Automation Factory

Ansible for Windows environment

Ganesh Palnitkar

17

Using Ansible to control remote windows server

Install Latest version of Ansible to have windows support and windows Modules available.

This can be done by updating the Linux repo with below command.,

```
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```

Or in case of RedHat,

```
# install the epel-release RPM if needed on CentOS, RHEL, or Scientific Linux
$ sudo yum install ansible
```

Instead of using ssh on windows, we will make use of 'winrm'.

To check if winrm is running on the remote windows server, use below command.

From windows PowerShell prompt, run,

C:\Test-WSMan 192.168..33.91 …. Here the IP address is for the remote windows server.

Once the Linux server is up and running, go ahead and install Ansible using either of the multiple ways explained in other document.

In order to make sure Ansible is able to communicate with Windows remote server over 'winrm', we must first install the 'pywinrm' python package on the Ansible server form which we want to control our remote windows server. This can be done on the Ansible server using below command.,

```
$ sudo pip install xmltodict
```

```
$ sudo pip install pywinrm
```

And then install the Ansible server using below command.

```
$ sudo pip install ansible
```

Once we have Ansible up and running, create an inventory file and mentioned the Windows Remote server ip address in it under the web group as [web].

After this, create a 'group_vars' folder and create a web.yml file to declare the variables.

```
ansible_user: vagrant
ansible_password: vagrant
ansible_port: 5985
ansible_connection: winrm
ansible_winrm_cert_validation: ignore
```

Save the file and to test the ansible connections with remote Windows server run the command,

```
$ ansible web –i inventory –m win_ping
```

Using Ansible Windows Module.

All of the windows modules start with the name as 'win_'

Try running the setup command which is command for all platforms.

$ ansible web –i inventory –m setup
$ ansible web –i inventory –m raw –a "ipconfig" …. Using the raw module one can run windows command on the remote windows server.

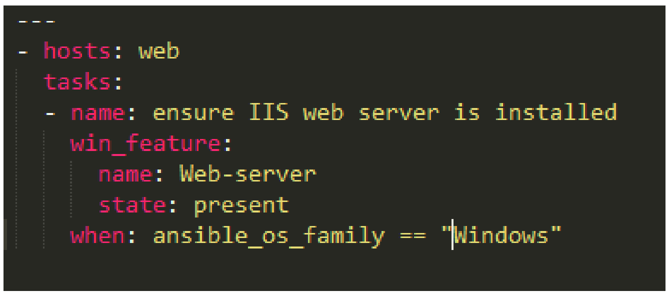
We can also start stop windows services using the 'win_service' module as shown below.,

$ ansible web –i inventory –m win_service –a "name=spooler state=started" … if we do not pass the state attribute we can get the status of the service.

**** If we want to install or enable a windows feature on the remote windows server we can use the module as mentioned below.

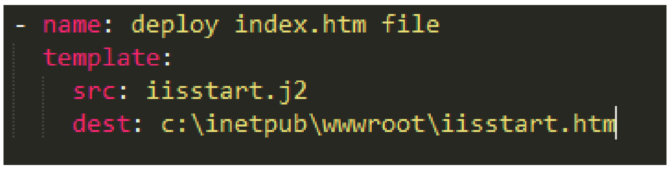
$ ansible web –i inventory –m win_feature –a "name=Telnet-Client state=present"

Let's now create a **playbook** to install iis server on the remote windows server. IIS being inbuilt feature of windows server, we can use the 'win_feature' module.



```
---
- hosts: web
  tasks:
  - name: ensure IIS web server is installed
    win_feature:
      name: Web-server
      state: present
    when: ansible_os_family == "Windows"
```

We can use the 'template' or 'copy' module to update the default page file.



```
- name: deploy index.htm file
  template:
    src: iisstart.j2
    dest: c:\inetpub\wwwroot\iisstart.htm
```

Now if we want to encrypt the files which are holding sensitive information, we can do that using the ansible-vault command as shown below.

$ ansible-vault encrypt group_vars/all.yml …. This will first prompt for setting up a vault password, once set, we can only run the ad-hoc commands or playbook with sensitive information with '—ask-vault-pass' command only.

So now if we try to run the ad-hoc command or running the playbook, it will return an error as decryption failed. So to use the vault, we can use the command,

$ ansible web –i inventory –m ping - -ask-vault-pass … this will prompt for entering vault password and then the command runs successfully.

Roles:

Roles are compartmentalized collections of tasks, templates, variables, and more. Ansible role represents and fully fledged server role, like a webserver, a database server, etc.

Ansible role follows a folder structure like one shown below. There will be tasks, templates and handlers and vars folders inside a role.

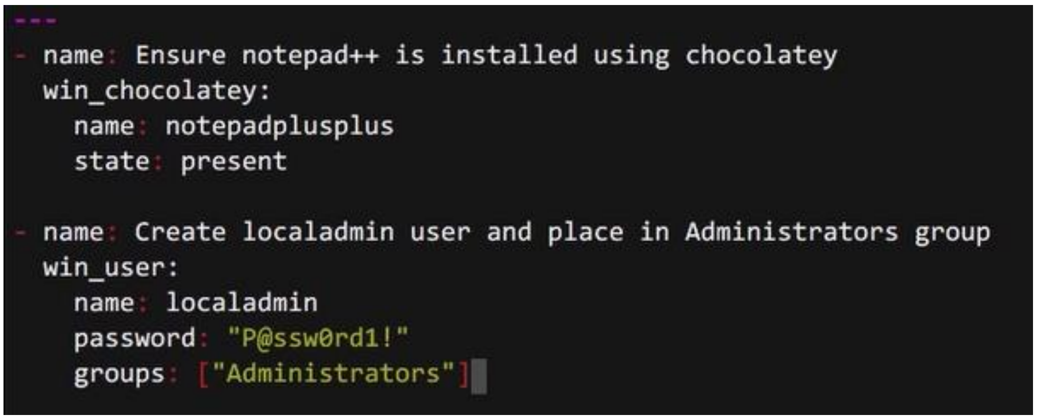
All the tasks that we might have defined inside a playbook, are written in the main.yml file inside the tasks folder.

All vars, handlers that we have written inside playbook are then written in the main.yml file inside the respective folders.

The folder structure of the role is somewhat as displayed below.

Also some of the common tasks that can be applied to all servers can be mentioned inside a ‘common’ role.

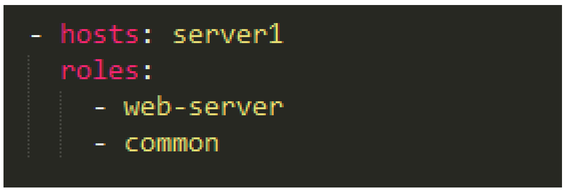
../roles/main.yml …. The tasks list. Here the win_chocolatey is a module which can be used for installing software packages like notepad++, 7ZIP… (Utility software)



```
---
- name: Ensure notepad++ is installed using chocolatey
  win_chocolatey:
    name: notepadplusplus
    state: present

- name: Create localadmin user and place in Administrators group
  win_user:
    name: localadmin
    password: "P@ssw0rd1!"
    groups: ["Administrators"]
```

../webserver.yml …. Calling the roles in he playbook placed outside the roles directory.



```
- hosts: server1
  roles:
    - web-server
    - common
```