IT 445 – Capstone Implementation

Lab 01 - Basic Automation Functions

1. **Learning Objectives:**
   - How to automate the creation of virtual machines
   - How do Static routes get created automatically?
   - How to automate connectivity tests

2. **Equipment/software:**
   - Each member should have access to
     - Virtual Machine Images (Ubuntu 20.04 Server/Ubuntu 20.04 Desktop)
     - VMWare Workstation
3. **Exercises**

   MAKE SURE TO PROVIDE NETWORK DIAGRAMs AND TABLEs CONTAINING INFORMATION ABOUT THE MACHINES INVOLVED (SUCH AS, IP ADDRESSES, MAC ADDRESSES, ETC.)

   NOTE THAT HAVING TO POST WIRESHARK PCAP/PCAPNG FILES IS NOT A SUBSITUTE FOR HAVING TO PROVIDE SCREEN SHOTS OF WIRESHARK CAPTURES AS EVIDENCE IN SUPPORT OF YOUR NARRATIVES AND ANSWERS.

   3.1 Exercise 1: Preparation (it should be assigned to the students before class)

   Before starting, make sure that you have VMWare Workstation installed onto the machine currently used, whether that is the lab machine or your personal one. If you don't have VMWare installed, then go to the link below and follow the steps to install it onto your specific machine:
   https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

   Step 1: Infrastructure Creation
   Introduction

   We will be creating and setting up the infrastructure needed for Lab 1 in IT 333. Four VMs, 2 Desktop VMs and 2 Server VMs will be created and set up using Vagrant, with its set ip addresses and characteristics being added to each VM.

   Download VirtualBox compatible to the specific host OS from the link given: https://www.virtualbox.org/wiki/Downloads

Download Vagrant compatible to your specific host OS from the link given: https://developer.hashicorp.com/vagrant/install

Create a directory that will hold the Vagrantfile and enter the created directory

- **mkdir VMs**
- **cd VMs**

Create a Vagrantfile in the directory named “vagrantfile” (no extension) that will create the vms needed for the lab. To Create the file, run the command ”type nul > vagrantfile”.

Write out the command set up for the control vm, which will be used as the hub for ansible, which can be found in Appendix A.

Q1: Why does the vagrantfile not have an extension attached to it?

Q2: Explain what each setting is doing in the line: “desktop1.vm.network "private_network", ip: "10.0.10.2", auto_config: true, nic_type: "virtio"”

Run the command ”vagrant up” to run the vms or the command ”vagrant up –provider=vmware desktop” to run the vms on vmware

If error occurs that states ”Vagrant encountered an error while attempting to load the utility service key file. This error can occur if the Vagrant VMware Utility has not yet been installed, or if it was installed incorrectly.”, check to make sure that the Vagrant VMWare utility is installed by running the command ”vagrant plugin install vagrant-vmware-desktop”, and if it is installed, uninstall and reinstall and run the commands again to see if the same error occurs. If the error still occurs, directly downloand the vagrant vmware utility file from the link given: https://developer.hashicorp.com/vagrant/install/vmware.

Q3: Why is the VMWare Utility file necessary?

The vagrant boxes don’t have to be downloaded from the hashicorp website, as they get downloaded when the “vagrant up…” command is run. Some vagrant boxes will not be compatible with the vagrant and give the error “The box you're attempting to add doesn't support the provider you requested. Please find an alternate box or use an alternate provider. Double-check your requested

provider to verify you didn't simply misspell it." Because of this, make sure that you are using the correct vagrant boxes.

Once the vagrantfile is successfully ran, add the 4 additional vms one by one to the vagrantfile and run the vagrantfile to individually bring up and create the VMs. The set up for each vm can be found in Appendix B.

Step 2: Ansible Playbook Creation

We now need to set up Ansible to help us automate some of the tasks that will help us with the process of setting up the VMs for this lab.

Install Ansible on Host A (this will be the control machine)

sudo apt update
sudo apt upgrade
sudo apt install ansible

Generate an SSH key

ssh-keygen

Copy the public key to each VM

ssh-copy-id (username of vm)@(ip address of vm)

Q4: What is the purpose of the SSH key? Is it possible to complete this lab without the key being copied to each vm?

Create inventory.ini file that contains the ip addresses of target machines

sudo vim inventory.ini

Add the following lines to the inventory.ini file:

[servers]

R1 ansible_host=192.168.21.144

R2 ansible_host=192.168.21.145

[desktops]

HostA ansible_host=192.168.21.142

HostB ansible_host=192.168.21.143

Now that Ansible is installed and the inventory.ini file is set up properly, let's learn from ansible by writing a simple playbook that installs the nginx package that can be seen in Appendix C:

Run the playbook:

ansible-playbook -i inventory.ini nginx.yml

You should see a changed state on the vm that nginx is being installed on. If an error occurs, then attempt to troubleshoot by going over the syntax of the playbook. If an error still occurs update the vm that the package is being installed onto by using the command:

sudo apt update –fix-missing

Check to see if nginx is officially installed onto the vm by running a command on the targeted vm:

sudo systemctl status nginx

You should see the package being up and running.

Q5: What does the "-i" option do the command that runs the playbook? Does the playbook still run if that option isn't present?

Step 3: Ansible and Vagrant integration through Control vm

Now that we got an idea of how these playbooks can be written and what these options do, let's see how we can use these playbooks to add connectivity between the 4 vms.

Instead of individually adding the routes that are needed in each vm to have universal connectivity, we can construct a playbook that adds the missing ip route to all of the vms.

Create a playbook, ip_routes.yml that adds the routes necessary for each vm to connect to each other. The yml file can be seen in Appendix D:

Run the playbook:

ansible-playbook -i inventory.ini ip_routes.yml

Q6: What is the purpose of each route added? How would the connectivity be affected if the default routes on HostA and HostB are removed? How would the connectivity be affected if the routes were removed from R1 and R2?

If successful, these routes should be added to each vm. To check, use the command “ip route” on each vm and find the added route, or run the playbook again. **Running the playbook again when the routes are added will give an error stating that the file exists, meaning that the routes are added in.**

Once the routes are added, create a playbook, forwarding.yml, that enables ip forwarding on the server vms, R1 and R2, which can be seen in Appendix E:

Run the playbook:

ansible-playbook -i inventory.ini forwarding.yml

This playbook changes the value of the 'net.ipv4.ip_forward’ line to make it equal 1, which enables ip forwarding, on R1 and R2.

Q7: What is the purpose of each line in the first task of the forwarding.yml?

Q8: What is the purpose for the “sysctl -p” command?

At this point, the new routes are added to the vms and ip forwarding is enabled on the server vms. Now, we will create a playbook, ping.yml that checks the

connectivity of these vms by pinging every vm listed in the inventory.ini file, which can be seen in Appendix F:

Run the playbook:

ansible-playbook -i inventory.ini ping.yml

The results of the playbook should be successful the results [OK] being present for each vm during each test. If errors occur, then troubleshooting is needed.

Q9: What would the result of the ping.yml playbook being ran be if the forwarding.yml playbook was never ran?

4. **Deliverables:**
   - Overleaf generated Lab Report.
   - Additional Evidence & FILES (such as Wireshark PCAP, iptables.rules, etc.)

5. **References:**

   1. VMWare Workstation Download https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion
   2. Virutal Box Downloads https://www.virtualbox.org/wiki/Downloads
   3. Ansible Documentation https://docs.ansible.com/
   4. Vagrant Documentation https://developer.hashicorp.com/vagrant/docs

Appendix A: Set up of the Control VM

```
Vagrant.configure("2") do |config|


  config.vm.define "Control" do |desktop1|
    desktop1.vm.box = "gusztavvargadr/ubuntu-desktop-2004-lts"
    desktop1.vm.hostname = "Control"
    desktop1.vm.network "private_network", ip: "10.0.10.10", virtualbox__intnet: "VMnet2",
auto_config: true, nic_type: "virtio"
    desktop1.vm.provider "vmware_desktop" do |v|
      v.gui = true
      v.memory = 1024
      v.cpus = 1
    end
  end
end
```

Appendix B: Set up of the Lab VMs

Vagrantfile:

```
Vagrant.configure("2") do |config|


 config.vm.define "Control" do |desktop1|

  desktop1.vm.box = "gusztavvargadr/ubuntu-desktop-2004-lts"

  desktop1.vm.hostname = "Control"

  desktop1.vm.network "private_network", ip: "10.0.10.10", virtualbox__intnet: "VMnet2",
auto_config: true, nic_type: "virtio"

  desktop1.vm.provider "vmware_desktop" do |v|

   v.gui = true

   v.memory = 1024

   v.cpus = 1

  end

 end



 config.vm.define "HostA" do |desktop2|

  desktop2.vm.box = "gusztavvargadr/ubuntu-desktop-2004-lts"

  desktop2.vm.hostname = "HostA"

  desktop2.vm.network "private_network", ip: "10.0.10.2", virtualbox__intnet: "VMnet2",
auto_config: true, nic_type: "virtio"

  desktop2.vm.provider "vmware_desktop" do |v|

   v.gui = true

   v.memory = 1024

   v.cpus = 1

  end
```

```
 end



 config.vm.define "HostB" do |desktop3|
  desktop3.vm.box = "gusztavvargadr/ubuntu-desktop-2004-lts"
  desktop3.vm.hostname = "HostB"
  desktop3.vm.network "private_network", ip: "10.0.30.2", virtualbox__intnet: "VMnet4",
auto_config: true, nic_type: "virtio"
  desktop3.vm.provider "vmware_desktop" do |v|
    v.gui = true
    v.memory = 1024
    v.cpus = 1
  end


 end



 config.vm.define "R1" do |r1|
  r1.vm.box = "gusztavvargadr/ubuntu-server-2004-lts"
  r1.vm.hostname = "R1"
  r1.vm.network "private_network", ip: "10.0.10.1", virtualbox__intnet: "VMnet2",
auto_config: true, nic_type: "virtio"
  r1.vm.network "private_network", ip: "10.0.20.1", virtualbox__intnet: "VMnet3",
auto_config: true, nic_type: "virtio"
  r1.vm.provider "vmware_desktop" do |v|
    v.gui = true
    v.memory = 1024
```

```
    v.cpus = 1
  end
 end



 config.vm.define "R2" do |r2|
  r2.vm.box = "gusztavvargadr/ubuntu-server-2004-lts"
  r2.vm.hostname = "R2"
  r2.vm.network "private_network", ip: "10.0.20.2", virtualbox__intnet: "VMnet3",
auto_config: true, nic_type: "virtio"
  r2.vm.network "private_network", ip: "10.0.30.1", virtualbox__intnet: "VMnet4",
auto_config: true, nic_type: "virtio"
  r2.vm.provider "vmware_desktop" do |v|
    v.gui = true
    v.memory = 1024
    v.cpus = 1
  end
 end
end
```

Appendix C: nginx.yml file

```
- name: Install nginx
  hosts: all
  become: yes
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
```

Appendix D: ip_route.yml file

```
- name: Add Routes to all VMs
  hosts: all
  become: yes
  tasks:

    - name: Add default route for HostA
      when: inventory_hostname == 'HostA'
      shell: sudo ip route add default via 10.0.10.1

    - name: Add default route for HostB
      when: inventory_hostname == 'HostB'
      shell: sudo ip route add default via 10.0.30.1

    - name: Add routes for R1
      when: inventory_hostname == 'R1'
```

```
    shell: sudo ip route add 10.0.30.0/24 via 10.0.20.2 dev eth2


  - name: Add routes for R2
    when: inventory_hostname == 'R2'
    shell: sudo ip route add 10.0.10.0/24 via 10.0.20.1 dev eth1
```

Appendix E: forwarding.yml

```
- name: Enable IP forwarding on R1 and R2
 hosts: servers
 become: yes
 tasks:


  - name: Ensure IP forwarding is enabled on boot
    lineinfile:
      path: /etc/sysctl.conf
      regexp: '^net.ipv4.ip_forward='
      line: 'net.ipv4.ip_forward=1'
      state: present


  - name: Apply sysctl changes
    shell: |
      sysctl -p
```

Appendix F: ping.yml

```
- name: Check connectivity between VMs
  hosts: all
  become: yes
  tasks:

    - name: Ping HostA from all VMs
      shell: ping -c 2 10.0.10.2
      ignore_errors: yes
      changed_when: false

    - name: Ping HostB from all VMs
      shell: ping -c 2 10.0.30.2
      ignore_errors: yes
      changed_when: false

    - name: Ping R1 eth1 from all VMs
      shell: ping -c 2 10.0.10.1
      ignore_errors: yes
      changed_when: false

    - name: Ping R1 eth2 from all VMs
      shell: ping -c 2 10.0.10.1
      ignore_errors: yes
      changed_when: false
```

```
  - name: Ping R2 eth1 from all VMs
    shell: ping -c 2 10.0.20.2
    ignore_errors: yes
    changed_when: false

  - name: Ping R2 eth2 from all VMs
    shell: ping -c 2 10.0.30.1
    ignore_errors: yes
    changed_when: false
```