

Guided Practice: Run Ansible Playbooks on Multiple Machines

Introduction

Ansible is best able to configure and manage systems that have been specified in an inventory file. Groups are used to identify specific servers and their functions. In this exercise you will configure Ansible to run on multiple machines with different operating systems and create Playbooks that configure your environment.

Outcome

In this Guided Practice, you will first edit the Ansible Inventory (hosts) file and then you will create and run Ansible playbooks from your CentOS computer to configure the Ubuntu and Windows systems on your network.

Resources Needed

- For this Guided Practice, we will use the CentOS 8, Windows 2019 server, and the Ubuntu 20.04 LTS machines in the VCastle pod configured for this class.

Level of Difficulty

Moderate to high

Deliverables

Deliverables are marked with a red border around the screenshot. Additionally, there are guide practice questions at the end which you must respond to. **Your username or studentID should be visible in all screenshots that you submit.**

General Considerations

You should be familiar with Linux networking. Secure Shell (ssh) should already be configured on your Ubuntu computer from a prior Guided Practice. Ansible should be installed and configured on your CentOS Computer from a prior Guided Practice. Optimizing your text editor for writing playbooks in YAML should make your job much easier. This was completed in a previous exercise. We suggest using either vim, nano, or gedit. There are also other editors like Atom or Emacs that are available and can be configured for writing YAML code.

Edit the Ansible Inventory (Host) File to Reflect your Network Architecture

1. Edit the inventory (hosts) file. Please note that **the IP addresses shown in the examples may not reflect your network configuration**. The webserver and dbserver IP address for your network will most likely be 192.168.1.3 for both and the win machine 192.168.1.2. The multiple entries are intended to demonstrate slightly more complex scenarios possible with Ansible. The [win:vars] section must be entered as shown. The examples have been developed using vim as an editor configured for YAML as shown in a previous Guided Practice.

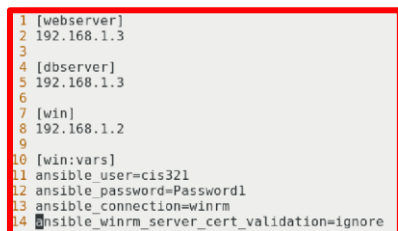
Type:

```
sudo rm /etc/ansible/hosts and press <ENTER>
sudo vim /etc/ansible/hosts
```

Press **i** to enter the insert mode, and type the following:

```
[webserver]
192.168.1.3
<ENTER>
[dbserver]
192.168.1.3
<ENTER>
[win]
192.168.1.2
[win:vars]
ansible_user=cis321
ansible_password=Password1
ansible_connection=winrm
ansible_winrm_server_cert_validation=ignore
```

Take a screenshot of your file. It should resemble the one below:



```
1 [webserver]
2 192.168.1.3
3
4 [dbserver]
5 192.168.1.3
6
7 [win]
8 192.168.1.2
9
10 [win:vars]
11 ansible_user=cis321
12 ansible_password=Password1
13 ansible_connection=winrm
14 ansible_winrm_server_cert_validation=ignore
```

Enter the following text to close your editor:

```
Press <ESC> and type  
:wq and press <ENTER>
```

Write a Playbook to Check for a Website

In this portion of the assignment, we will write a Playbook to test your new hosts file checking for the presence of the website you created in a previous Guided Practice.

NOTE: It is recommended that you create a directory under your user's home directory for all your playbooks.

1. Create the CheckWebsite.yml playbook to validate the new hosts file. Type the following:

```
vim CheckWebsite.yml <ENTER>  
i (to enter insert mode)  
---  
- hosts: webserver  
  tasks:  
    - name: Ensure website is present and updated  
      file:  
        path: /var/www/html  
        state: directory  
        owner: root  
        group: root  
        mode: 0655  
        recurse: yes  
      become: True  
<ESC> :wq
```

NOTE: Please remember that, depending upon your vim configuration, the spaces may show up as periods or as spaces. (Of course, the periods in the url always appear as periods.)

```

1 ---
2 -.hosts:.webserver
3 ..tasks:
4 ....- .name:.Ensure.website.is.present.and.updated
5 .....file:
6 .....path:./var/www/html
7 .....state:.directory
8 .....owner:.root
9 .....group:.root
10 .....mode:.0655
11 .....recurse:.yes
12 .....become:.True

```

Type:

```

ansible-playbook CheckWebsite.yml --syntax-check
ansible-playbook CheckWebsite.yml --ask-become-pass

```

Enter the password when asked. Note the IP address in the output below will be different and reflect your network configuration and hosts file.

```

[alex@localhost ansible]$ ansible-playbook CheckWebsite.yml --ask-become-pass
BECOME password:

PLAY [webserver] *****

TASK [Gathering Facts] *****
ok: [192.168.0.249]

TASK [Ensure website is present and updated] *****
ok: [192.168.0.249]

PLAY RECAP *****
192.168.0.249      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescue
d=0    ignored=0

```

Write a Playbook to Install mysql-server on Your Database Server and Start the mysql Service.

1. Type:

```

vim MySQLInstall.yml <ENTER>
i (to enter insert mode)
---
- name: Install Mysql and start mysql service
  become: yes
  hosts: dbserver

```

```

tasks:
  - name: Install Mysql
    apt: pkg=mysql-server state=latest

  - name: Start and enable mysql service
    service:
      name: mysql
      state: started
      enabled: yes

```

NOTE: Please remember that, depending upon your vim configuration, the spaces may show up as periods or as spaces. (Of course, the periods in the url always appear as periods.)

```

1 ---
2 -.name:.Install.Mysql.and.start.mysld.service
3 ..become:.yes
4 ..hosts:.dbserver
5 ..tasks:
6 ....-.name:.Install.Mysql
7 .....apt:.pkg=mysql-server.state=latest
8
9 ....-.name:.Start.and.enable.mysql.service
10 .....service:
11 .....name:.mysql
12 .....state:.started
13 .....enabled:.yes

```

```

ansible-playbook MySQLInstall.yml --syntax-check <ENTER>
ansible-playbook MySQLInstall.yml --ask-become-pass <ENTER>

```

```

PLAY [Install Mysql and start mysld service] *****
*****
TASK [Gathering Facts] *****
*****ok: [192.168.0.237]

TASK [Install Mysql] *****
*****ok: [192.168.0.237]

TASK [Start and enable mysql service] *****
*****ok: [192.168.0.237]

PLAY RECAP *****
*****192.168.0.237 : ok=3
changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

2. On the Ubuntu server, log in and enter the following code to ensure the mysql service is running. Type:

```
sudo systemctl status mysql.service
```

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2020-12-20 17:44:13 EST; 37min ago
     Process: 645 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 833 (mysqld)
      Status: "Server is operational"
        Tasks: 37 (limit: 4657)
       Memory: 388.6M
      CGroup: /system.slice/mysql.service
             └─833 /usr/sbin/mysqld
```

Write a Playbook to Access and Perform Simple File Commands on the Windows Machine using Ansible

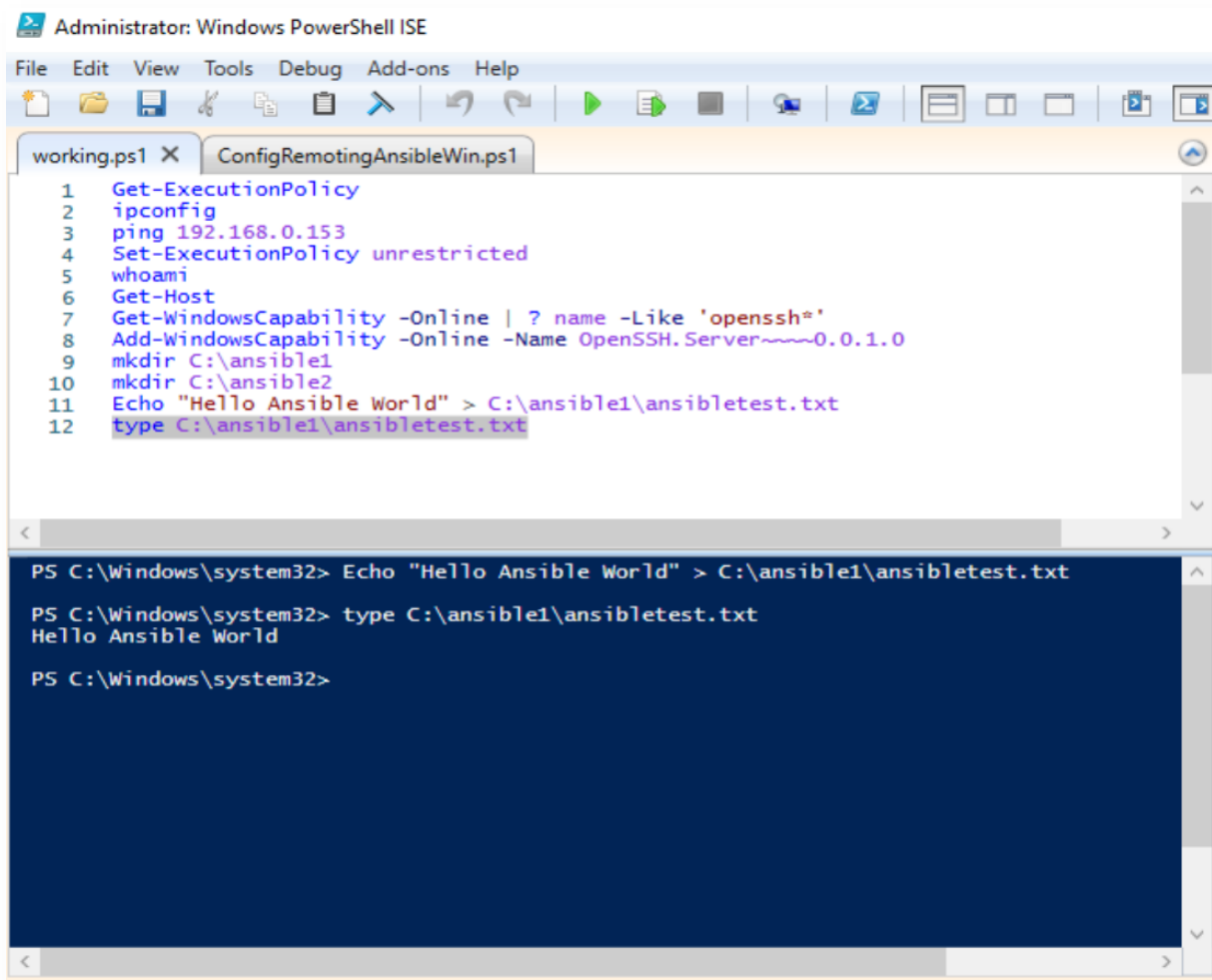
1. From your CentOS machine, ensure you can ping your Windows machine. Type the following:

```
ansible win -m win_ping (and press <ENTER>)
```

```
[alex@localhost ~]$ ansible win -m win_ping
192.168.0.77 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

2. On your Windows machine, create the directories **c:\ansible1** and **c:\ansible2** and populate the **ansible1** directory with a file name **ansibletest.txt** containing the text "Hello Ansible World."

The example below is using the PowerShell ISE, but you may use a different method to create the directories and the file.



The screenshot shows the Windows PowerShell ISE interface. The top pane displays a script named 'ConfigRemotingAnsibleWin.ps1' with 12 lines of PowerShell commands. The bottom pane shows the execution output of these commands.

```
1 Get-ExecutionPolicy
2 ipconfig
3 ping 192.168.0.153
4 Set-ExecutionPolicy unrestricted
5 whoami
6 Get-Host
7 Get-WindowsCapability -Online | ? name -Like 'openssh*'
8 Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
9 mkdir C:\ansible1
10 mkdir C:\ansible2
11 Echo "Hello Ansible World" > C:\ansible1\ansibletest.txt
12 type C:\ansible1\ansibletest.txt
```

```
PS C:\Windows\system32> Echo "Hello Ansible World" > C:\ansible1\ansibletest.txt
PS C:\Windows\system32> type C:\ansible1\ansibletest.txt
Hello Ansible World
PS C:\Windows\system32>
```

3. Create an Ansible playbook that will manipulate files on the Window server. Please note that your IP addresses may be different from those shown in the screen snapshots, reflecting your network topology. Additionally, when editing YAML files, remember that blank lines are permissible, but whitespace must adhere to strict formatting rules.

Type:

```
vim WinFileCopy.yml
i
---
- hosts: win
  tasks:
    - name: Copy File
      win_copy:
        src: C:\ansible1\ansibletest.txt
```

```
dest: C:\ansible2\ansibletest.txt
remote_src: yes
<ESC>:wq
```

NOTE: Please remember that, depending upon your vim configuration, the spaces may show up as periods or as spaces. (Of course, the periods in the url always appear as periods.)

```
1 ---
2
3 -.hosts:.win
4
5 ..tasks:
6
7 ..-.name:.Copy.File
8
9 ....win_copy:
10
11 .....src:.C:\ansible1\ansibletest.txt
12
13 .....dest:.C:\ansible2\ansibletest.txt
14
15 .....remote_src:.yes
```

Type:

```
ansible-playbook WinFileCopy.yml --syntax-check (and press <ENTER>)
ansible-playbook WinFileCopy.yml (and press <ENTER>)
```

```
[alex@localhost ~]$ vim WindowsFileCopy.yml
[alex@localhost ~]$ ansible-playbook WindowsFileCopy.yml --syntax-check

playbook: WindowsFileCopy.yml
[alex@localhost ~]$ ansible-playbook WindowsFileCopy.yml

PLAY [win] *****

TASK [Gathering Facts] *****
ok: [192.168.0.77]

TASK [Copy File] *****
changed: [192.168.0.77]

PLAY RECAP *****
192.168.0.77          : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. Verify that the file has been copied. You may use the command line, Windows Explorer, or PowerShell to validate the success of the playbook.

The screenshot shows the Windows PowerShell ISE interface. The top menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. Below the menu is a toolbar with icons for file operations and execution. Two tabs are open: 'working.ps1' and 'ConfigRemotingAnsibleWin.ps1'. The 'ConfigRemotingAnsibleWin.ps1' tab is active and contains a PowerShell script with 14 lines of code. The script performs various system tasks: getting execution policy, displaying IP configuration, pinging 192.168.0.153, setting execution policy to unrestricted, displaying the user name, getting the host name, checking Windows capabilities for OpenSSH, adding the OpenSSH.Server capability, creating two directories (C:\ansible1 and C:\ansible2), creating a text file in C:\ansible1, displaying the contents of the text file, and listing the contents of both directories. The output of the script is displayed in a dark blue console window at the bottom. It shows the directory listing for C:\ansible1, followed by the command to list C:\ansible2, and then the directory listing for C:\ansible2. Both listings show a file named 'ansibletest.txt' with a length of 44 bytes, created on 12/13/2020 at 2:58 PM.

```
1 Get-ExecutionPolicy
2 ipconfig
3 ping 192.168.0.153
4 Set-ExecutionPolicy unrestricted
5 whoami
6 Get-Host
7 Get-WindowsCapability -Online | ? name -Like 'openssh*'
8 Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
9 mkdir C:\ansible1
10 mkdir C:\ansible2
11 Echo "Hello Ansible World" > C:\ansible1\ansibletest.txt
12 type C:\ansible1\ansibletest.txt
13 dir C:\ansible1
14 dir C:\ansible2
```

Directory: C:\ansible1

Mode	LastWriteTime	Length	Name
-a----	12/13/2020 2:58 PM	44	ansibletest.txt

PS C:\Windows\system32> dir C:\ansible2

Directory: C:\ansible2

Mode	LastWriteTime	Length	Name
-a----	12/13/2020 2:58 PM	44	ansibletest.txt

Guided Practice Questions

In your **Guided Practice Lab Report**, answer the following questions about this learning activity. Some may require research.

1. What is idempotency, and why is this a desirable feature for configuration management applications?
2. Why is there a separate “ping” module for windows hosts?
3. Would you write separate playbooks to manage Linux and Windows configurations or use a single playbook? Explain.
4. When should the ask-become-pass option be invoked with a playbook?

References

- GeekFlare. (2020). *9 Ansible playbooks example for Windows administration*.
<https://geekflare.com/ansible-playbook-windows-example/>
- Henderson, B. (2019). Connecting to a Windows host.
<https://www.ansible.com/blog/connecting-to-a-windows-host>
- Hochstein, L. & Moser, R. (2017) *Ansible: Up and running*. [Kindle].
- JavaTpoint. (2018). *Ansible commands cheat sheets*. <https://www.javatpoint.com/ansible-commands-cheat-sheets>
- Kenlon, S. (2019). *10 YAML tips for people who hate YAML*.
<https://www.redhat.com/sysadmin/yaml-tips>
- LinuxBuz. (2020). *Ansible apt module – tutorial and examples*.
<https://linuxbuz.com/linuxhowto/ansible-apt-module>
- NGEL. (2020). *YAML Script: Second ansible playbook to get date and server uptime status*.
<https://ngelinux.com/yaml-script-first-ansible-playbook-to-get-date-and-server-uptime-status/>
- OS Radar. (2020). *How to enable SSH in Windows Server 2019*. <https://www.osradar.com/how-to-enable-ssh-in-windows-server-2019/>
- RedHat. (2020a). Using Ansible and Windows.
https://docs.ansible.com/ansible/latest/user_guide/windows_usage.html
- Redhat. (2020b). Drive automation across open hybrid cloud deployments.
<https://www.ansible.com/>
- RedHat. (2020c). Glossary.
https://docs.ansible.com/ansible/latest/reference_appendices/glossary.html
- Kelom. (2020). Ansible - MYSQL installation. <https://medium.com/@kelom.x/ansible-mysql-installation-2513d0f70faf>