MY REPOSITORY:
https://github.com/BoikenZ/network-automation-lab.git

Outline

- 01: Initial setup / installations
  - Deploy Ubuntu virtual machine
  - Clone Ansible AWX GitHub repository and run install script
  - Access AWX
- 02: Configuring AWX
  - Create an Inventory
  - Set up SSH credentials for network devices
  - Crate an AWX project from a GitHub repository.
- 03: Creating and Running jobs
  - Create a job template
  - Run the job
  - Create Workflow
  - Create a survey

Introduction

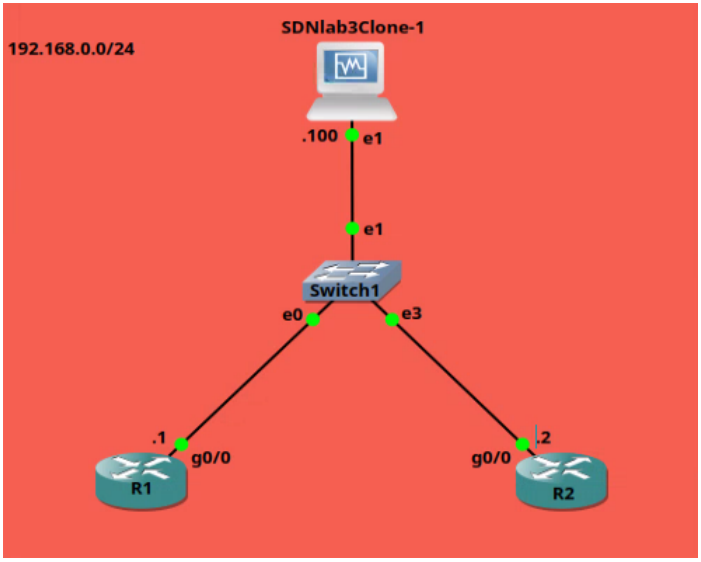
In this lab, we will be using an Ubuntu Virtual Machine running AWX to configure connected network devices. We will be using GNS3 to simulate the network devices (in our case, we will be using 2 cisco c7200 routers.) We will be using VirtualBox for the Virtual Machine. GNS3 allows us to import VirtualBox VMs to be used.

Step 1: Initial Setup and Installations

Step 1.1: Deploy Ubuntu Virtual machine

- Using virtual box, create a new Ubuntu virtual machine.
- After the virtual machine has been created, turn it off.
- Open GNS3, at the bottom of the All Devices tab, click on: New Template > Manually create a new template > Next > VirtualBox VMs > New. In the VM list, click and choose the VM we just created. Afterwards, click edit, go to the network tab and set the amount of adapters to 2.

- In GNS3, this is the topology ive created:



- make sure to connect the 2nd adapter to the router, as VirtualBox has the 1st adapter set to the bridged adapter.

Step 1.2: IP configurations

Now we are inside the Ubuntu VM. We should have internet connectivity, though, we want to assign an IP address to the adapter that is connected to the topology.

Edit Netplan

run `ip a` to see which of the 2 adapters dont have an ip address yet. In our case, its enp0s8

```
sudo vim /etc/netplan/01-network-manager-all.yaml
```

Configure your netplan file so it looks something like this:

```
network:
    version: 2
    renderer: NetworkManager
    ethernets:
        enp0s8:
            dhcp4: false
            addresses:
                - 192.168.0.100/24
```

After you've saved the file, run `sudo netplan apply` to apply the changes. run `ip a` to ensure the changes have been applied.

Now, we want to do basic configurations on the routers. All we need to do is set an IP address for the interface connect:

Router 1

```
conf t
int g0/0
ip address 192.168.0.1 255.255.255.0
no shut
```

Router 2

```
conf t
int g0/0
ip address 192.168.0.2 255.255.255.0
no shut
```

We also need to enable ssh on our routers:

both routers

```
ip domain-name www.example.com

crypto key generate rsa
ip ssh time-out 60
ip ssh authentication-retries 3

line vty 0 4
transport input ssh
login local

username root password telecomS144
```

this will generate rsa keys, disable telnet (only ssh), and set a username and password we can use to connect to it.

These configurations should suffice for this lab. Now, you should be able to ping the routers from the virtual machine.

1.3 AWX Install

Now we can install AWX using the provided install script.

Run the following commands to install the AWX:

1. `wget https://raw.githubusercontent.com/maniak-academy/guide-k3s-awx-tower/main/install.sh`
2. `chmod +x install.sh`
3. `sudo ./install.sh`

Now we want to run `kubectl -n awx get awx,all,ingress,secrets` to get the ip address its running on.

Then, we want to modify our `/etc/hosts` file to add a dns entry to access awx.

/etc/hosts

```
x.x.x.x    awx.example.com
```

add this entry (replace x.x.x.x with the ip address shown by the previous command)

Now, if we open firefox on the VM, and access `awx.example.com`, It should bring us to the awx landing page. Log in with the credentials:

- Username: `admin`
- Password: `Ansible123!`

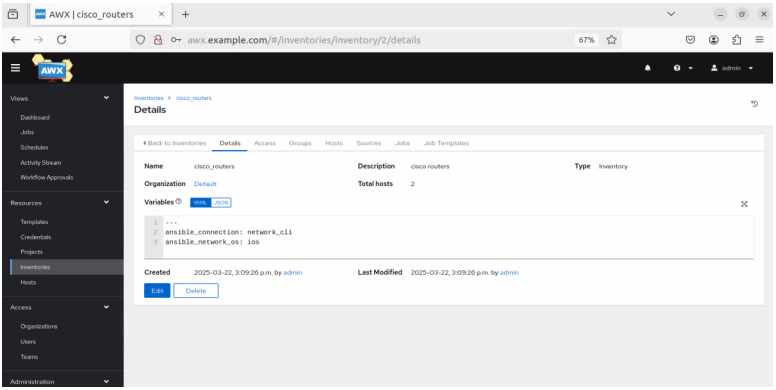
Step 2: Configuring Ansible AWX

This step is relatively straight forward.

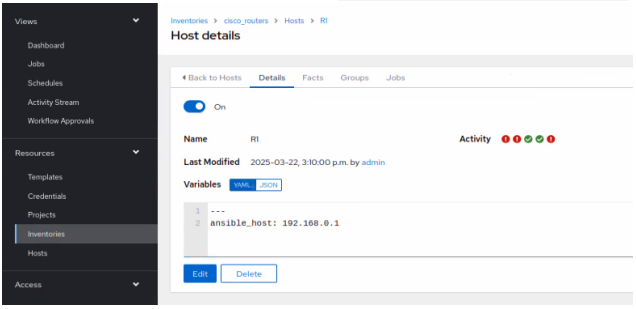
We create a Project, which we link to a GitHub repository that we have connected to it. for ease of use, ensure this github repository is public, if not, we need to generate a Personal Access Token to give the AWX project access to it.

We now create an inventory (mine is called cisco_routers), and in the variables section, i defined:

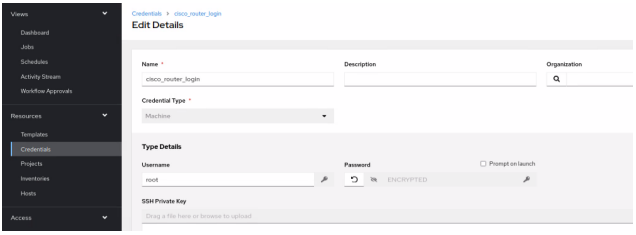
```
ansible_connection: network_cli
ansible_network_os: ios
```



Next, we click on the hosts tab, and define our 2 routers. I have 2. R1 and R2, and in the variables section on each, i have `ansible_host: 192.168.0.x` defined.



Next, we go to credentials, and create a credential called `cisco_router_login`. Set credential type to `machine`, and use the username and password that we set in the cisco router configurations. This credential will be used to connect when we run the playbooks.



Step 3: Creating and Running Ansible Jobs

Now we can almost start creating and running our jobs.

ensure, that in our GitHub repository that we have linked to the project, we have a `requirements.yml` file in it that reads the following:

```
collections:
    - name: cisco.ios
```

This ensures that when we run the job, we have the cisco.ios module installed.

In my case, i will be using this playbook:

Code Blame 18 lines (16 loc) · 356 Bytes

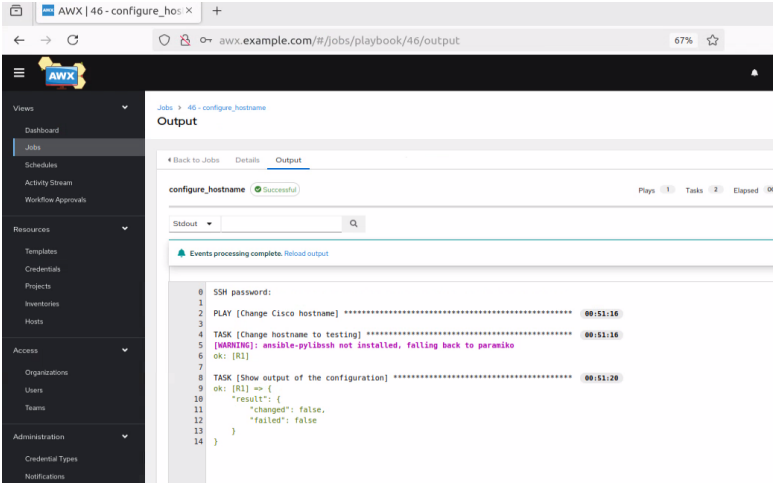
```
---
- name: Change Cisco hostname
  hosts: R1
  gather_facts: no
  tasks:

    - name: Change hostname to testing
      ios_config:
        lines:
          - "hostname testing"
        save_when: modified
      register: result
      become: yes
      become_method: enable

    - name: Show output of the configuration
      debug:
        var: result
```

This will change the hostname of the cisco router to `testing`

Now to create and run the job, we go projects > NetworkAutomation *(this is the name of the project i created)* > Job Templates > Add. Add the following:
Name: `hostname config`
Inventory: `cisco_routers`
Project: `NetworkAutomation`
Playbook: `change_hostname.yml` *if your playbook is not showing up in the dropdown menu on the playbook part, ensure to sync up your project.*
Credentials: `cisco_router_login`

and now save.

We can now click on the job template, and hit launch.



If we look at the output, we can see that it has executed successfully, though, nothing has changed because my router's hostname was already sent to testing.

Part 4: Reflection and Exploration

Reflective Essay:

Automation is essential in modern IT, as it enhances efficiency, consistency and scalability, Ansible AWX provides a web-based interface, api and task / inventory / job system for managing Ansible Playbooks. It has multiple possible use cases in IT automation. Including but not limited to:

1. Network Config Management:
    1. It allows for centralized management of network configs across multiple devices
    2. Reduces manual errors: Network device configuration is tedious and prone to human error, This reduces these errors.
2. Cloud Infrastructure:
    1. Automates the deployment and scaling of virtual machines and containers
    2. Has integration with cloud providers like AWS and Azure

Ansible AWX significantly enhances IT automation by providing a user friendly, centralized and scalable way to manage Ansible Playbooks. This ability to automate network configurations and enhance deployment of cloud infrastructure makes it an amazing too to increase efficiency when it comes to IT.

Exploring additional features:

Dynamic Inventory is one feature I took a look at. It allows admins to get host information from other sources (AWS/Azure, databases, etc) instead of relying on your static inventory files.

You can simply create one by going to inventory > Add Inventory, Give it a name.
Next, you click the Sources tab > Add source, and now select where to get the information from (Amazon ec2, VMware, Microsoft Azure Resource Manager, Google Compute engine, along with more sources.)

Then you just sync the inventory.

Troubleshooting

SSH error

```
Unable to negotiate with 192.168.0.1 port 22: no matching key exchange
method found. They offer: ........
```

This error happened because the cisco ios images we are using are old, and there were no key exchange method, ciphers, and host key algorithms shared between the router and the VM. To fix this, we edit the `./ssh/config` file and append the following lines:

./ssh/config

```
Host R1
    Hostname 192.168.0.1
    User root
    KexAlgorithms +diffie-hellman-group-exchange-sha1
    Ciphers +aes128-cbc
    HostKeyAlgorithms +ssh-rsa
```

Replace each of the fields with what the router was offering (for me, it was requesting those three.)

Error with running the job

I had 2 problems with this.
The first error i had was that it was unable to communicate with the cluster. This was due to a networking error. What caused the problem was that i had the VM hooked up to a GNS3 NAT when i ran the install script. The fix was to restart on a fresh VM, and just keep the default VirtualBox bridged adapter for internet connectivity.

The second error i encountered was because `cisco.ios` module was not resolved. This error was because i didn't have the `requirements.yml` file in my GitHub repository. Once i added it, the error went away.