IN730 Special Topic - Network Automation

Lab2 - Ansible Playbooks (Cloud Version)

Requirements

- Completion of lab 1
- Azure Subscription
- Terminal Emulator

Disclaimer

If you stopped your virtual machine that was running your gns3server and you didnt make your tap1 and ip routes persistent then you will need to run the following commands again to recreate those

```
sudo tunctl -t tap1
sudo ifconfig tap1 192.168.1.254 netmask 255.255.255.0 up
sudo ip route add 192.168.1.0/24 via 192.168.1.254 dev tap1
sudo ip route add 192.168.2.0/30 via 192.168.1.254 dev tap1
```

Ansible playbook to pull device information

We will create an ansible playbook that will pull configuration from our routers that we can use as a backup

In order to achieve this we first need to create a directory to be used to store the backups of the routers configuration

This will create a directory in our home directory

```
sudo mkdir ~/ansible-backups
```

Now we will create a playbook called backup.yaml

```
sudo vim /etc/ansible/backup.yaml
```

Insert the following

- Do note, make sure to edit the following

```
<YOUR HOME DIRECTORY> with the home directory of your user account your using
```

```
---
  - hosts: localhost

    tasks:
            - name: Get Date/Time
              setup:
                      filter: "ansible_date_time"
                      gather_subset: "!all"

            - name: Store Date/Time
              set_fact:
                      DTG: "{{ansible_date_time.date }}"

            - name: Create Directory {{hostvars.localhost.DTG}}
              file:
                      path: /home/<YOUR HOME DIRECTORY>/ansible-backups/{{hostvars.localhost.DTG}}
                      state: directory
    run_once: true

  - hosts: routers
    connection: local
    remote_user: admin
    gather_facts: false
    tasks:
            - name: backup running config
              block:
              - name:
                ios_command:
                  commands: show running-config
                register: config

              - name: save running config to backup folder
                copy:
                  content: "{{config.stdout[0]}}"
                  dest: "/home/<YOUR HOME DIRECTORY>/ansible-backups/{{hostvars.localhost.DTG}}/{{inventory_hostname}}-{{hostvars.localhost.DTG}}-config.txt"
```

You can run your ansible playbooks by being located in the directory where its located by using

```
ansible-playbook backup.yaml
```

Or you can provide the path to the playbook

```
ansible-playbook /etc/ansible/backup.yaml
```

Now that we have pulled the configuration and stored it on our machine lets confirm that it is there

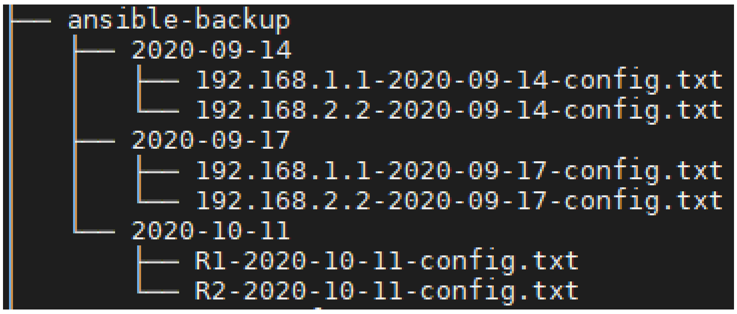
I order to do this we will download and install an application called tree which will help with displaying the contents of our directories

```
sudo apt-get install tree -y
```

We can now use the following command to list the home directory

```
tree ~/ansible-backup
```

you should have an output simillar to the following



```
├── ansible-backup
│   ├── 2020-09-14
│   │   ├── 192.168.1.1-2020-09-14-config.txt
│   │   └── 192.168.2.2-2020-09-14-config.txt
│   ├── 2020-09-17
│   │   ├── 192.168.1.1-2020-09-17-config.txt
│   │   └── 192.168.2.2-2020-09-17-config.txt
│   └── 2020-10-11
│       ├── R1-2020-10-11-config.txt
│       └── R2-2020-10-11-config.txt
```

Here we can see that ansible managed to pull configuration from the routers

Ansible playbooks to deploy configuration

For this section we will create a playbook that will push configuration to our routers this will allow us to maintain a uniform environment across many devices

```
sudo vim /etc/ansible/motd.yaml
```

Insert the following into our playbook

```
---
  - name: testbook
    hosts: network
    connection: local
    remote_user: admin
    gather_facts: false
    tasks:
            - name: configure login banner
              ios_banner:
                      banner: login
                      text: |
                              Here
                              Is
```

```
                             A
                             Test
                             Configuration
                             Banner
                     state: present
```

The purpose of this playbook is to setup a banner so that when we ssh onto our routers we will see it, you can use this banner to warn unauthorized users

You can run your ansible playbooks by being located in the directory where its located by using

```
ansible-playbook motd.yaml
```

Or you can provide the path to the playbook

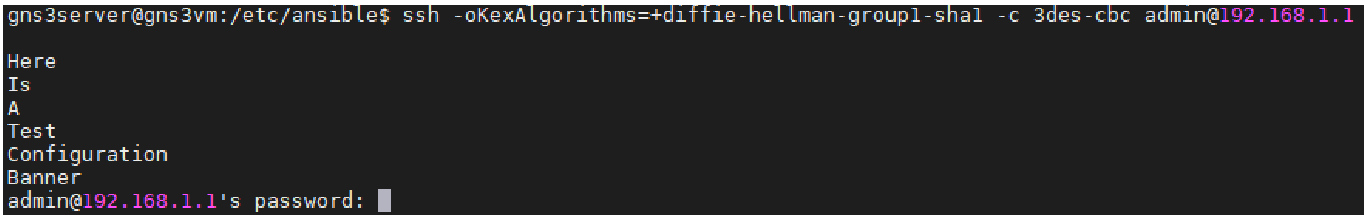
```
ansible-playbook /etc/ansible/motd.yaml
```

Now lets connect to our router to see and verify if the change we made took effect

- Do note that due to issues with gns3 and cloud we need to add a few additional options to our ssh command

```
ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 -c 3des-cbc admin@<router IP>
```

We should see the following when we connect



```
gns3server@gns3vm:/etc/ansible$ ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 -c 3des-cbc admin@192.168.1.1

Here
Is
A
Test
Configuration
Banner
admin@192.168.1.1's password:
```

Create additional VM's using Ansible

In order to create VM's in azure using ansible we need to download and install additional software

We will download curl which we require in order to get the azure command line

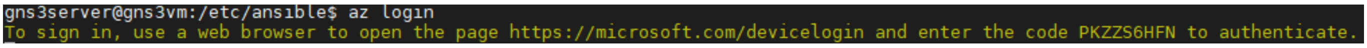
```
sudo apt install curl
```

Download and install azure command line

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
sudo apt install azure-cli
```

Now that we have downloaded and installed azure command line we need to connect to it

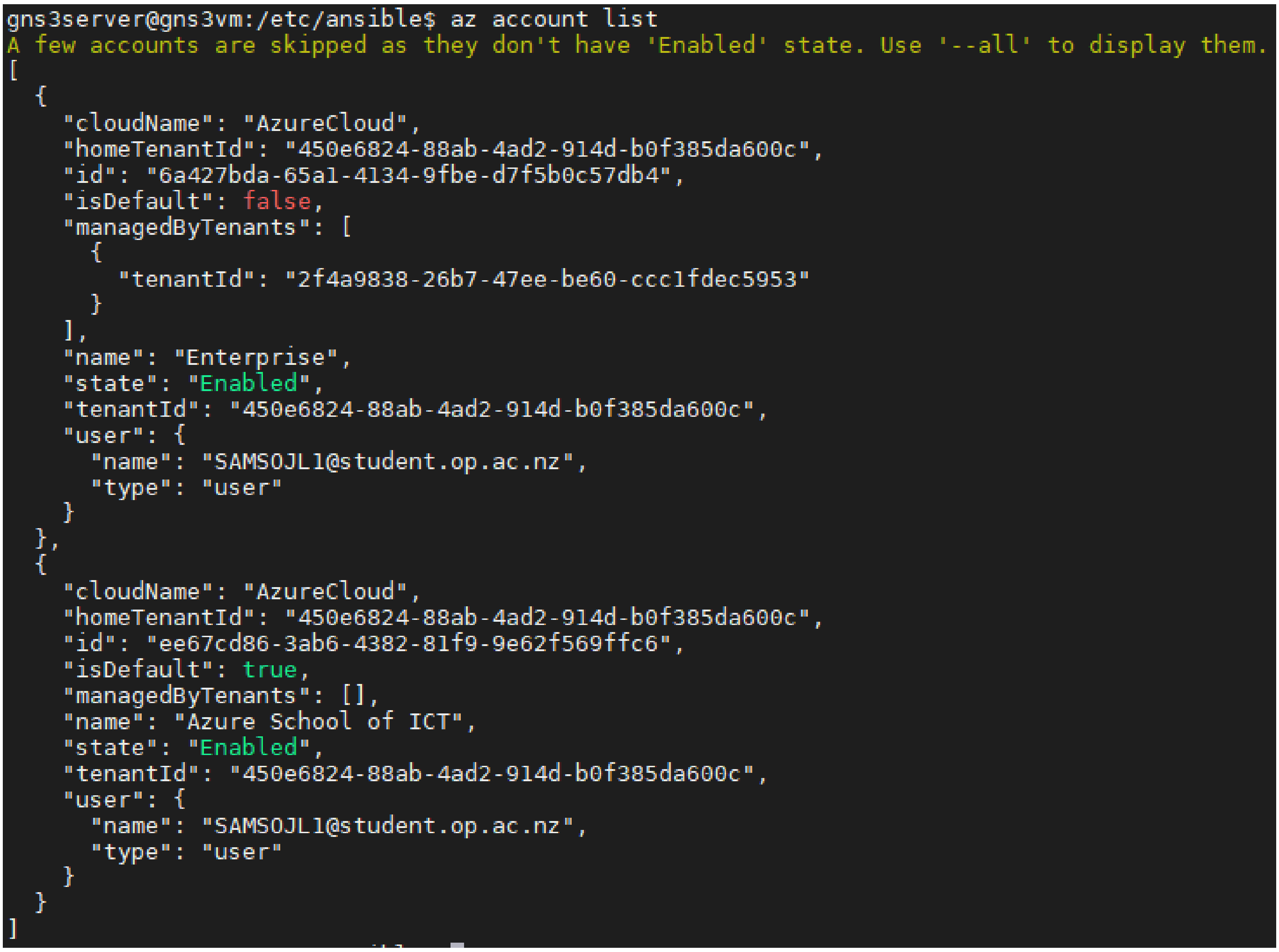
```
az login
```



```
gns3server@gns3vm:/etc/ansible$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code PKZZS6HFN to authenticate.
```

Follow the on screen prompt and input the code it provides on the website then select your microsoft account that you used for the previous lab

After you have logged in you will be given an output that contains the information about your microsoft azure account



```
gns3server@gns3vm:/etc/ansible$ az account list
A few accounts are skipped as they don't have 'Enabled' state. Use '--all' to display them.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "450e6824-88ab-4ad2-914d-b0f385da600c",
    "id": "6a427bda-65a1-4134-9fbe-d7f5b0c57db4",
    "isDefault": false,
    "managedByTenants": [
      {
        "tenantId": "2f4a9838-26b7-47ee-be60-ccc1fdec5953"
      }
    ],
    "name": "Enterprise",
    "state": "Enabled",
    "tenantId": "450e6824-88ab-4ad2-914d-b0f385da600c",
    "user": {
      "name": "SAMSOJL1@student.op.ac.nz",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "450e6824-88ab-4ad2-914d-b0f385da600c",
    "id": "ee67cd86-3ab6-4382-81f9-9e62f569ffc6",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure School of ICT",
    "state": "Enabled",
    "tenantId": "450e6824-88ab-4ad2-914d-b0f385da600c",
    "user": {
      "name": "SAMSOJL1@student.op.ac.nz",
      "type": "user"
    }
  }
]
```

When you use azure cli it will assign anything you create to your default subscription so if you have multiple subscriptions on your account we will need to set this

To change your default subscription is a quick task

```
az account set --subscription <ID>
```

After changing your subscription you wont be given a notification so you will need to verify that it has changed by using the following command

```
az account list
```

- sudo apt-get install python-pip
- pip install packaging
- pip install msrestazure
- pip install ansible[azure]

We will now create a playbook that will create a virtual machine in azure

```
sudo vim /etc/ansible/create.yaml
```

Insert the following into the playbook

```
- name: Create Azure VM
  hosts: localhost
  connection: local

  vars:
   vm_offer: "UbuntuServer"
   vm_pub: "Canonical"
   vm_sku: "18.04-LTS"

   vm_size: "Standard_E2s_v3"

   az: "australiaeast"
   net: "Ansible"
   vm_net: "AnsibleVNet"
   vm_subnet: "AnsibleSubnet"

   vm_publicIP: "AnsiblePublicIP"
   vm_NSG: "AnsibleNSG"
   vm_NIC: "AnsibleNIC"
   vm_Name: "ansibletest"

   resource_group: "ansible"

   vm_peer: "AnsiblePeer"

   os_user: "ansible"
   os_pass: "ansible@ssw0rd"

  tasks:
  - name: Create virtual network
    azure_rm_virtualnetwork:
      resource_group: "{{ resource_group }}"
      name: "{{ vm_net }}"
      address_prefixes: "10.0.0.0/16"

  - name: Add subnet
    azure_rm_subnet:
      resource_group: "{{ resource_group }}"
      name: "{{ vm_subnet }}"
      address_prefix: "10.0.1.0/24"
      virtual_network: "{{ vm_net }}"

  - name: Create public IP address
    azure_rm_publicipaddress:
      resource_group: "{{ resource_group }}"
      allocation_method: Static
      name: "{{ vm_publicIP }}"
      domain_name: gitlab-test
    register:  reg_publicIP

  - debug: var=reg_publicIP

  - name: Create Network Security Group that allows SSH
    azure_rm_securitygroup:
      resource_group: "{{ resource_group }}"
      name: "{{ vm_NSG }}"
      rules:
        - name: SSH
          protocol: Tcp
          destination_port_range: 22
          access: Allow
          priority: 1001
          direction: Inbound
        - name: HTTP
          protocol: Tcp
          destination_port_range: 80
          access: Allow
          priority: 1002
          direction: Inbound
        - name: HTTPS
          protocol: Tcp
          destination_port_range: 443
          access: Allow
          priority: 1003
          direction: Inbound

  - name: Create virtual network interface card
    azure_rm_networkinterface:
      resource_group: "{{ resource_group }}"
      name: "{{ vm_NIC }}"
      virtual_network: "{{ vm_net }}"
      subnet: "{{ vm_subnet }}"
      public_ip_name: "{{ vm_publicIP }}"
      security_group: "{{ vm_NSG }}"

  - name: Create VM
    azure_rm_virtualmachine:
      resource_group: "{{ resource_group }}"
      name: "{{ vm_Name }}"
      vm_size: "{{ vm_size }}"
      admin_username: "{{ os_user }}"
      admin_password: "{{ os_pass }}"
      ssh_password_enabled: true
      network_interfaces: "{{ vm_NIC }}"
      image:
        offer: "{{ vm_offer }}"
        publisher: "{{ vm_pub }}"
        sku: "{{ vm_sku }}"
        version: latest

  - name: Peer Old To New
    azure_rm_virtualnetworkpeering:
      resource_group: "{{ resource_group }}"
      virtual_network: "{{ net }}"
      name: "{{ vm_peer }}"
      remote_virtual_network:
              resource_group: "{{ resource_group }}"
              name: "{{ vm_net }}"
      allow_virtual_network_access: true
      allow_forwarded_traffic: true

  - name: Peer New To Old
    azure_rm_virtualnetworkpeering:
      resource_group: "{{ resource_group }}"
      virtual_network: "{{ vm_net }}"
      name: "{{ net }}"
      remote_virtual_network:
              resource_group: "{{ resource_group }}"
              name: "{{ net }}"
      allow_virtual_network_access: true
      allow_forwarded_traffic: true
```

At the end of this playbook we added the azure_rm_virtualnetworkpeering module, this will allow devices in different networks to communicate with each other