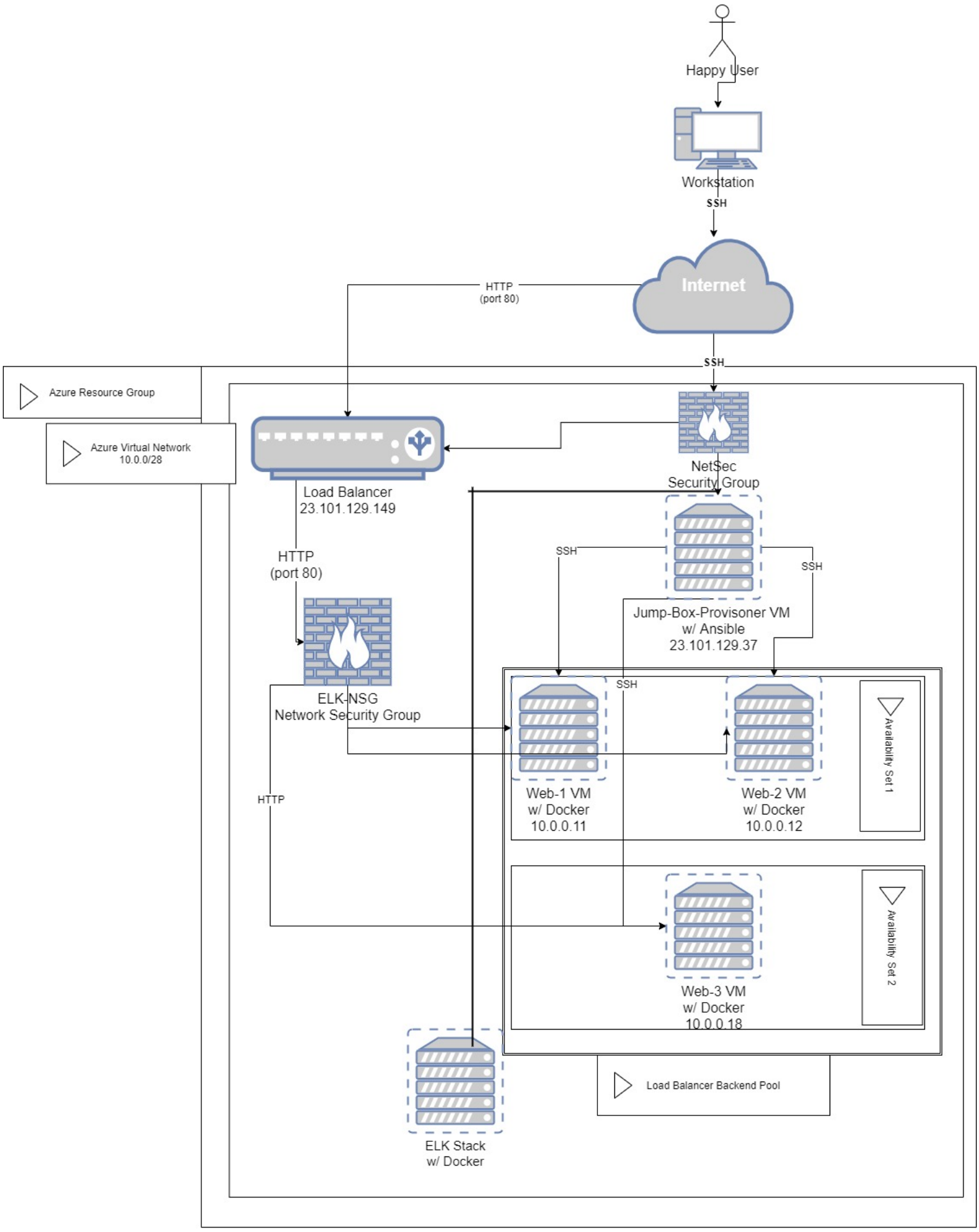
ELK-Stack_README

Automated ELK Stack Deployment

The files in this repository were used to configure the network depicted below.



These files have been tested and used to generate a live ELK deployment on Azure. They can be used to either recreate the entire deployment pictured above or, alternatively, select portions of the file may be used to install only certain pieces of it, such as Filebeat.

Playbook Snippet

```
---
  - name: Config web VM with Docker
    hosts: webservers
    become: true
    tasks:

    - name: docker.io
      apt:
        force_apt_get: yes
        update_cache: yes
        name: docker.io
```

```
      state: present

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

  - name: Install Python Docker Moudule
    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

  - name: enable docker service
    systemd:
      name: docker
      enabled: yes
```

elk-playbook snippet

```
---
- name: configure elk vm with docker
  hosts: elk
  remote_user: sysadmin
  become: true
  tasks:

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

    - name: docker.io
      apt:
        force_apt_get: yes
        update_cache: yes
        name: docker.io
        state: present

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

    - name: Install Python Docker Moudule
      pip:
        name: docker
        state: present

    - name: remove dvwa
      docker_container:
        name: dvwa
        image: cyberxsecurity/dvwa
        state: absent

    - name: download and launch a docker web container
      docker_container:
        name: elk
        image: sebp/elk:761
        state: started
        restart_policy: always
        published_ports:
          - 5601:5601
          - 9200:9200
          - 5044:5044

    - name: enable docker service
      systemd:
        name: docker
        enabled: yes
```

Description of the Topology

The main purpose of this network is to expose a load-balanced and monitored instance of DVWA, the Damn Vulnerable Web Application.

Load balancing ensures that the application will be highly resilient, in addition to restricting access to the network.

Integrating an ELK server allows users to easily monitor the vulnerable VMs for changes to file system and system metrics. For example, programs like *Filebeat* will generate log files about the file system, including when and which files have been changed. As where *Metricbeat* will collect machine metrics such as uptime.

The configuration details of each machine may be found below.

Configuration Details

Name	Function	IP Address	Operating System
Jump-Box-Provisioner	Gateway	23.101.129.37	Linux
RedTeam-LoadB	Load Balancer	23.101.129.149	
Elk-VM	Virtual Machine	52.250.52.70/(10.1.0.4)	Linux
Web-1	Virtual Machine	10.0.0.11	Linux
Web-2	Virtual Machine	10.0.0.12	Linux
Web-3	Virtual Machine	10.0.0.18	Linux

Access Policies

The machines on the internal network are not exposed to the public Internet.

Only the Jump box machine can accept connections from the Internet. Access to this machine is only allowed from the following IP addresses:

52.177.85.216

Machines within the network can only be accessed by 23.101.129.37 .

A summary of the access policies in place can be found in the table below.

Access Polices

Name	Publicly Accessible	Allowed IP Addresses
Jump Box	No	52.177.85.216
Untitled		
Untitled		

Elk Configuration

Ansible was used to automate configuration of the ELK machine. No configuration was performed manually, which is advantageous because it makes rendering new VM's as simple as running a script file.

The playbook implements the following tasks:

- Configures upgraded memory capacity
- Install Docker
- Install pip3
- Install Python Docker module
- Download and launch Docker container

The following screenshot displays the result of running `docker ps` after successfully configuring the ELK instance.

```
sysadmin@ELK-Vm:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
  NAMES
767ca00b9c09        sebp/elk:761        "/usr/local/bin/star…"   10 days ago         Up 15 seconds       0.0.0.0:5044->5044/tcp, 0.0.0.0:5601->5601/tcp, 0.0.0.0:9200->9200/tcp, 9300/tcp
  elk
sysadmin@ELK-Vm:~$
```

Target Machines & Beats

This ELK server is configured to monitor the following machines:

- 10.0.0.11
- 10.0.0.12
- 10.0.0.18

We have installed the following Beats on these machines:

- Filebeat
- Metricbeat

These Beats allow us to collect the following information from each machine:

- Filebeat:

  Simplifies the collection, parsing, and visualization of common log formats down to a single command from the Kibana Dashboard. Learn more here.

- Metricbeat:

  Monitors containers/consolidates data from hundreds of instances for issues such as uptime. Learn more here.

Using the Playbook

In order to use the playbook, you will need to have an Ansible control node already configured. Assuming you have such a control node provisioned:

SSH into the control node and follow the steps below:

- Copy the elk-playbook.yml file to /etc/ansible .
- Update the elk-playbook.yml file to include the 'elk' host
- Run the playbook, and navigate to ELK Server URL: http://51.141.160.207:5601/app/kibana#/home?_g=() to check that the installation worked as expected.