use for NVMe/TCP testing, we will make different choices that will have a different cost profile. Leave Architecture set to **64-bit (x86)**
4. Under “Instance type”, select **micro** again because it is Free tier eligible.
5. Under “key pair (login), select **Create new key pair**
  - a. In the Create key pair dialog, provide a Key Pair name, I chose to use **SANdbosBlog**
  - b. Select a “Key pair type” of **RSA**
  - c. Select a “Private key file format” of **.ppk**
  - d. Click **Create key pair** and a file (e.g., SANdbosBlog.ppk) will be downloaded to your system. You will need this file when we connect to the instance via PuTTY in a subsequent step.
  - e. Ensure that your key name (e.g., SANdbosBlog) is shown in the **Key pair name - required**
6. Under “Network Settings” click on **Edit**
  - a. Under “VPC”, ensure the correct VPC is selected (e.g., SANdbosBlog-vpc)
  - b. Under “Subnet”, ensure your public network is selected (e.g., **SANdbosBlog-subnet-public1-us-east-1a**)

- c. Under “Auto-assign public IP”, ensure it is set to **Disable**
  - d. Under “Firewall (security groups)”, click on “**Select existing security group**”
  - e. Under “Common security groups”, leave it **blank**
  - f. Click **Advanced network configuration**
  - g. Under “Network Interface 1”
    - a. Set the “Description” to **Public-LAN**
    - b. Under the “Security groups” pull down menu, choose **LAN-Public**
    - c. Leave everything else at the default setting
  - h. Click **Add network interface**
    - a. Set the “Description” to **SAN A**
    - b. Under the “Subnet” pull down menu, select the **SAN A subnet**
    - c. Under the “Security groups” pull down menu, choose **SAN A**
    - d. Leave everything else at the default setting
  - i. Note, this instance type only supports two network interfaces, so we will only configure SAN A for now.
7. Under the Summary tab (on the right side of the console)
- a. For Number of instances, specify 2
  - b. Then click **Launch instance**
  - c. Click **View all instances** when the button becomes available.

## Take an inventory of your Virtual Infrastructure

1. Now that your EC2 instances are up, ensure that you name them appropriately, I chose **Host 1** and **Host 2**. You can rename them in the EC2 Dashboard by clicking in the **Name** field.
2. For each Host, take note of at least the last four digits of the instance ID.
3. For each Host, click on the **Instance ID** and select the **Networking** tab
4. Scroll down to the Network Interfaces section and take note of the last four digits of each Interface ID (you may need to expand the Interface ID column to see the last four digits). You'll also want to understand what network each interface is associated with, and the Description field should answer this question for you.
5. You'll also want to take note of the IPv4 addresses for the SAN A interfaces. We will be using them to verify connectivity in the final step.
6. I highly recommend keeping track these instance and interface IDs because we will use them later. Here's what my configuration looks like at this point. Note that the IP address for the LAN-Public interfaces are Elastic 1 and Elastic 2, we will actually assign them next.



## Create and Assign Elastic IPs

If I was only using a single network interface per instance, I could have had a public IP Address assigned to each interface. However, because this functionality is not supported when you have more than one network interface on an instance, I've found the easiest way to get access to my instances is to use Elastic IPs. Unfortunately, Elastic IPs are not free, so please be advised.

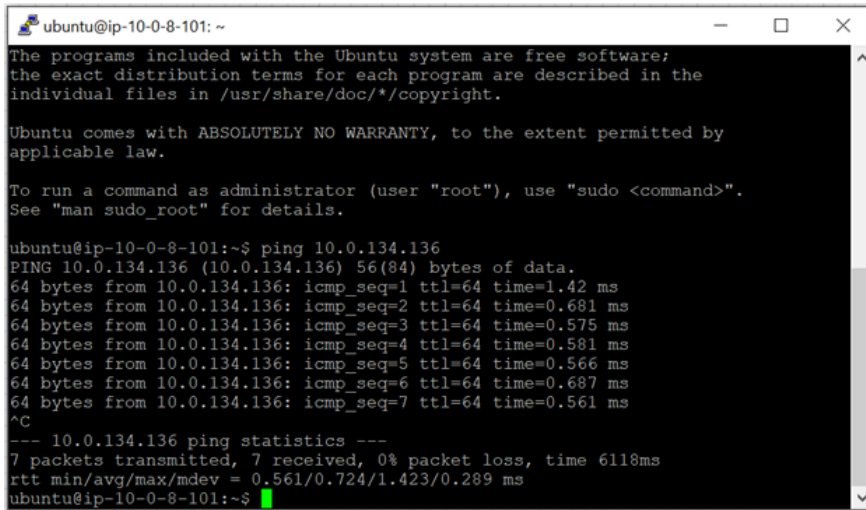
1. Under "Network & Security" select **Elastic IPs** and then click on **Allocate Elastic IP address**
2. Click Allocate and you will be returned to the Elastic IP Address screen. Take note of the Allocated IPv4 address. In my case it is 3.230.foo.bar
3. Select the IP Address (little check box to the left of the IP), click **Actions** and then **Associate Elastic IP address**
4. Under resource type, choose **Network interface** and then click inside of the Network Interface box and select the Public-LAN interface of Host 1. BTW, you need to use the resource ID of the interface and this is why it is so important to keep an updated diagram of your virtual infrastructure as you create it. In my case the last four digits of the interface on Host 1 that is connected to Public-LAN is 20d1. Clicked on the correct interface ID and then click **Associate**.
5. Because you need two Elastic IP addresses, click on **Allocate Elastic IP address** a second time and repeat the above steps for the Public-LAN interface on Host 2. My second Elastic IP was 34.234.you.too

## Connect to your Instances

To complete this next step you're going to need the SANdbosBlog.ppk file that was downloaded to your system when you were creating your instances. You're also going to need PuTTY installed on your system as well.

1. Open PuTTY
2. In the **Host Name (or IP Address)** field enter the Elastic IP Address associated with Host 1 (e.g., 3.230.foo.bar)
3. Under connection -> Data enter the default user name for ubuntu in AWS (**ubuntu**)
4. Under connection -> SSH -> Auth, browse to the location of the SANdbosBlog.ppk file.

5. Click Open
6. Accept the warning about the Host Key
7. And you should be logged into your Host 1 instance.
8. Ping the IP Address of the SAN A interface on Host 2, in my case, this IP Address is 10.0.134.136 and verify that it was successful

A terminal window titled 'ubuntu@ip-10-0-8-101: ~' with standard window controls. It displays the Ubuntu welcome message, warranty disclaimer, and instructions on using 'sudo'. The user then runs 'ping 10.0.134.136', showing successful results with 7 packets received and 0% loss. The window has a dark background and a light border.

```
ubuntu@ip-10-0-8-101: ~
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-8-101:~$ ping 10.0.134.136
PING 10.0.134.136 (10.0.134.136) 56(84) bytes of data.
64 bytes from 10.0.134.136: icmp_seq=1 ttl=64 time=1.42 ms
64 bytes from 10.0.134.136: icmp_seq=2 ttl=64 time=0.681 ms
64 bytes from 10.0.134.136: icmp_seq=3 ttl=64 time=0.575 ms
64 bytes from 10.0.134.136: icmp_seq=4 ttl=64 time=0.581 ms
64 bytes from 10.0.134.136: icmp_seq=5 ttl=64 time=0.566 ms
64 bytes from 10.0.134.136: icmp_seq=6 ttl=64 time=0.687 ms
64 bytes from 10.0.134.136: icmp_seq=7 ttl=64 time=0.561 ms
^C
--- 10.0.134.136 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6118ms
rtt min/avg/max/mdev = 0.561/0.724/1.423/0.289 ms
ubuntu@ip-10-0-8-101:~$
```

## Next steps

So far we've created a Free AWS account, configured MFA, setup billing alerts, created a VPC with three networks, created 3 security groups and verified basic connectivity to and within our VPC using a pair of free tier servers.

Next time we're going to deploy 3 larger EC2 instances, and install:

1. nvme-stas on our Host EC2 instance,
2. SmartFabric Storage Software on our Centralized Discovery Controller (CDC) EC2 instance, and
3. SPDK on our Storge EC2 instance.

Thanks for reading!

---

Posted at 04:42 PM | [Permalink](#)

 Reblog (0)

---

## Comments

 You can follow this conversation by subscribing to the [comment feed](#) for this post.

The comments to this entry are closed.

[Brass Tacks](#)   [Powered by TypePad](#)