NSX Infrastructure Deployment using Ansible

Up and Running

Joshua Newton
Staff Technical Account Manager

Sometime during 2022

Special Thanks

Recognition

- **Michel Berry** – Staff Technical Account Manager
  - Testing / Validation / Peer Review
  - Inspiration of this Deck
- **William Nead** – Senior Technical Account Manager
  - Testing / Validation / Peer Review
- **Steven Brown** – Staff Consulting Architect
  - Testing / Validation / Peer Review

Agenda

- Introduction
- What is Ansible
- Ansible - The Basics
- Ansible – Ad Hoc
- Ansible-Vault
- YAML
- Templates
- Ansible Installation
- NSX Ansible Modules
- Git / Github / SSH Keys
- GitHub Repository
- Visual Studio Code (VS Code)
- Deployment
- Resources

Introduction
Content Focus

This content is designed to get you up and running with Ansible for NSX as quickly as possible

- Understanding Basic Ansible
- Requirements / Installation
- Operations: Ansible Playbooks with NSX Modules
- Resources

This content is designed for people who are familiar with NSX but have little or no experience with Ansible or Configuration Management tools

Note
I am NOT an Ansible Expert but have spent a good amount of time learning / understanding Ansible
(trying to at least)

What is Ansible?

What is Ansible?

Provisioning

Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

It was created by Michael DeHaan in 2012 and later acquired by Red Hat, Inc in 2015.

Ansible is written in Python with playbooks in easy to read and work with YAML

https://docs.ansible.com/ansible/latest/index.html

What is Ansible?

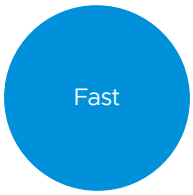
Benefits

Playbooks written in YAML

User friendly

Easy to understand

No need to learn programming languages like Ruby, HCL, etc



Fast to learn

Fast to setup

No agents or daemons to install

No extra software on NSX Managers or ESXi hosts

Re-use same file multiple times for different environments

HashiCorp Configuration Language (HCL)

Ansible Base Concepts
https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html

Modules
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html
https://docs.ansible.com/ansible/latest/collections/community/vmware/index.html
https://docs.ansible.com/ansible/latest/collections/community/network/nclu_module.html

Tasks / Playbooks
https://docs.ansible.com/ansible/latest/user_guide/index.html#writing-tasks-plays-and-playbooks

Inventories
https://docs.ansible.com/ansible/latest/getting_started/get_started_inventory.html#
https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#intro-inventory
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/host_group_vars_vars.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

Ansible – The Basics

Ansible – The Basics
Connections

Ansible is an agentless architecture that will use a trusted relationship between the Ansible control machine and the target (remote) host for automated passwordless connectivity

So how does Ansible connect to remote hosts?

- SSH + SSH Keys pairs (public and private keys)
  - Remote hosts will have the authorized_keys file populated with the SSH Public Key
  - Ansible control machine will use known_hosts + SSH Private Key
- Possible custom logins provided by modules
  - Modules like vmware.ansible_for_nsxt and community.vmware provide ability to login to NSX Manager / vCenter which do not use the SSH Keys
  - Which makes it easier to get started quickly

ssh-keygen <enter> will create a basic RSA-SHA2-512 public / private key. Passphrase is not needed.

To create a different security type key you can use the –t <type> and –C to add comments to it such as email address it's with.

ssh-keygen –t ed25519 –C "myemail@address.com"

Once a key has been created you can use the ssh-copy-id command to copy the key to your destination

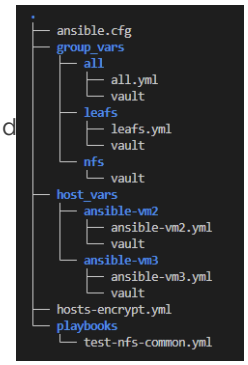
ssh-copy-id user@address (if you do not use the –i and say which file it will copy all .pub files over to target address)

Ansible – The Basics
Files

Ansible follows a directory structure – because of this it is easy to create / handle multiple different projects all in their own directories.

Ansible uses the following files or directories for configure, inventory, playbooks, roles and variables.

- Ansible.cfg – Defines the working area
- Hosts file – Defines inventory of objects and possible properties
- group_vars / host_var (directories) – Specific properties / variables for d inventory objects
- playbooks – Normal spot for playbooks (groups of tasks to run)
- roles (directory) – Allows for easily reusing of grouped content (Not going to cover roles in this session)



https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html#directory-layout

Ansible – The Basics

Ansible Configuration File

The ansible.cfg file can be used to define parameters, this can be global or local depending on where the ansible.cfg file is located. Oddly it's NOT required but will greatly limit your abilities with Ansible.

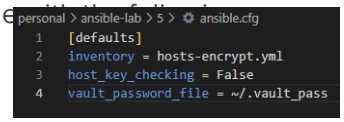
The Ansible Configuration file is created using the INI format

Example:

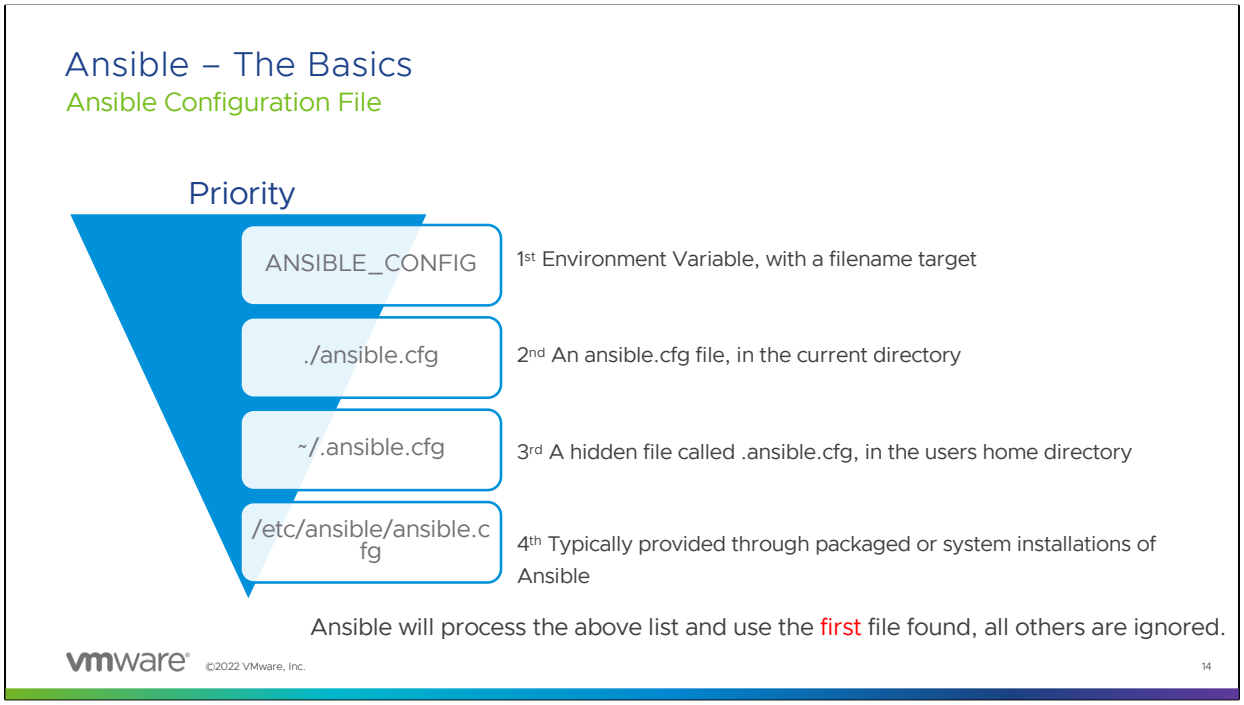
```
[defaults]                          ← Section
inventory = hosts                       ← Inventory file location / name (hosts)
host_key_checking = False           ← Variable
vault_password_file = ~/.vault_pass ← Location of Vault Password
```

You can generate a fully commented out example ansible.cfg file [illegible] command

ansible-config init --disabled > ansible.cfg



https://en.wikipedia.org/wiki/INI_file
https://docs.ansible.com/ansible/latest/reference_appendices/config.html



https://docs.ansible.com/ansible/latest/reference_appendices/config.html

The configuration file

Changes can be made and used in a configuration file which will be searched for in the following order:

ANSIBLE_CONFIG (environment variable if set) (E.G. "**export ANSIBLE_CONFIG=/home/ansible/this_is_my_example_ansible.cfg**" to set this variable and use "**unset ANSIBLE_CONFIG**" to clear)

ansible.cfg (in the current directory)

~/.ansible.cfg (in the home directory)

/etc/ansible/ansible.cfg

Ansible will process the above list and use the first file found, all others are ignored.

Running the command “**ansible --version”** will tell you which configuration file will be used from the directory that the command was run from. (E.G. **ansible --version** in the users home directory where a .ansible.cfg file is located will provide

Ansible – The Basics

Ansible Inventory File

Inventories organize managed nodes in centralized files that provide Ansible with system information and network locations. Using an inventory file, Ansible can manage a large number of hosts with a single command. Inventories also help you use Ansible more efficiently by reducing the number of command-line options you need to specify.

- Inventory files can be in INI or YAML format
- Variables can be assigned within the Inventory file, they can be per object (single host), applied to a group of objects (multiple hosts) or even nested groups.
- Variables can be plain text or encrypted using ansible-vault

https://docs.ansible.com/ansible/latest/getting_started/get_started_inventory.html#
https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html#intro-inventory

Ansible – The Basics

Ansible Inventory File Examples

ansible-lab > 0 > hosts

```
[all]
[all:vars]
ansible_become_pass='VMware1!'
[control]
taco ansible_connection=local
[nfs]
ansible-vm[1:2] ansible_user=ubuntu
ansible-vm3
[nfs:vars]
ansible_user=ansible
ansible_become_pass='VMware1!'
[leafs]
cumulus-leaf-[1:2]
[leafs:vars]
ansible_user=cumulus
ansible_become_pass='VMware1!'
```

ansible-lab > 4 > ! hosts-encrypt.yml

```
---
all:
  hosts:
  vars:
    ansible_user : ubuntu
    ansible_become_pass: "{{ vault_ansible_become_pass }}"
  children:
    control:
      hosts:
        taco:
          ansible_connection: local
    nfs:
      hosts:
        ansible-vm[1:2]:
        ansible-vm3:
          ansible_user: ansible
        #ubuntu-[00:01]:
      vars:
    leafs:
      hosts:
        cumulus-leaf-[1:2]:
      vars:
        ansible_user: cumulus
        ansible_become_pass: "{{ vault_cumulus_become_pass }}"

...
```


On left INI formatted host files
On right YAML formatted host file

Ansible – The Basics

group_vars and host_vars – Inventory expanded

The directories are used to add control variables to different objects, this allows your playbooks to be as generic as possible

group_vars

- Allows you to apply variables for playbooks or add SSH login information or sudo password for multiple objects

host_vars

- Same as group_vars but is based on a specific host being called

Example:

Hosts 1-3 are all Ubuntu Linux, but Host 3 uses a different user/password

You could supply the user / password for all the Hosts in the group_vars directory, but then in the hosts_vars directory you supply the correct user / password for Host 3.

https://www.youtube.com/watch?v=lYzbqaFB2ZM

Host_vars normally would not be used as much but if you have one off objects you can use this directory structure.

Ansible – The Basics
Playbooks

Ansible Playbooks offer a repeatable, re-usable, simple configuration management and multi-machine deployment system, one that is well suited to deploying complex applications. If you need to execute a task with Ansible more than once, write a playbook and put it under source control. Then you can use the playbook to push out new configuration or confirm the configuration of remote systems.

- Playbooks are expressed in YAML format with a minimum of syntax.
- A playbook is composed of one or more 'plays' in an ordered list. The terms 'playbook' and 'play' are sports analogies. Each play executes part of the overall goal of the playbook, running one or more tasks. Each task calls an Ansible module.
- A playbook runs in order from top to bottom. Within each play, tasks also run in order from top to bottom. Playbooks with multiple 'plays' can orchestrate multi-machine deployments, running one play on your webservers, then another play on your database servers, then a third play on your network infrastructure, and so on. At a minimum, each play defines two things:
  - the managed nodes to target, using a pattern
  - at least one task to execute

https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html
https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html#yaml-syntax
https://github.com/ansible/ansible-examples
https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html#using-patterns

Patterns – Calling out who the target is.

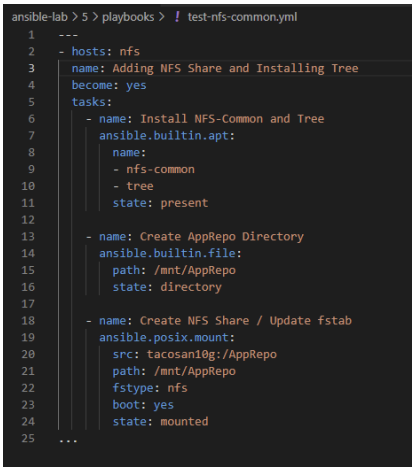
Example
Hosts: leafs
Hosts: nfs

https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html#using-patterns

Ansible – The Basics

Playbooks Example - Static

- **hosts:** nfs -Using the hosts file, it will execute on all hosts under the nfs group
- **name:** - When the playbook starts it will print this text out
- **become:** - Become root for the tasks (sudo <command>)
- **tasks:** - This is the start of the playbook
  - **name:** - This is the name of the task that is being executed
  - **ansible.builtin.apt:** - The module that's being use for this task.
    - **name:** - In THIS case, name is equal to the "apt" you'd like to install (sudo apt install nfs-common tree)- name is a call under the ansible.builtin.apt module
    - **state:** - present = create or absent = destroy/remove



This playbook is static – as the variables are called within the playbook, if you wanted to reuse this playbook over and over again for different deployments you'd have to change multiple lines

There are 3 tasks in this playbook example

ansible.builtin.apt (ansible core module)
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html#ansible-collections-ansible-builtin-apt-module

ansible.builtin.file (ansible core module)
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html#ansible-collections-ansible-builtin-file-module

ansible.posix.mount (ansible galaxy module)
https://docs.ansible.com/ansible/latest/collections/ansible/posix/mount_module.html

Ansible – The Basics

Playbooks Example – Little more dynamic

ansible-lab > 5 > group_vars > nfs > ! nfs.yml

```yaml
# apt packages to install
packages:
  - nfs-common
  - tree
# NFS Directory to create
nfs_path: '/mnt/AppRepo'
# NFS source
nfs_src: 'tacosan10g:/AppRepo'
...
```

ansible-lab > 5 > playbooks > ! test-nfs-common.yml

```yaml
---
#Playbook will use varibles from group_vars/nfs directory (nfs.yml)
- hosts: nfs
  name: Adding NFS Share and Installing Tree
  become: yes
  tasks:
    - name: Install NFS-Common and Tree
      ansible.builtin.apt:
        name: "{{ packages }}"
        state: present

    - name: Create AppRepo Directory
      ansible.builtin.file:
        path: "{{ nfs_path }}"
        state: directory

    - name: Create NFS Share / Update fstab
      ansible.posix.mount:
        src: "{{ nfs_src }}"
        path: "{{ nfs_path }}"
        fstype: nfs
        boot: yes
        state: mounted
...
```

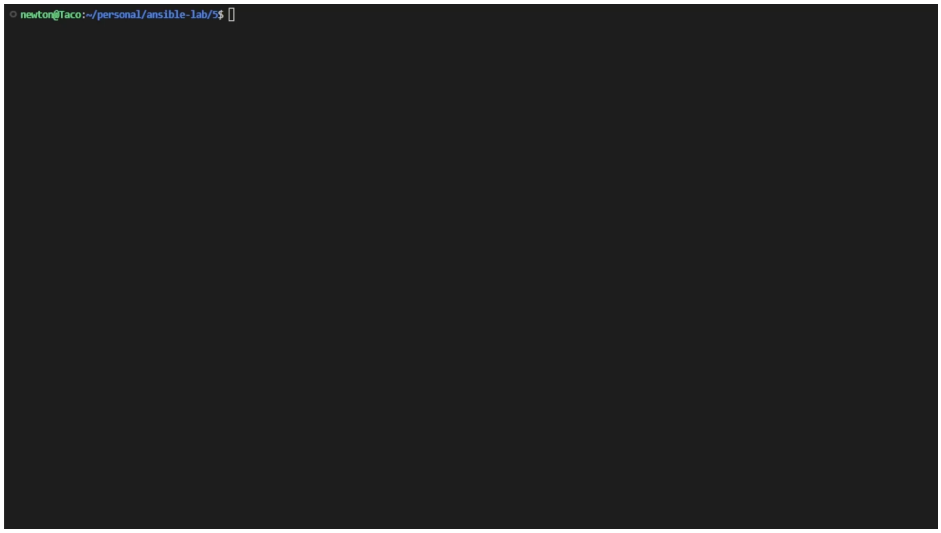

Same playbook – yet a little more dynamic.

Using Jinja2 templating

Using roles would make this playbook even more dynamic

Ansible – The Basics

Running Playbooks



ansible-playbook <playbook>

with Ubuntu 22.04 you might get an error like this with a playbook run.
"module_stderr": "/usr/bin/env: ‘python’: No such file or directory\n", "module_stdout": "", "msg": "MODULE FAILURE\nSee stdout/stderr for the exact error", "rc": 127}

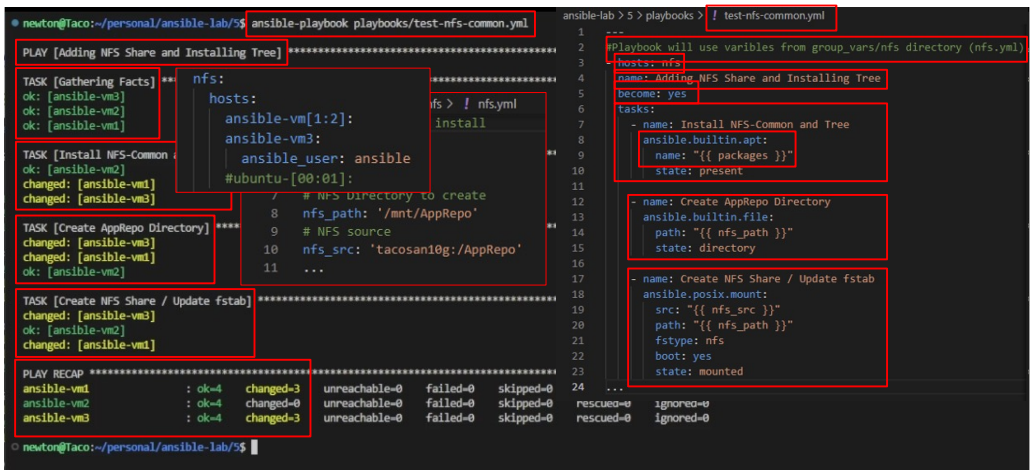
can do 1 of 2 things.
1) sudo ln --symbolic /usr/bin/python3 /usr/bin/python
2) sudo apt install python-is-python3

both should fix the problem

Green = Completed No Changes
Yellow = Changes Made
Red = Failure

Ansible – The Basics

Running Playbooks – The Break Down



Ansible will follow the supplied playbook – test-nfs-common.yml

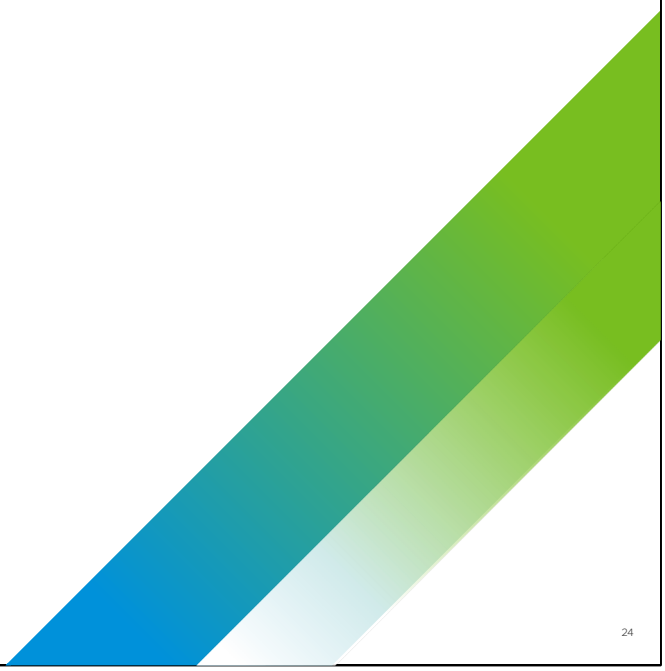
1) While name is not required – it helps to let you know WHICH playbook is being executed
2) gather_facts is not set to False (default true) so since it's not defined it will gather facts on the defined hosts. – nfs group. Hosts are pulled from our "hosts file" which is referenced in the ansible.cfg file. (Inventory=hosts-encrypt.yml) – Green = No changes made (normal for a gather_facts task)
3) First task is executed – Once again using name adds information to as to which task is running. All tasks will be run as root user because "become" is set to yes/true
4) This task is defined as an ansible.builtin.apt – This playbook is using the jinja2 syntax to pull a variable from the group_vars/nfs/nfs.yml file. State = present which means create / install
    1) Ansible-vm2 = green - no changes
    2) Ansible-vm1 and 3 = yellow – meaning that 1 or more changes needed to be made. nfs-common and/or tree needed to be installed
5) 2nd task is running to create a directory, once again jinja2 syntax is calling for the nfs_path variable from the nfs.yml file.
    1) Ansible-vm2 = green = no changes made

    2) Ansible-vm1 and 3 = yellow – changes made again, since this is a single task we know it was missing the /mnt/AppRepo directory
1) 3rd task running to update the /etc/fstab file so that this NFS share would connect on boot. In this case boot: yes IS the default but wanted to make sure it was understood.
2) Play RECAP – shows the final state of the playbook.
    1) Ok=4 says all four task ran without a problem. (Remember gather_facts = 1 task + 3 other defined tasks)
    2) Changed = 3 for Ansible-vm1/3 and nothing changed for Ansible-vm2
    3) Everything else is showing no errors, failed jobs or skipped jobs etc.

Green = Completed No Changes
Yellow = Changes Made
Red = Failure

https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#ansible-playbook

Ansible Ad Hoc



Ansible Ad Hoc

Doing something one at a time

An Ansible ad hoc command is used to automate a single task on one or more managed nodes. Ad hoc commands are quick and easy, but they are not reusable.

- Ad hoc command rely on inventory files
- The command module is the default module for ansible command
  - ansible.builtin.command module
  - does not support extended shell syntax like piping and redirects
- Other modules can be called using the –m <module> command
- Module arguments can be supplied with the -a <arg> command
  - Module arguments are used to filter data or provide how you'd like a command to look

Example:

ansible control -m setup -a 'filter=ansible_distribution*'

https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html#command-module

Example
ansible control -m setup -a 'filter=ansible_distribution
filter=ansible_distribution_release'

Ansible - command
control - which inventory hosts to run command on (control group)
-m setup – use the setup module
-a 'filter=ansible_distribution*' = filter the data and only provide information for fields that start with ansible_distribution

```
taco | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/os-release",
```

```
        "ansible_distribution_file_variety": "Debian",
        "ansible_distribution_major_version": "22",
        "ansible_distribution_release": "jammy",
        "ansible_distribution_version": "22.04",
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false
}
```


ansible control –m setup > <filename> - to see all information about system called

Test connectivity
ansible all -m ping -o

Ansible-Vault

Ansible-Vault
Encrypting Content

Ansible Vault encrypts variables and files so you can protect sensitive content such as passwords or keys rather than leaving it visible as plaintext in playbooks or roles. To use Ansible Vault you need one or more passwords to encrypt and decrypt content.

- The cipher algorithm used to encrypt the data is AES256
  - This is the only supported cipher algorithm
- The utility called ansible-vault is used for CRUD options
- Ansible-vault can encrypt variables or entire files and YAML playbooks
  - Files remain encrypted after use UNLESS you pass encrypted file as a SRC argument with copy, template, unarchive, script or assemble modules. The file will be NOT be encrypted on the target host
- Vault password can be supplied during Ad Hoc runs, before playbooks or supplied as a file

https://docs.ansible.com/ansible/latest/user_guide/vault.html

--ask-vault-password
--vault-password-file
--vault-id (if used during creation)

CRUD
- Create
- Decrypt
- Edit
- View
- Encrypt
- Encrypt_string
- Rekey (change password)

- [defaults]
- inventory = hosts-encrypt.yml
- host_key_checking = False
- vault_password_file = ~/.vault_pass ← This file is in plain text, has the vault

password needed to handle encrypted playbooks / variables

YAML

YAML
YAML Ain't Markup Language

- YAML is a human-readable data-oriented language
  - Easy to read / write vs XML and JSON
    - YAML Syntax
  - Used by Ansible / Kubernetes / AWS etc
  - Can convert between JSON and YAML
    - JSON not the easiest to read
- Supported by multiple Visual Editors
  - Visual Studio Code (VS Code / VSC)
  - Atom
  - Sublime Text

https://www.json2yaml.com/
https://code.visualstudio.com/
https://atom.io/
https://www.sublimetext.com/

YAML

YAML Syntax

- Every YAML file begins with 3 dashes
  - Indentation is used for objects or possible nested objects
    - Normally 2 spaces from line above
  - Tabs are not allowed
- YAML supports multiple value types
  - String, Floating-Point Numbers, Integers, Boolean, Array
- Using Visual Editor like VSC makes following this EASY
  - Can constantly monitor your input and show when errors are place

```
---
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
  - huey
  - dewey
  - louie
  - fred
xmas-fifth-day:
  calling-birds: four
  french-hens: 3
  golden-rings: 5
  partridges:
    count: 1
    location: "a pear tree"
  turtle-doves: two
```


https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
https://www.commonwl.org/user_guide/topics/yaml-guide.html
https://learnxinyminutes.com/docs/yaml/

*Picture / Text from – Great explaining YAML
https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started

The file starts with three dashes. These dashes indicate the start of a new YAML document. YAML supports multiple documents, and compliant parsers will recognize each set of dashes as the beginning of a new one. Next, we see the construct that makes up most of a typical YAML document: a key-value pair.
Doe is a key that points to a string value: **a deer, a female deer.** YAML supports more than just string values. The file starts with six key-value pairs. They have four different data types. **Doe** and **ray** are strings. **Pi** is a floating-point number. **Xmas** is a boolean. **French-hens** is an integer. You can enclose strings in single or double-quotes or no quotes at all. YAML recognizes unquoted numerals as integers or floating point. The seventh item is an array. **Calling-birds** has four elements, each denoted by an opening dash. I indented the elements in **calling-birds** with two spaces.
Indentation is how YAML denotes nesting. The number of spaces can vary from file to

file, but tabs are not allowed.
Finally, we see **xmas-fifth-day**, which has five more elements inside it, each of them indented. We can view **xmas-fifth-day** as a dictionary that contains two string, two integers, and another dictionary.
YAML supports nesting of key-values, and mixing types.

Note
Values may be wrapped in quotation marks but be aware that this may change the way that they are interpreted i.e. "1234" will be treated as a character string , while 1234 will be treated as an integer.

https://jinja.palletsprojects.com/en/3.1.x/intro/
https://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html

Templates

Templating with Jinja2

Jinja2 is a powerful and easy to use python-based templating engine

- Template files are simple text files that store variables that can change when playbooks are executed
- These variable get replaced by actual values defined during the playbook execution
- Using the following tags in a Jinja2 template file
  - {{ }} : Used for embedding variables which prints their actual value during code execution.
  - {% %} : Used for control statements such as loops and if-else statements.
  - {# #} : To specify the comments.
- Template files end with the .j2 extension

Great pages about Templates
https://www.linuxtechi.com/configure-use-ansible-jinja2-templates/

https://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#handling-undefined-variables

https://blog.learncodeonline.in/everything-about-ansible-jinja2-templating

Templates

Templating with Jinja2

Variables can be defined before being called or defined in another file.

Calling the variable in a task happens inside the {{ }} brackets and normally “ “

```
---
- hosts: all
  become: true
  gather_facts: false
  vars:
    j2_var: jinja2 variable

  tasks:
    - name: A Jinja2 example
      debug:
        msg: "A example of {{ j2_var }}"
```

```
bzala@linuxways:~/jinja2_variable$ ansible-playbook jinja2_var_example.yml

PLAY [all] *********************************************************************************************************

TASK [A Jinja2 example] ********************************************************************************************
ok: [server1] => {
    "msg": "A example of jinja2 variable"
}

PLAY RECAP *********************************************************************************************************
server1                    : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```


Here we have a YAML file with the j2_var being defined before the task.

During the task we call this variable within the {{ }} brackets

Sourced from:
https://linuxways.net/centos/how-to-use-the-jinja2-template-in-ansible/

Templates

Templating with Jinja2

Templates end in with the .j2 extension

```
server {
        listen {{ http_port }};
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name {{ http_host }};

        location / {
                try_files $uri $uri/ =404;
        }
}
```

```
---
- hosts: all
  become: true
  gather_facts: false
  vars:
    http_port: '80'
    http_host: '192.168.122.210'

  tasks:
    - name: A Jinja2 accessing the variable example
      template:
        src: nginx.conf.j2
        dest: "/etc/nginx/sites-enabled/nginx.conf"
      notify:
        - restart nginx

  handlers:
    - name: restart nginx
      service:
        name: nginx
        state: restarted
```

Defining the variables of **http_port** and **http_host**

Sourcing the template file during the task.

Outcome shows the port and server name populated

```
root@linuxways:/etc/nginx/sites-enabled# cat nginx.conf
server {
        listen 80;
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name 192.168.122.210;

        location / {
                try_files $uri $uri/ =404;
        }

}
```


Great pages about Templates
https://www.linuxtechi.com/configure-use-ansible-jinja2-templates/

https://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html

https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html#handling-undefined-variables

Sourced from:
https://linuxways.net/centos/how-to-use-the-jinja2-template-in-ansible/

Ansible Installation

Ansible Installation
Requirements

- Linux Server/VM for your Ansible Control Host
  - This is where you run commands and Playbooks
- Pip3
- Python3 >=3.6 (3.8+ for Ansible 2.11.0 or newer)
- Git: Used for cloning of code to deploy NSX
- Openssh-Client: Used for git
- OVFTool: Used for ovf deployment; must be 4.4 or newer
- PyVmOmi: Python library for vCenter api
- Ansible 2.9.0 or newer for python3

* As of March 2022 / Notes have updates as of October 2022

Ansible Installation

Installation

There are several ways to install Ansible on Linux, but we will use pip to ensure a clean install and Ubuntu 20.04 LTS (Ubuntu Server 20.04.4 LTS) also works for 22.04 LTS (Ubuntu Server 22.04.1 TLS)

What is Pip3?

- It's the standard package manager for Python3
- We will use it to install the latest version of Ansible on our Linux Host

Before we install pip3, we will uninstall Ansible if it's present on our Ansible Control Host and then use pip3 for a clean install of Ansible

Works for 22.04 – Tested Oct 2022

Ansible Installation

Update your system's packages

Log into your Ubuntu Host and update your system's packages index list

- Type the command: **sudo apt update**

Install updates for all packages on your system

- Type the command: **sudo apt upgrade**
- When Ubuntu asks you "Do you want to continue?" Type '**Y**'

Or use one line to do it all

- Type the command: **sudo apt update && sudo apt upgrade -y**

Ansible Installation

Uninstall ansible if part of base system install

Check if python3 and ansible are installed on your system and installed version

Run the commands

apt list –a python3

python3 --version

apt list –a ansible

Here we see python3 is installed with version 3.8.10 and ansible is NOT installed through apt

```
ansible@ansible-lab:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.4 LTS
Release:        20.04
Codename:       focal
ansible@ansible-lab:~$ apt list -a python3
Listing... Done
python3/focal,now 3.8.2-0ubuntu2 amd64 [installed,automatic]

ansible@ansible-lab:~$ python3 --version
Python 3.8.10
ansible@ansible-lab:~$ apt list -a ansible
Listing... Done
ansible/focal 2.9.6+dfsg-1 all
```


https://docs.ansible.com/ansible-core/devel/installation_guide/intro_installation.html

Ansible Installation

Install pip3

Python3 as we verified using apt list should be part of the Base System Install in Ubuntu 20.04

Ansible is not part of the Base System install

Ansible 2.5 and above works with python3

The easiest way to run ansible under python3 is to install it with pip3

Install pip3

- Run the command: **sudo apt install python3-pip**
- Validate installation: run the command: **pip3 --version**

```
ansible@ansible-lab:~$ pip3 --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
```


We're using PIP since we have to install pyvmomi and pyvim packages due to requirements for NSX

Ansible Installation

Install Ansible via pip3

Use pip3 to install ansible

Run the command: **sudo pip3 install ansible**

```
ansible@ansible-vm3:~$ sudo pip3 install ansible
Collecting ansible
  Downloading ansible-5.4.0.tar.gz (38.3 MB)
     |████████████████████████████████| 38.3 MB 10.0 MB/s
Collecting ansible-core~=2.12.2
  Downloading ansible-core-2.12.3.tar.gz (7.8 MB)
     |████████████████████████████████| 7.8 MB 21.5 MB/s
Requirement already satisfied: PyYAML in /usr/lib/python3/dist-packages (from ansible-core~=2.12.2->ansible) (5.3.1)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible-core~=2.12.2->ansible) (2.8)
Requirement already satisfied: jinja2 in /usr/lib/python3/dist-packages (from ansible-core~=2.12.2->ansible) (2.10.1)
Collecting packaging
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB 19.3 MB/s
Collecting resolvelib<0.6.0,>=0.5.3
  Downloading resolvelib-0.5.4-py2.py3-none-any.whl (12 kB)
Collecting pyparsing!=3.0.5,>=2.0.2
  Downloading pyparsing-3.0.7-py3-none-any.whl (98 kB)
     |████████████████████████████████| 98 kB 24.0 MB/s
Building wheels for collected packages: ansible, ansible-core
  Building wheel for ansible (setup.py) ... done
  Created wheel for ansible: filename=ansible-5.4.0-py3-none-any.whl size=63646986 sha256=65019b834204c18e460e8f3ad1c700a
  Stored in directory: /root/.cache/pip/wheels/24/c7/72/a881743995f29b94c46a417f2050481502af9b9a8785722ae0
  Building wheel for ansible-core (setup.py) ... done
  Created wheel for ansible-core: filename=ansible_core-2.12.3-py3-none-any.whl size=2076487 sha256=f92b29e4fa1375a8be9db
  Stored in directory: /root/.cache/pip/wheels/25/2d/a5/9ca95ba857e85692792dd14a19a81fb53f3f79cd4e49e457c1
Successfully built ansible ansible-core
Installing collected packages: pyparsing, packaging, resolvelib, ansible-core, ansible
Successfully installed ansible-5.4.0 ansible-core-2.12.3 packaging-21.3 pyparsing-3.0.7 resolvelib-0.5.4
```


To upgrade after install
pip3 install --upgrade ansible
Or
pip3 install –U ansible

Ansible Installation
Verify Ansible Installation

Verify installation

Run the command: **pip3 list | grep ansible**

```
ansible@ansible-lab:~$ pip3 list | grep ansible
ansible                 5.5.0
ansible-core            2.12.4
```

Ansible is installed and we can verify it will use python3, 3.8.10, from the python version value.

```
ansible@ansible-lab:~$ ansible --version
ansible [core 2.12.4]
  config file = None
  configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.8/dist-packages/ansible
  ansible collection location = /home/ansible/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.8.10 (default, Mar 15 2022, 12:22:08) [GCC 9.4.0]
  jinja version = 2.10.1
  libyaml = True
```

Also note we do NOT currently have a ansible.cfg file (config file = None)

This file will NOT be created when installing by pip3, however if you use apt or yum etc it will normally create this default cfg file in the /etc/ansible directory

Update 10/8/2022 Ubuntu 22.04.1
pip3 list | grep ansible

ansible	6.4.0
ansible-core	2.13.4

Ansible Installation

Install Ansible on specific operating systems

Ansible can be installed on multiple different operating systems, thankfully Ansible has documentation for Fedora / CentOS, Ubuntu, Debian and Windows

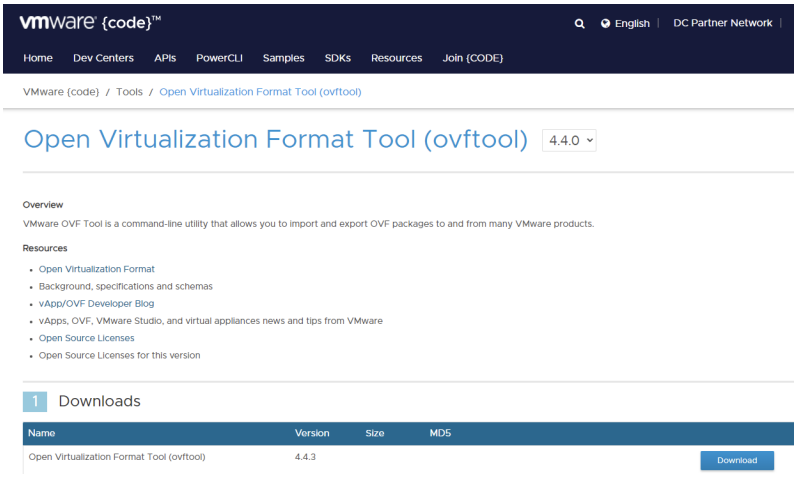
https://docs.ansible.com/ansible-core/devel/installation_guide/installation_distros.html

https://docs.ansible.com/ansible-core/devel/installation_guide/installation_distros.html

Ansible Installation

Requirements for NSX: Download OVFTool(Open Virtualization Format Tool)

https://code.vmware.com/web/tool/4.4.0/ovf



https://code.vmware.com/web/tool/4.4.0/ovf
Using 4.4.3 for these slides – anything 4.4+ should work.

Ansible Installation

Requirements for NSX: OVFTool

Move to the directory where you have the ovftool bundle

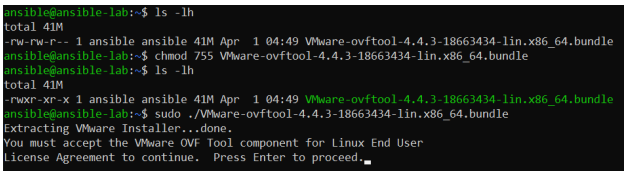
Set the permissions to 755

chmod 755 VMware-ovftool-4.4.3-18663434-lin.x86_64.bundle

Run the Installer

sudo ./VMware-ovftool-4.4.3-18663434-lin.x86_64.bundle

Press Enter to accept the EULA and begin the installation of OVFTool



Ansible Installation

NSX Requirements: packages pyvmomi pyvim

We will use pip3 to install the remaining package requirements

- Pyvmomi
- Pyvim
- Ansible-lint

Run the command:

sudo pip3 install pyvmomi pyvim ansible-lint

```
ansible@ansible-lab:~$ pip3 list | grep -E 'ansible|pyvmomi|pyvim'
ansible                 6.4.0
ansible-compat          2.2.1
ansible-core            2.13.4
ansible-lint            6.8.0
pyvim                   3.0.3
pyvmomi                 7.0.3
```


ansible-lint will check your playbooks for possible errors
https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#ansible-lint

Update 10/8/2022 Ubuntu 22.04.1
pip3 list | grep -E 'ansible|pyvmomi|pyvim'

ansible	6.4.0
ansible-compat	2.2.1
ansible-core	2.13.4
ansible-lint	6.8.0
pyvim	3.0.3
pyvmomi	7.0.3

NSX Ansible Modules

NSX Ansible Modules

Information: Github – ansible-for-nsxt

Official VMware Github repository for NSX is located at:

- https://github.com/vmware/ansible-for-nsxt

Github branch information – Can be used to grab Ansible Galaxy Collections

- Master: Latest code, under development
- V3.2.0: NSX 3.2.x and below
- V3.1.0: NSX 3.1.x and below
- V3.0.0: NSX 3.0.x and below
- V1.1.0: NSX 2.4, NSX 2.5
- V1.0.0: NSX 2.3

*FYI: only the Numbered Versions, not the Master branch, have official GSS Support

https://github.com/vmware/ansible-for-nsxt

NSX Ansible Modules

Installation: Ansible Galaxy Collection – ansible-for-nsxt

With NSX 3.2 VMware NSX Ansible Modules are now distributed as an Ansible Galaxy collection and completely supported by VMware on Github.

To install the 'ansible-for-nsxt' collection run this command

ansible-galaxy collection install git+https://github.com/vmware/ansible-for-nsxt.git,master

```
ansible@ansible-lab:~$ ansible-galaxy collection install git+https://github.com/vmware/ansible-for-nsxt.git,master
Cloning into '/home/ansible/.ansible/tmp/ansible-local-18683zjv16cky/tmpothi1s_r/ansible-for-nsxt3id197if'...
remote: Enumerating objects: 2863, done.
remote: Counting objects: 100% (504/504), done.
remote: Compressing objects: 100% (282/282), done.
remote: Total 2863 (delta 328), reused 345 (delta 218), pack-reused 2359
Receiving objects: 100% (2863/2863), 1.15 MiB | 3.81 MiB/s, done.
Resolving deltas: 100% (2016/2016), done.
Already on 'master'
Your branch is up to date with 'origin/master'.
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Installing 'vmware.ansible_for_nsxt:1.0.0' to '/home/ansible/.ansible/collections/ansible_collections/vmware/ansible_for_nsxt'
Created collection for vmware.ansible_for_nsxt:1.0.0 at /home/ansible/.ansible/collections/ansible_collections/vmware/ansible_for_nsxt
vmware.ansible_for_nsxt:1.0.0 was installed successfully
```

Note We're going to use the 'master' branch, there's currently a pyvim bug in the v3.2 branch (3/29/22)

Master branch
ansible-galaxy collection install git+https://github.com/vmware/ansible-for-nsxt.git,master
V3.2 branch
ansible-galaxy collection install git+https://github.com/vmware/ansible-for-nsxt.git,v3.2.0

To Upgrade / update a collection use the –-upgrade or –U after install.
ansible-galaxy collection install --upgrade git+https://github.com/vmware/ansible-for-nsxt.git,master

Corrected commit – Commits to V3.2 branch after March 29th 2022 SHOULD have this fix in it.
https://github.com/vmware/ansible-for-nsxt/commit/46dde5b11758732d09b1461dc0aee116f45a105f

https://docs.ansible.com/ansible/latest/cli/ansible-galaxy.html

Ansible-galaxy command provided by vmware github docs
https://github.com/vmware/ansible-for-nsxt

NSX Ansible Modules

Installation: Ansible Galaxy Collection – ansible-for-nsxt

Ansible Galaxy Collections will install into a hidden directory of the user that installed them
~/.ansible/collections/ansible_collections/vmware/ansible-for-nsxt

With the modules needed under the plugins directory

```
ansible@ansible-lab:~$ ls -lh ~/.ansible/collections/ansible_collections/vmware/ansible_for_nsxt/
total 128K
-rw-r--r-- 1 ansible ansible 2.5K Apr  1 05:34 CONTRIBUTING.md
-rw-r--r-- 1 ansible ansible  49K Apr  1 05:34 FILES.json
-rw-r--r-- 1 ansible ansible  37K Apr  1 05:34 LICENSE.txt
-rw-r--r-- 1 ansible ansible  987 Apr  1 05:34 MANIFEST.json
drwxr-xr-x 2 ansible ansible 4.0K Apr  1 05:34 meta
drwxr-xr-x 5 ansible ansible 4.0K Apr  1 05:34 plugins
-rw-r--r-- 1 ansible ansible  13K Apr  1 05:34 README.md
drwxr-xr-x 4 ansible ansible 4.0K Apr  1 05:34 tests
ansible@ansible-lab:~$
ansible@ansible-lab:~$ ls -lh ~/.ansible/collections/ansible_collections/vmware/ansible_for_nsxt/plugins/
total 12K
drwxr-xr-x 2 ansible ansible 4.0K Apr  1 05:34 doc_fragments
drwxr-xr-x 2 ansible ansible 4.0K Apr  1 05:34 modules
drwxr-xr-x 3 ansible ansible 4.0K Apr  1 05:34 module_utils
```


Note that under the plugins directory is the modules and module_utils directorys

Git / GitHub / SSH Keys

Git / GitHub / SSH Keys

You spelled “Get” wrong

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is a Source Code Management (SCM) that allows branching and merging of code.

Example of branches: Main, V3.2.0, V3.1.0 etc.

Git allows you clone / pull / push of software repositories from sites such as

- GitHub
- GitLab
- Bitbucket

Git is installed by default on Ubuntu, we will configure git so that we can use it to push / pull / clone operations

We’re also going to use GitHub to create and clone a repository

Github.com

Git / GitHub / SSH Keys

GitHub

Creating a GitHub or GitLab account will allow you to track changes that you make from a clone repository, while this is NOT mandatory, it's highly recommended.

Reasons for using an online repository:

- Creates a backup of any work done
- Allows you to move backward / forward in the life of the code
- Becomes highly portable, you can work from multiple computers, yet be on the "same page"
- Allows the ability to share / work with others in the devolvement of the code
- Free.99 in our case and easy to sign up

Github.com
Gitlab.com

Git / GitHub / SSH Keys
Generating SSH Key

Using the SSH protocol, you can connect and authenticate to remote servers and services. With SSH keys, you can connect to GitHub without supplying your username and personal access token at each visit.

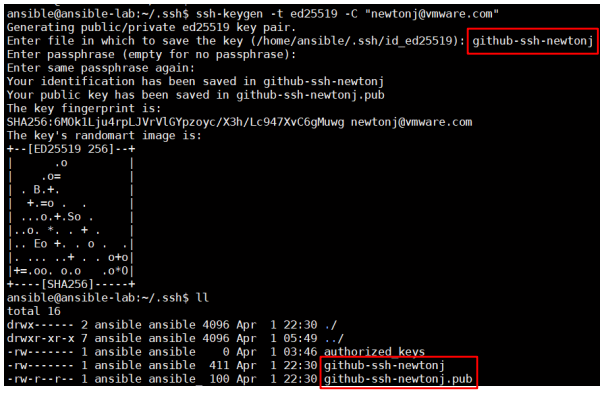
This will allow you to easily perform git CRUD operations

To generate a new SSH Key run the following command in the *~/.ssh* directory

ssh-keygen –t ed25519 –C "<sign-up email>"

Standard Warning
"do NOT lose or share private key file", we'll use the .pub file later on

Windows 10 in notes



Linux / MAC
https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
ssh-keygen –t ed255519 –C "<email used to sign up>"
Older systems might need
ssh-keygen –t rsa –b 4096 –C "<email used to sign up>"

For Windows 10 -
https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_keymanagement

Git / GitHub / SSH Keys

SSH Key Configuration

Now that you have a private key and public key we need to configure the system to use these keys when requested.

First we need to create a config file and give RW access

install –m 600 /dev/null config

Next edit the file and add the following

host github.com
HostName github.com
User git
IdentityFile ~/.ssh/<ssh-key-created>

SSH Keys must be owner RW ONLY (600) anything higher will generate an error

See Notes for Multiple Accounts

```
ansible@ansible-lab:~/.ssh$ install -m 600 /dev/null config
ansible@ansible-lab:~/.ssh$ vim config
ansible@ansible-lab:~/.ssh$ cat config
host github-work
        HostName github.com
        User git
        IdentityFile ~/.ssh/github-ssh-newtonj
ansible@ansible-lab:~/.ssh$ ll
total 28
drwx------ 2 ansible ansible 4096 Oct 17 21:13 ./
drwxr-x--- 9 ansible ansible 4096 Oct 17 21:13 ../
-rw------- 1 ansible ansible  711 Oct  5 02:07 authorized_keys
-rw------- 1 ansible ansible   88 Oct 17 21:13 config
-rw------- 1 ansible ansible  411 Oct  5 02:23 github-ssh-newtonj
-rw------- 1 ansible ansible  100 Oct  5 02:23 github-ssh-newtonj.pub
-rw------- 1 ansible ansible 2102 Oct  5 20:56 known_hosts
ansible@ansible-lab:~/.ssh$
```


55

Note SSH Keys must be RW owner ONLY (chmod 600)
https://chmod-calculator.com/

Linux / MAC / Windows
http://manpages.ubuntu.com/manpages/trusty/man5/ssh_config.5.html

If using multiple accounts you'll need to set them up somewhat like this. *Note for 2nd account you should drop the .com as that might create a problem. E.G. github-work

```
#Default github account
host github.com
      HostName github.com
      User git
      IdentityFile ~/.ssh/github-ssh-key
#2nd  github account (work)
host github-work
      Hostname github.com
      User git
      IdentityFile ~/.ssh/github-ssh-newtonj-new
```

```
#Gitlab account
host gitlab.com
        HostName gitlab.com
        User git
        IdentityFile ~/.ssh/gitlab-ssh-key
```

Git / GitHub / SSH Keys

GitHub SSH Key Configuration

Next up we're going to copy the contents of the .pub file to GitHub. Login to Github select settings, SSH and GPG Keys

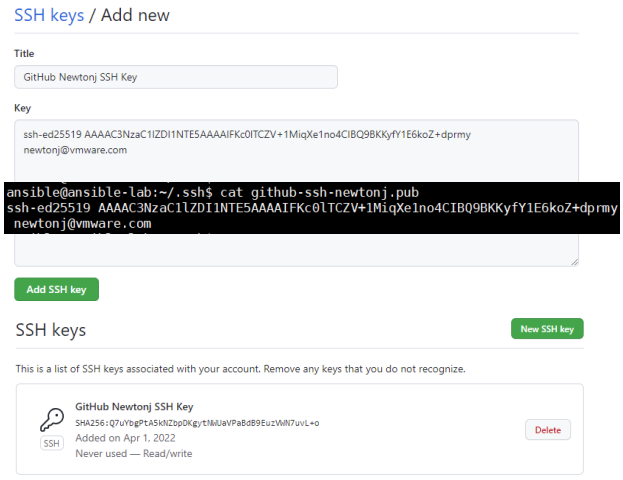
Click on "New SSH Key"

In the "Title" area provide a meaningful name

Under the "Key" area you're going to copy contents of the .pub file you created earlier

Click "Add SSH Key"

Congrats you know have a SSH Key assigned to your account!



SSH and GPG Keys
https://github.com/settings/keys

Git / GitHub / SSH Keys
Test SSH Key Connection

To verify we've configured everything correctly GitHub has provided a quick and easy test

Run the command:

ssh –T git@github.com

A successful connection will return:

"Hi <username>! You've successfully authenticated, but GitHub does not provide shell access."

Note – Since this is your first time connecting, you'll need to allow GitHub to be saved to the known_hosts file

Windows 10 in notes

```
ansible@ansible-lab:~/.ssh$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.113.4)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.113.4' (ECDSA) to the list of know
n hosts.
Hi newtonj-vmware! You've successfully authenticated, but GitHub does not provid
e shell access.
```


https://docs.github.com/en/authentication/connecting-to-github-with-ssh/testing-your-ssh-connection
ssh –T git@github.com

If using 2nd account call the url as the same name as the "host" line
ssh –T git@github-work

Also for GitLab
ssh –T git@gitlab.com

SSH on Window 10

Add SSH Keys to \Users\<User>\.ssh

Extra Windows SSH Commands

(Elevated Rights)

Check to see if ssh-agent is running or DISABLED
<powershell> Get-Service ssh-agent | select *
PS C:\Windows\system32> Get-Service ssh-agent | select *

Name : ssh-agent
RequiredServices : {}
CanPauseAndContinue : False
CanShutdown : False
CanStop : False
DisplayName : OpenSSH Authentication Agent
DependentServices : {}
MachineName : .
ServiceName : ssh-agent
ServicesDependedOn : {}
ServiceHandle : SafeServiceHandle
Status : Stopped
ServiceType : Win32OwnProcess
StartType : Disabled <-- If set to Disabled you will fail to start service (See below)
Site :
Container :

(Elevated Rights)
PS C:\Windows\system32> start-service ssh-agent
start-service : Service 'OpenSSH Authentication Agent

(ssh-agent)' cannot be started due to the following error: Cannot start service ssh-agent on computer '.'.
At line:1 char:1
+ start-service ssh-agent
+ ~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo   : OpenError: (System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId : CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand

Enable service (Elevated Rights)
Get-Service -Name ssh-agent | Set-Service -StartupType Manual

Start Service (Elevated Rights)
Start-Service ssh-agent

***Some of this can be skipped if you've configured your "config" file inside the .ssh/ folder**

Once that is done you can use the ssh-add to add the key to windows 10
ssh-add .ssh/github-ssh

Example:
C:\Users\Newton>ssh-add .ssh/github-ssh
Identity added: .ssh/github-ssh (newtonj@vmware.com)

Confirm it's working
C:\Users\Newton>ssh -T git@github.com
Hi newtonj-vmware! You've successfully authenticated, but GitHub does not provide shell access.

Or first time
C:\Users\Newton>ssh -T git@github.com
The authenticity of host 'github.com (140.82.112.3)' can't be established.
ECDSA key fingerprint is SHA256:p2QAMXNIC1TJYWeIOttrVc98/R1BUFWu3/LiyKgUfQM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,140.82.112.3' (ECDSA) to the list of known hosts.
Hi newtonj-vmware! You've successfully authenticated, but GitHub does not provide shell access.

<Remove from known_host file> - If there's a problem – try again
ssh-keygen -R "hostname"

Example:

ssh-keygen –R github.com

C:\Users\Newton\.ssh> ssh-keygen -R github.com
Host github.com found: line 12
C:\Users\Newton/.ssh/known_hosts updated.
Original contents retained as
C:\Users\Newton/.ssh/known_hosts.old

Remove SSH Key
ssh-add –d <file>

Example:
C:\Users\Newton\.ssh> ssh-add -d .\github-ssh-newtonj
Identity removed: .\github-ssh-newtonj (newtonj@vmware.com)

Git / GitHub / SSH Keys

Git – One Last Thing Before We Go

The last thing we need to do is set the global git identity configuration – This is needed as a "First-Time Git Setup"

The global git username and email address are associated with commits on all repositories on your system that don't have repository-specific values.

To set your global commit name and email address run the git config command with the --global option:

git config --global user.name "Your Name"

git config --global user.email "youremail@yourdomain.com"

Saves values in the ~/.gitconfig file

```
ansible@ansible-lab:~/.ssh$ git config --list
ansible@ansible-lab:~/.ssh$ git config --global user.name "Joshua Newton"
ansible@ansible-lab:~/.ssh$ git config --global user.email "newtonj@vmware.com"
ansible@ansible-lab:~/.ssh$ git config --list
user.name=Joshua Newton
user.email=newtonj@vmware.com
ansible@ansible-lab:~/.ssh$ cat ~/.gitconfig
[user]
        name = Joshua Newton
        email = newtonj@vmware.com
```


All OS's
https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

Git / GitHub / SSH Keys

Git – Useful Commands

git config – Configuration parameters

git init – Creates a new local repository

git clone <path> - Creates a working copy of a local repository

git add – Add one or more files to staging

git commit –m "Commit Message" – Commits changes (staging) to head (The current commit)

git push – Sends committed changes to current branch on remote repository

git status – Show status of changed local files compared to remote repository

git pull – Fetch and merge changes on the remote repository to your local working repository

git remote - Manage the set of repositories ("remotes") whose branches you track.

git branch – Lists all branches in your repo also tells you which branch you're currently in

git checkout – Used to switch between branches or create a new branch

git diff - Show changes between commits, commit and working tree, etc

git merge – Join two or more development histories together

git stash – Temporarily store modified, tracked files

Each of these commands have multiple flags that can be used see notes for multiple helpful sites / cheat sheets etc

Official Docs

https://git-scm.com/docs

Helpful Pages

https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html
https://dzone.com/articles/top-20-git-commands-with-examples
https://www.atlassian.com/git/tutorials/syncing

Great "quick access" page

https://education.github.com/git-cheat-sheet-education.pdf
https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet

GitHub Repository

GitHub Repository

Installation: Picking A Repository

ansible-for-nsxt	ansible-nsx-deployment
Uses Ansible Galaxy Collection	Uses Ansible Galaxy Collection
Generic Playbooks	Detailed / Complex Playbooks
Generic Buildout	Complex Buildout
Notes? Not sure what those are	Detailed Notes throughout
Ramp time – LONG	Ramp time – Short
Developed by vmware	Inspired by ansible-for-nsxt but completely rewrote for quick production / lab deployments – Developed by me
More MEH	SUCH WOW
	*I'm bias

ansible-for-nsxt - vmware official NSX repository – Answerfile and playbooks not built out as much
https://github.com/vmware/ansible-for-nsxt

ansible-nsx-deployment – Online repository built for deployment of NSX. Answerfile and playbooks built
https://github.com/newtonj-vmware/ansible-nsx-deployment

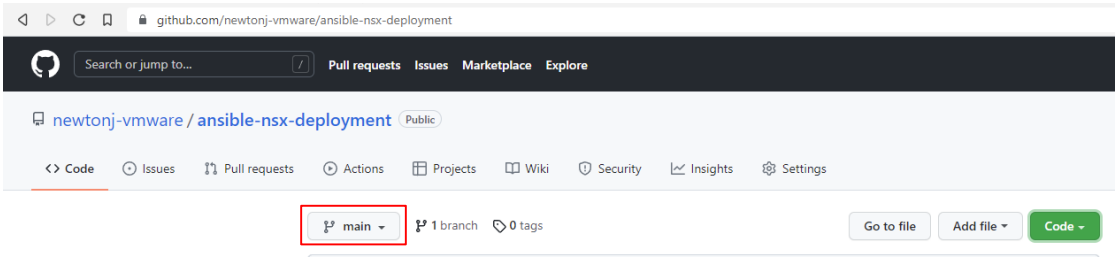
GitHub Repository

Installation: Github – ansible-nsx-deployment

Launch your browser to the GitHub repro URL:

- https://github.com/newtonj-vmware/ansible-nsx-deployment

Make sure 'Main' is selected for the branch in the UI:



Note the Code box, we'll click on this next

Duh of course I'd use mine! :P

GitHub Repository

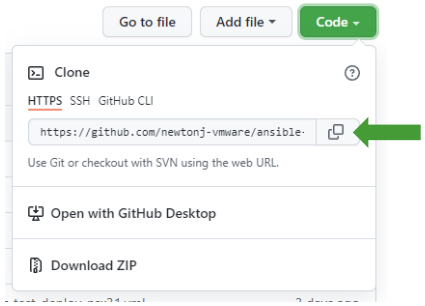
Clone Repository URL: Git Clone

Finally, we're ready to clone a repository – The steps are the same for pretty much any repository you find.

To clone a repository, we just need to know the URL, thankfully GitHub have made this very simple but providing the code box and copy to clipboard.

Click on the "Code" box then make sure "HTTPS" is selected then click on the copy to clipboard icon (green arrow)

https://github.com/newtonj-vmware/ansible-nsx-deployment.git



https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories

Copied text
https://github.com/newtonj-vmware/ansible-nsx-deployment.git

We'll use this copied text to clone this repository to your local system next

GitHub Repository

Clone Repository: Git Clone To Local System

Once you have URL we'll use git clone to copy the online repository to your local system – We're using the HTTPs however you could use the SSH since we've built that out already.

git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git

```
ansible@ansible-lab:~$ ls -lh
total 41M
-rwxr-xr-x 1 ansible ansible 41M Apr  1 04:49 VMware-ovftool-4.4.3-18663434-lin.x86_64.bundle
ansible@ansible-lab:~$ git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git
Cloning into 'ansible-nsx-deployment'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 46 (delta 26), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (46/46), 54.73 KiB | 1.01 MiB/s, done.
ansible@ansible-lab:~$ ls -lh
total 41M
drwxrwxr-x 5 ansible ansible 4.0K Apr  2 04:54 ansible-nsx-deployment
-rwxr-xr-x 1 ansible ansible  41M Apr  1 04:49 VMware-ovftool-4.4.3-18663434-lin.x86_64.bundle
```

We can see that the cloning operation created a new directory in our home directory
If cloning your own repository use SSH method

Normally you'll want to do the git clone operation from your home directory, using shared locations like an NFS share can create errors in ansible between windows / linux systems

If cloning your own repository, you should use the SSH method – This will use your SSH Key to clone the repository, thus giving you full access to push/pull etc
e.g. git clone git@github.com:newtonj-vmware/ansible-nsx-deployment.git

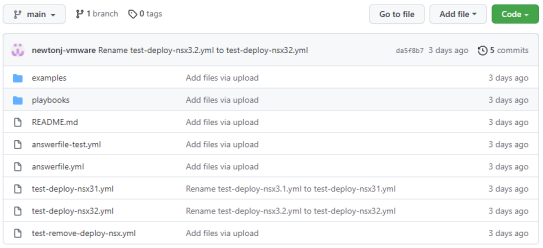
GitHub Repository

Clone Repository: Confirmation Of Local Files

Local files match the same file structure as online

```
ansible@ansible-lab:~$ pwd
/home/ansible
ansible@ansible-lab:~$ cd ansible-nsx-deployment/
ansible@ansible-lab:~/ansible-nsx-deployment$ ls -l
total 56
-rw-rw-r-- 1 ansible ansible 14096 Apr  2 04:40 answerfile-test.yml
-rw-rw-r-- 1 ansible ansible 14096 Apr  2 04:40 answerfile.yml
drwxrwxr-x 2 ansible ansible  4096 Apr  2 04:40 examples
drwxrwxr-x 5 ansible ansible  4096 Apr  2 04:40 playbooks
-rw-rw-r-- 1 ansible ansible  1486 Apr  2 04:40 README.md
-rw-rw-r-- 1 ansible ansible  1092 Apr  2 04:40 test-deploy-nsx31.yml
-rw-rw-r-- 1 ansible ansible  1159 Apr  2 04:40 test-deploy-nsx32.yml
-rw-rw-r-- 1 ansible ansible  1082 Apr  2 04:40 test-remove-deploy-nsx.yml
```

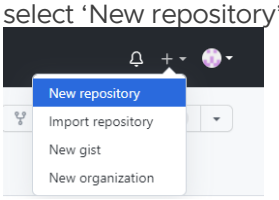
Online files match the same file structure as local files



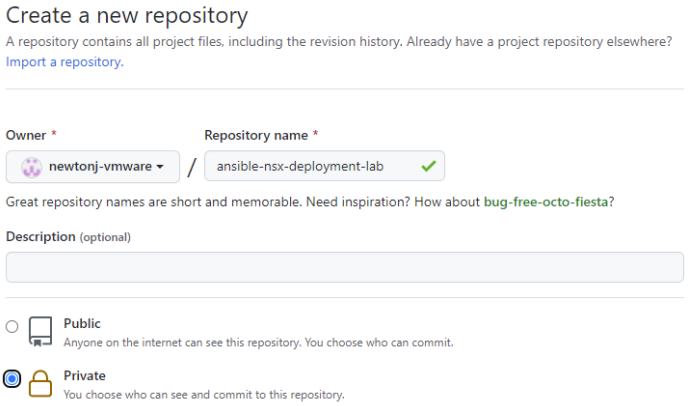
GitHub Repository

Create Your Repository: GitHub

Head back to GitHub and if you're still logged in click on the + in the top right corner and select 'New repository'



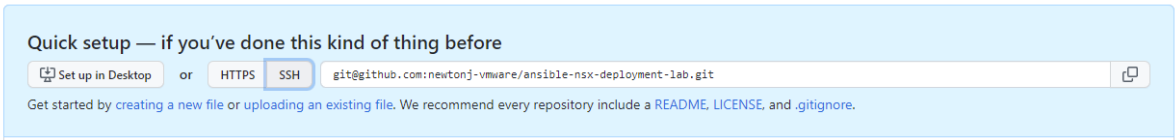
Give your new repository a meaningful name and select who can see this repository – Once you're done click 'Create repository'



GitHub Repository

Modify Original Clone: GitHub

On the next page inside the “Quick setup” you’ll want to click on ‘SSH’ then the ‘copy to



In the cloned repository we need to remove the old metadata that contains the original location and changes. This information is in a hidden directory called ‘.git’

```
ansible@ansible-lab:~/ansible-nsx-deployment$ ll
total 68
drwxrwxr-x 5 ansible ansible  4096 Apr  2 18:53 ./
drwxr-xr-x 8 ansible ansible  4096 Apr  2 18:53 ../
-rw-rw-r-- 1 ansible ansible 14096 Apr  2 18:53 answerfile-test.yml
-rw-rw-r-- 1 ansible ansible 14096 Apr  2 18:53 answerfile.yml
drwxrwxr-x 2 ansible ansible  4096 Apr  2 18:53 examples/
drwxrwxr-x 8 ansible ansible  4096 Apr  2 18:53 .git/
drwxrwxr-x 5 ansible ansible  4096 Apr  2 18:53 playbooks/
-rw-rw-r-- 1 ansible ansible  1486 Apr  2 18:53 README.md
-rw-rw-r-- 1 ansible ansible  1092 Apr  2 18:53 test-deploy-nsx31.yml
-rw-rw-r-- 1 ansible ansible  1159 Apr  2 18:53 test-deploy-nsx32.yml
-rw-rw-r-- 1 ansible ansible  1082 Apr  2 18:53 test-remove-deploy-nsx.yml
```

```
ansible@ansible-lab:~/ansible-nsx-deployment$ pwd
/home/ansible/ansible-nsx-deployment
ansible@ansible-lab:~/ansible-nsx-deployment$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = https://github.com/newtonj-vmware/ansible-nsx-deployment.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "main"]
        remote = origin
        merge = refs/heads/main
```

GitHub Repository

Modify Original Clone: GitHub

First, remove the original git information

rm –rf .git

Next, use 'git init' – This command creates an empty Git repository

git init

Now that the repository is initialized, we need to use the information saved to clipboard

git remote add origin git@github.com:newtonj-vmware/ansible-nsx-deployment-lab.git

'git add .' will add all changes to staging area – Since this is a newly init repository all files will be added

git add .

After files have been added to staging area we need to commit the changes with

```
ansible@ansible-lab:~/ansible-nsx-deployment$ git init
Initialized empty Git repository in /home/ansible/ansible-nsx-deployment/.git/
ansible@ansible-lab:~/ansible-nsx-deployment$ git remote add origin git@github.com:newtonj-vmware/ansible-nsx-deployment-lab.git
ansible@ansible-lab:~/ansible-nsx-deployment$ git add .
ansible@ansible-lab:~/ansible-nsx-deployment$ git commit -m "Initial commit"
[master (root-commit) 7a7a602] Initial commit
 42 files changed, 2150 insertions(+)
 create mode 100644 README.md
```


See Notes for multiple accounts

https://git-scm.com/docs/git-init
https://www.atlassian.com/git/tutorials/saving-changes
https://docs.github.com/en/get-started/importing-your-projects-to-github/importing-source-code-to-github/adding-locally-hosted-code-to-github

Github uses their own API and do not allow git init of new repos

Example: gh repo create

Gitlab WILL allow repo to be created on the fly

Clone and push to new repro (GH)

git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git
cd ansible-nsx-deployment
rm -rf .git
git init <Initialized empty Git repository>

git remote add origin <ssh info> (e.g. git@github.com:newtonj-vmware/ansible-nsx-deployment-lab.git>

Note

If you're using multiple accounts you'll call the URL that matches your "host" line in ~/.ssh/config

e.g. git remote add origin <ssh info> (e.g. git@**github-work**:newtonj-vmware/ansible-nsx-deployment-lab.git>

git add . <adds all the files>
git commit -m "Initial Commit" <give it a comment>
git branch -M main <apply main branch>
git push -u origin main <push to GitHub>

Gitlab – Will automatically create the new repository
git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git
cd ansible-nsx-deployment
rm -rf .git
git init <Initialized empty Git repository>

git remote add origin <ssh info> (e.g. git@gitlab.com:newtonj-

vmware/ansible-nsx-deployment-lab.git>

git add . <adds all the files>
git commit -m "Initial Commit" <give it a comment>
git branch -M main <apply main branch>
git push -u origin main <push to GitLab>

To confirm remote URL information just look at the .git/config file in the root directory of cloned repository

GitHub Repository

Push Modified Clone: GitHub

Next up we set the default branch information to main (or what ever you'd like yours to be)

git branch –M main

Last thing we do is push the originally cloned repository to our newly created repository / branch

```
 create mode 100644 test-deploy-nsx31.yml
 create mode 100644 test-deploy-nsx32.yml
 create mode 100644 test-remove-deploy-nsx.yml
ansible@ansible-lab:~/ansible-nsx-deployment$ git branch -M main
ansible@ansible-lab:~/ansible-nsx-deployment$ git push -u origin main
Warning: Permanently added the ECDSA host key for IP address '140.82.114.3' to the list of known hosts.
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 2 threads
Compressing objects: 100% (37/37), done.
Writing objects: 100% (37/37), 10.20 KiB | 2.55 MiB/s, done.
Total 37 (delta 23), reused 0 (delta 0)
remote: Resolving deltas: 100% (23/23), done.
To github.com:newtonj-vmware/ansible-nsx-deployment-lab.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```


https://docs.github.com/en/get-started/importing-your-projects-to-github/importing-source-code-to-github/adding-locally-hosted-code-to-github

Github uses their own API and do not allow git init of new repos

Example: gh repo create

Gitlab WILL allow repo to be created on the fly

Clone and push to new repro (GH)
git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git
cd ansible-nsx-deployment
git remote remove origin
git remote add origin git@github.com:<user>/<repro-name>.git <tells it where to send it>
git branch -M main <sets to main branch>
git push -u origin main <pushes update>

Gitlab – Will automatically create the new repository
Git remote remove origin

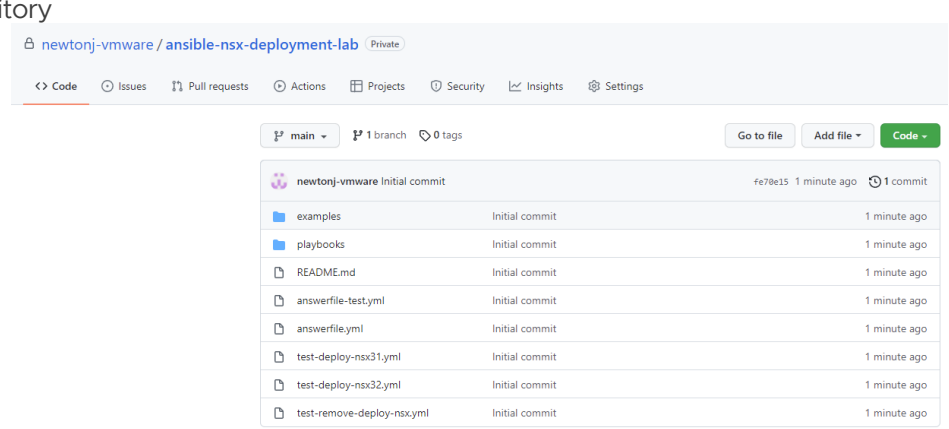
git remote add origin git@:<user>/<repro-name>.git
git branch -M main <sets to main branch>
git push -u origin main <pushes update>

To confirm remote URL information just look at the .git/config file in the root directory of cloned repository

GitHub Repository

Verify GitHub Repository

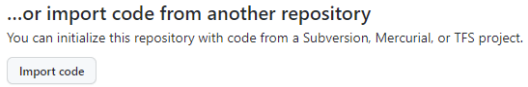
At this point if you go back to your GitHub page you'll see that all the files show up in your repository



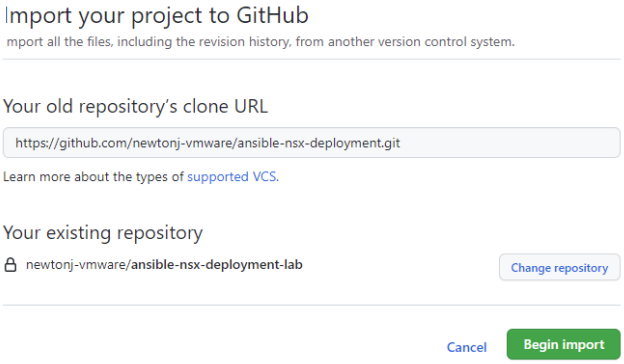
GitHub Repository

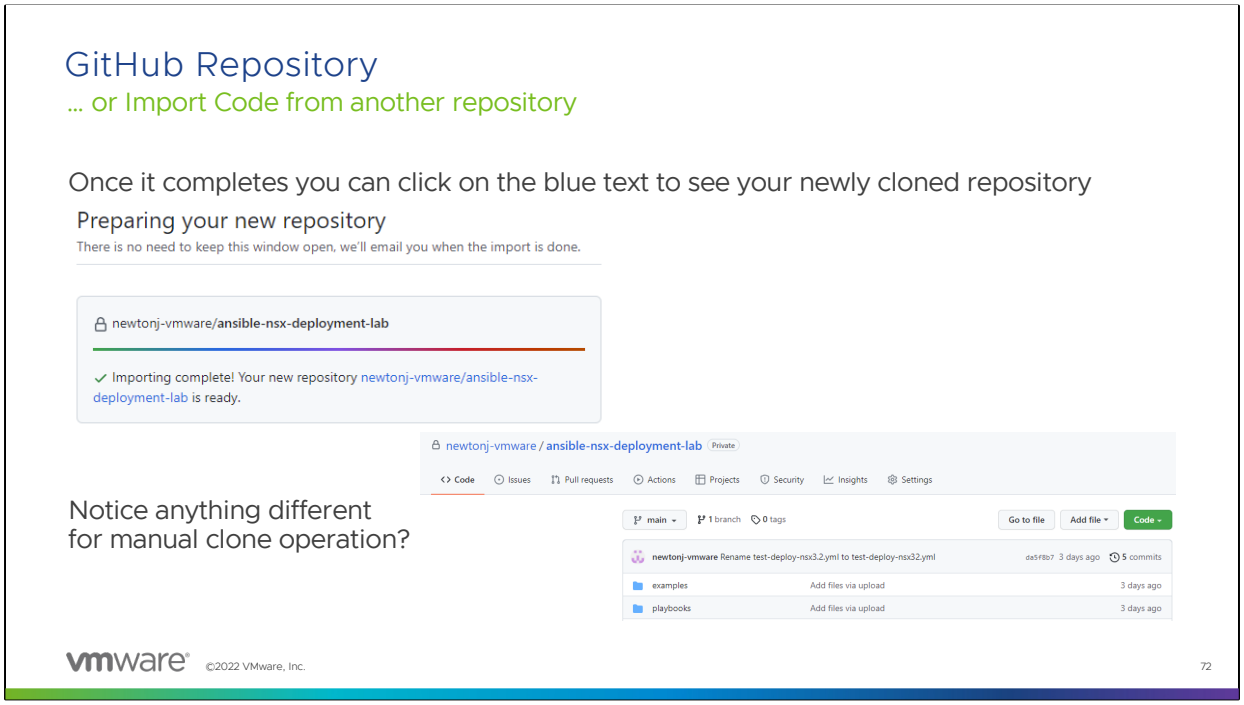
… or Import Code from another repository

Of course, you could just use the “Import code” easy button…



One the next page enter the same URL from our ‘git clone’ command, then click “Begin import”





Using the web based import method keeps all the old commit information

Manual process to create the same information

Clone and push to new repro (GH)

git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git

cd ansible-nsx-deployment

git remote remove origin

git remote add origin git@github.com:<user>/<repro-name>.git <tells it where to send it>

git branch -M main <moves to main branch>

git push -u origin main <pushes update>

Gitlab
Git remote remove origin
git remote add origin git@gitlab.com:<user>/<repro-name>.git
git branch -M main
git push -u origin main

GitHub Repository

But wait… There's more!

If you wanted to you could also JUST completely skip creating your own repository and JUST clone / pull from the original repository.

- You would only have to ensure that you do NOT use any of the file names
- You have zero control of the code, but code could always be up to date
- There is no code tracking or backup for what you write
- Not ideal but very simple

```
ansible@ansible-lab:~$ git clone https://github.com/newtonj-vmware/ansible-nsx-deployment.git
Cloning into 'ansible-nsx-deployment'...
remote: Enumerating objects: 91, done.
remote: Counting objects: 100% (91/91), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 91 (delta 51), reused 45 (delta 22), pack-reused 0
Unpacking objects: 100% (91/91), 76.76 KiB | 1.42 MiB/s, done.
ansible@ansible-lab:~$ cd ansible-nsx-deployment/
ansible@ansible-lab:~/ansible-nsx-deployment$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 687 bytes | 687.00 KiB/s, done.
From https://github.com/newtonj-vmware/ansible-nsx-deployment
   73a073b..ade5973  main       -> origin/main
Updating 73a073b..ade5973
Fast-forward
 answerfile-test.yml | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
```


git clone - that uses the basic HTTPS URL

git pull – will refresh local data with the latest data from repository

We can see that 1 file has changed from last time we pulled data. – answerfile-test.yml

Visual Studio Code (VS Code)

Visual Studio Code (VS Code)

Your new best friend just walked into the room!

Visual Studio Code

Visual Studio Code (VS Code)

The multitool that IS useful

What can VS Code do

- Free.99 open source application on Linux / Windows / macOS
- Text editor
- Supports wide array of programming languages
  - Java, C++, Python, CSS, GO, YAML, Ruby, Powershell
- Allows for remote development
  - SSH, WSL, Containers
- Works with git
- 1000s of extensions available
  - Rest API, Rest Client, Graph, Themes
- A M$ product that IS good

Intro Videos
https://code.visualstudio.com/docs/getstarted/introvideos

Using Git with Visual Studio Code (Official Beginner Tutorial)
https://www.youtube.com/watch?v=i_23KUAEtUM

Git: Commits in Visual Studio Code
https://www.youtube.com/watch?v=E6ADS2k8oNQ

Git: branches in Visual Studio Code
https://www.youtube.com/watch?v=b9LTz6joMf8

Keyboard short cuts
https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf

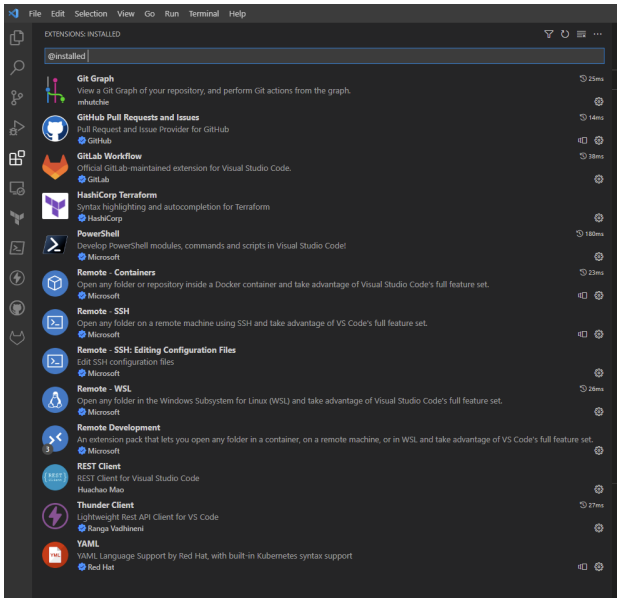
https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf

https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf

Visual Studio Code (VS Code)
Extensions

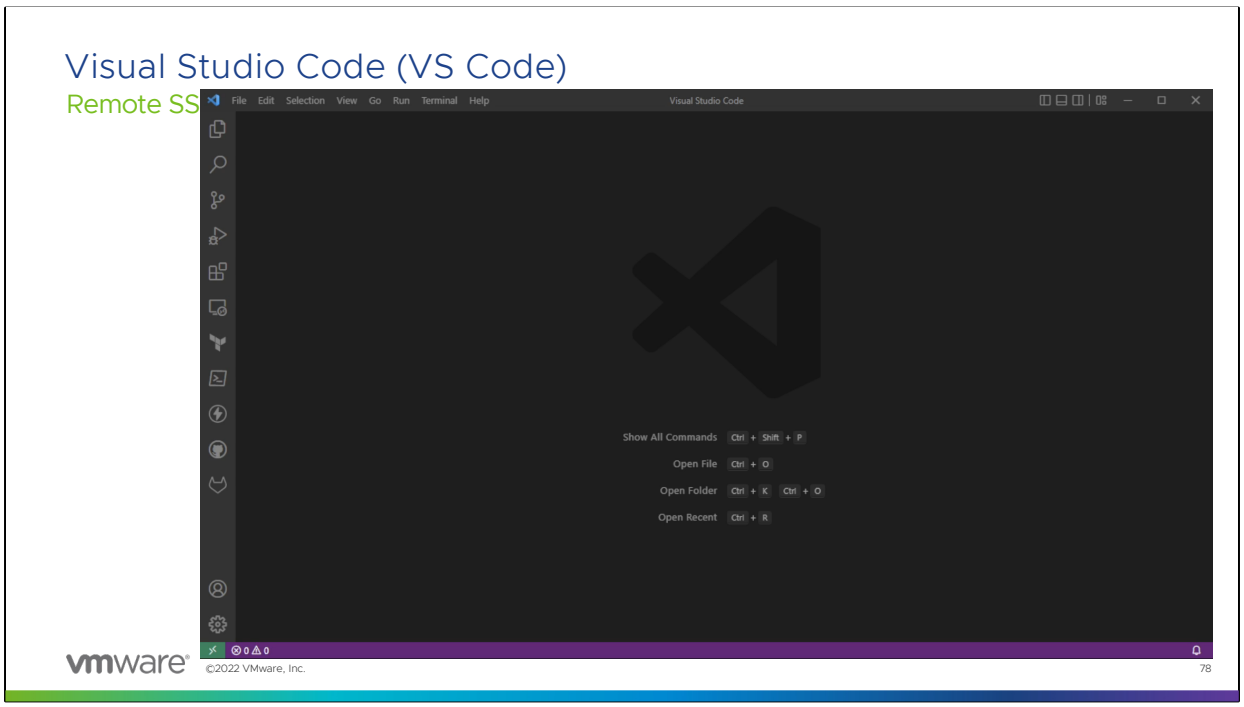
Adding features to VS Code with Extensions

- Ansible – Syntax highlighting / validation
- YAML – Provides YAML Language support to VSC
- Remote – SSH – Allows remote SSH
- Remote – WSL – Allows working with WSL VM (Windows)
- REST Client – Allows inline REST API Calls
- Thunder Client – Think “Postman” but built into VSC
- Git Graph – Visual git tracker
- Themes



77

Recommended Extensions
Remote – SSH – Allows remote SSH
Remote – WSL – Allows working with WSL VM (Windows)
REST Client – Allows inline REST API Calls
Thunder Client – Think “Postman” but built into VSC
YAML - Provides YAML Language support to VSC
Git Graph – Visual git tracker



Using Ctrl+Shift+p to open “Command Palette” – Remote-SSH command
Using Ctrl+, to open Settings – SSH Show Login Terminal

You can add SSH connections to your ~/.ssh/config file (like earlier) also if you add the public or private key to your ~/.ssh directory you will not be prompt for password.

Some errors could be due to known_hosts (in case you redeploy VMs w/ same name)
ssh-keygen –R <hostname>
To remove old hosts from your known_host files.

In video I am using this command between connections to get the thumbprint question again.

https://code.visualstudio.com/docs/remote/ssh#_getting-started

Deployment

Deployment

Time to push button, receive bacon

Everything is based off the answer file, the easiest way to use this file is to copy it to a deployment name answer file then call it inline with your ansible-playbook command using the -e EXTRA_VARS option.

Example – **ansible-playbook –e @production-answerfile.yml deploy-nsx.yml**

You can run a single section of playbook by just calling that playbook and not the full deploy-nsx.yml file.

Example - **ansible-playbook –e @production-answerfile.yml playbooks/16-create-segments.yml**

Modify the deploy-nsx.yml file if you'd like to limit the deployment scope.

Inside the answerfile.yml and answerfile-example.yml file (both the same) there are comment lines for every section and variable plus hints if the section relies on something defined / created before.

Also within the answer file are URLs that call back to the NSX python modules being used for each playbook section, inside these files are all the usable options that can be use within a playbook and answer file.

Adding -v or multiple -vvv / -vvvv will increase the verbosity of the playbook being executed

When using a file for EXTRA_VARS you must add a @ in front of the file name, no space. Multiple EXTRA_VARS can be used, just add -e in front of each.

Example - **ansible-playbook -e @answerfile-lab1.yml -e nsx_license="XXXXX-XXXXX-XXXXX-XXXXX-XXXXX" deploy-nsx.yml - adding a different nsx_license**

Troubleshooting –

-v - Causes Ansible to print more debug messages. Adding multiple -v will increase the verbosity, the builtin plugins currently evaluate up to -vvvvvv. A reasonable level to start is -vvv, connection debugging might require –vvvv

Example - **ansible-playbook -e @answerfile-lab1.yml deploy-nsx.yml -vvv**

Deployment

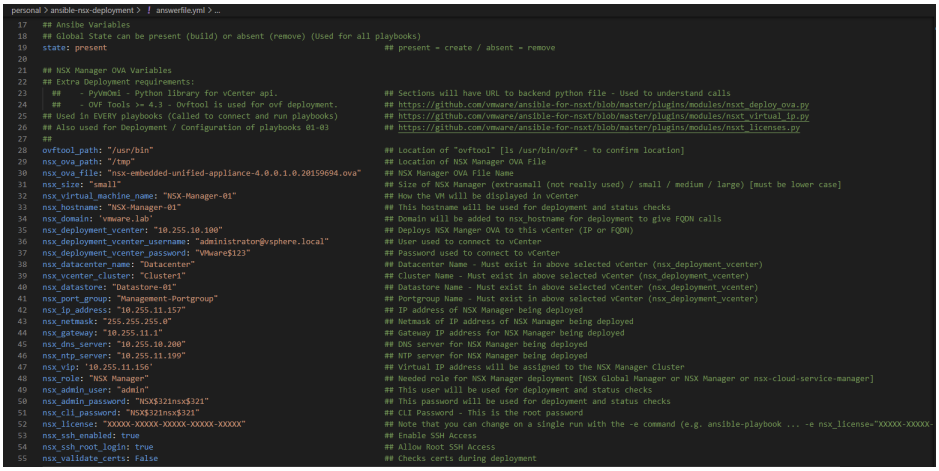
deploy-nsx.yml – A Playbook Calling Playbooks

work > ansible-nsx-deployment > deploy-nsx.yml

```yaml
---
- hosts: localhost
  name: NSX 3.2.x + 4.x Deployment
  gather_facts: false
  vars_files:
    - answerfile.yml


# Infrastructure Deployment #

- ansible.builtin.import_playbook: playbooks/01-deploy-nsxmanager.yml
- ansible.builtin.import_playbook: playbooks/02-configure-nsxmanager-vip.yml
- ansible.builtin.import_playbook: playbooks/03-configure-nsxmanager-license.yml
- ansible.builtin.import_playbook: playbooks/04-configure-compute-managers.yml
- ansible.builtin.import_playbook: playbooks/05-configure-nsxmanager-uplinkprofiles.yml
- ansible.builtin.import_playbook: playbooks/06-configure-nsxmanager-ippools.yml
- ansible.builtin.import_playbook: playbooks/07-configure-nsxmanager-transportzones.yml
- ansible.builtin.import_playbook: playbooks/08-configure-nsxmanager-transportnodesprofiles.yml
- ansible.builtin.import_playbook: playbooks/09-configure-nsxmanager-transportnodes.yml
- ansible.builtin.import_playbook: playbooks/10-deploy-edgenodes.yml
- ansible.builtin.import_playbook: playbooks/11-configure-nsxmanager-edgeclusters.yml

```

Deployment

answerfile.yml – OVA Deployment



```
personal > ansible-nsx-deployment > ! answerfile.yml > ...
17  ## Ansibe Variables
18  ## Global State can be present (build) or absent (remove) (Used for all playbooks)
19  state: present                                          ## present = create / absent = remove
20
21  ## NSX Manager OVA Variables
22  ## Extra Deployment requirements:
23  ##     - PyVmOmi - Python library for vCenter api.      ## Sections will have URL to backend python file - Used to understand calls
24  ##     - OVF Tools >= 4.3 - Ovftool is used for ovf deployment.  ## https://github.com/vmware/ansible-for-nsxt/blob/master/plugins/modules/nsxt_deploy_ova.py
25  ## Used in EVERY playbooks (Called to connect and run playbooks)  ## https://github.com/vmware/ansible-for-nsxt/blob/master/plugins/modules/nsxt_virtual_ip.py
26  ## Also used for Deployment / Configuration of playbooks 01-03  ## https://github.com/vmware/ansible-for-nsxt/blob/master/plugins/modules/nsxt_licenses.py
27  ##
28  ovftool_path: "/usr/bin"                                ## Location of "ovftool" [ls /usr/bin/ovf* - to confirm location]
29  nsx_ova_path: "/tmp"                                    ## Location of NSX Manager OVA File
30  nsx_ova_file: "nsx-embedded-unified-appliance-4.0.0.1.0.20159694.ova"  ## NSX Manager OVA File Name
31  nsx_size: "small"                                       ## Size of NSX Manager (extrasmall (not really used) / small / medium / large) [must be lower case]
32  nsx_virtual_machine_name: "NSX-Manager-01"              ## How the VM will be displayed in vCenter
33  nsx_hostname: "NSX-Manager-01"                          ## This hostname will be used for deployment and status checks
34  nsx_domain: 'vmware.lab'                                ## Domain will be added to nsx_hostname for deployment to give FQDN calls
35  nsx_deployment_vcenter: "10.255.10.100"                 ## Deploys NSX Manger OVA to this vCenter (IP or FQDN)
36  nsx_deployment_vcenter_username: "administrator@vsphere.local"  ## User used to connect to vCenter
37  nsx_deployment_vcenter_password: "VMware$123"           ## Password used to connect to vCenter
38  nsx_datacenter_name: "Datacenter"                       ## Datacenter Name - Must exist in above selected vCenter (nsx_deployment_vcenter)
39  nsx_vcenter_cluster: "Cluster1"                         ## Cluster Name - Must exist in above selected vCenter (nsx_deployment_vcenter)
40  nsx_datastore: "Datastore-01"                           ## Datastore Name - Must exist in above selected vCenter (nsx_deployment_vcenter)
41  nsx_port_group: "Management-Portgroup"                  ## Portgroup Name - Must exist in above selected vCenter (nsx_deployment_vcenter)
42  nsx_ip_address: "10.255.11.157"                         ## IP address of NSX Manager being deployed
43  nsx_netmask: "255.255.255.0"                            ## Netmask of IP address of NSX Manager being deployed
44  nsx_gateway: "10.255.11.1"                              ## Gateway IP address for NSX Manager being deployed
45  nsx_dns_server: "10.255.10.200"                         ## DNS server for NSX Manager being deployed
46  nsx_ntp_server: "10.255.11.199"                         ## NTP server for NSX Manager being deployed
47  nsx_vip: '10.255.11.156'                                ## Virtual IP address will be assigned to the NSX Manager Cluster
48  nsx_role: "NSX Manager"                                 ## Needed role for NSX Manager deployment [NSX Global Manager or NSX Manager or nsx-cloud-service-manager]
49  nsx_admin_user: "admin"                                 ## This user will be used for deployment and status checks
50  nsx_admin_password: "NSX$321nsx$321"                    ## This password will be used for deployment and status checks
51  nsx_cli_password: "NSX$321nsx$321"                      ## CLI Password - This is the root password
52  nsx_license: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXX"           ## Note that you can change on a single run with the -e command (e.g. ansible-playbook ... -e nsx_license="XXXXX-XXXXX-
53  nsx_ssh_enabled: true                                   ## Enable SSH Access
54  nsx_ssh_root_login: true                                ## Allow Root SSH Access
55  nsx_validate_certs: False                               ## Checks certs during deployment
```


URLs are provided to the python code that runs on the backend. Looking inside these files will provide you with other options that you can use to modify the playbooks / answer file. These are at the start of every section.

Deployment

answerfile.yml – OVA Deployment

personal > ansible-nsx-deployment > ! answerfile.yml > ...

```yaml
## Ansibe Variables
## Global State can be present (build) or absent (remove) (Used for all p
state: present

## NSX Manager OVA Variables
## Extra Deployment requirements:
##    - PyVmOmi - Python library for vCenter api.
##    - OVF Tools >= 4.3 - Ovftool is used for ovf deployment.
## Used in EVERY playbooks (Called to connect and run playbooks)
## Also used for Deployment / Configuration of playbooks 01-03
##
ovftool_path: "/usr/bin"
nsx_ova_path: "/tmp"
nsx_ova_file: "nsx-embedded-unified-appliance-4.0.0.1.0.20159694.ova"
nsx_size: "small"
nsx_virtual_machine_name: "NSX-Manager-01"
nsx_hostname: "NSX-Manager-01"
nsx_domain: 'vmware.lab'
nsx_deployment_vcenter: "10.255.10.100"
nsx_deployment_vcenter_username: "administrator@vsphere.local"
nsx_deployment_vcenter_password: "VMware$123"
nsx_datacenter_name: "Datacenter"
nsx_vcenter_cluster: "Cluster1"
nsx_datastore: "Datastore-01"
nsx_port_group: "Management-Portgroup"
nsx_ip_address: "10.255.11.157"
nsx_netmask: "255.255.255.0"
nsx_gateway: "10.255.11.1"
nsx_dns_server: "10.255.10.200"
nsx_ntp_server: "10.255.11.199"
nsx_vip: '10.255.11.156'
nsx_role: "NSX Manager"
nsx_admin_user: "admin"
nsx_admin_password: "NSX$321nsx$321"
nsx_cli_password: 'NSX$321nsx$321'
nsx_license: "XXXXX-XXXXX-XXXXX-XXXXX-XXXXX"
nsx_ssh_enabled: true
nsx_ssh_root_login: true
nsx_validate_certs: False
```

personal > ansible-nsx-deployment > playbooks > 01-deploy-nsxmanager.yml

```yaml
---
- hosts: localhost
  name: 01 - Deploy the NSX Manager to a vCenter environment
  gather_facts: false
  vars_files:
    - ../answerfile.yml
  tasks:
    - name: Deploy NSX Manager OVA
      vmware.ansible_for_nsxt.nsxt_deploy_ova:
        ovftool_path: "{{ ovftool_path }}"
        path_to_ova: "{{ nsx_ova_path }}"
        ova_file: "{{ nsx_ova_file }}"
        deployment_size: "{{ nsx_size }}"
        vmname: "{{ nsx_virtual_machine_name }}"
        hostname: "{{ nsx_hostname }}.{{ nsx_domain }}"
        dns_domain: "{{ nsx_domain }}"
        vcenter: "{{ nsx_deployment_vcenter }}"
        vcenter_user: "{{ nsx_deployment_vcenter_username }}"
        vcenter_passwd: "{{ nsx_deployment_vcenter_password }}"
        portgroup: "{{ nsx_port_group }}"
        ip_address: "{{ nsx_ip_address }}"
        netmask: "{{ nsx_netmask }}"
        gateway: "{{ nsx_gateway }}"
        dns_server: "{{ nsx_dns_server }}"
        ntp_server: "{{ nsx_ntp_server }}"
        role: "{{ nsx_role }}"
        admin_password: "{{ nsx_admin_password }}"
        cli_password: "{{ nsx_cli_password }}"
        ssh_enabled: "{{ nsx_ssh_enabled }}"
        allow_ssh_root_login: "{{ nsx_ssh_root_login }}"
        datacenter: "{{ nsx_datacenter_name }}"
        cluster: "{{ nsx_vcenter_cluster }}"
        datastore: "{{ nsx_datastore }}"

    - name: Checking NSX Manager Status - Waiting until NSX Manager is Up
      vmware.ansible_for_nsxt.nsxt_manager_status:
        hostname: "{{ nsx_hostname }}.{{ nsx_domain }}"
        username: "{{ nsx_admin_user }}"
        password: "{{ nsx_admin_password }}"
        validate_certs: "{{ nsx_validate_certs }}"
        # Amount of time (minutes) the task will wait for the NSX Manager to come back ready
        wait_time: 30
...
```


Answerfile on left
Playbook on right

Deployment

answerfile.yml – Adding Compute Managers

personal > ansible-nsx-deployment > answerfile.yml > [ ] compute_managers > { } 0

```
57  ## Compute Managers Section (dynamic)
58  ## Used for playbook 04
59  compute_managers:
60  - display_name: "nsx-vcenter"
61    server: "nsx-vcenter.vmware.lab"
62    credentials:
63      credential_type: UsernamePasswordLoginCredential
64      username: "administrator@vsphere.local"
65      password: "VMware1!"
66  ## Start of 2nd Compute Manager
67  - display_name: "dev-vcenter"
68    server: "dev-vcenter.vmware.lab"
69    credentials:
70      credential_type: UsernamePasswordLoginCredential
71      username: "administrator@vsphere.local"
72      password: "VMware$123"
73
```

personal > ansible-nsx-deployment > playbooks > 3.1 > 04-configure-compute-managers.yml

```
1   ---
2   - hosts: localhost
3     name: 04 - Register a vCenter as a Compute Manager in NSX
4     gather_facts: false
5     vars_files:
6       - ../../answerfile.yml
7
8     tasks:
9       - name: Register Compute Manager
10        vmware.ansible_for_nsxt.nsxt_fabric_compute_managers:
11          hostname: "{{ nsx_hostname }}.{{ nsx_domain }}"
12          username: "{{ nsx_admin_user }}"
13          password: "{{ nsx_admin_password }}"
14          validate_certs: "{{ nsx_validate_certs }}"
15          display_name: "{{ item.display_name }}"
16          server: "{{ item.server }}"
17          origin_type: vCenter
18          credential: "{{ item.credentials }}"
19          state: "{{ state }}"
20        with_items:
21          - "{{ compute_managers }}"
22  ...
23
```


Answerfile on left – 04 - Compute Managers Section

Playbook on right – 04 Configure Compute Managers

Note
Playbook – line 20 – with_items: compute_managers
This is making a reference back to the answerfile, looking for section called "compute_managers"

In these pictures we can see that 2 different compute managers have been assigned, they start over with the – display_name section.
nsx-vcenter and dev-vcenter

Resources

Resources
URLs or BS

The BEST place to start is the main Ansible Documentation site.

https://docs.ansible.com/

https://docs.ansible.com/ansible/latest/

https://docs.ansible.com/ansible-core/devel/index.html

https://docs.ansible.com/galaxy.html

https://github.com/ansible/ansible-examples

https://github.com/vmware/ansible-for-nsxt

https://github.com/newtoni-vmware/ansible-nsx-deployment

Github / git

https://docs.github.com/en

https://git-scm.com/docs

https://dzone.com/articles/top-20-git-commands-with-examples

https://github.com/rutgerblom/SDDC.Lab (REALLY nice pod deployment – Advance)

Visual Studio Code

https://code.visualstudio.com/docs

https://code.visualstudio.com/docs/getstarted/introvideos

Getting Started with Visual Studio Code (Videos)

SSH

http://manpages.ubuntu.com/manpages/trusty/man5/ssh_config.5.html

https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_keymanagement

YAML / Templates

https://www.commonwl.org/user_guide/topics/yaml-guide.html

https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started

https://jinja.palletsprojects.com/en/3.1.x/intro/

https://www.linuxtechi.com/configure-use-ansible-jinja2-templates/

https://blog.learncodeonline.in/everything-about-ansible-jinja2-templating

https://linuxways.net/centos/how-to-use-the-jinja2-template-in-ansible/

All these URLs are within the notes of slides. Pictures used from other sites have their URLs in the notes of that slide.

Resources

URLs or BS

Online Classes

https://www.udemy.com/course/diveintoansible/

https://www.udemy.com/course/ansible-masterclass/

https://www.udemy.com/course/cumulus-linux-fundamentals-plus-ansible-automation/

https://www.udemy.com/course/learning-path-automation-with-ansible-puppet-and-salt/

Learning Ansible – Linkedin Learning

Ansible Essential Training – Linkedin Learning

Udemy has a lot of good classes, some will give discount codes, do a search for the instructor + coupon etc. I found a few on github or udemy has sales all the time. Spent under $50 for all classes total.