# General overview

# Definitions: jump_box = provisioner VM; ansible container = control container node
# Definitions: webservers = Web1 and Web2 VM's running in ansible container
# Definitions: ELK-SERVER = Elk VM part of vNet Elk; user = source computer

****
# Setup ansible container control node and webserver VMs 1 & 2 within jump_box
# provisioner VM

# All VMs started and running in Azure
# Use sudo for docker commands unless you are root
# Any instructions not given for Azure may be assumed to be left as default or configured as
# needed

# In Azure
Network Security Group>Give your computer's public IP from any of your ports permission to ssh to jump_box private IP only to port 22;
Network Security Group>Give ansible control container node within jump_box permission to ssh into Web1 & Web2 private IP's only to port 22;

# In Git Bash
# Create ansible container control node
ssh into jump_box;
# ! You are now user: jump_box !
# This link provides commands for the instructions that follow resulting in new user: ansible
# control container node
# install-launch-start-attach-ansible
Install docker.io;
Pull ansible container
List out ansible container (note name of container);
Run ansible container;
Start ansible container;
Attach ansible container;
# ! You are now user: ansible container !
# This link gives commands for following instructions: generate_ssh_cmds
Generate ssh key;
cat public ssh key and copy;

# In Azure
# Prohibited: ssh from jump_box to webservers
# Allowed: ssh from ansible control container node within jump_box to webservers internal IP
# only to port 22
VM reset password>Give Web1 & 2 access to public ssh key; paste public ssh key;

# In Git Bash
# As user: ansible container
Test ssh key: test_ssh;
nano to configure files: /etc/ansible/ansible.cfg (allow <admin username> as remote user with SSH connection) & /etc/ansible/hosts (allow Web1 & 2 private IPs with python3 interpreter);
ping newly added webservers with this command: $ ansible all -m ping;
Run ansible playbook to configure ansible container: ansible_playbook_ansible;
Test connection for each webserver VM (ssh <admin name>@<webserver1 or 2 internal IP>; then $ curl localhost/setup.php for each VM)

****

# General access

# All VMs started and running in Azure
# Use sudo for docker commands unless you are root

# In Git Bash
ssh <admin username>@<jump_box public IP>;
# ! You are now user: jump_box !
# This is a link to commands for the following instruction that result in connecting the jump_box
# with the ansible container: list-start-attach-ansible
List out local container(s)
Start ansible container
Attach ansible container
# ! You are now user: ansible container !
ssh <admin username>@Web1 or 2
# ! You are now user: <admin user>@Web1 or 2 !

****

# Add Load Balancer to vNet with jump_box and webservers to regulate inbound/outbound
# webserver1 & 2 traffic;

# All VMs started and running in Azure
# Use sudo for docker commands unless you are root
# Any instructions not given for Azure may be assumed to be left as default or configured as
# needed

# Add security rule to forward port 80 from Load Balancer to vNet; webserver VMs will have
# HTTP access;

# At any point a new VM can be added behind the Load Balancer by following in Azure: Virtual
# Machine>(add VM and configure availability options: availability set and availability set: [name
# of vNet availability set])>Networking>(suggested configuration: public IP: none, NIC:
# advanced, configure network security group: [choose name of vNet security group] and Load
# Balancing: no (because VM should now be part of an availability set));
# Update /etc/ansible/hosts with any newly added VM with it's internal IP;
# Configure any new VMs added to the vNet by running the ansible playbook:
[ansible_playbook_ansible]()

# In Azure
Load Balancers>Add load balancer (with same resource group as vNet & static public IP)>Add health probe (suggested configuration TCP/80/5/2)
Load Balancers>Backend pools>Add backend pool (add webservers names and internal IP addresses add name of vNet with jump_box and webservers)
Load Balancers>Frontend IP Configuration>[name of load balancer]>PentestLBR>(suggested configuration: IPv4/TCP/80/80/session persistence = Client IP and protocol)
Network Security Group>[name of vNet security group]>inbound security rules>(suggested configuration: Source: your external IPv4 address, source port ranges: * , destination: VirtualNetwork, destination port ranges: 80, Protocol: Any, and Action: Allow)
Network Security Group>[name of vNet security group]>inbound security rules>(remove default deny all rule)
Test connection: from user browser enter http://<insert Load Balancer public IP>/setup.php

****

# Create vNet for ELK server

# All VMs started and running in Azure
# Use sudo for docker commands unless you are root
# Any instructions not given for Azure may be assumed to be left as default or configured as
# needed

# In Azure
Virtual Network>Create VM ELK-SERVER in region different from jump_box;
Virtual Network>Settings>Peerings>Add peering to and from ELK-SERVER and jump_box vNets;

# In Git Bash
# As user: ansible container
# This is a link providing commands to ssh from jump_box to user ansible container:
[ssh-user2provisioner]()
Cat public ssh key and copy

# In Azure
Virtual Machine>Add new Ubuntu VM (image with >=4GiG) with name ELK-SERVER, public IP address, same region as vNet ELK-SERVER and paste the public ssh key (follow instructions below for copy of ssh key);
Network security group>[choose Elk server group]>inbound security rules>(suggested configuration: source: IP addresses, source IP address: [insert user public IP], source ports: * , destination: VirtualNetwork, destination port: 5601 (or whichever port ELK-SERVER is running on), protocol: any, and action: allow);

# In Git Bash
# As user: ansible container
nano to configure files: /etc/ansible/hosts (allow host/grup Elk private IP with python3 interpreter);
Test ssh from ansible control container to ELK-SERVER;
Create and run an ansible playbook to install Elk: create_launch_ansible_playbook_elk_cmds;

****

# Add Filbeat to Elk monitoring for improved log monitoring of webservers 1 & 2
# containers

# All VMs started and running in Azure
# Use sudo for docker commands unless you are root
# Any instructions not given for Azure may be assumed to be left as default or configured as
# needed

# In user browser
Check ELK-SERVER is connected search: http://[ELK-SERVER public IP]:5601/app/kibana;
Install up-to-date Filebeat: in Kibana web page> Add Log Data>System Logs and choose DEB tab under Getting Started;

# In Git Bash
# As user: ansible container
Create filebeat configuration file: filebeat_configuration_yml;
Create and run filebeat playbook: filebeat_playbook_create+run_yml;

# In user browser
Verify installation: In Elk stack Kibana filebeat installation web page complete steps 1-5;

****

# Install and configure metricbeat to specialize filebeat and Elk monitoring

# In user browser

Search http://[ELK-SERVER public IP]:5601/app/kibana;
Install up-to-date metricbeat: Elk stack Kibana web page> Add Metric Data>Docker Metrics and choose DEB tab under Getting Started;

# In Git Bash
# This link has commands to instructions below: [metricbeats_install_configuration]
Install Metricbeat;
Create Metricbeat configuration file;
Update Metricbeat;

# In user browser
# In Elk stack Kibana docker metrics web page
Validate metricbeat is enabled: complete step 5;

****
# Edit /etc/ansible/ansible.cfg file by updating the following:
# uncomment remote_user & set equal to remote_user = <admin name>;
# In this project example: remote_user = RedAdmin;

****
# Edit /etc/ansible/hosts file to add Elk group:
# Updated configured section of /etc/ansible/hosts file appears as follows:
[webservers]
<Insert internal IP of Web1 VM> ansible_python_interpreter=/usr/bin/python3
<Insert internal IP of Web2 VM> ansible_python_interpreter=/usr/bin/python3

# List the IP address of your ELK server
# There should only be one IP address
[elk]
<Insert internal IP of ELK-SERVER> ansible_python_interpreter=/usr/bin/python3

****
# This is a link for a pre-configured ansible playbook to configure the ansible container within
# jump_box provisioner VM: [ansible_playbook_ansible]
# Run ansible playbook with command: ansible-playbook <name of playbook>.yml
# This yamil ansible playbook will: install docker.io, python3-pip, docker, cyberxsecurity/dvwa
# (specific for this project)
# container, publish port 80 on the container to port 80 on the host, and start docker service on
# boot

****
# The following is a link for a pre-configured ansible playbook for Elk installation:
ansible_playbook_Elk

****
# Adding Filebeat to Elk monitoring

# If curl command does not work then filebeat configuration file template:
https://github.com/the-Coding-Boot-Camp-at-UT/UTA-VIRT-CYBER-PT-09-2021-U-LOL/blob/master/1-Lesson-Plans/13-Elk-Stack-Project/Activities/Stu_Day_2/Solved/config_files/filebeat-configuration.yml
# nano /etc/ansible/files/filebeat-config.yml and copy and paste the above template
# Configure filebeat-config.yml as follows: