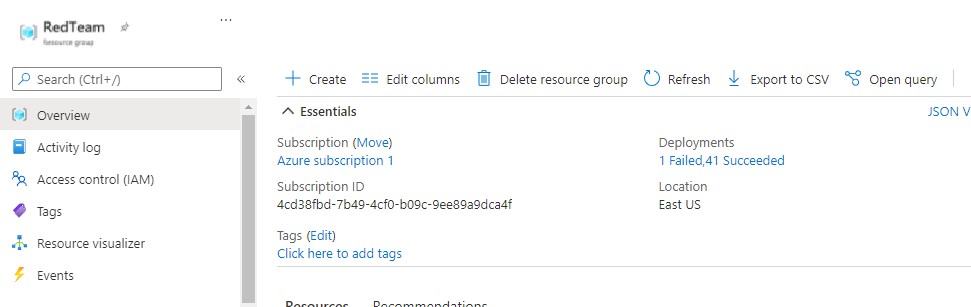
**A description of deployment**

# Setting up the Cloud Environment

## Create Resource Group

● Create a resource group and a region, I chose US East region:

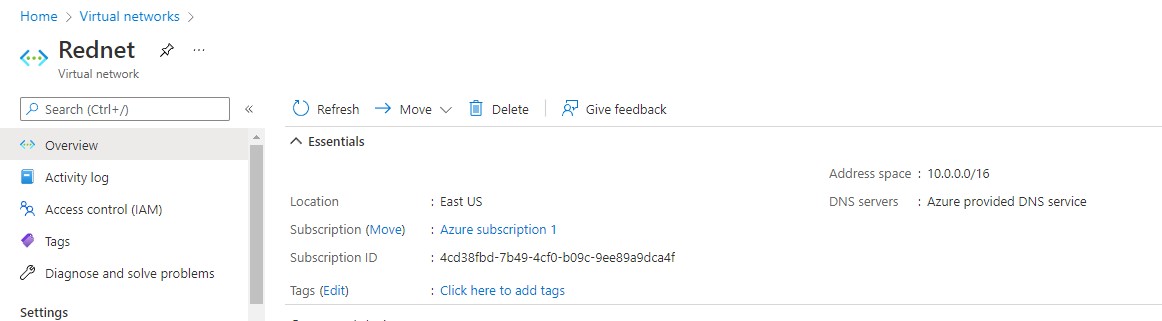
○ Named as Resource Group RedTeam.



## Create Vnet (Virtual Network)

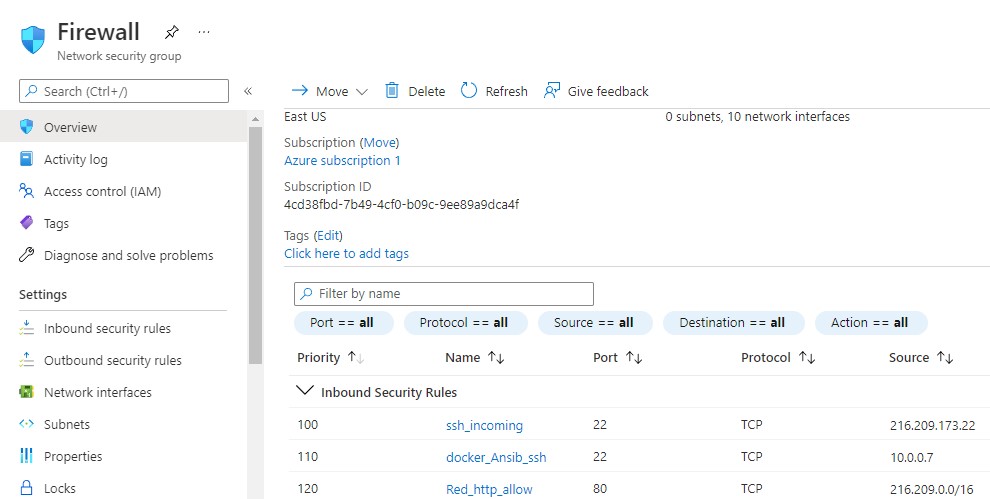
* Create a new Virtual Network.
* Make sure to select the resource group we created previously as well as the same region:

○ Also, use the default IP and subnet settings.



## Create a Network Security Group (NSG)

* Create a Network Security Group that is part of the Resource Group RedTeam.
* Named my NSG "Firewall".
* Make sure the NSG is in the same region as everything else we've created thus far.

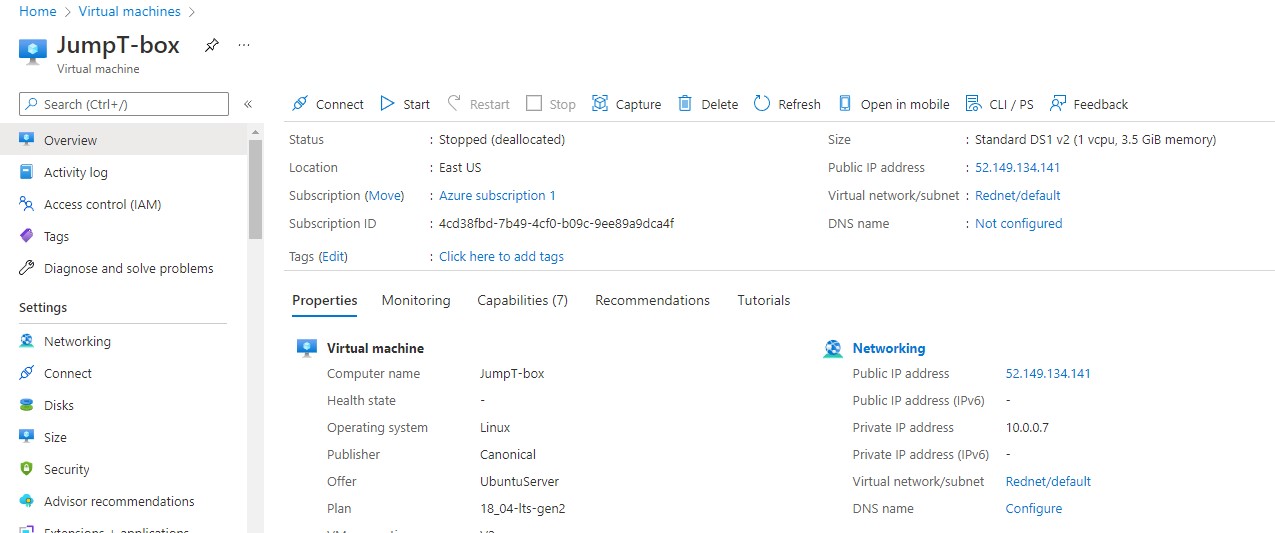


# Create the Virtual Machines

## Create Jump Box VM

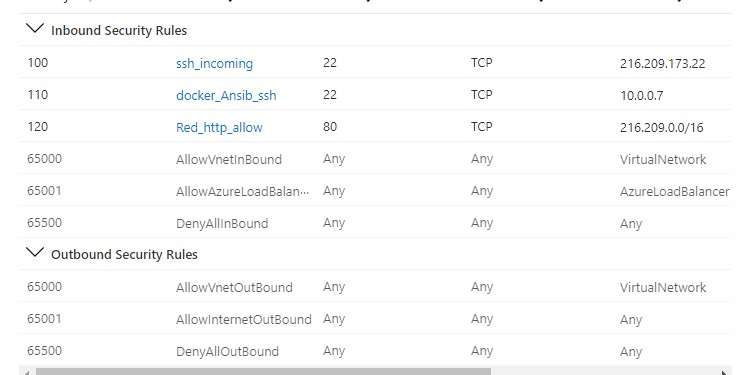
* Log in to Azure account.
* Click on the virtual machines box and then click new.
* Under the resource group, select the group RedTeam.
* Named the VM JumpT-Box.
* Use a Ubuntu server with at least 1GB of memory.
* Use a public SSH key from from local computer and give it a username ● Use "ssh-keygen" to create a public key that doesn't have one.

○ My username is sysadmin.



## Network Security Group Rules

● This is an overview of all of the inbound and outbound rules for the Firewall NSG:



## Set up Docker.io on the Jump Box VM

* SSH into your Jump-Box VM, turn on your machine on Azure before that: ssh sysadmin@[public IP]
* Once logged in, implement the following:

○ sudo apt install docker.io

○ sudo docker pull cyberxsecurity/ansible

○ Launch the ansible container: docker run -ti cyberxsecurity/ansible:latest bash to make sure it works

○ type exit

## Config and Hosts File

* cd into the /etc/ansible/ directory and nano ansible.cfg file
  1. scroll to the remote\_user section and update to include sysadmin instead of root. Save and exit.
* nano /etc/ansible/hosts file
  1. Uncomment the [webservers] header

○ Under the header, add the internal IP address of the 3 VMs:

■ 10.0.0.11 ansible\_python\_interpreter=/usr/bin/python3

■ 10.0.0.12 ansible\_python\_interpreter=/usr/bin/python3

■ 10.0.0.13 ansible\_python\_interpreter=/usr/bin/python3

## Create 3 Virtual Machines

* We will create 3 additional virtual machines that will be web servers.
* We will name them Web-1, Web-2, and Web-3.
* Follow this criteria:
  1. Allow no public IP address.

○ Create a new availability set, I called mine webset, set the 3 VMs to this.

○ Connect your VMs to the RedNet VNet and to the Firewall NSG.

* Use a public SSH key from the JumpT-Box VM docker container and give it a username .
  1. Use "ssh-keygen" to create a public key if you don't have one. ○ username is sysadmin.
* To make sure it works:
  1. SSH into the Jump Box VM

○ Start and attach your docker container: sudo docker start “name of the container” && sudo docker attack “name of the container”

○ Once in the container, SSH into each VM to make sure they work:

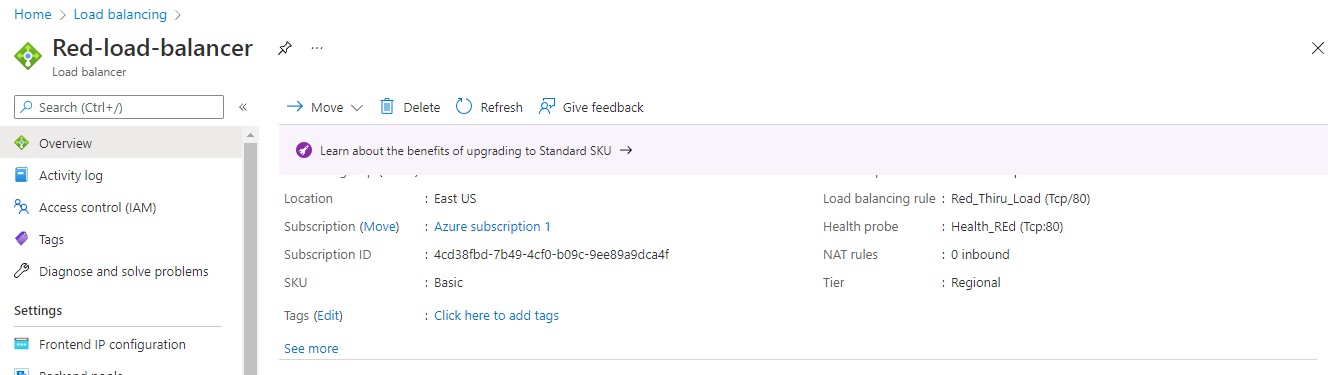
■ ssh sysadmin@10.0.0.12

■ ssh sysadmin@10.0.0.13

■ ssh sysadmin@10.0.0.14

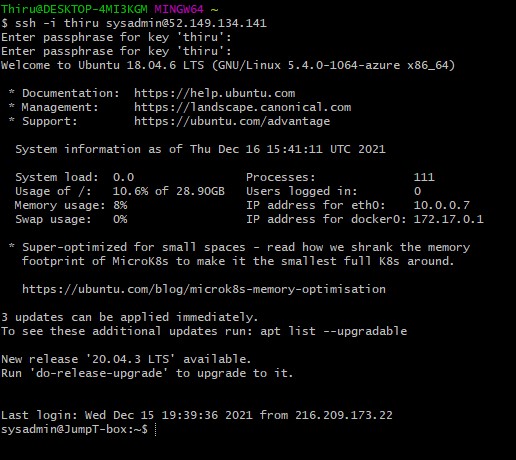
# Load Balancer

* Create a new Load Balancer in Azure.
* Select static IP address and select the same Resource Group and region.
* Select create new public IP address.
* I named my Load Balancer: Red-Team-LB
* Create a new Backend Pool and add the 3 VMs to it.



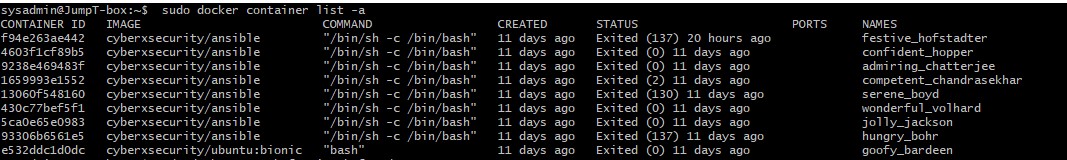
# Logging into Jump Box Provisioner

* Log in to Azure and turn on your Jump Box Provisioner virtual machine.
* Open your personal computer terminal

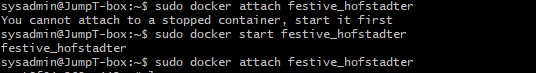


# Starting Docker

* Check to see which containers you have:



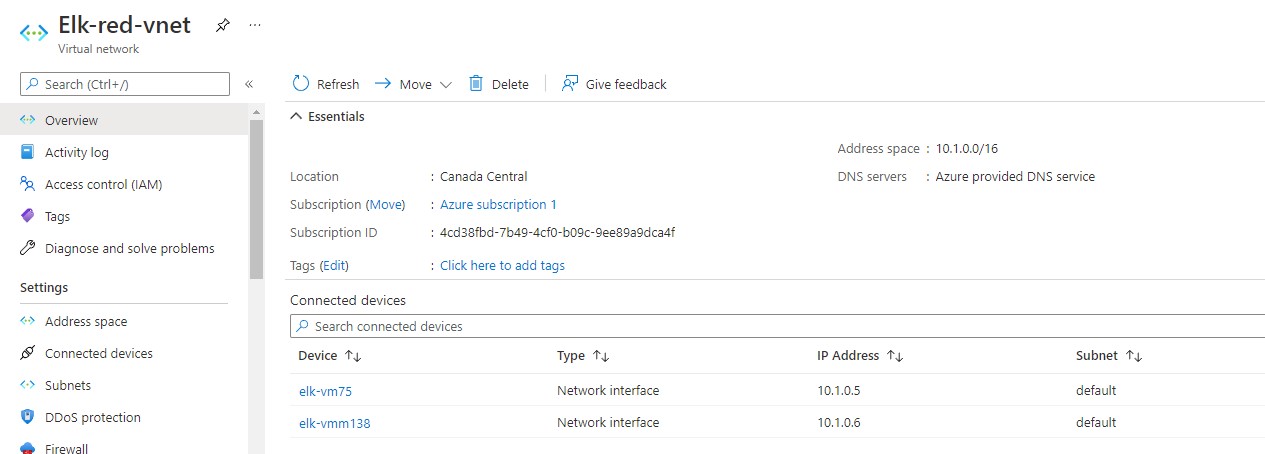
* my container is: festive\_hofstadter
* To start your container, use the following commands:



# Install ELK Stack

## Create a new VNet (Virtual Network)

* Create a new VNet that is in the same resource group (RedTeam) but in a different region.
* I created one and named it Canadacentral:



Create a peer connection between the two VNets (RedNet and Elk-red-net)

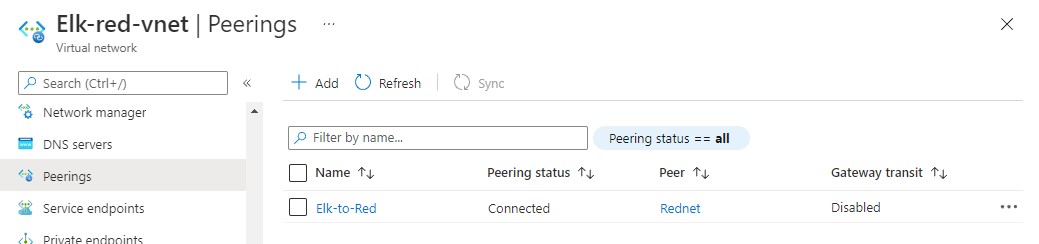
* In my Elk-red-vnet page, I clicked on Peerings and added a new Peering with the following settings:

○ I created a new connection from EastNet to RedNet and called it:

Elk-to-Red, connecting to my RedNet VNet.

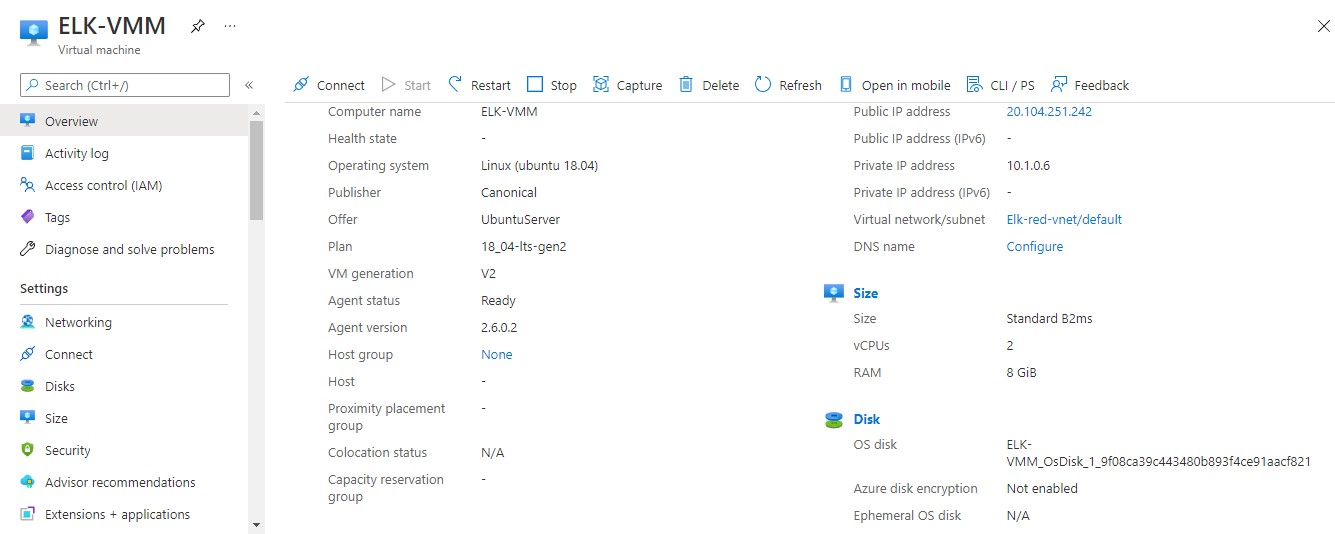
○ I created another connection from RedNet to EastNet and called it:

Red-to-Elk.

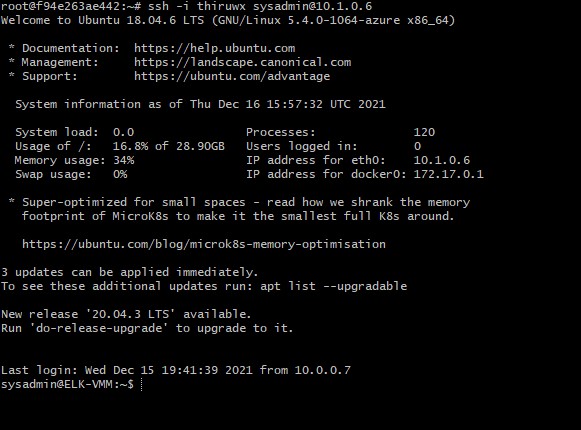


## Create a new Virtual Machine

* Create a new Ubuntu VM with a 8GB minimum RAM size.
* The IP address will be public.
* The VNet will be Elk-red-vnet, and the resource group will be the one we have for the other VMs, which is RedTeam.
* I used the public key from the ansible container and my username as sysadmin.
* I named my VM: ELK.

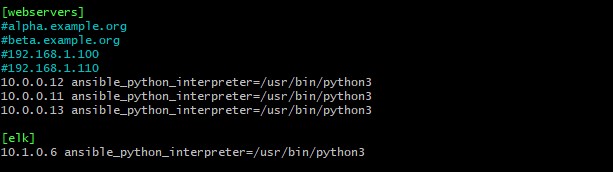


* Once the VM is created, I ssh'd into the VM to make sure it works:



## Download and Configure the Container

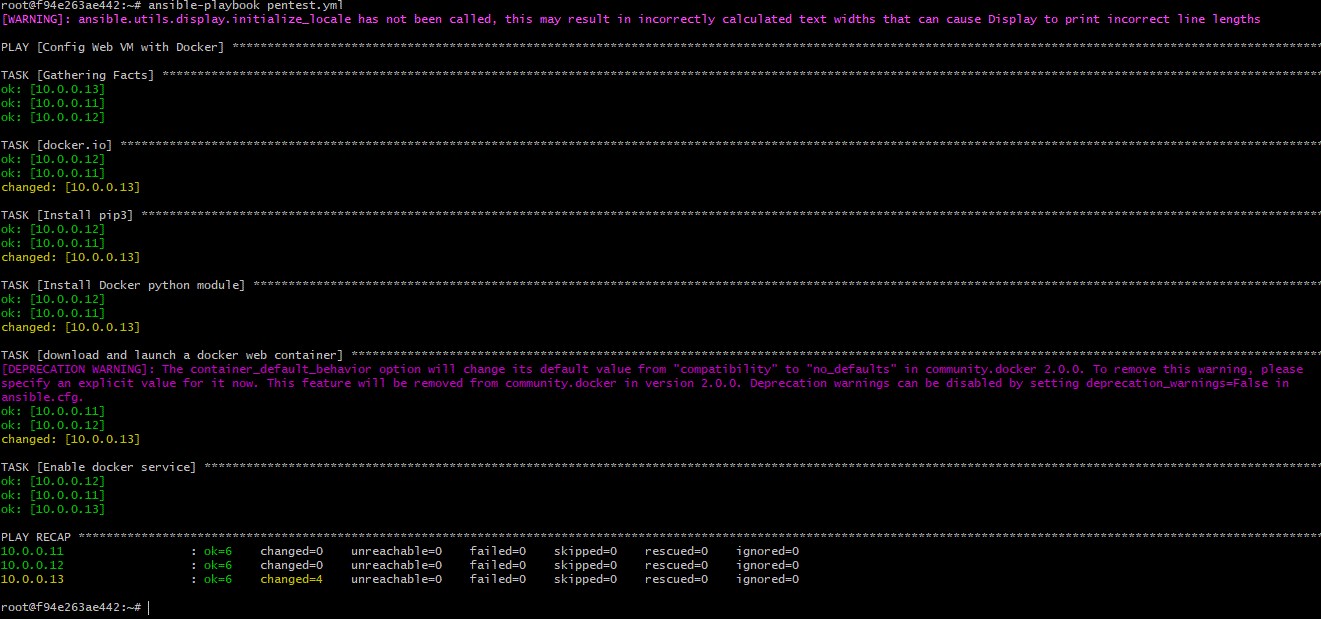
● We will add the new VM IP address to the hosts file in the ansible container ● Edited the hosts file and added the elk IP address under the [elk] group:



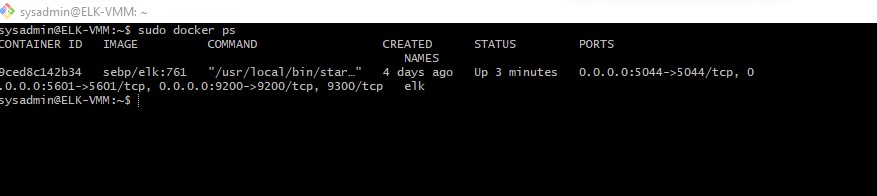
Created a playbook that will configure the ELK server, named it "install-elk.yml":

## Launched and Exposed the Container

* Ran the yaml playbook, made sure it worked:

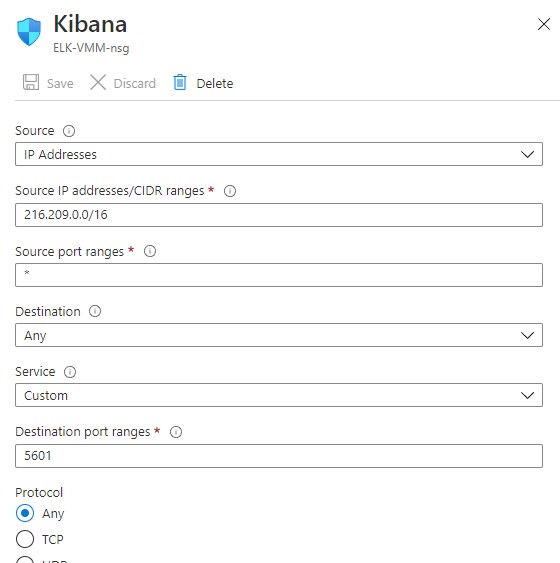


* As we can see from above, the container was successfully created, with the image "sebp/elk:761".



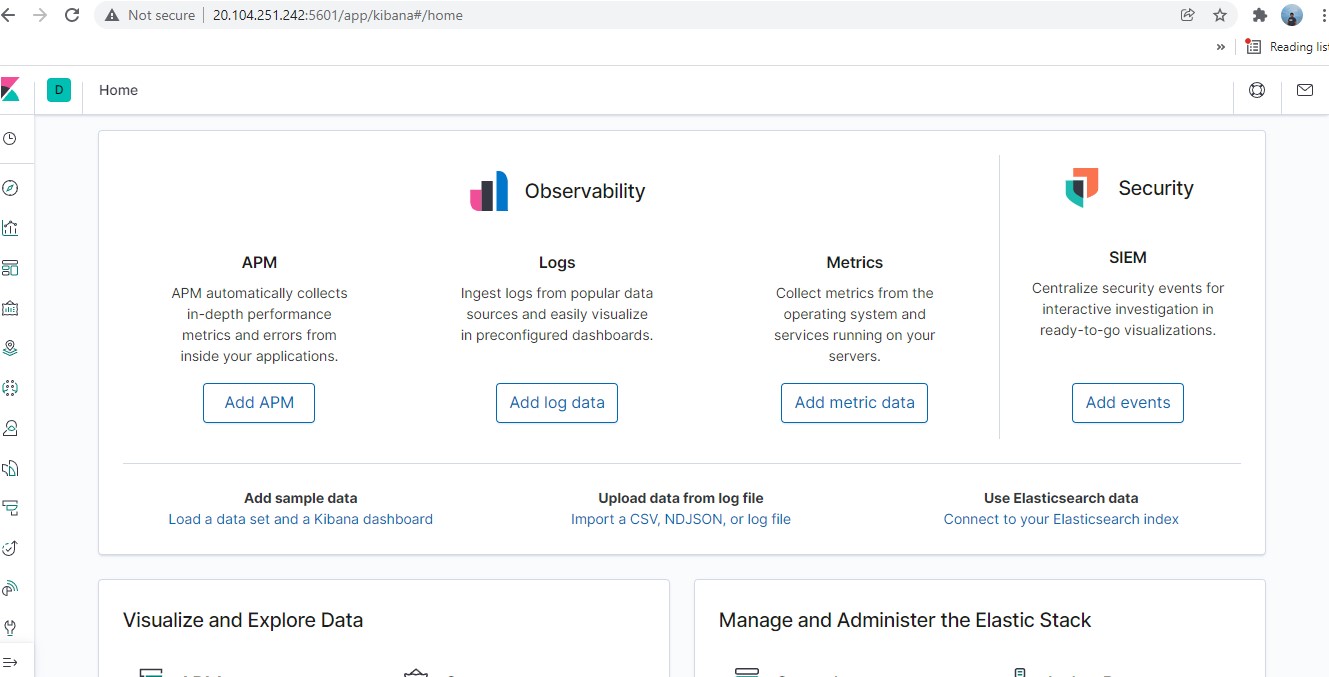
## Identity and Access Management

* We are going to restrict access to the ELK-VMM through the ELK Network Security Group:
* Once in the NSG for ELK, we are going to add an inbound rule that will allow access from our computer to the ELK server on port 5601:



Finally, we will verify that we can log into the server by accessing on our browser, [ELK-public-IP]:5601/app/kibana:

○ Note: the public IP will always change every time we restart it.

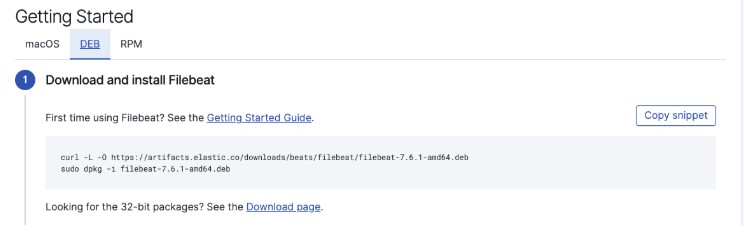


# Install Filebeat

● We can use Filebeat to collect, parse, and visualize ELK logs in a single command. This will help us better track our organizational goals.

## Install Filebeat on the container

* First, we will start the virtual machines (including the ELK server) on Azure.
* Then, we will access the kibana page and make sure it works.
* We will start the container within the jump box vm:
* I jumped back to the kibana page and found the DEB page for creating a system log and will use this guide to create our filebeat playbook.



## Create Filebeat configuration file

* Once the folder is created, I will download the filebeat-config.yml:
* I opened the filebeat-confi.yml file and added the ELK server private IP address in two areas:

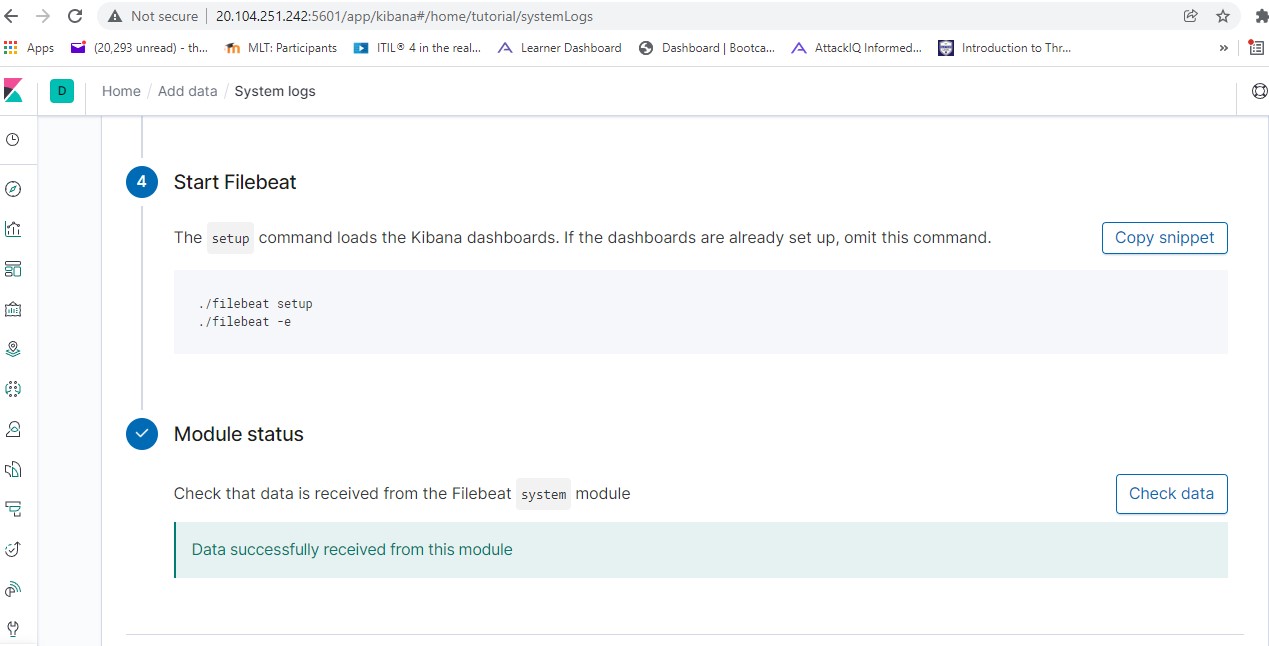
output.elasticsearch: hosts: ["10.1.0.6:9200"] username: "elastic" password: "changeme"

setup.kibana:

host: "10.1.0.6:5601"

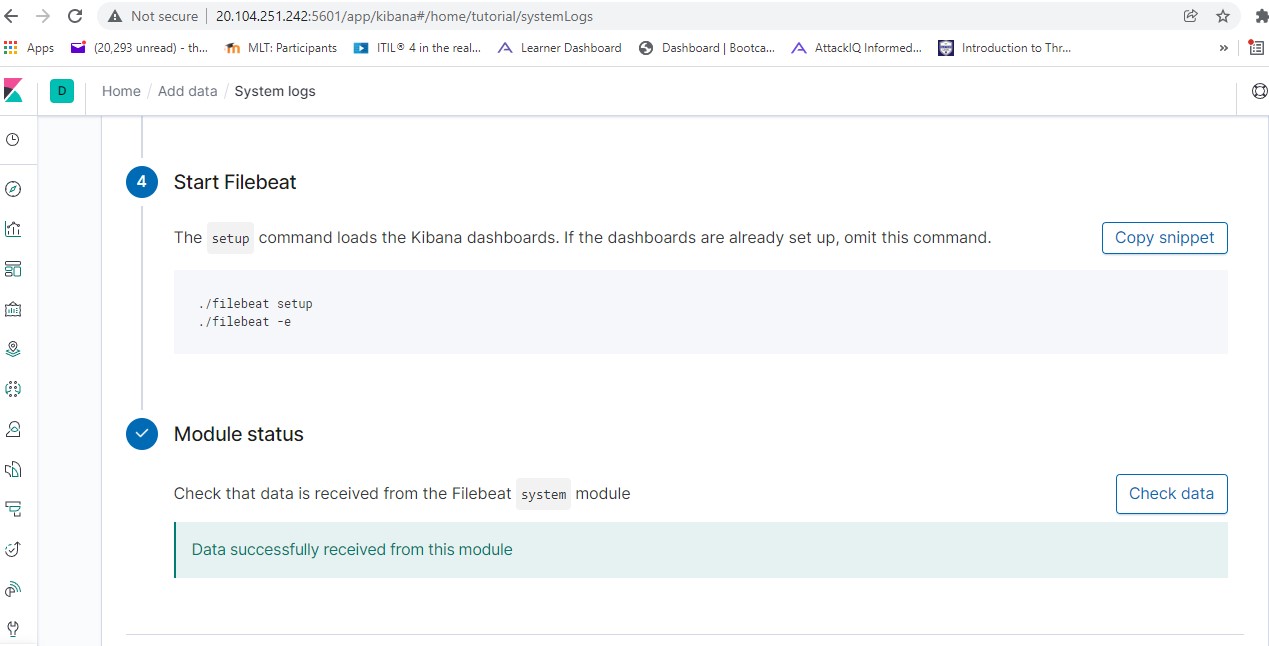
## Create Filebeat Installation Playbook

* I created the filebeat-playbook.yml under /etc/ansible/files.
* Once created, using the DEB page, I added the needed commands:

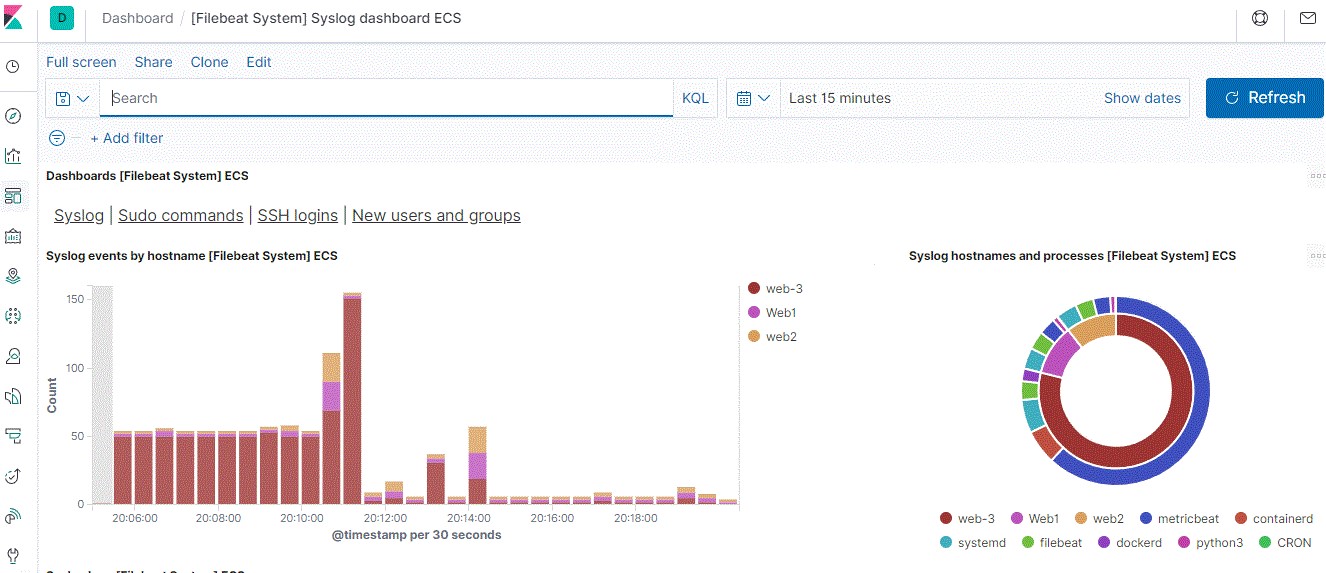


## Verify Installation and Playbook

* Back on the kibana page, I will check the data on the instruction page:



* Finally, I checked the dashboard to make sure things are running:



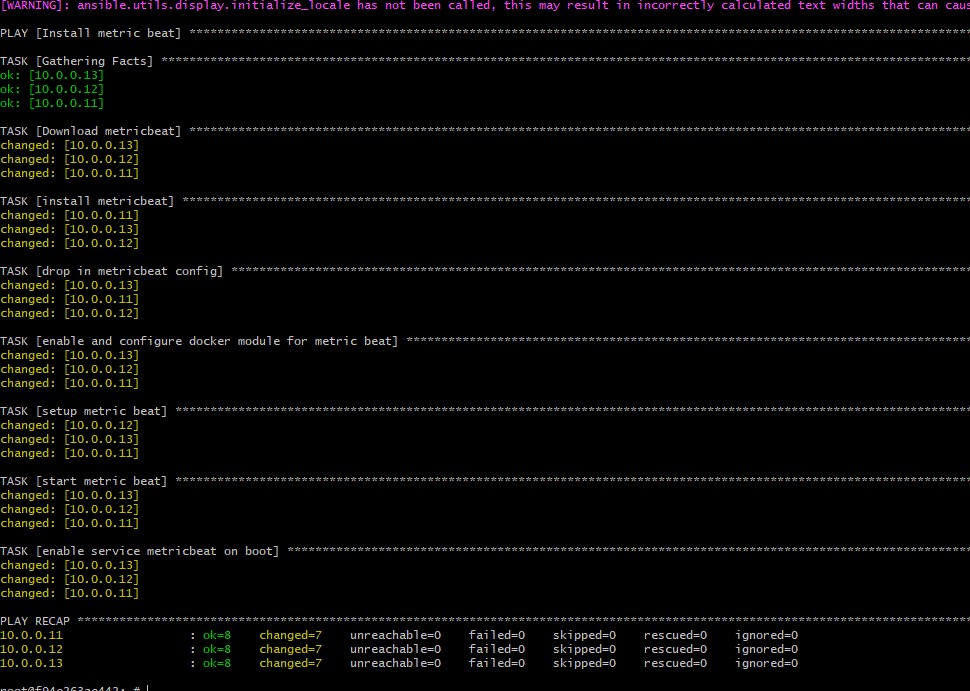
## Create a play to install Metricbeat

* Using the same steps for Filebeat, I created a config file for Metricbeat and added ELK's private IP in two areas:

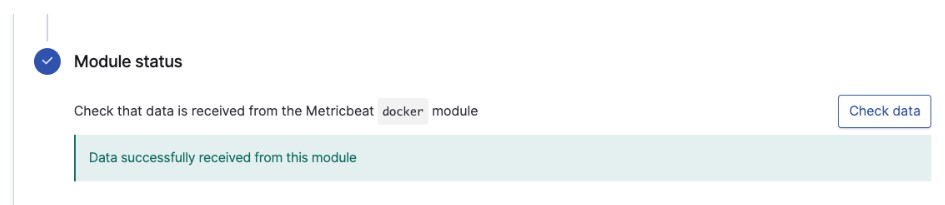
output.elasticsearch: hosts: ["10.1.0.6:9200"] username: "elastic" password: "changeme"

setup.kibana: host: "10.1.0.6:5601"

* Under the files folder, I will create the metricbeat-playbook.yml:
* Once both playbooks were created, I ran the metribeat-playbook.yml:



* Now that the playbook ran successfully, I will check the data on the kibana page:



* Finally, we will check the dashboard, to make sure it works:

