

The goal of this activity is to create a highly available web server for XCorp's Red Team for training and testing purposes. We will implement IaaS to create a cloud infrastructure environment that will consist of a jump box provisioner and 2 virtual machines running DVWA that are both receiving traffic from a load balancer. In addition, we will create a cloud monitoring system by installing and configuring an ELK stack server.

I.) Creating the Virtual Network Environment

A.) Setting up the Resource Group

- 1.) We will create a logical grouping of network assets in Microsoft Azure that will help incorporate automatic provisioning, network monitoring, and access control in our virtual environment by first creating a resource group. This resource group will include a logical network, firewall, virtual computer, and other network resources that will provide the Red Team with the components required to maintain a secure cloud infrastructure.
- 2.) Within the Azure portal, we will click on Resource Groups on the home screen, select the create resource group button, specify the name of our resource group as RedTeam, and ensure the region is selected as East US.

Create a resource group ...

[Basics](#) [Tags](#) [Review + create](#)

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more ↗](#)

Project details

Subscription *	<input type="text" value="Azure subscription 1"/>
Resource group *	<input type="text" value="RedTeam"/> ✓

Resource details

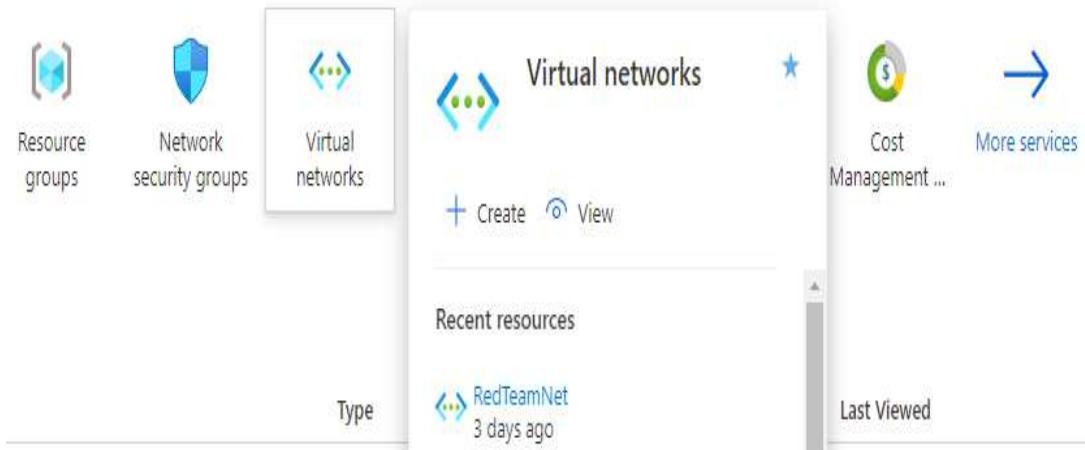
Region *	<input type="text" value="(US) East US"/>
----------	---

B.) Setting up the Virtual Network

- 1.) We will now create a virtual network within Azure that will provide an isolated environment for our virtual machines to communicate securely with each other and outside networks. This

virtual network will allow us to deploy servers, virtual machines, and services securely from an outside network.

- 2.) From the home screen within the Azure portal, select the Virtual networks icon and click on the create button.



- 3.) Within the Basics tab of the VM creation page, we will attach our subscription to the RedTeam resource group created in the first step and ensure it is in the same East US region.

The screenshot shows the 'Create virtual network' wizard with the 'Basics' tab selected. The page has a header 'Create virtual network' with a '...' button. Below the header are five tabs: 'Basics' (selected), 'IP Addresses', 'Security', 'Tags', and 'Review + create'. A descriptive paragraph explains that Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure, enabling communication between Azure resources and on-premises networks. It links to 'Learn more about virtual network'. The 'Project details' section contains fields for 'Subscription' (set to 'Azure subscription 1') and 'Resource group' (set to 'RedTeam'). The 'Instance details' section contains fields for 'Name' (set to 'RedTeamNet') and 'Region' (set to 'East US').

- 4.) Next, we will define our network and subnet within the IP Addresses tab and use the default settings provided by Azure. Our virtual network will be defined by the IP range 10.0.0.0/16. Within that network we will also have a subnet that will be defined by the IP range 10.0.0.0/24.

- 5.) The last step in creating our virtual network is to configure the security settings of the network. Based on our current subscription, we will incorporate the default settings provided by Azure and disable the BastionHost, DDoS Protection Standard, and Firewall settings. In a real-world setting, consideration should be given to enabling these settings to provide enhanced network security to the environment based on the security requirement needs of the organization.

Create virtual network



- 6.) We will click on create and review to review to ensure our configuration of the virtual network is correct. Once everything looks sufficient, we will click on the create button to deploy our virtual network.

C.) Creating the Network Security Group

- 1.) Our goal is to have virtual machines running inside our virtual network. Before deploying any VMs in our cloud environment, we must ensure that we have a mechanism in place that will monitor the flow of inbound and outbound traffic in our newly created virtual network and the virtual machines running on that network. In Azure, a network security group (NSG) will provide the same functionality as an on-premises firewall by blocking and allowing specified traffic to our virtual network and virtual machines. The steps to follow will be used to create a NSG and configure rules to control traffic coming into the network.

- 2.) We will click on the create network security group option within the Azure portal, attached the security group to our RedTeam resource group, and name our NSG RedTeamSG.

The screenshot shows the 'Create network security group' wizard in the Azure portal. The top navigation bar includes 'Home > Network security groups >' followed by the title 'Create network security group' and a '...' button. Below the title are three tabs: 'Basics' (which is selected), 'Tags', and 'Review + create'. Under the 'Project details' section, 'Subscription *' is set to 'Azure subscription 1' and 'Resource group *' is set to 'RedTeam' (with a 'Create new' link). In the 'Instance details' section, 'Name *' is set to 'RedTeamSG' and 'Region *' is set to 'East US'.

- 3.) The previous step helped create the NSG for our RedTeam resource group. In this step we will apply inbound security rules to restrict the flow of traffic coming into our network. We will first create a default deny inbound security rule that will block all source IP addresses and all source port ranges to any destination addresses on the network using any protocol. This default deny rule will ensure that no malicious actors can access the network while it is safely being configured to our required specifications.

The screenshot shows the configuration interface for an inbound security rule. The fields are as follows:

- Source: Any
- Source port ranges: *
- Destination: Any
- Service: Custom
- Destination port ranges: *
- Protocol: Any (selected)
- Action: Deny (selected)
- Priority: 4020
- Name: Default_Deny
- Description: Deny All inbound Traffic

- 4.) In the previous inbound security rule created, we denied all traffic to our virtual network. However, we will need access to our virtual network in order to configure and deploy virtual machines and services in our environment. To accomplish this, we will create a new inbound security rule that will allow traffic only from our external host workstation to connect to the virtual network via RDP. This rule will specify the source as our public IP address and allow access from any source port range. At the same time, we will specify the destination as our virtual network using port 3389 for HTTP traffic. We want this rule to be recognized before our default deny rule so we will set a lower priority number than 4020, which was our priority number for our default deny rule, to ensure it is applied first.

 Add inbound security rule X

Source (i)
IP Addresses ▼

Source IP addresses/CIDR ranges * (i)
100.15.100.74 ✓

Source port ranges * (i)
* ▼

Destination (i)
VirtualNetwork ▼

Service (i)
RDP ▼

Destination port ranges (i)
3389 ▼

Protocol
 Any
 TCP
 UDP
 ICMP

Action
 Allow
 Deny

Priority * (i)
4000 ✓

Name * (i)
Allow_RDP ✓

Description
Allow RDP from local workstation over port 3389 ✓

D.) Setting up the Jump Box Provisioner

- 1.) At this point we have created a virtual network that is protected by our NSG that allows traffic from our local workstation while blocking all other incoming traffic to the network. In the steps to follow we will create a virtual machine that will be our jump box provisioner that will allow us to configure other virtual machines in the network.

- 2.) We will create our jump box provisioner by allocating a software version of various components consisting of RAM, storage, OS disks, and a CPU. Within the Azure portal, we will click on the create VM button to begin the creation of our jump box provisioner and use the following settings to configure the VM.

- Assign the jump box VM to our East US region
- Specify no infrastructure redundancy required
- Select the Ubuntu Server 18.04 LTS image
- Allocate 1 vCPU and 1 GiB of memory to the VM
- Select SSH public key as authentication type
- Assign the VM to our RedTeam virtual network created
- Select the subnet created within the VNet created
- Use the default public IP address provided by Azure
- Choose the advanced option within the NIC network security group setting
- Assign the created RedTeam security group to the VM
- Ensure accelerated networking is kept at its default setting of off
- Confirm the load balancing selection is kept off

Once all these settings have been configured properly, the review and create option should resemble the following:

The screenshot shows the Azure portal interface for managing a virtual machine named 'Jump-Box-Provisioner'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Networking, Connect, Disks, Size, Security, Advisor recommendations, Extensions, Continuous delivery, Availability + scaling, Configuration, Identity, Properties, Locks, Operations, Bastion, Auto-shutdown, Backup, Disaster recovery, Guest + host updates, Inventory, Change tracking, and Configuration management (Preview).

The main content area displays the 'Essentials' tab with the following details:

Setting	Value
Resource group (Move)	: REDTEAM
Status	: Stopped (deallocated)
Location	: East US
Subscription (Move)	: Azure subscription 1
Subscription ID	: 38cb30e6-ce8b-411d-a2d6-cba69411c14c
Tags (Edit)	: Click here to add tags
Operating system	: Linux
Size	: Standard B1s (1 vcpus, 1 GiB memory)
Public IP address	: 40.112.62.73
Virtual network/subnet	: RedTeamNet/default
DNS name	: Not configured

The 'Properties' tab is selected, showing the following configuration details:

Category	Setting	Value	
Virtual machine	Computer name	Jump-Box-Provisioner	
	Health state	-	
	Operating system	Linux	
	Publisher	Canonical	
	Offer	UbuntuServer	
	Plan	18.04-LTS	
	VM generation	V1	
	Host group	None	
	Host	-	
	Proximity placement group	-	
Colocation status	N/A		
Capacity reservation group	-		
Networking	Public IP address	40.112.62.73	
	Public IP address (IPv6)	-	
	Private IP address	10.0.0.4	
	Private IP address (IPv6)	-	
	Virtual network/subnet	RedTeamNet/default	
	DNS name	Configure	
	Size	Standard B1s	
	vCPUs	1	
	RAM	1 GiB	
	Disk	OS disk	Jump-Box-Provisioner_disk1_89b154a29d904ea1bae6032911b174fa
Azure disk encryption		Not enabled	
Ephemeral OS disk		N/A	
Data disks		0	
Security type		Azure Spot	-
		Azure Spot	-
		Azure Spot eviction policy	-
		Trusted launch	Disabled
		Guest + host updates	-
		Inventory	-
	Change tracking	-	
	Configuration management (Preview)	-	
	enablevmaccess	-	

E.) Establishing an SSH Connection to the Jump Box

- 1.) We will need a mechanism that will allow us to use the jump box to access our virtual environment and configure all virtual machines inside our cloud infrastructure. Due to the security limitations of password authentication and their susceptibility to be brute forced, we will control access to our jump box server with SSH keys. To accomplish this, we will first run the command ssh-keygen within our gitbash terminal to create our SSH Key pair.

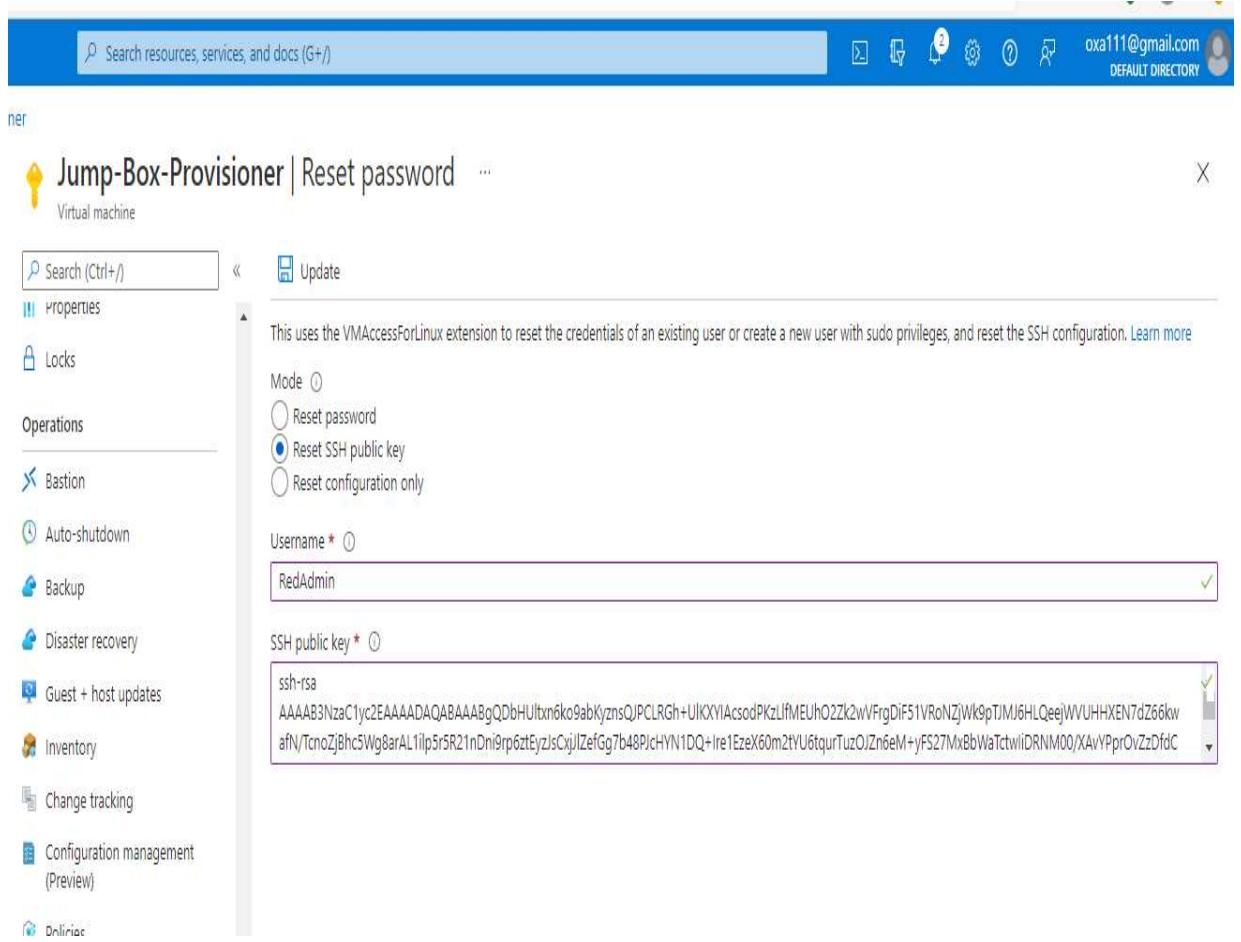
```
Omar@DESKTOP-6IMQLP0 MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Omar/.ssh/id_rsa):
/c/Users/Omar/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Omar/.ssh/id_rsa
Your public key has been saved in /c/Users/Omar/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Jh5iAWD1f6xmqis3+mCxObchPXiNKFj6VcTIHCQG4yo Omar@DESKTOP-6IMQLP0
The key's randomart image is:
+-- [RSA 3072] ---+
|+++=oo
|oo =.+
| . =.o
| . o. .
|E.. o +.so
|ooB.o+ +o
|+X B...+
|oo*++ +
| .*=+.
+--- [SHA256] -----+
```

Now we will run the cat ~/ssh/id_rsa.pub command within the terminal to display our public key.

```
Omar@DESKTOP-6IMQLP0 MINGW64 ~
$ cat ~/ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQDbHU1txn6ko9abKyznsQJPCLRGh+U1kXYIAcsodPKzL1fMEUhO2Zk2wFrgDiF51VRoNZjwk9ptTJMj6HLQeejwVUHHXEN7dZ66kwafN/TcnoZjBhc5Ng8arAl1iP
5r521nDni9rp6ztEyzJsCxjJ1ZefGg7b48PJcHYN1DQ+Ire1EzeX6m2tYU6tqurTuZ0JZn6eM+yFS27MxBbWaTctwLiDRNM00/XAvYPPr0vZzDfdC8rOR4NogJMdexBzlF5qUG580UKo5/FB+DA0u8p6Nj5WoUMk4
08B107D0MnTCm1Z9W0E2trocE7aD6p101jEYmZ7N/u1nYyZrvCAufyCV06iz81KBVfH4icb2aKwC3yy3xhYJLYgyV5T8Z28F02Tu7augxJ0XGHRB1ZUfTwq76FzR0mCmW11ebY2Hoqzc5/81Nx+zbbU5CjEMvScnP
jHTFg1V1QeVhSAgoyakMWuPUX7Hdiah0PEhs20b31iQ84ggM7oHxbryU1Q0U= Omar@DESKTOP-6IMQLP0

Omar@DESKTOP-6IMQLP0 MINGW64 ~
$ ssh RedAdmin@40.112.62.73
RedAdmin@40.112.62.73: Permission denied (publickey).
```

To ensure this newly created public key can we used to access our jump box provisioner, we will copy and paste the key to the SSH public key field within the jump box VM



- 2.) We now have successfully added our SSH Key to our jump box provisioner VM. Even though our key has been added to the VM, we still cannot access the machine since we have not updated our NSG to allow inbound SSH connections. It is important to know that this was by design to prevent any malicious attackers to access the VM during our configuration process. Now that the SSH key has been added to our jump box VM, we will now update our NSG to allow inbound SSH traffic from only the IP address of our local workstation. To ensure this is configured properly, we will add an inbound network security rule to allow only our local workstation IP address from any source port to the virtual network destination using SSH port 22 with TCP traffic.

SSH_Allow

RedTeam-SG

Save Discard Delete

Source IP Addresses: 100.15.100.74

Source port ranges: *

Destination VirtualNetwork

Service SSH

Destination port ranges: 22

Protocol: TCP

Action: Allow

Priority: 490

Name: SSH_Allow

Description: Allow SSH to the network

Now we are able to successfully access our jump box provisioner VM through a SSH connection from our local workstation. At the same time, we can ensure that even if our key has been compromised by a malicious attacker, our jump box VM cannot be accessed unless our local machine has also been compromised.

F.) Creating Two Additional Virtual Machines in the Virtual Network and Allowing SSH Connection From the Jump Box Provisioner

- 1.) We will now create 2 additional Ubuntu Server 18.04 LTS VMs inside our Red Team resource group and same US Region. Both of these machines will utilize the same Standard B1ms offering that allocates 1 CPU and 2 GiBs of RAM. We must also ensure that these VMs are in the same availability set so that they both can be added to the load balancer that we will later incorporate into our network. These VMS will also use SSH public key for authentication. In addition, they will be assigned to the same RedTeamNSG and will not have a public IP address. The configuration of the newly created Web-1 and Web-2 should have the below configuration:

Connect Start Restart Stop Capture Delete Refresh Open in mobile CLI / PS
JSON Vie

Essentials

Resource group (Move) : RedTeam	Operating system : Linux (ubuntu 18.04)
Status : Running	Size : Standard B1ms (1 vcpus, 2 GiB memory)
Location : East US	Public IP address : 13.68.250.112
Subscription (Move) : Azure subscription 1	Virtual network/subnet : RedTeamNet/default
Subscription ID : 38cb30e6-ce8b-411d-a2d6-cba69411c14c	DNS name : Not configured
Tags (Edit) : Click here to add tags	

[Properties](#) [Monitoring](#) [Capabilities \(7\)](#) [Recommendations](#) [Tutorials](#)

Virtual machine	Networking
Computer name : Web-1	Public IP address : 13.68.250.112
Health state : -	Public IP address (IPv6) : -
Operating system : Linux (ubuntu 18.04)	Private IP address : 10.0.0.5
Publisher : Canonical	Private IP address (IPv6) : -
Offer : UbuntuServer	Virtual network/subnet : RedTeamNet/default
Plan : 18.04-LTS	DNS name : Configure
VM generation : V1	
Agent status : Ready	Size
Agent version : 2.5.0.2	Size : Standard B1ms
Host group : None	vCPUs : 1
Host : -	RAM : 2 GiB
Proximity placement group : -	Disk
Colocation status : N/A	OS disk : Web-1_OsDisk_1_866002e1f0404456a4b413fa9cf71a72
Capacity reservation group : -	Azure disk encryption : Not enabled
Availability + scaling	Ephemeral OS disk : N/A
Availability zone : -	Data disks : 0
Scale Set : -	Azure Spot
Security type	Azure Spot : -
Trusted launch : Disabled	Azure Spot eviction policy : -
Extensions	
enablevmacces	

- 2.) We will now configure both the Web-1 VM and the Web-2 VM so that they can be accessed and configured via the jump box provisioner by generating a new SSH key pair within the jump box provisioner and adding the key to each of the VMs.

```

RedAdmin@Jump-Box-Provisioner:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/RedAdmin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/RedAdmin/.ssh/id_rsa.
Your public key has been saved in /home/RedAdmin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:j5/1MYVvK/D647IlUs5stNUgBkTlpgjetbxhDATEvHs RedAdmin@Jump-Box-Provisioner
The key's randomart image is:
+---[RSA 2048]---+
| oo o+.. |
| .. o |
| o.. . = . |
| ..+ B = . + |
| ..+ S o o o |
| . E. @.o o |
| . + Oo= o |
| +.*++ . |
| =*+o. |
+---[SHA256]---+
RedAdmin@Jump-Box-Provisioner:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDIAJ+315QhZRMhSnWftFdianKDGNtOk6xPrNaI2Mc2+VvedLWcf728
kLu5rWrRzqbe4Jniuaz70V3QHBBc+20V+ls/Ci6Qi86wlCcM2b0lkzqWp6lDN6323TSCKbxT6jfsgjdHvYA+Xa3UQDp
4cuUWLmb5oWSFRLmazNM7d6/Cm4fVUyvWmLsscXOZpbqBTVD94hd/jMsG7+kLM5wNK/M2r09f/PRFYIUuvAF RedAdmin
RedAdmin@Jump-Box-Provisioner:~$ ssh RedAdmin@192.168.0.6

```

The key has been generated within the jump box provisioner.

The screenshot shows the configuration interface for the VM Access for Linux extension. The main title is "Web-1 | Reset password". On the left, there is a sidebar with various navigation options: Home, Web-1, Properties, Locks, Operations, Bastion, Auto-shutdown, Backup, Disaster recovery, Guest + host updates, Inventory, Change tracking, and Configuration management (Preview). The "Properties" option is currently selected.

The main content area displays the extension settings:

- Mode:** The "Reset SSH public key" option is selected.
- Username:** The value "RedAdmin" is entered in the input field.
- SSH public key:** A large text input field contains the public key string: "AAAAB3NzaC1yc2EAAAQABAAQDIAJ+315QhZRMhSnWftFdianKDGNtOk6xPrNaI2Mc2+VvedLWcf7287ExIGVqUVIZQSQQpZmNyKfgxv7A/O18cdcnkcmQnaYCU5WshMlxGY2kLu5rWrRzqbe4Jniuaz70V3QHBBc+20V+ls/Ci6Qi86wlCcM2b0lkzqWp6lDN6323TSCKbxT6jfsgjdHvYA+Xa3UQDpSjRfpsZ+bi+7+w6CE3vkdzYbzQGDIO3gY/rSDBeegTU2X8ui4prOj84cuUWLmb5oWSFRLmazNM7d6/Cm4fVUyvWmLsscXOZpbqBTVD94hd/jMsG7+kLM5wNK/M2r09f/PRFYIUuvAF RedAdmin@Jump-Box-Provisioner".

The key has been added to Web-1 and Web-2 to allow access from the jump box provisioner.

- 3.) Now that both newly created VMs have been configured with newly generated SSH key, we must now reconfigure our RedTeamSG to allow SSH connections from our jump box provisioner to our Web-1 and Web-2 VMs. To accomplish this, we will create a new inbound security rule that will specify the source IP private address of the jump box provisioner (10.0.0.4), allow traffic from any source port, designating the service as SSH on port 22, and selecting the virtual network as the destination. The newly configured rule allowing SSH from the jump box should resemble the following:

The screenshot shows the configuration of an inbound security rule named "SSH_From_JumpBox". The rule is set to "Allow" and has a priority of 480. It specifies the source IP as 10.0.0.4 and allows traffic on port 22 using the TCP protocol. The destination is set to the "VirtualNetwork". The rule is currently in edit mode, indicated by the "Edit" icon in the top bar.

Source
IP Addresses: 10.0.0.4
Source IP addresses/CIDR ranges: 10.0.0.4
Source port ranges: *

Destination
VirtualNetwork

Service
SSH
Destination port ranges: 22
Protocol: TCP

Action
Allow

Priority: 480

Name: SSH_From_JumpBox

Description: SSH_From_JumpBox

- 4.) Now that the SSH keys have been added to our web server VMs, we will now test out the connectivity from our jump box to each of the VMs to ensure they are properly configured.

From the jump box, we will attempt to SSH to each of the Web-1 and Web-2 using their private IP addresses of 10.0.0.5 and 10.0.0.6.

```
RedAdmin@Jump-Box-Provisioner:~$ ssh RedAdmin@10.0.0.5
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Fri Oct 29 20:09:28 UTC 2021

 System load:  0.0          Processes:           123
 Usage of /:   13.4% of 28.90GB  Users logged in:     0
 Memory usage: 20%
 Swap usage:  0%          IP address for eth0:   10.0.0.5
                           IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Oct 29 19:14:43 2021 from 10.0.0.4
RedAdmin@Web-1:~$ exit
logout
Connection to 10.0.0.5 closed.
RedAdmin@Jump-Box-Provisioner:~$ ssh RedAdmin@10.0.0.6
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Fri Oct 29 20:11:11 UTC 2021

 System load:  0.0          Processes:           123
 Usage of /:   13.4% of 28.90GB  Users logged in:     0
 Memory usage: 20%
 Swap usage:  0%          IP address for eth0:   10.0.0.6
                           IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Oct 29 18:59:38 2021 from 10.0.0.4
RedAdmin@Web-2:~$ -
```

II.) Configuring the Jump Box VM to Provision Docker Containers

A.) Installing Docker Containers

- 1.) At this point we have created a resource group with a virtual network, 1 jump box provisioner, 2 addition VM web servers, and a network security group within our cloud infrastructure. The NSG has been properly configured to only allow access from the public IP address of our workstation while denying all other incoming traffic into the network. In addition, inbound rules

have been created to allow an SSH connection from our local workstation to our jump box provisioner and an SSH connection from the jump box provisioner to the Web-1 and Web-2 VMs. Our cloud environment has been properly setup and securely configured so that we can now install and run containers from our jump box to configure our Web-1 and Web-2 VMs. This will be accomplished by first installing Docker on our jump box provisioner and downloading an Ansible container that will act as a provisioner to our Web-1 and Web-2 VMs.

- 2.) After establishing an SSH connection to our jump box provisioner, we will run the apt-get update to ensure we are running the latest images and install docker with the sudo apt install docker.io command.

```
logout
Connection to 10.0.0.6 closed.
RedAdmin@Jump-Box-Provisioner:~$ apt-get update
Reading package lists... Done
E: Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)
E: Unable to lock directory /var/lib/apt/lists/
W: Problem unlinking the file /var/cache/apt/pkgcache.bin - RemoveCaches (13: Permission denied)
W: Problem unlinking the file /var/cache/apt/srcpkgcache.bin - RemoveCaches (13: Permission denied)
RedAdmin@Jump-Box-Provisioner:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [2277 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1759 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 4288 kB in 1s (3493 kB/s)
Reading package lists... Done
RedAdmin@Jump-Box-Provisioner:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (20.10.7-0ubuntu1~18.04.2).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
RedAdmin@Jump-Box-Provisioner:~$
```

- 3.) We will verify the Docker server is running by running the sudo systemctl status docker command.

```

RedAdmin@Jump-Box-Provisioner:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2021-10-29 18:14:36 UTC; 2h 4min ago
    Docs: https://docs.docker.com
 Main PID: 1461 (dockerd)
   Tasks: 13
  CGroup: /system.slice/docker.service
          └─1461 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Oct 29 18:14:35 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:35.527219900Z" lev
Oct 29 18:14:35 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:35.527334800Z" lev
Oct 29 18:14:35 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:35.527785800Z" lev
Oct 29 18:14:35 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:35.869198500Z" lev
Oct 29 18:14:35 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:35.932188700Z" lev
Oct 29 18:14:36 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:36.326211757Z" lev
Oct 29 18:14:36 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:36.326846240Z" lev
Oct 29 18:14:36 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:36.329960177Z" lev
Oct 29 18:14:36 Jump-Box-Provisioner systemd[1]: Started Docker Application Container Engine.
Oct 29 18:14:36 Jump-Box-Provisioner dockerd[1461]: time="2021-10-29T18:14:36.416384074Z" lev
Lines 1-19/19 (END)

```

- 4.) After verifying that the Docker service is up and running, we will now download a Docker image with the sudo docker pull cyberxsecurity/ansible command.

- 5.) Now that the Docker image has been downloaded, we will run the docker container list -a command to view the name of our container.

```

RedAdmin@Jump-Box-Provisioner:~$ sudo docker container list -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
115036c12357 cyberxsecurity/ansible "/bin/sh -c /bin/bas..." 10 days ago Exited (127) 10 days ago objective_khorana
2bf428ca9a7d cyberxsecurity/ansible:latest "/bin/sh -c /bin/bas..." 12 days ago Exited (127) 12 days ago gracious_banzai
3eab3f49ae85 cyberxsecurity/ansible:latest "/bin/sh -c /bin/bas..." 12 days ago Exited (127) 12 days ago focused_roentgen
83f2a245eaf2 cyberxsecurity/ansible:latest "/bin/sh -c /bin/bas..." 12 days ago Exited (127) 12 days ago pensive_solomon
10483885e930 cyberxsecurity/ansible "/bin/sh -c /bin/bas..." 13 days ago Exited (127) 13 days ago strange_shaw
11f0f2047a1c cyberxsecurity/ansible:latest "/bin/sh -c /bin/bas..." 13 days ago Exited (137) 19 hours ago goofy_easley
RedAdmin@Jump-Box-Provisioner:~$
```

The name of our container has been identified as goofy_easley.

- 6.) We will now start the container with the sudo docker start goofy_easley command and confirm it is running with the sudo docker ps command.

```
RedAdmin@Jump-Box-Provisioner:~$ sudo docker start goofy_easley
goofy_easley
RedAdmin@Jump-Box-Provisioner:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND
11f0f2047a1c        cyberxsecurity/ansible:latest   "/bin/sh -c /bin/l
RedAdmin@Jump-Box-Provisioner:~$
```

B.) Provisioning the Ansible Container

- 1.) Now that we have started the container, our next task is to activate a shell and connect to the Ansible Docker Container with the sudo docker attach goofy_easley command.

```
RedAdmin@Jump-Box-Provisioner:~$ sudo docker attach goofy_easley
root@11f0f2047a1c:~#
```

The change in the prompt means we are now connected to the SSH container.

- 2.) To establish a connection from the Ansible Container to Web-1 and Web-2 we will first need to create an ssh key from inside the ansible container with the ssh-keygen command and run the cat .ssh/id_rsa.pub command to view the key.

```

RedAdmin@10.0.0.6: Permission denied (publickey).
root@11f0f2047a1c:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yV6R9MYLQfWef4qGV LHjDvXb5IM16N+wJG5gAgqT5fA root@11f0f2047a1c
The key's randomart image is:
+---[RSA 3072]---+
|          .+.. |
|          . =.. |
|          * + o. |
| + E . . . +=o . |
| o . . S .+.o+ |
| . . . . = .+.+ |
| .+ =o ==+ |
| ..+* .==|
| oo +.o |
+---[SHA256]---+
root@11f0f2047a1c:~# cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQCyJgYWkHKlnfHzA5rk6rzissewzbDL2xOF/QRW
6VIDeucOcdvpgrr86n7jyIhMK/Xx7Xmaj+sKDERSkk7SN1j6Gq4YLwdWm4ruafKIT7XL91OXuEJb
RhCehr/+XFYlzD5Jt1ToL8GPCDrPrzuNKTTXL5uMz1+Kc1z5a2iY+ERUs1r6x9f5XoDabVJluH9u
3jzCzbCCeefQz+RVLF1tSt7DGu8PWlbdEJ8s9V0FVZd/fKHOJ5ggmKzne664J9D2CRyKjuL9WzMG
h4l72ss= root@11f0f2047a1c
root@11f0f2047a1c:~#

```

- 3.) We will now use the newly generated key and install it on our Web-1 and Web-2 VMs.



This uses the VM Access for Linux extension to reset the credentials of an existing user or create a new user with sudo privileges, and reset the SSH configuration. [Learn more](#)

Mode

Reset password

Reset SSH public key

Reset configuration only

Username * ✓

SSH public key * ✓

- 4.) Verify the VMs have been properly configured with the SSH key by establishing a SSH connection to each VM.

```

Connection to 10.0.0.6 closed.
root@11f0f2047a1c:~# ssh RedAdmin@10.0.0.5
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Fri Oct 29 20:53:41 UTC 2021

 System load:  0.0          Processes:      123
 Usage of /:   13.4% of 28.90GB  Users logged in:  0
 Memory usage: 20%          IP address for eth0:  10.0.0.5
 Swap usage:   0%          IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Oct 29 20:50:21 2021 from 10.0.0.4
RedAdmin@Web-1:~$ exit
logout
Connection to 10.0.0.5 closed.
root@11f0f2047a1c:~# ssh RedAdmin@10.0.0.6
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Fri Oct 29 20:53:45 UTC 2021

 System load:  0.0          Processes:      124
 Usage of /:   13.4% of 28.90GB  Users logged in:  0
 Memory usage: 20%          IP address for eth0:  10.0.0.6
 Swap usage:   0%          IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

```

- 5.) Our VMs have now been properly configured with the SSH key generated from within the Ansible container. To ensure the Ansible Container can connect to the Web-1 and Web-2 VMs, we must configure the ansible.cfg file to add our RedAdmin username to allow Ansible to make the proper connections. First, we will navigate into the /etc/ansible directory and use the nano ansible.cfg command to edit the configuration file and add our RedAdmin username.

```
# default user to use for playbooks if user is not specified
# (/usr/bin/ansible will use current user as default)
remote_user = RedAdmin

# logging is off by default unless this path is defined
# if so defined, consider logrotate
#log_path = /var/log/ansible.log

# default module name for /usr/bin/ansible
#module_name = command

# use this shell for commands executed under sudo
# you may need to change this to bin/bash in rare instances
# if sudo is constrained
#executable = /bin/sh

# if inventory variables overlap, does the higher precedence one win
```

- 6.) Next, we will alter the hosts configuration file to add the private IP addresses of Web-1 and Web-2. After using the nano hosts command to enter the hosts file, we will locate the web servers group of IP address and add in 10.0.0.5 for Web-1 and 10.0.0.6 for Web-2. In addition, we will also add the python line ansible_python_interpreter=/usr/bin/python3 line to the right of each IP address.

```
[webservers]
#alpha.example.org
#beta.example.org
#192.168.1.100
#192.168.1.110
10.0.0.5 ansible_python_interpreter=/usr/bin/python3
10.0.0.6 ansible_python_interpreter=/usr/bin/python3
```

- 7.) We will now test our Ansible connection to our VMs with the following command ansible webservers -m ping.

```
root@11f0f2047a1c:/etc/ansible# ansible webservers -m ping
10.0.0.6 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.0.5 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Based on the output we can see that the Ansible container was able to successfully connect to our VMs.

III.) Create an Ansible Playbook to Configure Each VM with the DVWA Web Application

A.) Creating an Ansible Playbook

- 1.) In the previous step, we deployed an Ansible container that can connect and access our Web-1 and Web-2 VMs. Using the concept of Infrastructure as a Code, we will now create an Ansible playbook that installed Docker in order to configure our Web-1 and Web-2 VMs with DVWA. The playbook will be written using YAML code that will be used to run our “plays” to run and configure each of our servers with the web application. The code that will be used is as follows:

```
---  
- name : Config Web VM with Docker  
  hosts: webservers  
  become: true  
  tasks:  
  
  - name: docker.io  
    apt:  
      update_cache: yes  
      name: docker.io  
      state: present  
  
  - name: Install pip3  
    apt:  
      name: python3-pip  
      state: present  
  
  - name: Install Python Docker Module  
    pip:  
      name: docker  
      state: present  
  
  - name: download and launch a docker web container  
    docker_container:  
      name: dvwa  
      image: cyberxsecurity/dvwa  
      state: started  
      restart_policy: always  
      published_ports: 80:80
```

- 2.) We will now run our Ansible playbook using the ansible-playbook pentest.yml command. In this case pentest.yml is the name of our Ansible playbook that will configure each of the VMs with DVWA.

```

root@11f0f2047a1c:~# ansible-playbook pentest.yml
[WARNING]: ansible.utils.display.initialize_locale has not been called, this may result in incorrectly calculated text widths that can cause
Display to print incorrect line lengths

PLAY [Config Web VM with Docker] ****
TASK [Gathering Facts] ****
ok: [10.0.0.6]
ok: [10.0.0.5]
ok: [10.0.0.7]

TASK [docker.io] ****
ok: [10.0.0.6]
ok: [10.0.0.5]
ok: [10.0.0.7]

TASK [Install pip3] ****
ok: [10.0.0.6]
ok: [10.0.0.7]
ok: [10.0.0.5]

TASK [Install Python Docker Module] ****
ok: [10.0.0.6]
ok: [10.0.0.7]
ok: [10.0.0.5]

TASK [download and launch a docker web container] ****
[DEPRECATION WARNING]: The container_default_behavior option will change its default value from "compatibility" to "no_defaults" in

```

- 3.) We will now ensure that DVWA is running on Web-1 and Web-1 by establishing an SSH connection from the Ansible Container to each VM.

```

Select RedAdmin@Web-1: ~
10.0.0.5      : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=
10.0.0.6      : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=
10.0.0.7      : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=

root@11f0f2047a1c:~# ssh RedAdmin@10.0.0.5
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Fri Oct 29 21:34:18 UTC 2021

 System load:  0.11           Processes:          123
 Usage of /:   13.4% of 28.90GB  Users logged in:   0
 Memory usage: 20%            IP address for eth0:  10.0.0.5
 Swap usage:   0%             IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

```

Web-1 with Private IP address of 10.0.0.5 is up and running.

```
Connection to 10.0.0.5 closed.
root@11f0f2047a1c:~# ssh RedAdmin@10.0.0.6
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri Oct 29 21:39:34 UTC 2021

System load: 0.03          Processes:      123
Usage of /: 13.4% of 28.90GB Users logged in: 0
Memory usage: 21%
Swap usage:  0%           IP address for eth0: 10.0.0.6
                           IP address for docker0: 172.17.0.1

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Oct 29 21:33:10 2021 from 10.0.0.4
RedAdmin@Web-2:~$
```

Web-2 with Private IP address of 10.0.0.6 is up and running.

V.) Deploying a Load Balancer in the Virtual Network

A.) Setting up the Load Balancer

- 1.) Now that Web-1 and Web-2 have been configured with DVWA, we will now add a load balancer in front of our VMs. A load balancer will help accomplish two goals. First it will create network redundancy by distributing network traffic and ensuring the services offered by the VMs has high availability. Second, the addition of a load balancer to the network will help mitigate against Denial of Service attacks by allocating traffic across the pool of servers to ensure the network isn't overwhelmed with incoming traffic.
- 2.) From the Azure Portal we will search for load balancers and hit create. To configure the load balancer properly, we will assign it to the same RedTeam resource group and same East US region. We will also assign the Load Balancer with a static IP public address, select Basic for SKU, and click on the review + create button. The deployed load balancer will have the following settings:

Screenshot of the Azure portal showing the Azure Load Balancer (RedTeamLB) configuration page.

Overview

Essentials

Resource group (Move)	Backend pool
RedTeam	RedTeamLB (3 virtual machines)
Location	Load balancing rule
East US	PenTest (Tcp:80)
Subscription (Move)	Health probe
Azure subscription 1	RedTeamProbe (Tcp:80)
Subscription ID	NAT rules
38cb30e6-ce8b-411d-a2d6-cba69411c14c	0 inbound
SKU	Tier
Basic	Regional

Tags

See more

Configure high availability and scalability for your applications

Create highly-available and scalable applications in minutes by using built-in load balancing for cloud services and virtual machines. Azure Load Balancer supports TCP/UDP-based protocols and protocols used for real-time voice and video messaging applications. [Learn more](#)

Balance IPv4 and IPv6 addresses

Native dual-stack endpoints help meet regulatory requirements and address the fast-growing number of devices in mobile and IoT. [Learn more](#)

View frontend IP configuration

Build highly reliable applications

Load Balancer improves application uptime by routing traffic to healthy nodes. [Learn more](#)

View health probes

Secure your networks

Control network traffic and protect private networks using built-in network address translation (NAT). [Learn more](#)

View inbound NAT rules

- 3.) A health probe will next be added to the load balancer to ensure both Web-1 and Web-2 are able to safely receive traffic from the load balancer. The health probe will be configured to receive TCP traffic to port 80. In addition, the health probe will be run in 5 second intervals and will be designated as unhealthy if 2 consecutive failures are recorded.

Home > Load balancing > RedTeamLB >

RedTeamProbe

RedTeamLB

Name	RedTeamProbe
Protocol *	TCP
Port *	80
Interval *	5 seconds
Unhealthy threshold *	2 consecutive failures
Used by	PenTest

- 4.) We now need to ensure that our load balancer has a pool of servers it can distribute traffic to. This will be accomplished by adding both Web-1 and Web-2 to the backend server pool.

Microsoft Azure oxa1110 DEFA

Search resources, services, and docs (G+ /)

Home > RedTeamLB

RedTeamLB | Backend pools

Load balancer

Backend pool == all	Resource Name == all	Resource Status == all	IP address == all	Network interface == all
<input type="button" value="Add"/>	<input type="button" value="Refresh"/>	<input type="button" value="Give feedback"/>		
<input type="text" value="Filter by name..."/>				
<input type="button" value="Overview"/>				
<input type="button" value="Activity log"/>				
<input type="button" value="Access control (IAM)"/>				
<input type="button" value="Tags"/>				
<input type="button" value="Diagnose and solve problems"/>				
<input type="button" value="Frontend IP configuration"/>				
<input checked="" type="checkbox"/> <input type="button" value="Backend pools"/>				
<input type="button" value="Health probes"/>				
<input type="button" value="Load balancing rules"/>				
<input type="button" value="Group by Backend pool"/>				
Backend pool	Resource Name	Resource Status	IP Address	Network interface
✓ RedTeamLB				
RedTeamLB	Web-2	Stopped (deallocated)	10.0.0.6	web-2297
RedTeamLB	Web-1	Stopped (deallocated)	10.0.0.5	web-1947

- 5.) With the load balancer deployed on the network we now have a mechanism that will allow us to distribute traffic and ensure the availability of our services. At this point the load balancer has been created, but not configured to allow traffic to the VM backend pool. In this step we will configure our load balancer by creating a load balancing rule to ensure traffic is correctly taken from the load balancer and distributed to the Web-1 and Web-2 VMs located in the backend pool. This rule will specify that we are using IPv4 traffic from our load balancer using standard TCP website traffic on port 80. In addition, we will specify the backend pool consisting of Web-

1 and Web-2 VMs, include the health probe created, and enable session persistence for Clint IP and protocol to ensure an open session does not change to a different server.

PenTest ...

RedTeamLB

Info A load balancing rule distributes incoming traffic that is sent to a selected IP address and port combination across a group of backend pool instances. Only backend instances that the health probe considers healthy receive new traffic.

Name	PenTest
IP Version *	<input checked="" type="radio"/> IPv4 <input type="radio"/> IPv6
Frontend IP address * ⓘ	RedTeamLB (13.68.250.112) ▼
Protocol *	<input checked="" type="radio"/> TCP <input type="radio"/> UDP
Port *	80
Backend port * ⓘ	80
Backend pool *	RedTeamLB ▼
Health probe *	RedTeamProbe (TCP:80) ▼ Create new
Session persistence ⓘ	Client IP and protocol ▼
Idle timeout (minutes) *	<input type="range" value="4"/> 4
Floating IP ⓘ	<input checked="" type="radio"/> Disabled <input type="radio"/> Enabled

- 6.) In the previous step we created a load balancing rule that took TCP web traffic sent to the load balancer and distributed the traffic to Web-1 and Web-2. In this step we will add a new security group rule that will allow incoming web traffic sent to port 80 to flow into our virtual network. The newly created rule to allow traffic to come into our virtual network will be configured with the following settings.

 **Port_80**
RedTeam-SG

Save Discard Delete

Source ⓘ
IP Addresses

Source IP addresses/CIDR ranges * ⓘ
100.15.100.74 ✓

Source port ranges * ⓘ
*

Destination ⓘ
VirtualNetwork

Service ⓘ
HTTP

Destination port ranges ⓘ
80

Protocol
 Any
 TCP
 UDP
 ICMP

Action
 Allow
 Deny

Priority * ⓘ
4010

Name
Port_80

Description
Allow Port 80 traffic from the internet to the Vnet. ✓

- 7.) In the RedTeam NSG, we will now remove the incoming security rule that blocked all traffic coming into our Vnet to allow traffic to flow through our load balancer. In this step we will remove the previously defined default deny rule to now allow traffic to go through.
- 8.) Our load balancer has now been deployed and configured with the proper specifications. The next step we will take is verifying our DVWA application can be accessed from our browser via the internet. To accomplish this, we will type in the front-end IP address of our load balancer and add /setup.php to the URL to see if it accessible over the internet.



Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
 If you get an error make sure you have the correct user credentials in: `/var/www/html/config/config.inc.php`

If the database already exists, it will be cleared and the data will be reset.
 You can also use this to reset the administrator credentials ('admin // password') at any stage.

Setup Check

Operating system: *nix
 Backend database: MySQL
 PHP version: 7.0.33-0+deb9u10

Web Server SERVER_NAME: 13.68.250.112

PHP function display_errors: Disabled
 PHP function safe_mode: Disabled
 PHP function allow_url_include: Enabled
 PHP function allow_url_fopen: Enabled
 PHP function magic_quotes_gpc: Disabled
 PHP module gd: Installed
 PHP module mysql: Installed
 PHP module pdo_mysql: Installed

MySQL username: app
 MySQL password: *****
 MySQL database: dvwa
 MySQL host: 127.0.0.1

reCAPTCHA key: Missing

[User: www-data] Writable folder /var/www/html/hackable/uploads: Yes
 [User: www-data] Writable file /var/www/html/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt: Yes

[User: www-data] Writable folder /var/www/html/config: Yes
Status in red, indicate there will be an issue when trying to complete some modules.

If you see disabled on either `allow_url_fopen` or `allow_url_include`, set the following in your `php.ini` file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

[Create / Reset Database](#)

Damn Vulnerable Web Application (DVWA) v1.10 "Development"

Connectivity to the DVWA application has been verified.

VI.) Deploying the ELK Stack Server and Creating an Ansible Playbook to install Filebeat and Metricbeat to the DVWA VMs

A.) Creating a New VNet and Connecting it to the Current VNet

- 1.) The goal is to have the Elk stack server set up in a different virtual network to receive logs from the VMs in our RedTeam virtual network. To accomplish this we must first create a new virtual network that has the same configuration as our RedTeam resource group in a different region. The address space allocated to the newly created VM is 10.1.0.0/16.

A screenshot of the Azure portal's Virtual Networks blade. The navigation bar shows 'Home > Virtual networks > ELK-Net'. The main content area is titled 'ELK-Net' and 'Virtual network'. The 'Overview' tab is selected. On the left, there's a sidebar with links like Activity log, Access control (IAM), Tags, Diagnose and solve problems, Address space, Connected devices, Subnets, DDoS protection, and Firewall. The 'Essentials' section contains resource group, location, subscription, address space, and DNS server information. Below that is a 'Connected devices' table with one row.

2.) Now that the new virtual network has been created, we must ensure a connection can be established between the two virtual networks. In order to allow traffic to pass from our first vNet to our new vNet, we will create a peer connection between both vNets. Our peer connection will be established by selecting our new vNet and selecting the new peering option. Within the setup menu, we will choose our original RedTeam vNet from the Virtual Network dropdown box to specify this is the vNet we will be connecting with. The configuration of the peer network should resemble the following:

Home > ELK-Net >

Elk-to-Red ...

ELK-Net

This virtual network

Peering link name

Elk-to-Red

Peering status

Fully Synchronized

Peering state

Succeeded

Traffic to remote virtual network ⓘ

Allow (default)

Block all traffic to the remote virtual network

Traffic forwarded from remote virtual network ⓘ

Allow (default)

Block traffic that originates from outside this virtual network

Virtual network gateway or Route Server ⓘ

Use this virtual network's gateway or Route Server

Use the remote virtual network's gateway or Route Server

None (default)

Remote virtual network

Remote Vnet Id

/subscriptions/38cb30e6-ce8b-411d-a2d6-cba69411c14c/resourceGroups/RedTeam/providers/Microsoft.Network/virtu... ⓘ

Address space

10.0.0.0/16

- 3.) We will now create a new network security group called Elk-Server-NSG and create an inbound security rule that will allow us to SSH into the network.

B.) Create a new VM and Designate it as the Elk Stack Server

- 1.) The Elk Stack Server will be created by creating a new VM that is deployed in the same region as our newly created vNet, which is in the US Central region. We will then select an image that provides us with 4 GiBs of memory to allow the Elk server to properly run and deploy it with a public IP address and basic security group setting. The configuration of the new Elk Stack Server with private IP address of 10.1.0.4 is referenced below:

2.) The new Elk server just created must be accessible from within the Ansible container. To ensure we can connect to the Elk server we will add our SSH key generated from within our Ansible container and add it to the server.

```
root@11f0f2047a1c:~# cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQCyJgYwHKlnfHzA5rk6rzissewbDL2xOF/QRWR+BUcwuFcuhMDYqBtS1k9yzr0tRz1m0yNifNwrOKGK4gdNPL71x/fogCnzXD
6VIDeucOcdvpgrr86n7jyIhMK/Xx7Xmaj+sKDErSkK7SN1j6Gq4YLwdWm4ruafKIT7XL910XuEJbpBCvWU/jdXupxjPdm5pcJ3axXrMPaBU0dtJaFNYKEJ1aqwLbF+aXIgqENpra
RhCehr+XFY1zD5jt1ToL8GPCDrPrzuNKTTL5uMz1+Kc1z5a2iY+ERUs1r6x9f5XoDabVJiuH9u+OCBwriNs7BabLhjbdtk7WFg66ga1BnXTertP+c5sxix/pf6JYg1VU2Rqfgu
3jzCzbCCeefQz+RVLFltSt7DGu8Pw1bdEJ8s9v0FVzd/fKH0J5ggmKzne664J9D2CRyKjuL9WzMGIjqZbm300h1LF8uRwzSUltSU38KVzWKYwjH+IVN8m5Jfp1e1/WxM8uMsFOYs
h4l72ss= root@11f0f2047a1c
root@11f0f2047a1c:~#
```

SSH key generated in the Ansible container.

The screenshot shows the 'Elk-Server | Reset password' page within the VM Access for Linux extension. The 'Mode' is set to 'Reset SSH public key'. The 'Username' field contains 'RedAdmin'. The 'SSH public key' field displays a long string of characters, which is the copied public key from the previous step.

SSH key added to the Elk server

- 3.) After the Elk stack server has been created, we must ensure that we can establish connectivity to it from our Ansible container installed on our jump box virtual machine. To determine if the connection has been set up correctly, we will attempt to SSH into the Elk Server from our Ansible container.

```
root@11f0f2047a1c:~# ssh RedAdmin@10.1.0.4
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1062-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information as of Sat Oct 30 13:47:14 UTC 2021

 System load:  0.12              Processes:           132
 Usage of /:   18.5% of 28.90GB  Users logged in:      0
 Memory usage: 35%              IP address for eth0:  10.1.0.4
 Swap usage:   0%                IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '20.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sat Oct 30 13:43:02 2021 from 10.0.0.4
RedAdmin@Elk-Server:~$
```

The connection was successful.

C.) Downloading and Configuring the Container

- 1.) At this point we have completed the following tasks:

- Created a vNet that will host our ELK server in a different region from our web servers and other resources
- Created a peer connection from the new vNet and the RedTeam vNet
- Configured the ELK Server with the same SSH key shared by our Web VMs, which were generated from within our Ansible container
- Verified that we can connect to the ELK server from within our Ansible Container

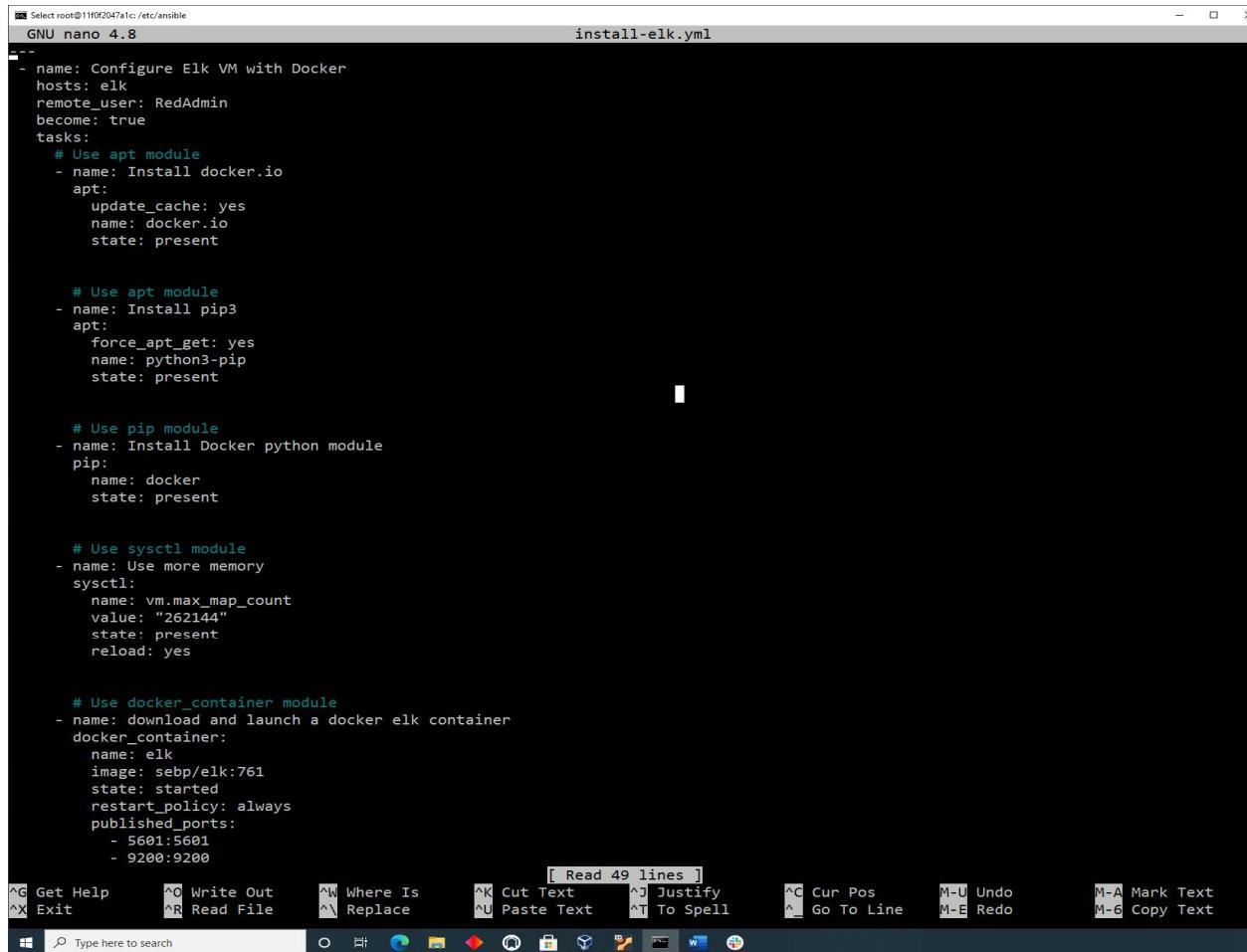
At this point we will use Ansible to configure our newly created ELK Server.

- 2.) In order for the ELK server to be recognized and configured by Ansible, we must first add it to the ELK inventory list from within the ansible hosts file. A line in the file will be added below the ELK group list that references its private IP address of 10.1.0.4 with the ansible_python_interpreter=/usr/bin/python3 line.

```
[webservers]
#alpha.example.org
#beta.example.org
#192.168.1.100
#192.168.1.110
10.0.0.5 ansible_python_interpreter=/usr/bin/python3
10.0.0.6 ansible_python_interpreter=/usr/bin/python3
10.0.0.7 ansible_python_interpreter=/usr/bin/python3

[elk]
10.1.0.4 ansible_python_interpreter=/usr/bin/python3
```

- 3.) We will now use YAML to create a playbook that will configure our ELK server. Within the playbook, we will install docker.io to install the required sebp/elk:761 Docker container and python3-pip to install Python software. In addition, the playbook will include a line to ensure the ELK server uses more memory to run properly by setting the vm.max_map_count line to 262144. In addition, the playbook will include the following port mappings for the container to start: 5601:5601, 9200:9200, 5044: 5044. A sample of the playbook is referenced below:



```

root@11f0f2047a1c:/etc/ansible          install-elk.yml
GNU nano 4.8

- name: Configure Elk VM with Docker
  hosts: elk
  remote_user: RedAdmin
  become: true
  tasks:
    # Use apt module
    - name: Install docker.io
      apt:
        update_cache: yes
        name: docker.io
        state: present

    # Use apt module
    - name: Install pip3
      apt:
        force_apt_get: yes
        name: python3-pip
        state: present

    # Use pip module
    - name: Install Docker python module
      pip:
        name: docker
        state: present

    # Use sysctl module
    - name: Use more memory
      sysctl:
        name: vm.max_map_count
        value: "262144"
        state: present
        reload: yes

    # Use docker_container module
    - name: download and launch a docker elk container
      docker_container:
        name: elk
        image: sebp/elk:761
        state: started
        restart_policy: always
        published_ports:
          - 5601:5601
          - 9200:9200

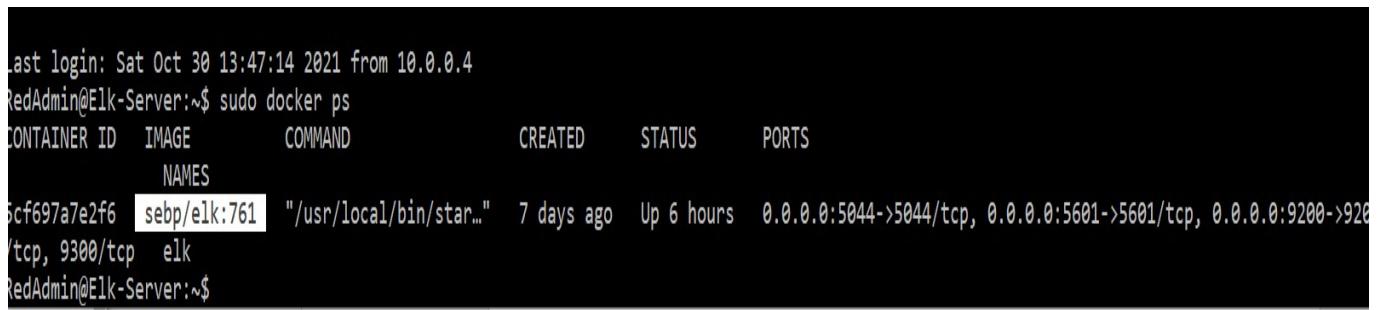
```

[Read 49 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
 ^X Exit ^R Read File ^A Replace ^U Paste Text ^T To Spell ^G Go To Line M-E Redo
 M-A Mark Text M-B Copy Text

- 4.) The playbook created in the previous step will be run to install and configure the container.

After the playbook has been run and executed, we will now SSH into the ELK server from our jump box to verify it is running the sebp/elk:761 container.



```

last login: Sat Oct 30 13:47:14 2021 from 10.0.0.4
RedAdmin@Elk-Server:~$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
5cf697a7e2f6 sebp/elk:761 "/usr/local/bin/star..." 7 days ago Up 6 hours 0.0.0.0:5044->5044/tcp, 0.0.0.0:5601->5601/tcp, 0.0.0.0:9200->920
/tcp, 9300/tcp elk
RedAdmin@Elk-Server:~$
```

The sebp/elk:761 container is properly running on the ELK server.

- 5.) With the new vNet and ELK stack server created, we will now add an inbound security rule to our Elk-server network security group to restrict access to the ELK VM. This inbound rule will contain our public IP address to ensure it is added to the security group's whitelist and allow TCP

traffic from our local workstation IP address over port 5601, which is the port running the ELK web server.

The screenshot shows a configuration interface for a Network Security Group (NSG). The rule is titled "Port_5601" and is associated with the "Elk-Server-nsg" security group. The rule details are as follows:

- Source:** IP Addresses (100.15.100.74)
- Source port ranges:** *
- Destination:** Any
- Service:** Custom (5601)
- Protocol:** TCP (selected)
- Action:** Allow (selected)
- Priority:** 310
- Name:** Port_5601
- Description:** Expose Elk Server

- 6.) After the ELK server has been installed and configured and the NSG updated to allow access to the server from our local workstation, we will attempt to access the server by typing in [http://\[ELK server public IP\]:\[Designated Port Number\]/app/kibana](http://[ELK server public IP]:[Designated Port Number]/app/kibana).

The screenshot shows the Elastic Stack interface with two main sections: Observability and Security.

Observability

- APM**: APM automatically collects in-depth performance metrics and errors from inside your applications. Buttons: [Add APM](#).
- Logs**: Ingest logs from popular data sources and easily visualize in preconfigured dashboards. Buttons: [Add log data](#).
- Metrics**: Collect metrics from the operating system and services running on your servers. Buttons: [Add metric data](#).

Security

- SIEM**: Centralize security events for interactive investigation in ready-to-go visualizations. Buttons: [Add events](#).

Visualize and Explore Data

- APM**: Automatically collect in-depth performance metrics and errors from inside your applications.
- Dashboard**: Display and share a collection of visualizations and saved searches.
- Graph**: Surface and analyze relevant relationships in your Elasticsearch data.
- Machine Learning**: Automatically model the normal behavior of your time series data to detect anomalies.
- Canvas**: Showcase your data in a pixel-perfect way.
- Discover**: Interactively explore your data by querying and filtering raw documents.
- Logs**: Stream logs in real time or scroll through historical views in a console-like experience.
- Maps**: Explore geospatial data from Elasticsearch and the Elastic Maps Service.

Manage and Administer the Elastic Stack

- Console**: Skip cURL and use this JSON interface to work with your data directly.
- Monitoring**: Track the real-time health and performance of your Elastic Stack.
- Saved Objects**: Import, export, and manage your saved searches, visualizations, and dashboards.
- Spaces**: Organize your dashboards and other saved objects into meaningful categories.
- Index Patterns**: Manage the index patterns that help retrieve your data from Elasticsearch.
- Rollups**: Summarize and store historical data in a smaller index for future analysis.
- Security Settings**: Protect your data and easily manage who has access to what with users and roles.
- Watcher**: Detect changes in your data by creating, managing, and monitoring alerts.

We are able to access the server from our local workstation.

VII.) Installing Filebeat and Metricbeat on the DVWA Containers

A.) Installing Filebeat on our DVWA VMs

- 1.) In the previous steps we have created an ELK monitoring server and used an Ansible playbook to install and configure an ELK container. In the steps to follow we will install Filebeat on our DVWA VMs to monitor database logs generated by Web-1 and Web-2. To accomplish this we will download the Filebeat configuration file and change the hosts line to include the private IP address and designated port number of our ELK server on lines #1106 and #1806 of the configuration file. A copy of the configuration file will be added to the /etc/filebeat directory.

```

# In case you specify an additional path, the scheme is required: http://localhost:9200/path
# IPv6 addresses should always be defined as: https://[2001:db8::1]:9200
hosts: ["10.1.0.4:9200"]
username: "elastic"
password: "changeme" # TODO: Change this to the password you set

# Set gzip compression level.

# You disable this check if you want to overwrite it. If you do, the lifecycle policy
# can be installed.
#setupilm.check_exists: false

# Overwrite the lifecycle policy at startup. The default is false.
#setupilm.overwrite: false

#===== Kibana =====

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:
  host: "10.1.0.4:5601" # TODO: Change this to the IP address of your ELK server
  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)

```

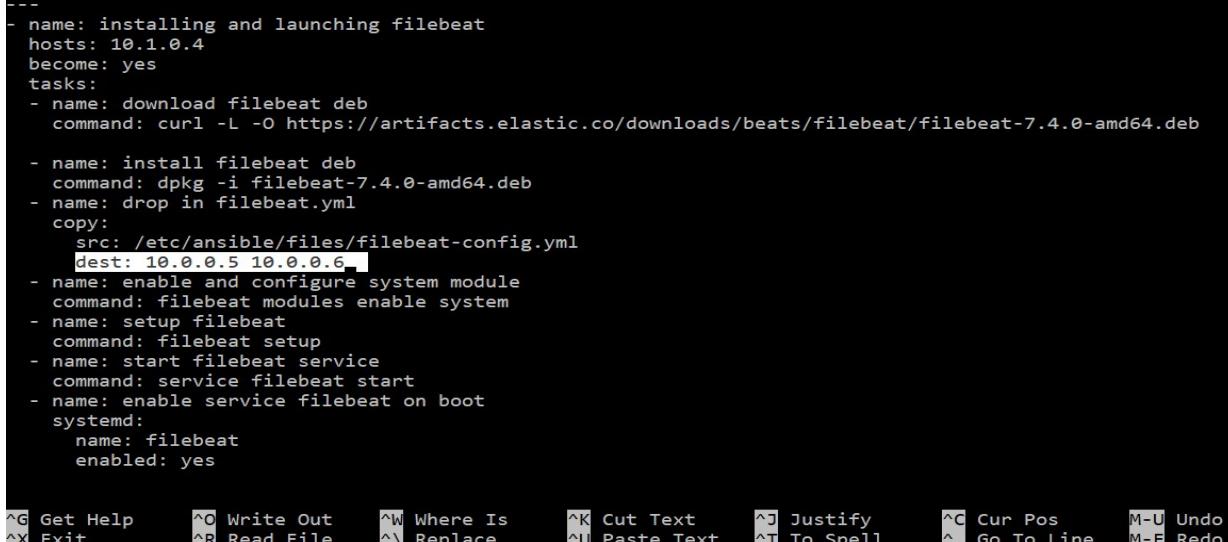
- 2.) Next we will create a new playbook that will download and install filebeat on our VMs running DVWA.

```

---
- name: installing and launching filebeat
  hosts: 10.1.0.4
  become: yes
  tasks:
    - name: download filebeat deb
      command: curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.4.0-amd64.deb

    - name: install filebeat deb
      command: dpkg -i filebeat-7.4.0-amd64.deb
    - name: drop in filebeat.yml
      copy:
        src: /etc/ansible/files/filebeat-config.yml
        dest: 10.0.0.5 10.0.0.6
    - name: enable and configure system module
      command: filebeat modules enable system
    - name: setup filebeat
      command: filebeat setup
    - name: start filebeat service
      command: service filebeat start
    - name: enable service filebeat on boot
      systemd:
        name: filebeat
        enabled: yes

```

A screenshot of a terminal window showing a text file named 'filebeat.yml'. The file contains the Ansible playbook code above. The status bar at the bottom shows the file is 10 lines long and has a size of 1024 bytes.

- 3.) After the configuration file has been correctly altered and the playbook created. We will now run the playbook and verify our ELK stack is able to receive logs from both Web-1 and Web-2 servers running DVWA. Within the ELK server Gui, we will navigate to add log data → System Logs → DEB tab → Module Status → Check Data → Verify Incoming Data.

2 Edit the configuration

Modify `filebeat.yml` to set the connection information:

[Copy snippet](#)

```
output.elasticsearch:  
  hosts: ["<es_url>"]  
  username: "elastic"  
  password: "<password>"  
setup.kibana:  
  host: "<kibana_url>"
```

Where `<password>` is the password of the `elastic` user, `<es_url>` is the URL of Elasticsearch, and `<kibana_url>` is the URL of Kibana.

3 Enable and configure the system module

From the installation directory, run:

[Copy snippet](#)

```
./filebeat modules enable system
```

Modify the settings in the `modules.d/system.yml` file.

4 Start Filebeat

The `setup` command loads the Kibana dashboards. If the dashboards are already set up, omit this command.

[Copy snippet](#)

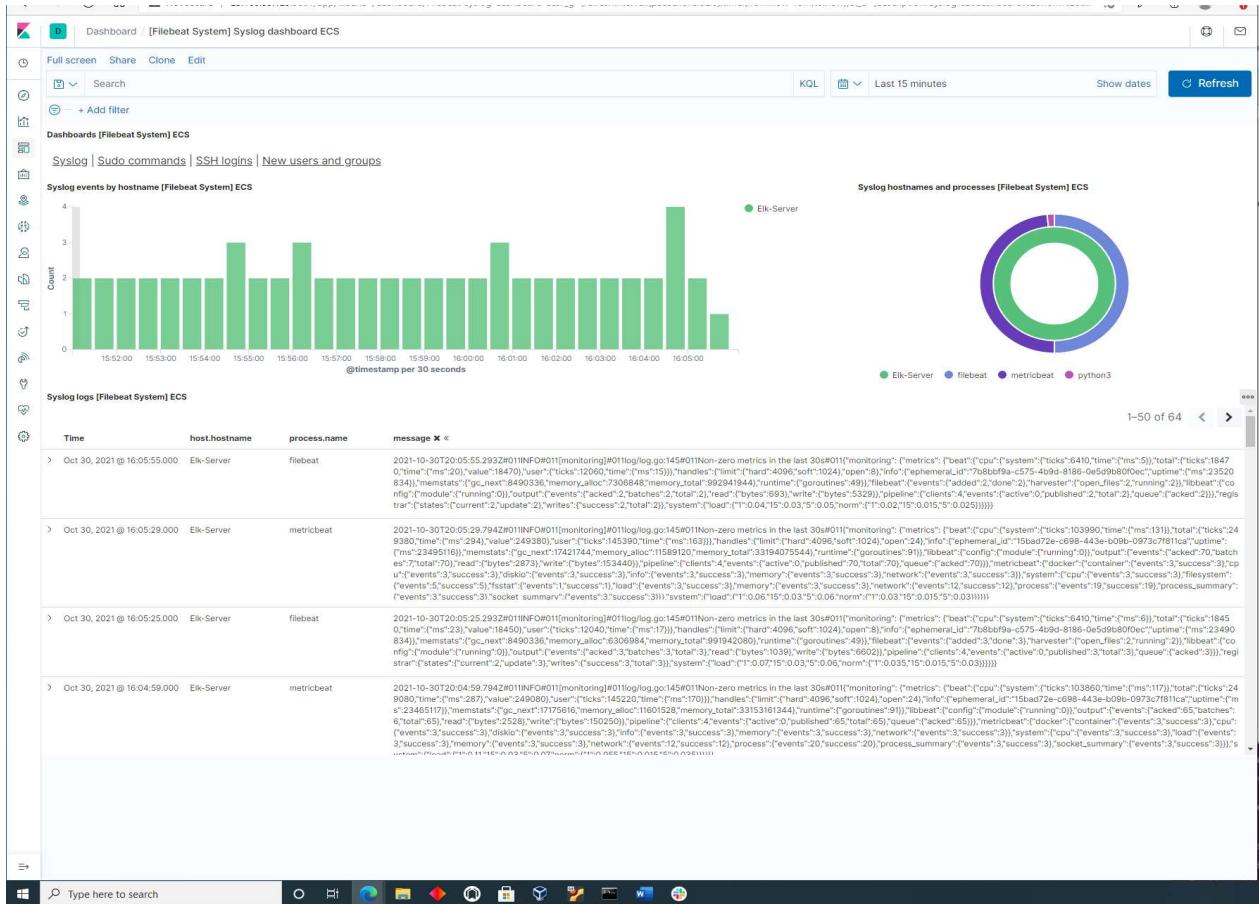
```
./filebeat setup  
./filebeat -e
```

Module status

Check that data is received from the Filebeat `system` module

[Check data](#)

Data successfully received from this module



The data has been successfully received by the ELK stack from Web-1 and Web-2.

B.) Installing Metricbeat on our DVWA VMs

- 1.) We will now run similar steps from above to install metricbeat on Web-1 and Web-2 to collect machine metrics such as CPU usage and uptime from our VMs. We will first download the metricbeat configuration file and alter it to include the private IP address of our ELK server.

```

# setup.kibana.host` options.
# You can find the `cloud.id` in the Elastic Cloud web UI.
#cloud.id:
# The cloud.auth setting overwrites the `output.elasticsearch.username` and
# `output.elasticsearch.password` settings. The format is `:`.
#cloud.auth:
===== Outputs =====
# Configure what output to use when sending the data collected by the beat.
----- Elasticsearch output -----
output.elasticsearch:
    # TODO: Change the hosts IP address to the IP address of your ELK server
    # TODO: Change password from 'changeme' to the password you created
    hosts: ["10.1.0.4:9200"]
    username: "elastic"
    password: "changeme"
----- Logstash output -----
#output.logstash:
    # The Logstash hosts
    #hosts: ["localhost:5044"]
    # Optional SSL. By default is off.
    # List of root certificates for HTTPS server verifications
    #ssl.certificateAuthorities: ["/etc/pki/root/ca.pem"]
    # Certificate for SSL client authentication
    #ssl.certificate: "/etc/pki/client/cert.pem"
    # Client Certificate Key

```

Keyboard Shortcuts:

- ^G Get Help
- ^O Write Out
- ^W Where Is
- ^K Cut Text
- ^J Justify
- ^C Copy
- ^X Exit
- ^R Read File
- ^V Replace
- ^I Paste Text
- ^T To Spell
- ^A Go

2.) Next, we will create a playbook that will be used to install metricbeat-7.6.1 and run the playbook to start the installation process on both vms.

```

---
- name: Install metric beat
hosts: 10.1.0.4
become: true
tasks:
    # Use command module
    - name: Download metricbeat
        command: curl -L -O https://artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-7.6.1-amd64.deb
    # Use command module
    - name: install metricbeat
        command: dpkg -i metricbeat-7.6.1-amd64.deb
    # Use copy module
    - name: drop in metricbeat config
        copy:
            src: /etc/ansible/files/metricbeat-config.yml
            dest: 10.0.0.5 10.0.0.6
    # Use command module
    - name: enable and configure docker module for metric beat
        command: metricbeat modules enable docker
    # Use command module
    - name: setup metric beat
        command: metricbeat setup
    # Use command module
    - name: start metric beat
        command: service metricbeat start
    # Use systemd module
    - name: enable service metricbeat on boot
        systemd:
            name: metricbeat
            enabled: yes

```

3.) After the playbook has been run and metricbeat installed on our web VMS, we will now verify that the Elk server is receiving log data from Web-1 and Web-2 by navigation from within the ELK Stack GUI to Add Metric Data → Docker Metrics → Deb →Module Status → Check Data

2 Edit the configuration

Modify `metricbeat.yml` to set the connection information:

[Copy snippet](#)

```
output.elasticsearch:  
  hosts: ['<es_url>']  
  username: "elastic"  
  password: "<password>"  
setup.kibana:  
  host: '<kibana_url>'
```

Where `<password>` is the password of the `elastic` user, `<es_url>` is the URL of Elasticsearch, and `<kibana_url>` is the URL of Kibana.

3 Enable and configure the docker module

From the installation directory, run:

[Copy snippet](#)

```
./metricbeat modules enable docker
```

Modify the settings in the `modules.d/docker.yml` file.

4 Start Metricbeat

The `setup` command loads the Kibana dashboards. If the dashboards are already set up, omit this command.

[Copy snippet](#)

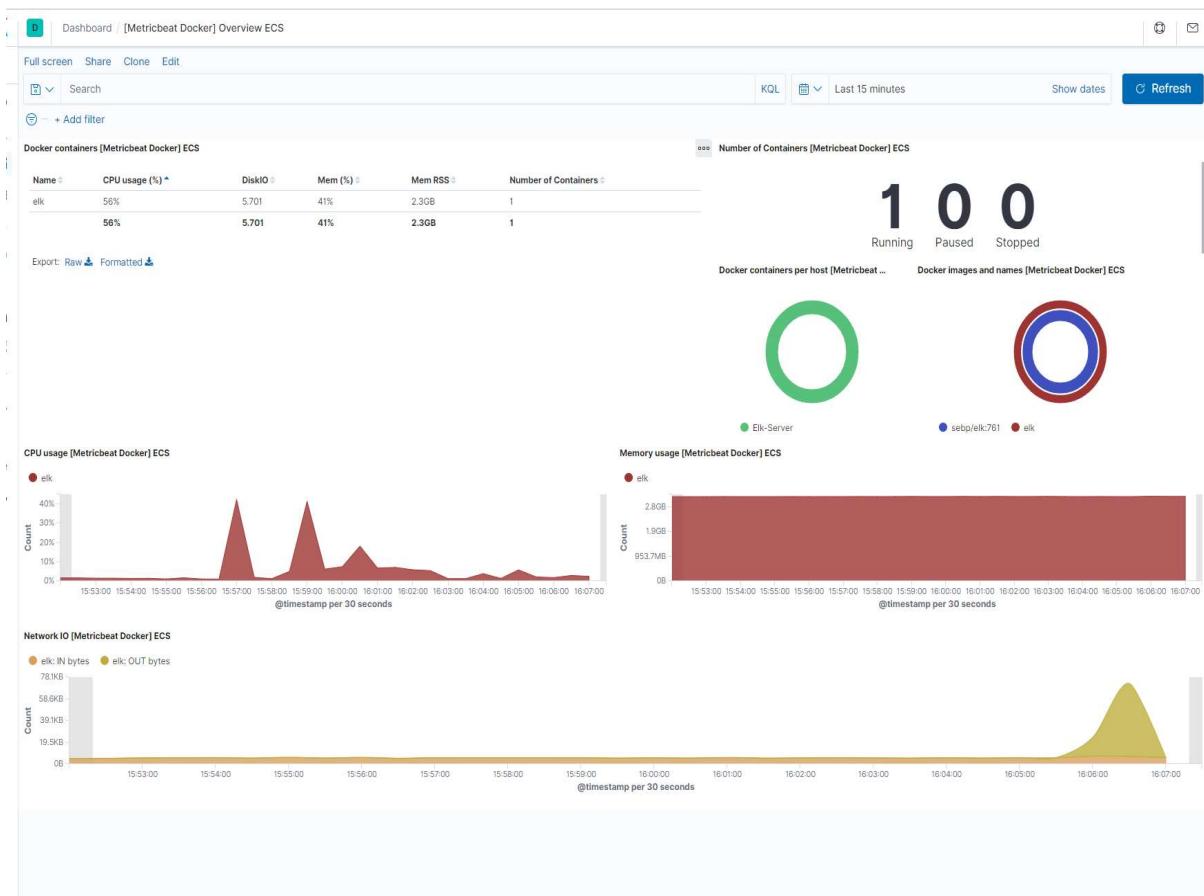
```
./metricbeat setup  
./metricbeat -e
```

Module status

Check that data is received from the Metricbeat `docker` module

[Check data](#)

Data successfully received from this module



The virtual network has been fully deployed with a cloud monitoring system.