>>> network .toCode()

Extending Ansible

Workshop

Agenda

- Introduction
- The Ansible Plugin System
- Custom Filters
  - Demo
  - Hands-on Labs!
- Custom Modules
  - Demo
  - Hands-on Labs!

Ansible Plugins

- A lot of Ansible functionality comes from its Plugins
- **Vars Plugins**
  - Inject data from other sources apart from inventory/playbook/cmdline
  - host_vars, group_vars are loaded via plugin
- **Inventory Plugins**
  - Code that pulls dynamic inventory data from other sources (e.g. NetBox, APIs, Databases etc.)
- **Callback Plugins**
  - Control how Ansible responds to events. Output coming from the CLI.
  - Make Ansible output in JSON or even send API calls to Slack
- **Other: Action, Cache, Lookup, Strategy, Become, Connection**

Ansible Plugins

Connection Plugins

- How Ansible connects to managed devices
- Most common is SSH via Paramiko or the native SSH client

```
> ansible-doc -t connection -l
docker       Run tasks in docker containers
httpapi      Use httpapi to run command on network appliances
kubectl      Execute tasks in pods running on Kubernetes
local        execute on controller
napalm       Provides persistent connection using NAPALM
netconf      Provides a persistent connection using the netconf protocol
network_cli  Use network_cli to run command on network appliances
paramiko_ssh Run tasks via python ssh (paramiko)
saltstack    Allow ansible to piggyback on salt minions
ssh          connect via ssh client binary
vmware_tools Execute tasks inside a VM via VMware Tools
... and many more!
```

Ansible Plugins

Become Plugins

- Help Ansible use various privilege escalation systems
- As in “become” another user

```
> ansible-doc -t become -l
doas        Do As user
dzdo        Centrify's Direct Authorize
enable      Switch to elevated permissions on a network device
ksu         Kerberos substitute user
machinectl  Systemd's machinectl privilege escalation
pbrun       PowerBroker run
pfexec      profile based execution
pmrun       Privilege Manager run
runas       Run As user
sesu        CA Privileged Access Manager
su          Substitute User
sudo        Substitute User DO
```

Filter Plugins

- **Filter Plugins** manipulate data and are a feature of Jinja
  - **Jinja Built-in Filters**
    - **https://jinja.palletsprojects.com/en/2.11.x/templates/#builtin-filters**
  - **Ansible Filters**
    - **https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html**
  - **Your Custom Filters**
- They are an integral part of Ansible's DSL (Domain Specific Language) - i.e. the YAML playbooks and its templating engine (including the **template** module!)
  - https://github.com/ansible/ansible/tree/devel/lib/ansible/plugins/filter

Ansible Plugins

Where does Ansible look for plugins?

- Any folder added to the ANSIBLE_PLUGIN_TYPE_PLUGINS environment variable (it's a colon-separated list like your system PATH)
    - e.g. ANSIBLE_FILTER_PLUGINS
    - Change the plugin search path using a custom **ansible.cfg** file
- Loaded from specific folders found next to the playbook
    - e.g. **filter_plugins,** lookup_plugins, connection_plugins etc.
    - this makes for simple packaging and distribution with your plays

- https://docs.ansible.com/ansible/latest/dev_guide/developing_locally.html

LAB TIME!

Labs: 01-04

Ansible Modules

- A **Module** is a reusable, standalone script that Ansible runs (via API, **ansible** or **ansible-playbook**)
  - Ansible provides the runtime framework, the input parameters, and captures the results (or the output)
  - Thousands of built-in modules - either from Redhat or third parties like NetworkToCode, Cisco, Juniper, Amazon, VMware etc.
- If you need functionality that's not available you can write your own
  - Others might find it useful so consider open sourcing it! :)

Ansible Modules

- A **Module** is a python script - its filename is the module name!
  - A large part will be documentation… make sure you include some with your module to make it easy to use it.
  - Documentation on writing Module Documentation
- Modules take arguments, which with their options = argument_spec
  - **required**, default, choices, aliases
  - **type**: str (default), list, dict, bool, int, float, path, json, bytes (+others)
- Modules can and *should* support check_mode (aka dry run)
- Ansible expects modules to output JSON
  - `{ 'changed': false, 'failed': true, 'msg': 'host unreachable' }`

Ansible Modules

- Where does Ansible look for modules?
  - Any folder added to the ANSIBLE_LIBRARY environment variable (it's a colon-separated list like your system PATH)
  - **(default)** $HOME/.ansible/plugins/modules
  - **(default)** /usr/share/ansible/plugins/modules/
  - Change the module search path using a custom **ansible.cfg** file

```
> ansible --version
ansible 2.9.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/etc/ntc/ansible/library']
  ansible python module location = /usr/local/lib/python3.6/site-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.6.8 (default, Jun 11 2019, 01:16:11) [GCC 6.3.0 20170516]
```

LAB TIME!

Labs: 05 and 06