

Ansible AWX Training

- [Introduction](#)
- [Installation](#)
 - [CentOS Red Hat Installation](#)
 - [Notes](#)
 - [Docker](#)
 - [Docker Getting Started](#)
 - [Docker Useful Commands](#)
 - [Running Ansible on your laptop](#)
 - [Installing Vagrant](#)
 - [Vagrant Basic Commands](#)
- [Quick Start](#)
 - [Concepts](#)
 - [Ansible Engine](#)
 - [Inventory](#)
 - [YAML](#)
 - [Configuration File](#)
 - [Playbooks](#)
 - [Modules](#)
 - [Ansible Tower - AWX](#)
 - [Ad-hoc Commands](#)
 - [Ansible Engine](#)
 - [AWX](#)
 - [Ansible Engine](#)
- [Inventories](#)
 - [Ansible Engine](#)
 - [AWX](#)
 - [Bulk Import](#)
 - [Dynamic Inventory](#)
 - [vmware](#)
- [Credentials](#)
- [Playbooks](#)
 - [Running Playbooks Deep Dive](#)
 - [Ansible Engine](#)
 - [Variables](#)
 - [AWX](#)
 - [Organizations](#)
 - [User Management](#)
 - [Project](#)
 - [Manual Local Repo](#)

- [Git Repo](#)
- [Templates](#)
 - [Job Template](#)
 - [Workflow Template](#)
- [Notifications](#)
 - [Email](#)
 - [Twilio](#)
- [Advanced Playbooks Features](#)
 - [Error Handling](#)
 - [Handlers](#)
 - [Templates and Variables](#)
 - [Loops](#)
 - [With_items](#)
 - [With_nested](#)
 - [File Manipulation](#)
- [Modules](#)
 - [Files](#)
 - [Archive](#)
 - [Lineinfile](#)
 - [Replace](#)
 - [Template](#)
 - [Yum](#)
 - [Command](#)
 - [System](#)
 - [Firewalld](#)
 - [Timezone](#)
 - [Jira](#)
 - [Notifications](#)
 - [Custom Modules: Writing your own module](#)
 - [Folder Structure](#)
- [Roles](#)
 - [Molecule](#)
- [Ansible Galaxy](#)
 - [Search](#)
 - [List installed roles](#)
 - [Remove installed roles](#)
 - [Import a role](#)
- [UCS Manager Management](#)
- [Cisco NXOS Management](#)
 - [Playbooks](#)
 - [Create VLANs](#)

- [Configure Portchannels](#)
- [Feature Enablement](#)
- [Cisco ASA Management](#)
 - [Playbooks](#)
 - [Create Network object](#)
- [Cisco IOS Management](#)
 - [Playbooks](#)
- [Deploying to Amazon Web Services](#)
 - [Inventory](#)
 - [Playbooks](#)
- [Deploying to Google Cloud Platform](#)
 - [Inventory](#)

Introduction

This document provides an overview on Ansible and AWX features and hands on tutorials.

Installation

Please refer to the following link: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

CentOS Red Hat Installation

These are the steps to install AWX/Ansible on top of CentOS/RHEL:

#EPEL Repo

```
yum -y install epel-release
```

Install Additional Packages

```
yum -y install git gettext ansible docker nodejs npm gcc-c++ bzip2
```

```
yum -y install python-docker-py
```

Run Docker

```
systemctl start docker
```

```
systemctl enable docker
```

```
# Clone AWX Repo and Deploy
git clone https://github.com/ansible/awx.git
cd awx/installer/
ansible-playbook -i inventory install.yml
```

Notes

You might need to modify git commands that access remote repositories if your internet access is through a proxy server. Once you have obtained the proxy settings (server URL, port, username and password); you need to configure your git as follows:

```
# git config --global http.proxy http://<username>:<password>@<proxy-server-url>:<port>
```

You would need to replace <username>, <password>, <proxy-server-url>, <port> with the values specific to your proxy server credentials.

If you do not specify “project_data_dir=/var/lib/awx/projects” during the deployment, you can’t use Manual SCM. You might face issue while creating the new project.

If you are behind an HTTP or HTTPS proxy server, you need to add the proxy configuration in the Docker systemd service file.

Docker

AWX relies on five Docker containers provided by Red Hat Ansible in Docker Hub (<https://hub.docker.com/u/ansible>).

The Ansible installation creates a PostgreSQL database that will be in one container and will create the AWX one that contains the web tier, the engine, a cache and a queue:

- AWX Web. Server which provides HTTP GUI access.
- AWX Task. Engine responsible of providing API interworking for solution components.
- RabbitMq. Acting as a messaging broker between solution components.
- Memcached. Memcached is a general-purpose distributed memory caching system. It is used to speed up dynamic database-driven websites by caching data and objects in RAM to reduce the number of times an external data source (such as a database or API) must be read.
- Postgres database. AWX requires access to a PostgreSQL database, and by default, one will be created and deployed in a container, and data will be persisted to a host volume.

Docker Getting Started

Once Docker is installed, open a terminal and type the following commands:

```
#docker info
#docker -v
```

If the installation worked, you will see a bunch of information about your Docker installation and docker version running:

```
# docker -v
```

```
# docker info

Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 1.13.1
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
Logging Driver: journald
Cgroup Driver: systemd
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Authorization: rhel-push-plugin
Swarm: inactive
Runtimes: docker-runc runc
Default Runtime: docker-runc
```

You can list the containers that are running by issuing the following command:

```
# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
28143772a881        ansible/awx_task:2.1.2   "sh ..."          19 minutes ago    Up 19 minutes     8052/tcp           awx_task
ded05efa006d        ansible/awx_web:2.1.2    "sh ..."          19 minutes ago    Up 19 minutes     0.0.0.0:80->8052/tcp   awx_web
a76b6a798a5b        memcached:alpine      "memcached"       43 hours ago     Up 6 hours        11211/tcp          memcached

```

Use the command "docker exec -it <container ID> /bin/bash" to get a bash shell in the container:

```
[root@RedhatNewAnsible projects]# docker exec -it ded05efa006d /bin/
bash
[root@awxweb awx]# ls
awxfifo favicon.ico projects public supervisord.log supervisord.pid
venv wsgi.py
[root@awxweb awx]# exit
exit
[root@RedhatNewAnsible projects]#
```

Docker Useful Commands

Stops one or more containers.

docker stop my_container

Stop all running containers:

docker stop \$(docker ps -a -q)

The following command does not attempt to shut down the process gracefully first:

docker kill my_container

Start one or more containers:

docker start my_container

This command displays the logs of a container:

docker logs --follow my_container

Remove one or more containers:

docker rm my_container

Remove one or more images:

```
docker rmi my_image
```

Running Ansible on your laptop

Vagrant is a tool for building and managing virtual machine environments. Vagrant provides easy to configure, reproducible, and portable work environments built on top of industry-standard technology.

Installing Vagrant

First of all, request local admin rights in advance to install the required software.

Download Virtualbox: <https://www.virtualbox.org> and install it.

The screenshot shows the 'VirtualBox' website's download page. On the left, there's a sidebar with links like 'News', 'Downloads', 'Overview', 'Documentation', 'End-user docs', 'Technical docs', 'Centaurus', and 'Community'. The main content area has a large 'VirtualBox' logo at the top. Below it, a section titled 'Download VirtualBox' contains a note about the page containing links to VirtualBox binaries and source code. It then lists 'VirtualBox binaries' with a note that by downloading, you agree to the terms and conditions of the respective license. It also mentions that if you're looking for the latest VirtualBox 6.0 packages, see [VirtualBox 6.0 binaries](#). A note states that version 6.0 will receive support until July 2023. Below this, there's a section for 'VirtualBox 6.0.4 platform packages' with a list of operating systems: Windows host, OS X host, Linux distributions, and Solaris hosts.

Download Vagrant: <https://www.vagrantup.com> for your operating system.

The screenshot shows the 'Vagrant' website's download page. At the top, there's a navigation bar with links for 'Downloads', 'Documentation', 'Community', 'Issues', 'Pull Requests', and 'Search'. Below the navigation, there's a search bar and a link to 'Enter your GitHub Project Here'. The main content area has a 'Download Vagrant' heading. It says: 'Below are the available downloads for the latest version of Vagrant (2.2.4). Please download the proper package for your operating system and architecture.' It includes a note about finding the SHA256 checksum for Vagrant 2.2.4 and verifying it with the download's signature file. It also links to the Vagrant GitHub repository. Below this, there are two download options: 'Debian' (32-bit | 64-bit) and 'Windows' (32-bit | 64-bit).

Ensure that Vagrant is properly installed by issuing the "vagrant" command as shown below:

```

Usage: vagrant [options] commands [args...]
  -h, --help           Prints this message and exits.
  -v, --version        Prints the version and exits.
  -V, --verbose        Prints lots of help.

Common commands:
  box                 Manages Boxes (Vagrantfile, Vagrant, etc).
  cloud               Manages everything related to Vagrant Cloud.
  destroy              Deletes all traces of the vagrant machine.
  status              Prints status against environments for this box.
  halt                Pauses the vagrant machine.
  help                Shows the help for a subcommand.
  init                Initializes a new Vagrant environment by creating a Vagrantfile.
  provision           Packages a running vagrant environment into a box.
  provisioner         Manages provisioner (VirtualBox, VMWare, etc).
  port                Displays information about guest port mappings.
  provider            Converts to another via provider-level networking.
  push                Deploys existing environment to a configured destination.
  rbd                Connects to RADOS RBD.
  reload              Restarts vagrant machine. Loads new Vagrantfile configuration.
  resume              Resumes a suspended vagrant machine.
  snapshot            Manages snapshots (saving, reverting, etc).
  syncbox             Syncs files between host and guest.

```

Create a local directory and then make a shared folder.

```

sudo mkdir centos7
cd centos7
sudo mkdir shared

```

Download the boxes you need to deploy from Vagrant Cloud: <https://app.vagrantup.com/boxes/search> (i.e. centos/7)



Discover Vagrant Boxes				
<input type="text" value="Search for boxes by operating system, included software, architecture and more."/> ✖				
		Sort By Downloads Recently Created Recently Updated		
	ubuntu/trusty64 13.10 (2013-10-17)	VirtualBox	Downloads: 38,385,370	Released: 10 days ago
	laravel/laravel 5.0.1-alpine	VirtualBox parallels VirtualBox VMware Windows	Downloads: 13,500,183	Released: 5 days ago
	hashicorp/precise64 1.1.0	VirtualBox parallels VirtualBox VMware Windows	Downloads: 6,737,697	Released: about 5 years ago
	centos/7 18.04	VirtualBox parallels VirtualBox VMware Windows	Downloads: 4,097,146	Released: 5 months ago

The command below downloads a centos 7 box

```
sudo vagrant box add centos/7
```

In this case, I am downloading a box called "jumperfly/ansible-2.8" which provides Ansible already installed.

NOTE: This is not an official box so be aware of the security risks.

```
# Shared code Vagrant box add jumperfly/ansible-2.8
--> box: Loading metadata for box 'jumperfly/ansible-2.8'
box: URL: https://vagrantcloud.com/jumperfly/ansible-2.8
--> box: Adding box 'jumperfly/ansible-2.8' (v0.4) to provider: virtualbox
box: Box loaded: https://vagrantcloud.com/jumperfly/ansible-2.8/ansible-2.8-v0.4.vmdk at /Users/steve/Downloads/jumperfly-ansible-2.8-v0.4.vmdk
--> box: Successfully added box 'jumperfly/ansible-2.8' (v0.4) for 'virtualbox'
```

Next step is to initialise the box.

sudo vagrant init centos/7

```
# Shared code Vagrant init jumperfly/ansible-2.8
# Vagrantfile has been placed in this directory. You can edit it if you
# need to. It contains all the settings for your box. Please refer
# to the documentation on 'vagrant init' for more information on what
# Vagrantfile does for more information on using Vagrant.
```

Launch your VM by issuing the command below:

sudo vagrant up

```
# Shared code vagrant up
--> default: Starting machine 'default'...
--> default: Importing base box 'jumperfly/ansible-2.8'...
--> default: Adding NAT adapter for VM networking...
--> default: Checking if box 'jumperfly/ansible-2.8' version '0.4' is up to date...
--> default: Setting the name of the VM: shared/default_28594a657f933609
--> default: Clearing any previously set network interfaces...
--> default: Preparing network interfaces based on configuration...
--> default: Bridge: br0
--> default: Forwarding ports...
--> default: 22 (guest) -> 2222 (host) (adapter 1)
--> default: Resizing disk...
--> default: Building box image (this may take a few minutes...)
--> default: 250M disk at: /Users/steve/.vagrant.d/boxes/jumperfly/ansible-2.8/v0.4/disk.vmdk
--> default: 599M swap swapfile swapfile
--> default: 
--> default: vagrant insecure key detected. Vagrant will automatically replace
--> default: this with a newly generated keypair for better security.
--> default: 
--> default: Generating public key within guest...
--> default: Removing insecure key from the guest of its current...
--> default: Key 'vagrant' disconnected and reconnected using new SSH key...
--> default: Public key copied and ready!
--> default: Checking for guest additions in VM...
--> default: No guest additions found.
--> default: Guest will not be clean/shutdown
```

Access the VM initiated. In this case, Ansible is already installed ("ansible --version").

sudo vagrant ssh

```
# Shared code vagrant ssh
Shared among environments: (1) (VAGRANT) to save file or directory
vagrant@centos-7-vm-15:~$ ansible --version
ansible 2.4.3
  config file = None
  configured module search path = [u'/usr/share/virtualbox/VagrantPlugins/Ansible/Ansible/ModuleLister/modulelist']
  available python modules = AnsibleModuleLister, AnsibleModuleListerLister
  executable = /usr/bin/ansible
  python version = 2.7.5 (Ubuntu 2.7.5-15ubuntu10.22.4-0ubuntu1.14.04.1)
  python version = 2.7.5 (CentOS., Apr  9 2016, 16:29:42) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]
```

Vagrant Basic Commands

Navigate to the folder which stores your Vagrantfile and pause the VM

sudo vagrant suspend

Switch off the VM

```
sudo vagrant halt
```

Destroy the VM

```
sudo vagrant destroy
```

Quick Start

Concepts

Ansible Engine

Inventory

The Ansible inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The file can be in one of many formats depending on your Ansible environment and plugins. The default location for the inventory file is /etc/ansible/hosts. If necessary, you can also create project-specific inventory files in alternate locations.

The inventory file can list individual hosts or user-defined groups of hosts. For example, if you are managing one or more data centers, you can create Ansible groups for those components that require the same set of operations.

YAML

YAML stands for "YAML Ain't Markup Language" (Please refer to: <https://yaml.org>).

It is basically a human-readable structured data format. It is less complex and ungainly than XML or JSON, but provides similar capabilities.

There are some rules that YAML has in place to avoid issues related to ambiguity. These rules make it possible for a single YAML file to be interpreted consistently, regardless of which library is being used to interpret it.

- YAML files should end in .yaml.
- YAML is case sensitive.
- YAML uses a fixed indentation scheme to represent relationships between data layers.
- Dictionary keys are represented in YAML as strings terminated by a trailing colon. Values are represented by either a string following the colon, separated by a space.
- To represent lists of items, a single dash followed by a space is used. Multiple items are a part of the same list as a function of their having the same level of indentation.

The following example represents a YAML playbook:

```
- name: configure interface settings
  ios config:
    lines:
      - description
```

Configuration File

Certain settings in Ansible are adjustable via a configuration file (ansible.cfg). The stock configuration should be sufficient for most users, but there may be reasons you would want to change them.

Configuration file which will be processed in the following order:

1. Environment variable: ANSIBLE_CONFIG
2. Actual directory "ansible.cfg" file.
3. Home directory "ansible.cfg" file.
4. Configuration file stored in "/etc/ansible/ansible.cfg".

Option	Default Value	Description
inventory	/etc/ansible/hosts	Inventory location
forks	5	Specify number of parallel processes to use
remote_port	22	Remote SSH port
host_key_checking	true	Check host key installed
timeout	10	SSH connection timeout in seconds
remote_user	root	Remote connection user
become	false	Run operations with become (does not imply password prompting)
become_method	sudo	Privilege escalation method to use
pipelining	false	Reduces the number of network operations required to execute a module on the remote server, by executing many Ansible modules without actual file transfer. This can result in a very significant performance improvement when enabled

Please refer to Ansible documentation for more details: https://docs.ansible.com/ansible/latest/reference_appendices/config.html

Playbooks

Playbooks are Ansible's configuration, deployment, and orchestration language. Playbooks are designed to be human-readable and are developed in a basic text language. There are multiple ways to organize playbooks and the files they include.

Modules

Ansible ships with a number of modules (called the 'module library') that can be executed directly on remote hosts or through playbooks. Users can also write their own modules. These modules can control system resources, like services, packages, or files , or handle executing system commands.

Ansible modules: https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html

Documentation

ANSIBLE

PRODUCTS

COMMUNITY

WEBINARS & TRAINING

FAQ

Ansible 2.8

If you're looking for Ansible 2.9, see the documentation archive.

SEARCH DOCUMENTATION

INSTALLATION, UPGRADE & CONFIGURATION

Ansible Guide

Ansible Porting Guide

ANSIBLE GLOSSARY

Getting Started

Working with Command Line Tools

Introducing The Ansible Community

Working with Inventory

Working With Playbooks

Understanding Hostfile Locations

Ansible 2.8 modules

All modules

- + `ad6_server` - Manage AD6 Networks AD6/S-NAS/Thunder devices' server objects
- + `ad6_server_group` - Manage AD6 Networks AD6/S-NAS/Thunder devices' Thunder devices' service groups
- + `ad6_virtual_server` - Manage AD6 Networks AD6/S-NAS/Thunder devices' Thunder devices' virtual servers
- + `ad6_user` - Manage AD6 users (authenticated)
- + `ad6c_user_certificate` - Manage AAA user certificates (Basic/GenCert)
- + `ad6access_port_block_to_access_port` - Manage port blocks of fabric interface policy leaf profile Interface selection (Infra-IPoE, Infra-PortBlocks)
- + `ad6access_port_to_interface_policy_leaf_profile` - Manage fabric interface policy leaf profile Interface selection (Infra-IPoE, InfraAccessPorts_infraPortBlocks)
- # `ad6access_port_to_leaf_access_port` - Manage sub port blocks of static interface policy leaf profile Interface selection (Infra-IPoE, InfraAccessPorts)
- + `ad6_ipo` - Manage interface Access Profile Profile (AIP) objects (InfraAIPProfile, P_AIPProfile)
- + `ad6_ipo_to_domain` - Bind AIPs to Physical or Virtual Domains (InfraIPDomain)
- + `ad6_ipo` - Manage top level Application Profile (AIP) objects (lv_AIP)
- + `ad6_ipo` - Manage Bridge Domains (BD) objects (lv_BD)
- + `ad6_ipo_subnet` - Manage Subnets (InfraSubnet)
- + `ad6_ipo_to_domain` - Link Bridge Domains to IP oIP objects (lv_BDInfraSubnet)

The following command displays the list of available modules:

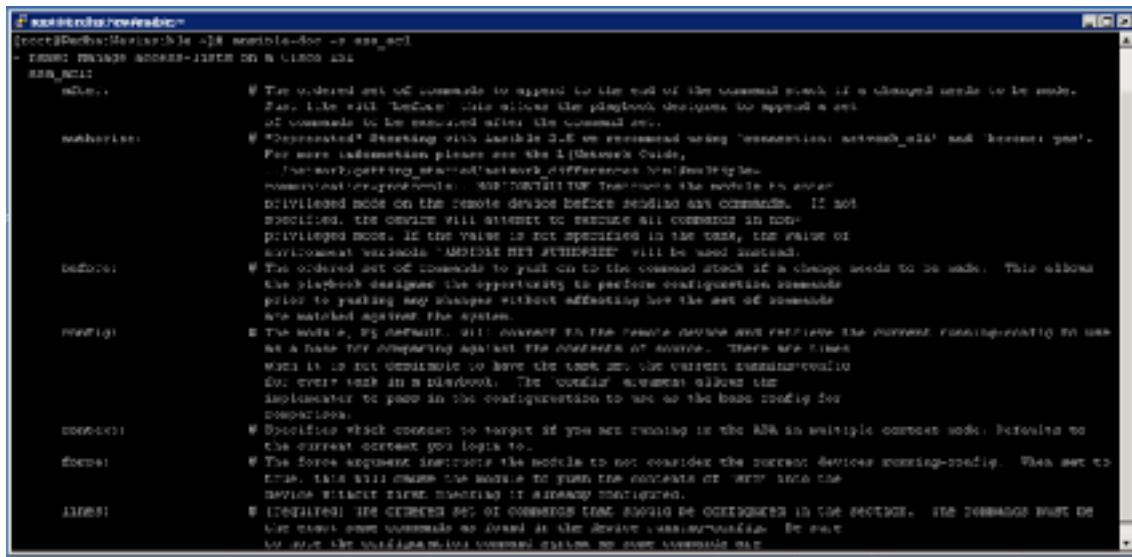
```
#ansible-doc -l | more
```

Narrow down modules related to ASA devices:

```
#ansible-doc -l | more | grep asa
```

Show actions that a module can perform:

ansible-doc -s module_name



Ansible Tower - AWX

The concepts across Ansible Engine and AWX (Ansible Tower) are the same. However, the GUI provided by AWX makes workflows being configured in a different way.

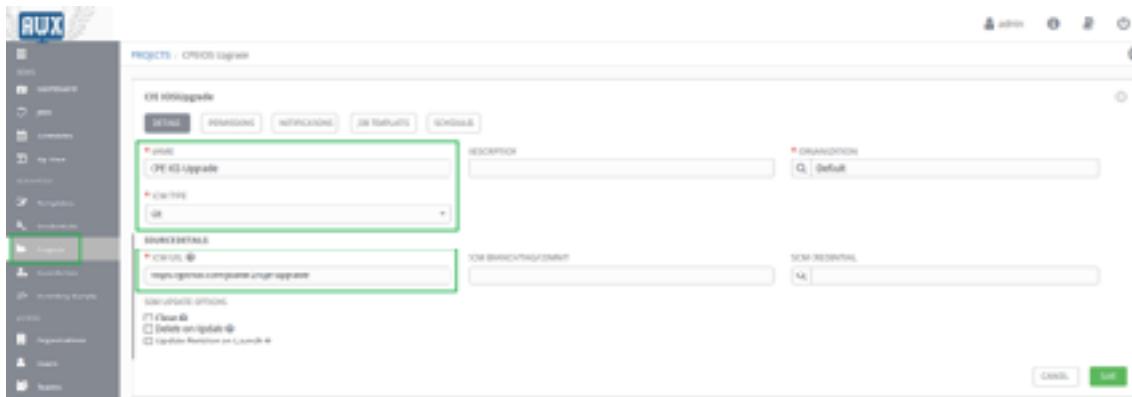
For running an Ansible Playbook with AWX, you need to configure the following items:

1. Credentials: User name/password or ssh key to connect to remote component.
 2. Project: It contains the Ansible playbook, config, roles, templates etc.
 3. Inventories: What servers the playbook will run against and connection specific configuration.
 4. Templates: Job template to associate all of the above and run the playbook
 5. Launch Templates: Launching current project.

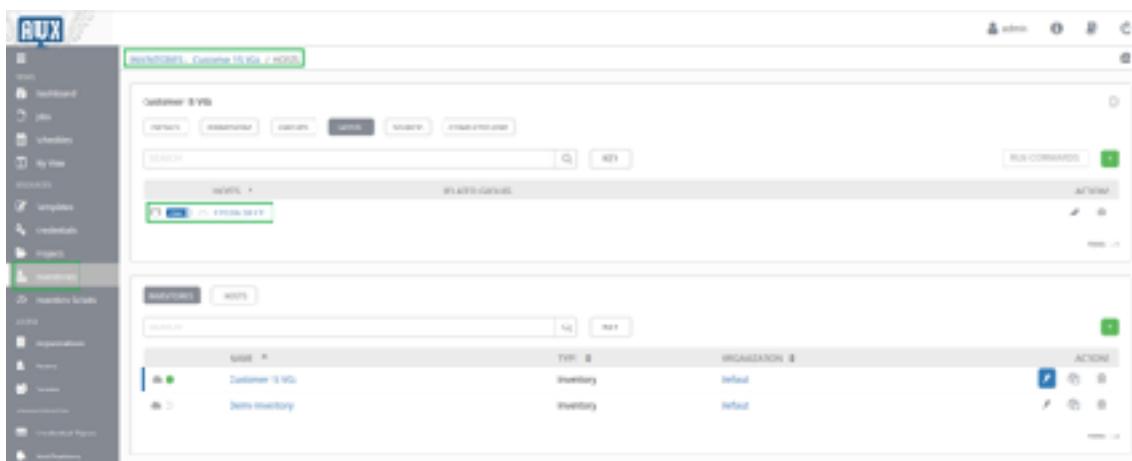
For instance, credentials (1) for a CPE device (Credential Type = Network) are created by typing the username and password value pair.



Next step requires creating a new Project (2) which is using Git as source.



Once the Project is saved, click on Inventories (3), create the corresponding one and add Hosts to it as shown below:



A new Hosts (i.e CPE IP address) is added to the "Customer 15 VGs" inventory.



Lets create a Job Template (4) which associated all the above: the inventory, credential and the Project which stores the list of playbooks.

The screenshot shows the 'TEMPLATES' tab in the Ansible AWX interface. A specific template named 'Retrieve Current IOS Version for IPB' is selected. The 'Run' button for this template is highlighted with a green box.

Finally, click on the Templates tab and launch it.

The screenshot shows the 'TEMPLATES' list page in the Ansible AWX interface. A template named 'Demo Job Template' is selected. The 'Run' button for this template is highlighted with a green box.

Ad-hoc Commands

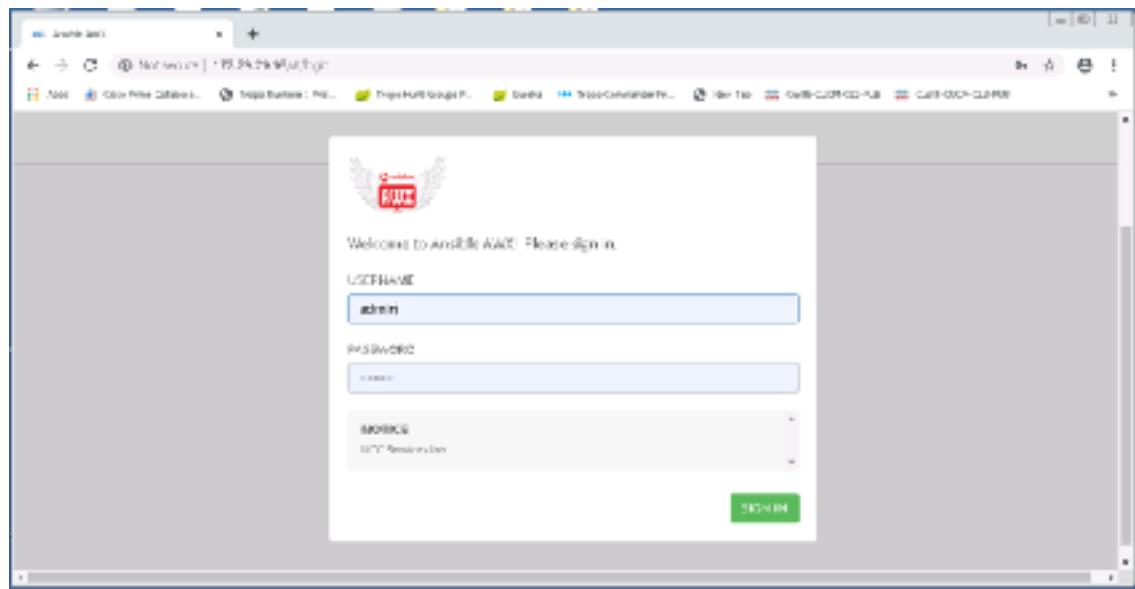
An Ansible Ad-hoc command is a one-liner Ansible command that performs a single task on the target host. It allows you to execute simple one-line task against one or group of hosts defined on the inventory file configuration.

An Ad-Hoc command will only have two parameters, the group of a host that you want to perform the task and the Ansible module to run.

Command	Description
#ansible all -m ping	The basic command of ansible ad-hoc against 'all' hosts on the inventory file and using the 'ping' module
#ansible localhost -m copy -a 'src=/home/myfile dest=/home/mydestinationfolder/myfilecopied'	The command copies a file to a destination folder
#ansible all -m yum -a "name=telnet state=present" -- become	The command installs the "telnet" package against 'all' hosts on the inventory file
#ansible all -m service -a "name=nginx state=started enabled=yes" --become	The commands starts the nginx service
#ansible localhost -m setup more	The command retrieves a bunch of system data from the remote host

AWX

Browse the corresponding URL as shown below.



Sign in with your credentials and you will be redirected to the AWX dashboard.



Click on Users tab, choose your username and click on Edit.

Edit your password, add you to specific organizations and teams, manage permissions and click on Save.

Ansible Engine

You can access Ansible Engine via SSH using any SSH client.

As a brief example, the /home/AnsibleTraining/ folder contains a inventory file called "hosts".

The /home/AnsibleTraining/playbooks/ folder stores the playbook below:

```
[root@RedhatNewAnsible playbooks]# cat asa_show_version.yml
---
- hosts: asa-lab
  gather_facts: false
  connection: network_cli

  tasks:
  - name: SHOW ASA VERSION
    asa_command:
      commands:
        - show version
    register: version

  - debug: var=version.stdout_lines
```

Let's analize the anatomy of the playbook above:

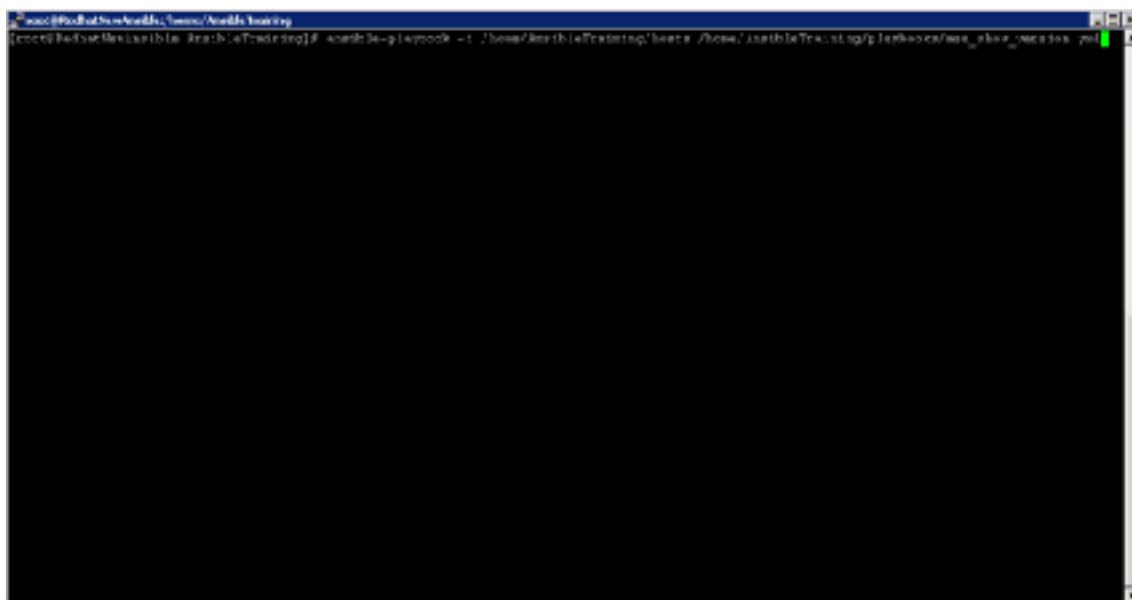
- hosts: Instructs the playbook to be run againsts the "asa-lab" group in the inventory file.
- gather_facts will connect to the managed host, run a script that collects a bunch of data (system, version, environment variables, etc.).
- Ansible uses the "connection" setting to determine how to connect to a remote device. When working with Ansible Networking, set this to network_cli so Ansible treats the remote

node as a network device with a limited execution environment. Without this setting, Ansible would attempt to use ssh to connect to the remote and execute the Python script on the network device, which would fail because Python generally isn't available on network devices.

- The tasks section includes two modules: "asa command" and "debug".
- The "asa_command" module is called in order to issue a "show version" command and store the fetched data in a register.
- "The "debug" commands prints the register output.

The following command runs the mentioned playbook:

```
#ansible-playbook -i /home/AnsibleTraining/hosts /home/AnsibleTraining/playbooks/asa_show_version.yml
```



Inventories

The inventory file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate. The inventory file can be in one of many formats depending on your Ansible environment and plugins.

The default location for the inventory file is `/etc/ansible/hosts`. If necessary, you can also create project-specific inventory files in alternate locations.

The inventory file can list individual hosts or user-defined groups of hosts. This enables you to define groups of devices with similar roles upon which to perform the same operational and configuration tasks. For example, if you are managing one or more data centers, you can create Ansible groups for those elements that require the same set of operations.

Ansible Engine

The following INI-formatted sample inventory file defines an individual host, called "webserver", and two groups of devices, `ios` and `nxos`.

```
$ cat /etc/ansible/hosts
```

The inventory file includes variables also:

```
[cisco-ios-devices:vars]
ansible_port=22
ansible_user=netadmin
```

Groups of groups are also supported:

```
[cisco-ios-devices]
10.10.0.1
10.10.0.2

[cisco-nxos-devices]
10.10.20.1
10.10.20.2

[network-devices:children]
cisco-ios-devices
cisco-nxos-devices
```

Patterns can be used as shown below:

```
[cisco-nexus-7000]
nexus[01:04].companydomain.com
```

The pattern above corresponds with the following bunch of hosts:

```
nexus01.companydomain.com
nexus02.companydomain.com
nexus03.companydomain.com
nexus04.companydomain.com
```

AWX

AWX stores the inventory in the backend database. To create an inventory list, use the option “Inventories” and click Add and name the Inventory list and save. Click the Add Host button to add a host, the host name can be a DNS resolvable name or an IP address.

Create a new inventory by navigating to Inventories tab and clicking on .

The screenshot shows the AWX web interface with the 'INVENTORY' page selected. The top navigation bar has several tabs: 'HOSTS' (highlighted with a green box), 'BRANCHES', 'VERSIONS', 'ROLES', and 'CLOUD PROVIDERS'. Below the tabs, there are sections for 'INSTANCES' (with a count of 17,000) and 'UNMANAGED'. There are also fields for 'INSTANCE CREDENTIAL' and 'INSTANCE GROUPS'. At the bottom, there are 'CREATE' and 'EDIT' buttons.

Once the new inventory is saved , click on the “Hosts” tab and click to add new hosts.

The screenshot shows the AWX web interface with the 'HOSTS' page selected. The top navigation bar has tabs: 'HOSTS' (highlighted with a green box), 'BRANCHES', 'VERSIONS', 'GROUPS', 'ROLES', 'SERVICES', and 'CONNECTED DEVS'. Below the tabs, there is a search bar and a 'RUN COMMANDS' button. The main area displays a table with columns 'HOSTS' and 'RELATED GROUPS'. One host entry is visible: '172.29.1.9'. At the bottom, there are 'CREATE' and 'EDIT' buttons.

Bulk Import

AWX offers a utility called “awx-manage” to perform several of operations via CLI . One of the most important features is bulk hosts import.

Login to the awx task container and execute the following command to import hosts to inventory:

You can list the containers that are running by issuing the following command:

```
# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
PORTS					
28143772a881	ansible/awx_task:2.1.2	"sh ..."	19 minutes ago	Up 19 minutes	/tini -- /bin/tcp
ded05efa006d	ansible/awx_web:2.1.2	"sh ..."	19 minutes ago	Up 19 minutes	/tini -- /bin/tcp

Use the command "docker exec -it <container ID> /bin/bash" to get a bash shell in the container:

```
[root@RedhatNewAnsible projects]# docker exec -it ded05efa006d /bin/
bash
[root@awxweb awx]# ls
awxfifo favicon.ico projects public supervisord.log supervisord.pid
venv wsgi.py
[root@awxweb awx]# exit
exit
[root@RedhatNewAnsible projects]#
```

Create a file which contains the list of hosts to add:

```
[root@awx awx]# more customer_15_vgs
172.29.240.20
172.29.240.21
172.29.240.22
172.29.240.23
172.29.240.24
```

Once the list is ready in the file, identify the inventory name in which you need to add the hosts from AWX GUI. In the example, I am adding the hosts to “Customer 15 VGs” inventory.

```
[root@awx awx]# awx-manage inventory_import --inventory-name 'Customer
15 VGs' --source customer_15_vgs
2.326 INFO Updating inventory 2: Customer 15 VGs
2.496 INFO Reading Ansible inventory source: /var/lib/awx/
customer_15_vgs
3.532 INFO Processing JSON output...
3.533 INFO Loaded 0 groups, 5 hosts
2018-12-17 21:57:24,015 DEBUG awx.main.models.inventory Going to
update inventory computed fields
2018-12-17 21:57:24,114 DEBUG awx.main.models.inventory Finished
updating inventory computed fields
3.702 INFO Inventory import completed for (Customer 15 VGs - 12)
in 1.4s
```

The imported hosts are displayed now in AWX GUI.

Dynamic Inventory

Please refer to the Ansible documentation: https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html

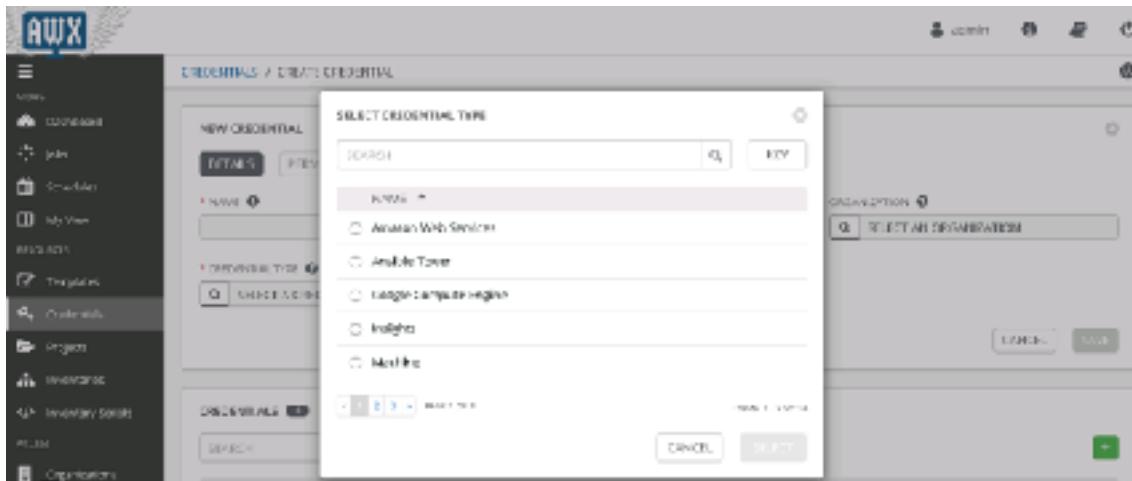
vmware

https://docs.ansible.com/ansible/latest/scenario_guides/vmware_scenarios/vmware_inventory.html

Credentials

For AWX, click on Credentials tab and add button

A bunch of credentials type are supported such as AWS, Machine, Google Cloud, Azure, Network, VMware, Vault, etc.



Playbooks

Playbooks are the heart of Ansible. They can contain other playbooks, roles and/or tasks.

```

- hosts: webservers
  become: yes
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: latest
      notify: restart apache

    - name: Create an index
      file:
        state: touch
        path: /var/www/html/index.html

    - name: Add line to index if not present
      lineinfile:
        state: present
        path: /var/www/html/index.html
        line: '<html>My hostname is {{ansible_hostname}}</html>'
```

Let's review the code line by line.

```
---
```

This line starts every playbook. The three dashes tell the interpreter that this is a YAML document.

```
- hosts: webservers
```

At this point starts the first play in the playbook. Each play is defined in a YAML as a list entry and has to have at least two keys, hosts and tasks (or roles).

These entries can be in any order as long as they are present. The hosts entry can either be a individual host in the inventory or an inventory group.

```
become: yes
```

This line will have the playbook run as root.

```
tasks:
```

```
- name: Install Apache
```

```
apt:
```

```
  name: apache2
```

```
  state: latest
```

```
  notify: restart apache
```

The tasks section contains all of the tasks that will run on the hosts defined earlier. Each separate task is a list entry under this section and contain one or more modules.

```
- name: Add line to index if not present
```

```
lineinfile:
```

```
  state: present
```

```
  path: /var/www/html/index.html
```

```
  line: '<html>My hostname is {{ansible_hostname}}</html>'
```

This is an example of a variable ({{ansible_hostname}}). Variables can be set by a variety of sources including the command line and the inventory file.

This particular variable will be set when Ansible runs through the fact gathering phase of the host, which provides the playbook run with a wide range of information that you can use in your playbooks.

Finally, the playbook uses handlers. The line "notify: restart apache" in the first task triggers this handler if there was a change that was required to be made by that task. This means if Apache wasn't installed or a new version was found the handler will be run and apache will be restarted.

Running Playbooks Deep Dive

Ansible Engine

```
#ansible-playbook -i /path/inventory_file /path/playbook.yaml
```

```
[root@centos-7-aws ~]# ansible-playbook -i localhost localhost.yml
[WARNING]: No inventory was parsed, only localhost is available
[WARNING]: provided plays file is empty, only localhost is available. Note that the localhost inventory does not match 'all'
PLAY [localhost]
  TASK [Gathering Facts]
    ok: [localhost]

  TASK [Set timezone to Europe/Madrid]
    changed: [localhost]

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

One of the first options anyone picks up is the debug option. To understand what is happening when you run the playbook, you can run it with the verbose (-v) option. Every extra v will provide the end user with more debug output.

```
[root@centos-7-aws ~]# ansible-playbook -i localhost localhost.yml -v
[WARNING]: No inventory was parsed, only localhost is available
[WARNING]: provided plays file is empty, only localhost is available
[WARNING]: enviroinfo facts that is empty, only localhost is available. Note that the localhost factdict doesn't match 'all'

PLAY [localhost]
  TASK [Set timezone to Europe/Madrid]
    changed: [localhost] => {"changed": true, "msg": "Fact 'tz' has been set to 'Europe/Madrid'"}

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

```

(base) [root@centos-7-aws ~]# ansible-playbook -i hosts/base/library.yml
[WARNING]: No inventory was parsed, only localhost is available. Note that the implicit localhost does not match 'all'
PLAYBOOK: base_library.yml
  1 play(s) in 1 hosts(s)

PLAY [localhost]
  HETC: main handlers
    (base) [root@centos-7-aws ~]# echo $?
    0
    base_main: >>> /usr/local/lib/python2.7/dist-packages/ansible/module_utils/basic.py
    changed: [localhost] => {"changed": true, "msg": "Test command 'echo $?' successful", "rc": 0}
    HETC: main handlers
    HETC: main handlers
  PLAY RECAP
  localhost      : ok=0   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
(base) [root@centos-7-aws ~]#

```

```

(base) [root@centos-7-aws ~]# ansible-playbook -i hosts/base/library2.yml
[WARNING]: No inventory was parsed, only localhost is available. Note that the implicit localhost does not match 'all'
PLAY [localhost]
  HETC: main handlers
    (base) [root@centos-7-aws ~]# echo $?
    0
    base_main: >>> /usr/local/lib/python2.7/dist-packages/ansible/module_utils/basic.py
    changed: [localhost] => {"changed": true, "msg": "Test command 'echo $?' successful", "rc": 0}
    HETC: main handlers
    HETC: main handlers
  PLAY RECAP
  localhost      : ok=0   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
(base) [root@centos-7-aws ~]#

```

If you run the playbook again, the execution is successful but no changes are made (changed=0).

```

(base) [root@centos-7-aws ~]# ansible-playbook -i hosts/base/library3.yml
[WARNING]: No inventory was parsed, only localhost is available.
[WARNING]: previous hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
PLAY [localhost]
  HETC: main handlers
  HETC: main handlers
  PLAY RECAP
  localhost      : ok=0   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
(base) [root@centos-7-aws ~]#

```

Variables

You define variable in files as mentioned above when using ansible-vault and in the vars section of a play, where you are defining the variable for the set of hosts in the play.

```
(localhost:~/base/library$ ansible-playbook variables.yaml
[WARNING]: No inventory was parsed, falls back to Ansible's internal hostlist.
[WARNING]: No inventory was provided, therefore falling back to Ansible's internal hostlist.
localhost:~/base/library$
```

```
services:
  - name: httpd
    port: 80
    state: started
    type: http
  - name: https
    port: 443
    state: started
    type: https
  - name: ssh
    port: 22
    state: started
    type: ssh
```

This playbook just fetches the variables from the file and displays the output.

```
(localhost:~/base/library$ ansible-playbook variables.yaml --check
[WARNING]: No inventory was parsed, falls back to Ansible's internal hostlist.
[WARNING]: No inventory was provided, falls back to Ansible's internal hostlist.
localhost:~/base/library$
```

```
PLAY [localhost]
ANSI [localhost]
ok: [localhost]

PLAY [localhost]
ANSI [localhost]
ok: [localhost]

PLAY [localhost]
ANSI [localhost]
ok: [localhost]
```

```
[WARNING]: The value "[{'name': 'httpd', 'port': 80}, {'name': 'https', 'port': 443}, {'name': 'ssh', 'port': 22}]" in a string field was converted to an integer. This might be a bug. If this does not look like what you expect, write the entire value to ensure it does not change.

changed: [localhost]
PLAY RECAP
localhost      : ok=3   changed=0   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0
```

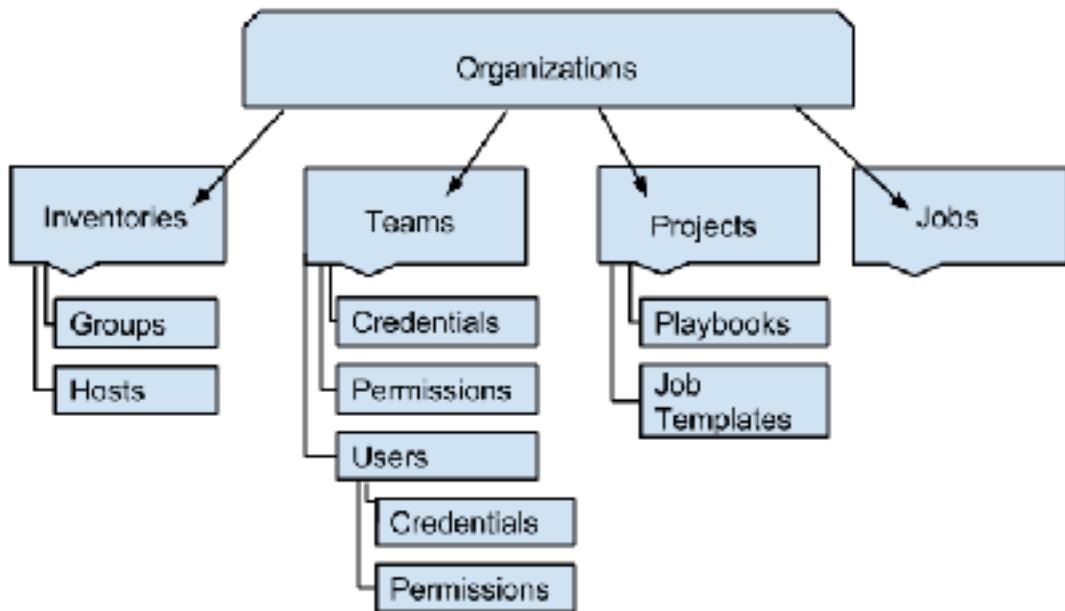
```
(localhost:~/base/library$ ls
variables.yaml  file1.log  file2.log  file3.log  file4.log  file5.log  file6.log  file7.log  file8.log  file9.log
(localhost:~/base/library$ rm variables.yaml
(localhost:~/base/library$ ls
file1.log  file2.log  file3.log  file4.log  file5.log  file6.log  file7.log  file8.log  file9.log
(localhost:~/base/library$)
```

AWX

Let's introduce some concepts before getting deeper on how to run playbooks with AWX.

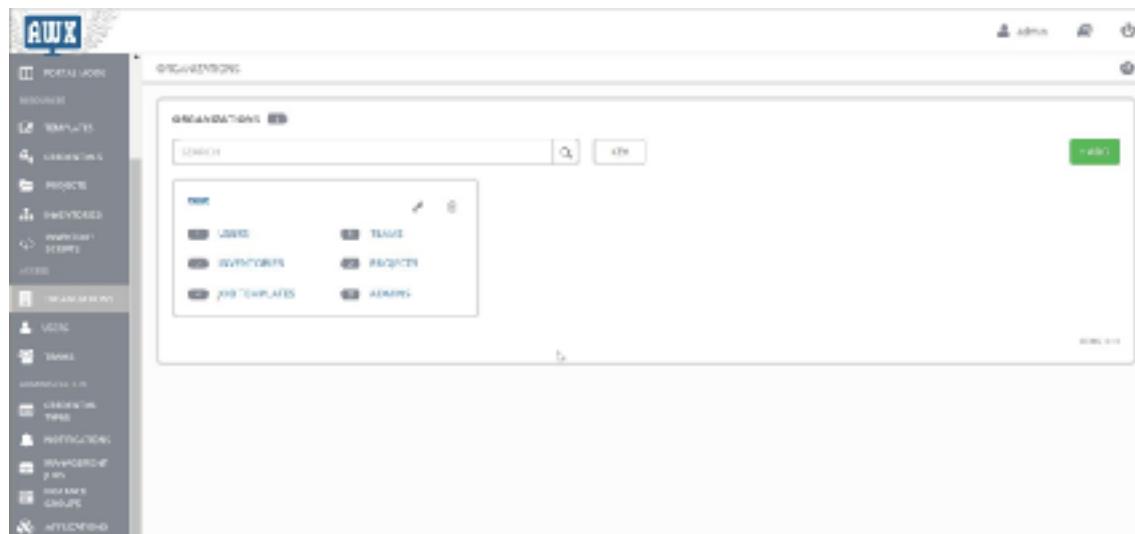
Organizations

Ansible AWX offers multilevel access delegation and role-based access control to the organization. An organization is a logical collection of Users, Teams, Projects, and Inventories. It is the highest level in the AWX object hierarchy. To support multiple clients from one umbrella, you could create an organization for each client and manage multiple teams under that. The picture below depicts the AWX organization hierarchy:



In order to create a new organization, login to AWX console with admin privileges.

Select “Organization” from the navigation and click on add . Enter the organization name and click on Save. Selecting the instance group (requires clustering) might be useful on large deployment.

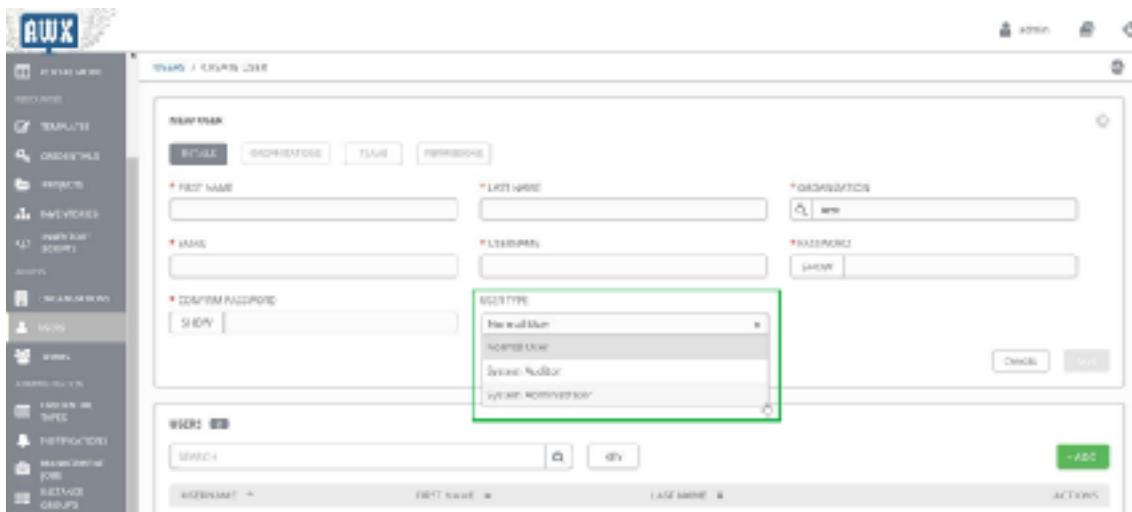


In order to create a new team, navigate to “Teams” from the menu. Click on to add a new team. Enter the team name and select the organization (you can add multiple teams based on the requirements).

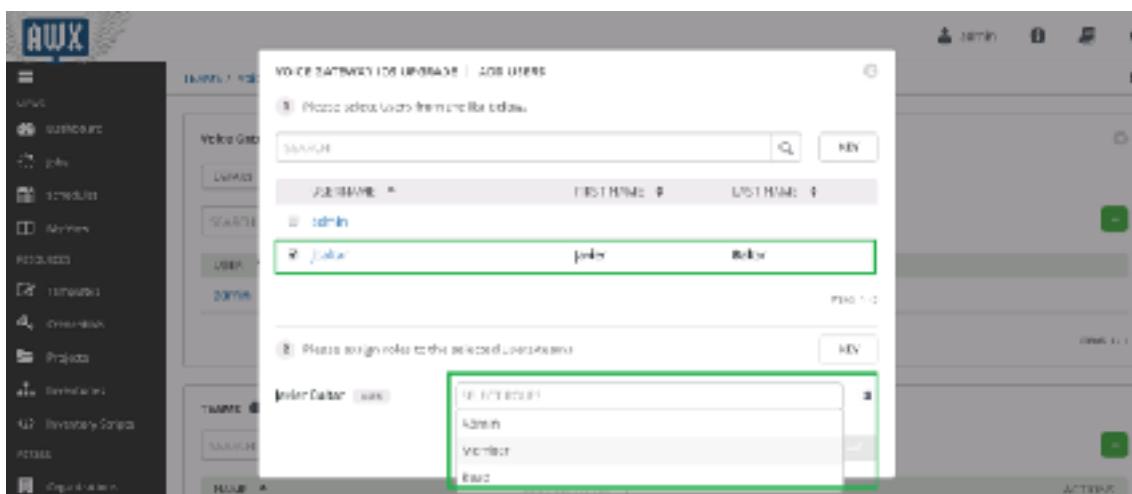
User Management

From the AWX main menu, you can add users to current organization. Select “users” and click on . Enter the user details and select the destination organization.

There are three user types: Normal User, System Auditor and System Administrator as shown below:



From the Teams menu, you can add users to teams assigning the proper role (admin, member or read only).



Project

Next, you configure a project, the project contains the SCM type, and playbook directory which you must select. The SCM type can be manual but other options such as Git are available for central management of scripts.

The playbook directory drop down should contain the sub directories in the root projects folder. Select the relevant folder for this project that contains all the playbooks required for jobs connected to this project.

The screenshot shows the AWX interface with the sidebar navigation open. Under the 'PROJECTS' section, the 'PROJECTS' option is selected. A modal window titled 'PROJECTS / CREATE PROJECT' is open, showing the 'NEW PROJECT' form. The 'NAME' field contains 'awx', the 'DESCRIPTION' field contains 'go to the moon', and the 'DESCRIPTION' field contains 'test'. The 'New Type' dropdown is set to 'Choose an SCM Type' and has 'Local Repo' selected. Below the form is a table listing existing projects: 'Basic Project' and 'awx'. The 'awx' project is highlighted.

Manual Local Repo

You can create the project folder to store the playbooks in the following local path:

```
cd /var/lib/awx/projects/
mkdir my_project_name
```

The screenshot shows the AWX interface with the sidebar navigation open. Under the 'PROJECTS' section, the 'PROJECTS' option is selected. A modal window titled 'PROJECTS / Local Repo' is open, showing the 'Local Repo' configuration form. The 'NAME' field contains 'Local Repo', the 'DESCRIPTION' field contains 'PROJECTS PATH', and the 'ORGANIZATIONS' field contains 'Default'. The 'SCM TYPE' dropdown is set to 'Manual' and has 'Ansible' selected. The 'REPO URL' dropdown is set to 'choose a playbook directory' and has 'local_repo' selected. The 'PROJECTS' table at the bottom shows a single entry: 'local_repo'.

Git Repo

A external Git repository can be configured at Project level as shown below:

The screenshot shows the AWX web interface with the 'Projects / HESTI' page selected. The 'HESTI' project is detailed, showing its name, description, organization, and repository URL. The 'Sync Now' button is highlighted with a green box.

The Projects tab displays the status of the repo, last revision and the sync button.

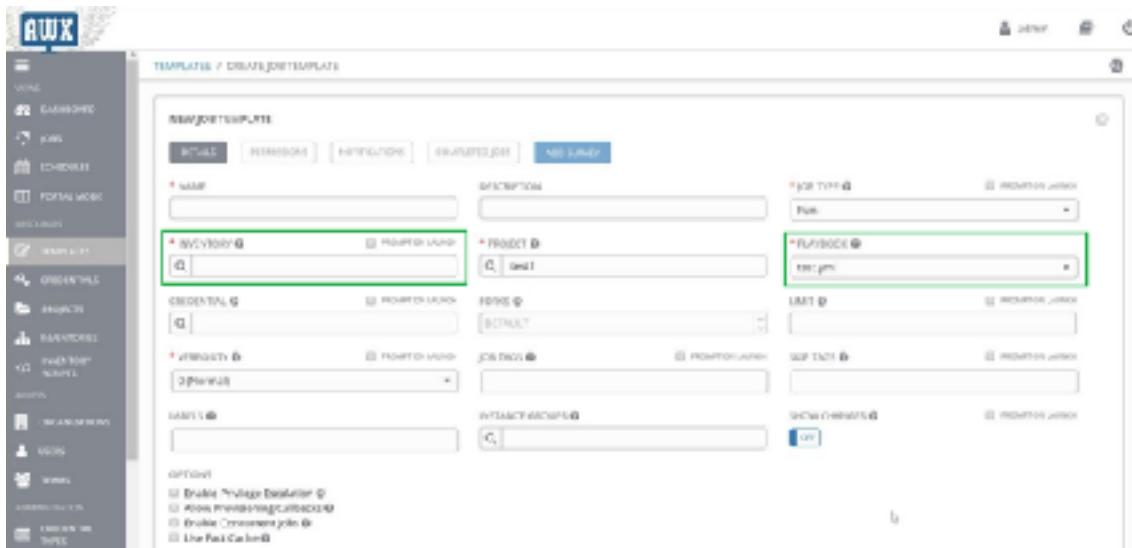
The screenshot shows the AWX web interface with the 'PROJECTS' page selected. It lists two projects: 'HESTI' and 'Level 10'. Each project card shows its name, organization, last modified date, and last sync date. The 'HESTI' project card is highlighted with a green box.

Templates

Job Template

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times and reuse of Ansible playbook content between teams.

To create a new job template, click the add button then select "Job Template" from the menu list.

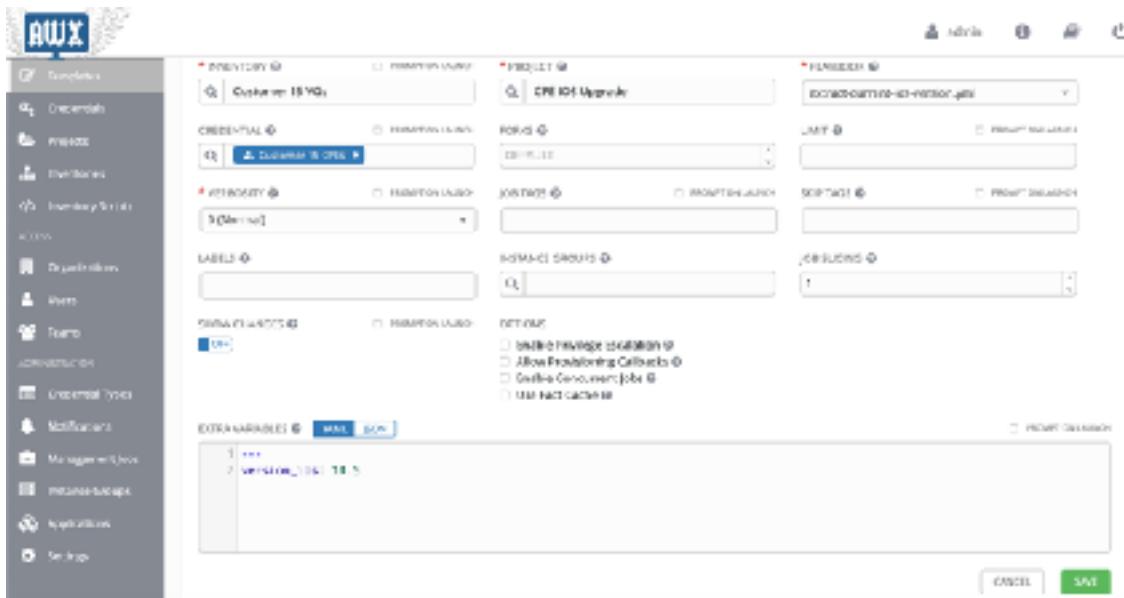


Enter the appropriate details into the following fields:

- Name: Enter a name for the job.
- Description: Enter an arbitrary description as appropriate (optional).
- Job Type:
 - Run: Execute the playbook when launched, running Ansible tasks on the selected hosts.
 - Check: Perform a “dry run” of the playbook and report changes that would be made without actually making them. Tasks that do not support check mode will be skipped and will not report potential changes.
 - Prompt on Launch – If selected, even if a default value is supplied, you will be prompted upon launch to choose a job type of run or check.
- Inventory: Choose the inventory to be used with this job template from the inventories available to the currently logged in AWX user.
 - Prompt on Launch – If selected, even if a default value is supplied, you will be prompted upon launch to choose an inventory to run this job template against.
- Project: Choose the project to be used with this job template from the projects available to the currently logged in AWX user.
- Playbook: Choose the playbook to be launched with this job template from the available playbooks. This menu is automatically populated with the names of the playbooks found in the project base path for the selected project.
- Credential: Click the search button to open a separate window. Choose the credential from the available options to be used with this job template. Use the drop-down menu list to filter by credential type if the list is extensive.
 - Prompt on Launch: If selected, upon launching a job template that has a default machine credential, you will not be able to remove the default machine credential in the Prompt dialog without replacing it with another machine credential before it can launch.
- Forks: The number of parallel or simultaneous processes to use while executing the playbook. A value of zero uses the Ansible default setting, which is 5 parallel processes unless overridden in /etc/ansible/ansible.cfg.
- Limit: A host pattern to further constrain the list of hosts managed or affected by the playbook. Multiple patterns can be separated by colons (:). As with Ansible engine, “a:b” means “in group a or b”, “a:b:&c” means “in a or b but must be in c”, and “a!:b” means “in a, and definitely not in b”.

- Prompt on Launch: If selected, even if a default value is supplied, you will be prompted upon launch to choose a limit.
- Verbose: Control the level of output Ansible produces as the playbook executes. Set the verbosity to any of Default, Verbose, or Debug. This only appears in the “details” report view. Verbose logging includes the output of all commands. Debug logging is exceedingly verbose and includes information on SSH operations that can be useful in certain support instances.
 - Prompt on Launch: If selected, even if a default value is supplied, you will be prompted upon launch to choose a verbosity.
- Job Tags: Provide a comma-separated list of playbook tags to specify what parts of the playbooks should be executed.
 - Prompt on Launch – If selected, even if a default value is supplied, you will be prompted upon launch to choose a job tag.
- Skip Tags: Provide a comma-separated list of playbook tags to skip certain tasks or parts of the playbooks to be executed.
 - Prompt on Launch – If selected, even if a default value is supplied, you will be prompted upon launch to choose tag(s) to skip.
- Labels: Supply optional labels that describe this job template, such as “dev” or “test”. Labels can be used to group and filter job templates and completed jobs in the AWX display.
 - Labels are created when they are added to the Job Template. Labels are associated to a single Organization using the Project that is provided in the Job Template. Members of the Organization can create labels on a Job Template if they have edit permissions (such as admin role).
 - Once the Job Template is saved, the labels appear in the Job Templates overview.
 - Click on the “x” beside a label to remove it. When a label is removed, and is no longer associated with a Job or Job Template, the label is permanently deleted from the list of Organization labels.
 - Jobs inherit labels from the Job Template at the time of launch. If a label is deleted from a Job Template, it is also deleted from the Job.
- Instance Groups: Click the search button to open a separate window. Choose the instance groups on which you want to run this job template. If the list is extensive, use the search to narrow the options.
- Show Changes: Allows you to see the changes made by Ansible tasks.
 - Prompt on Launch – If selected, even if a default value is supplied, you will be prompted upon launch to choose whether or not to show changes.
- Options: Supply optional labels that describe this job template, such as “dev” or “test”. Labels can be used to group and filter job templates and completed jobs in the AWX display.
 - Enable Privilege Escalation: If enabled, run this playbook as an administrator. This is the equivalent of passing the --become option to the ansible-playbook command.
 - Allow Provisioning Callbacks: Enable a host to call back to AWX via the AWX API and invoke the launch of a job from this job template.
 - Enable Concurrent Jobs: Allow jobs in the queue to run simultaneously if not dependent on one another.
 - Use Fact Cache: When enabled, AWX will activate an Ansible fact cache plugin for all hosts in an inventory related to the job running.
- Extra Variables:
 - Pass extra command line variables to the playbook. This is the “-e” or “–extra-vars” command line parameter for ansible-playbook that is documented in the Ansible documentation at Passing Variables on the Command Line.

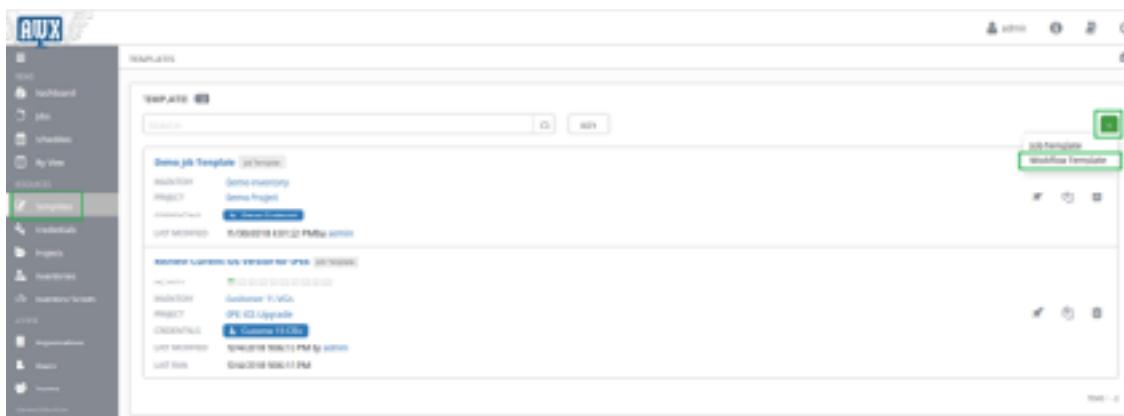
- Provide key/value pairs using either YAML or JSON. These variables have a maximum value of precedence and overrides other variables specified elsewhere.



Workflow Template

This feature in AWX enables users to create sequences consisting of any combination of job templates, project syncs, and inventory syncs that are linked together in order to execute them as a single unit. Another reason workflows are useful is because they allow the user to take any number of playbooks with the ability to make a decision tree depending on a job's success or failure and sending notifications also.

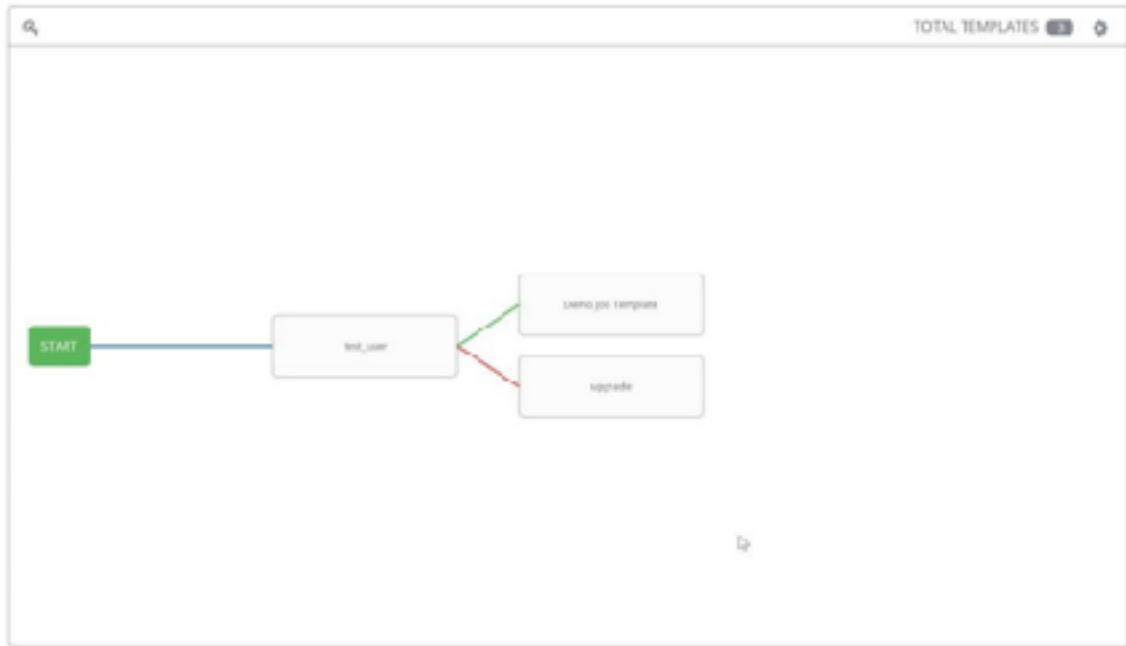
From the Templates menu, click on and make sure to select “Workflow Template”.



Once the mandatory fields are filled, click on Save and on Workflow Visualizer:

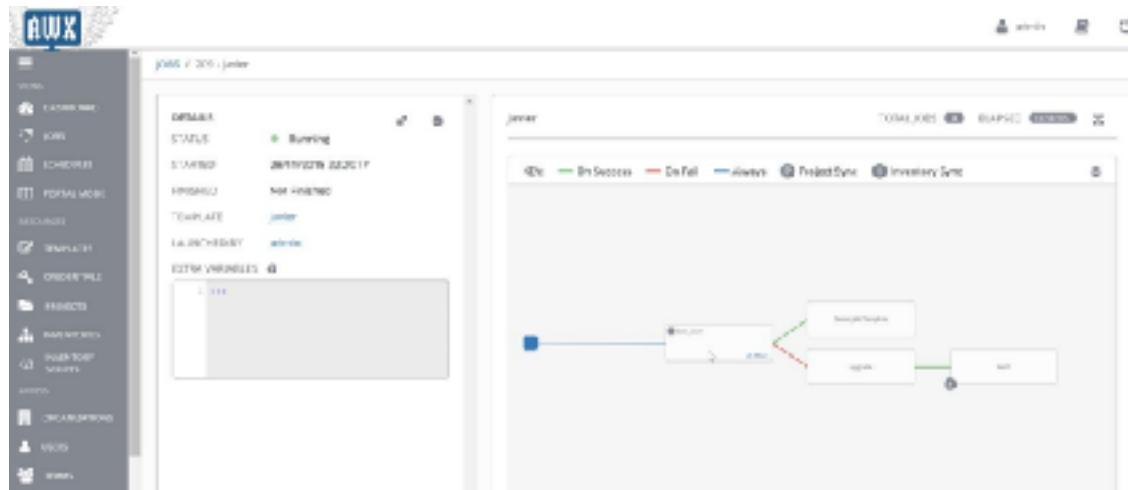


This screen will come up, where you can add different job templates and make sure they run on failure, success, or with either outcome.



Note that you can decide if things run on success, on failure, or always.

After everything is set and saved, you are ready to launch your template, which you can do by clicking on the rocket icon next to the workflow you would like to run.



You can schedule your workflows to run when you need them to. Click on the calendar icon next to any workflow job template:

CREATE SCHEDULE

NAME Schedule name:	STARTDATE 10/06/2018	START TIME 10:00:00
LOCATION , time zone: Europe/London	REPEAT FREQUENCY None (in weeks)	

CANCEL **SAVE**

If you need to set extra variables for the playbooks involved in a workflow template and/or allow for authorization of user input, then setting up surveys is the way to go.

In order to set one up, select a workflow template and click on the “Add Survey” button:

Survey Item | SURVEY [1]

ADD SURVEY ITEM

QUESTION

PLEASE ADD A SURVEY ITEM.

DESCRIPTION

ANSWER LABEL

ANSWER TYPE

IS REQUIRED

SUMMARY

DETAILS

Notifications

AWX provides a bunch of notification channels: email, Slack, Twilio, etc.

NEW NOTIFICATION TEMPLATE

NAME:

DESCRIPTION:

ORGANIZATION:

TYPE: Choose a type

Choose a type

- Email
- Slack
- Twilio
- PagerDuty
- HipChat

SAME	TYPE	ACTIONS
Twilio	Twilio	
Workmail	Email	

Email

Click on Notification and add a new one:

NAME: WORKMAIL

DESCRIPTION:

ORGANIZATION:

TYPE: Email

TYPE DETAILS:

TO: WORKMAIL

FROM: WORKMAIL

SUBJECT: WORKMAIL

BODY: WORKMAIL

OPTIONS:

USE SSL

USE TLS

From the Templates tab, you can control the notifications that are triggered by an specific template when it is successfully completed or fails.

The screenshot shows the AWX interface with the URL [https://127.0.0.1:8443/awx/api/job_template/1/](#). The left sidebar includes options like 'Dashboard', 'Jobs', 'Schedules', 'My Now', 'Template', 'Credentials', 'Projects', 'Inventory', 'Inventory Keys', 'Organizations', 'Users', 'Teams', 'Administrative', and 'Change types'. The main content area is titled 'Basic Job Template' and contains tabs for 'EMAIL', 'PERMISSIONS', 'NOTIFICATIONS' (which is selected), 'DOCUMENTATION', and 'SCHEDULES'. Under 'NOTIFICATIONS', there is a 'NAME' field with 'Twilio' and a 'DESCRIPTION' field with 'Twilio'. Below these are fields for 'TWILIO FAX' (set to 'OFF') and 'TWILIO SMS' (set to 'ON'). A 'NOTIFICATION' section at the bottom right indicates '0 TO NOTIFICATION' and '0 PENDING' notifications. Below this is a 'SAMPLES' section with a search bar and a list item 'Demo Job Template (Job Template)'.

This is an example of email notification received to your mailbox.

Twilio

To trigger a Twilio API call, you need to configure the following items in your account:

Click on dashboard > Twilio account details, where Account SID and AUTH TOKEN are listed.

Click on Twilio verified called IDs, which contains the allowed destination numbers.

Click on Geo Restrictions in order to ensure your destination number is not restricted:

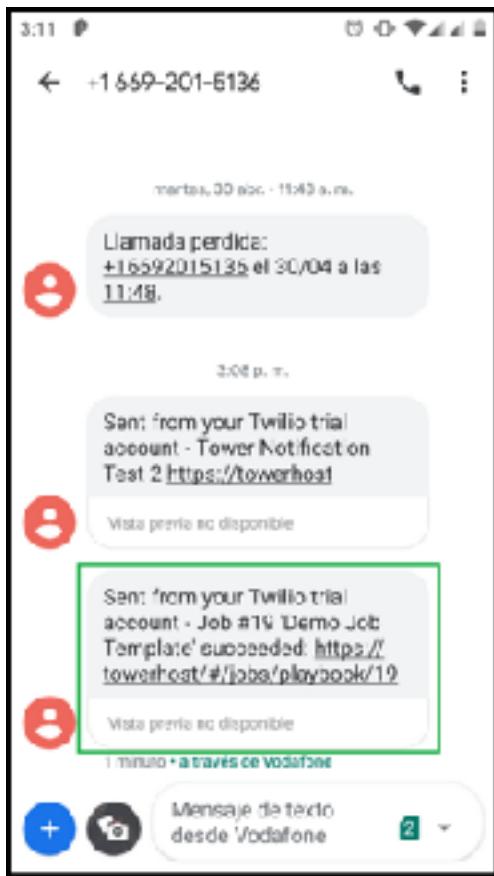
You have to create your Twilio notification in AWX:

The screenshot shows the AWX interface with the URL [https://127.0.0.1:8443/awx/api/notification/1/](#). The left sidebar includes options like 'Dashboard', 'Jobs', 'Schedules', 'My Now', 'Template', 'Credentials', 'Projects', 'Inventory', 'Inventory Keys', 'Organizations', and 'Users'. The main content area is titled 'Notifications / Twilio Jailer' and contains a form for creating a 'Twilio Jailer'. The form fields include: 'NAME' (Twilio Jailer), 'DESCRIPTION' (empty), 'ORGANIZATION' (Cofair), 'TYPE' (Twilio), 'TWILIO FAX' (OFF), 'TWILIO SMS' (ON), 'ACCOUNT TOKEN' (SHOW *****) (Account SID: ACd818169791636f2cc3e98d8a77), 'SOURCEPHONENUMBER' (-162975135), 'DESTINATIONSMSNUMBER' (+346358049), and 'ACCOUNTSID' (ACd818169791636f2cc3e98d8a77). At the bottom are 'CANCEL' and 'CREATE' buttons.

Associate your template with the notification:

The screenshot shows the Ansible AWX web interface. On the left, there's a sidebar with navigation links like 'Dashboard', 'Jobs', 'Shows', 'My Now', 'Templates', 'Calendars', 'Projects', and 'Invoices'. The main area is titled 'TIMEOUTS / Demo Job Template / NOTIFICATIONS'. It displays a 'Basic Job Template' configuration with tabs for 'EMAIL', 'SMS', 'WEBHOOK', 'DYNAMIC JOBS', and 'SCHEDULED'. Under the 'SMS' tab, there's a table with two rows: 'NAME' (19891967) and 'TYPE' (WHL), and another row for 'Work-email' (Email). Buttons for 'OK' and 'CANC' are visible. A link 'AD TO NOTIFICATION JOB ANNOTATION' is at the top right.

After running your job, the SMS notification below is dropped to my mobile:



Advanced Playbooks Features

Error Handling

Blocks allow you to group related tasks together and apply particular task parameters on the block level. They also allow you to handle errors inside the blocks in a way similar to most programming languages exception handling.

This example playbook that uses blocks to run group of tasks specific to one platform. If you want to perform a series of tasks with one set of task parameters (e.g. with_items, when, or sudo) applied, blocks are very handy.

Blocks are also useful if you want to be able to gracefully handle failures. There might be a non-critical task that is not essential for a deployment to succeed, so it would be better to gracefully handle a failure than to stop the entire deployment rollout.

This is an example of how to use a block to gracefully handle task failures:

```
tasks:
  - block:
      - name: Shell script to deploy a critical service.
        script: deploy_application.sh
    rescue:
      - name: This will only run in case of an error in the block.
        debug: msg="There was an error in the block."
    always:
      - name: This will always run
        debug: msg="This always executes."
```

Tasks inside the block will be run first. If there is a failure in any task in block, tasks inside rescue will be run. The tasks inside always will always be run, whether or not there were failures in either block or rescue.

Please refer to Ansible documentation: https://docs.ansible.com/ansible/latest/user_guide/playbooks_blocks.html#error-handling

Handlers

A Handler is exactly the same as a task, but it will run when called by another task. A Handler will take an action when called by an event it listens for.

This is useful for secondary actions that might be required after running a task, such as starting a new service after installation or reloading a service after a configuration change.

```
---
- hosts: webservers
  tasks:
    - name: Install Nginx
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start Nginx
  handlers:
    - name: start Nginx
      service: name=nginx state=started
```

Templates and Variables

A template in Ansible is a file which contains all your configuration parameters, but the dynamic values are given as variables. During the playbook execution, depending on the conditions like which cluster you are using, the variables will be replaced with the relevant values.

```
- hosts: webservers
  vars:
    variable_to_be_replaced: 'value1'
    inline_variable: 'value2'
  tasks:
    - name: Ansible using templates
      template:
        src: template_example.j2
        dest: /userA/docs/personal_data.txt
```

Template file

```
template_example.j2
{{ variable_to_be_replaced }}
Variable given as inline - {{ inline_variable }} -
```

This is the output of the playbook above:

```
value1
Variable given as inline - value2 -
```

Loops

With_items

```
- name: Remove users from the system.
  user:
    name: "{{ item }}"
    state: absent
    remove: yes
  with_items:
    - userA
    - userB
```

With_nested
Define variables

```
users_with_items:
  - name: "userA"
    personal_directories:
      - "old_files"
```

Playbook

```
- name: Create common users directories using
  file:
    dest: "/home/{{ item.0.name }}/{{ item.1 }}"
    owner: "{{ item.0.name }}"
    group: "{{ item.0.name }}"
    state: directory
  with_nested:
    - "{{ users_with_items }}"
    - "{{ common_directories }}"
```

File Manipulation

Useful ad-hoc commands for file manipulation:

Delete a File

```
#Delete the file /backups/tmp/nodelist.txt on all servers
ansible all -b -m file -a "state=absent path=/backups/tmp/nodelist.txt"
```

Update a Line in File

```
#Update the line of text "MY_SETTING" to "BLUE" in /opt/configuration.txt
on all servers
ansible all -b -m lineinfile -a "regexp=MY_SETTING line=BLUE path=/opt/
configuration.txt"
```

Extract /tmp/package.tgz to /opt/ on all hosts in webservers

```
ansible webservers -b -m unarchive -a "src=/tmp/package.tgz dest=/opt/
remote_src=yes"
```

Set the group ownership of a directory on each host in webservers

```
ansible webservers -b -m file -a "recurse=yes state=directory path=/opt/
automators/secrets group=protected"
```

Modules

Please refer to https://docs.ansible.com/ansible/latest/modules/modules_by_category.html for Ansible module index.

Files

Archive

Some examples using the archive modules:

tasks:

```
- name: Backup Directory /var/log/application01/
- archive:
  path: /var/log/application01/
  dest: "/var/backups/application01-{{ ansible_date_time.date }}.tgz"
```

Copy

tasks:

```
- name: Copy File
- copy:
  src: /var/log/application01/filename
  dest: "/var/backups/filename"
  owner: root
  group: root
  mode: u=r,g=r,o=
```

Fetch

tasks:

```
- name: Copy File from Remote Node
- fetch:
  src: /var/log/application01/filename
  dest: "/var/backups/
```

Lineinfile

The following playbook add a line after a string:

The following playbook deletes a line from a file:

```

C:\Program Files\Tortoise\TortoisePebble\Library\test
C:\Program Files\Tortoise\TortoisePebble\Library\test
Listed 00
Listed 3000
requirementstatus: no
Inventory: no
C:\Program Files\Tortoise\Tortoise\Library\0
C:\Program Files\Tortoise\Tortoise\Library\0
C:\Program Files\Tortoise\Tortoise\Library\0 more files listed (use -l to get all)
- status: localhost
  tasks:
    - name: remove file - PBK File
      tasktype:
        path: C:\Program Files\Tortoise\Tortoise\Library\0
        replace: "My requirement status"
        action: absent
  (C:\Program Files\Tortoise\Tortoise\Library\0
  C:\Program Files\Tortoise\Tortoise\Library\0
  C:\Program Files\Tortoise\Tortoise\Library\0 enable-pebble.turtle (use -e to get all)
  [WARNING] No inventory was present, only implicit localhost is available

[WARNING] www нет на localhost, используйте --use-host или --use-localhost
Pebble [14448960]
-----
```

DATA [Gathering facts]

ok [localhost]

TASK [Move file from 'src' to 'dest']

changed [localhost]

Pebble [14448960]

localhost	used	changed	unreachable	failed	skipped	rescued	ignorable

C:\Program Files\Tortoise\Tortoise\Library\0

Listed 00
Listed 3000
Starting 00
C:\Program Files\Tortoise\Tortoise\Library\0

Replace

This is an example of playbook using the replace module.

Template

Example playbook using the template module.

Yum

Example playbook using the yum module for installing https and mariadb packages.

```
root@centos-7-distro:~/home/playbooks/library (107)
[root@centos-7-distro:~/home/playbooks/library (107)]# ./test-ansible-test
[DEPRECATION WARNING]: you're using the 'installed' module Ansible's 2.8 will "Installed" | echo "Not Installed"
ok: [localhost]
  tasks:
    - name: ansible package install test
      yum:
        name: "{{ item }}"
        state: present
      with_items:
        - httpd
        - curl
[DEPRECATION WARNING]: you're using the 'library' module Ansible's 2.8 will "library" | echo "Not Installed"
[DEPRECATION WARNING]: The inventory was passed, only 'localhost' is available
[DEPRECATION WARNING]: provided hosts that is empty, only localhost is available. Note that the user 'root' does not match 'all'.
PLAY [localhost]
ok: [localhost]
  tasks:
    - name: [deleting facts]
      ok: [localhost]
    - name: [installing packages]
      yum:
        name: "{{ item }}"
        state: present
      with_items:
        - httpd
        - curl
      changed: [localhost] > [item='httpd', state='present']
[DEPRECATION WARNING]: running 'yum' only once while using a loop via 'with_items' is deprecated. Instead of using a loop to supply multiple items and specifying 'name: "{{ item }}", please use 'name: 'changed', 'state='present'' and reuse the loop. This feature will be removed in version 2.11. Deprecation warnings can be disabled by setting
deprecation_warnings_in_user_code=yes.
changed: [localhost] > [item='httpd', state='present']

PLAY RECAP
localhost:               100%  0m0s  0m0s  0m0s  0m0s  0m0s  0m0s
[root@centos-7-distro:~/home/playbooks/library (107)]#
```

Command

Example playbook using the raw command module.

```
root@centos-7-distro:~/home/playbooks/library (108)
[root@centos-7-distro:~/home/playbooks/library (108)]# ./test-ansible-test
[DEPRECATION WARNING]: more command test
ok: [localhost]
  tasks:
    - name: execute a command
      raw:
        cmd: /usr/bin/ls -l
        register: raw_ls_out
      changed: true
      register: raw_ls_out
[DEPRECATION WARNING]: you're using the 'library' module Ansible's 2.8 will "library" | echo "Not Installed"
[DEPRECATION WARNING]: The inventory was passed, only 'localhost' is available
[DEPRECATION WARNING]: provided hosts that is empty, only localhost is available. Note that the user 'root' does not match 'all'.

PLAY [localhost]
ok: [localhost]
  tasks:
    - name: [deleting facts]
      ok: [localhost]
    - name: [execute a command]
      raw:
        cmd: /usr/bin/ls -l
        register: raw_ls_out
      changed: true
      register: raw_ls_out
      failed: true
      msg: /bin/sh: 1: /usr/bin/ls: not found
      stderr: /bin/sh: 1: /usr/bin/ls: not found
      stdout: total 0
      rc: 127
      _raw_rc: 127
      _raw_stderr: /bin/sh: 1: /usr/bin/ls: not found
      _raw_stdout: total 0
      _rc: 127
      _stderr: /bin/sh: 1: /usr/bin/ls: not found
      _stdout: total 0
      _failed: true
      _msg: /bin/sh: 1: /usr/bin/ls: not found
      _register: raw_ls_out
      _register_name: raw_ls_out
      _register_value: {u'cmd': u'/usr/bin/ls -l', u'changed': true, u'failed': true, u'msg': u'/bin/sh: 1: /usr/bin/ls: not found', u'rc': 127, u'_raw_rc': 127, u'_raw_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_raw_stdout': u'total 0', u'_rc': 127, u'_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_stdout': u'total 0', u'_failed': true, u'_msg': u'/bin/sh: 1: /usr/bin/ls: not found', u'_register': u'raw_ls_out', u'_register_name': u'raw_ls_out', u'_register_value': {u'cmd': u'/usr/bin/ls -l', u'changed': true, u'failed': true, u'msg': u'/bin/sh: 1: /usr/bin/ls: not found', u'rc': 127, u'_raw_rc': 127, u'_raw_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_raw_stdout': u'total 0', u'_rc': 127, u'_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_stdout': u'total 0'}}
[DEPRECATION WARNING]: more command test
ok: [localhost]
  tasks:
    - name: [execute a command]
      raw:
        cmd: /usr/bin/ls -l
        register: raw_ls_out
      changed: true
      register: raw_ls_out
      failed: true
      msg: /bin/sh: 1: /usr/bin/ls: not found
      stderr: /bin/sh: 1: /usr/bin/ls: not found
      stdout: total 0
      rc: 127
      _raw_rc: 127
      _raw_stderr: /bin/sh: 1: /usr/bin/ls: not found
      _raw_stdout: total 0
      _rc: 127
      _stderr: /bin/sh: 1: /usr/bin/ls: not found
      _stdout: total 0
      _failed: true
      _msg: /bin/sh: 1: /usr/bin/ls: not found
      _register: raw_ls_out
      _register_name: raw_ls_out
      _register_value: {u'cmd': u'/usr/bin/ls -l', u'changed': true, u'failed': true, u'msg': u'/bin/sh: 1: /usr/bin/ls: not found', u'rc': 127, u'_raw_rc': 127, u'_raw_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_raw_stdout': u'total 0', u'_rc': 127, u'_stderr': u'/bin/sh: 1: /usr/bin/ls: not found', u'_stdout': u'total 0'}}

PLAY RECAP
localhost:               100%  0m0s  0m0s  0m0s  0m0s  0m0s  0m0s
[root@centos-7-distro:~/home/playbooks/library (108)]#
```

System

Firewalld

Example playbook using the firewalld module allowing ports 80 and 443.

```
[root@centos-7 ~]# curl -v http://127.0.0.1:8080/test.html
* Rebuilt URL to: http://127.0.0.1:8080/test.html
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
* HTTP request sent, awaiting response... 200 OK
* Recv buffer size: 4096
* [HTTP/1.1 200 OK]
* Content-Type: text/html; charset=UTF-8
* Content-Length: 10
* Date: Mon, 12 Dec 2016 10:30:20 GMT
<!DOCTYPE html>
<html>
<head>
<title>Test</title>
</head>
<body>
<h1>Hello World</h1>
</body>
</html>
```

```
[root@centos6-7-disk1 ~]# rpm -q libcurl curl curl-devel curl-devel-devel
libcurl-7.29.0-10.el7_2.1.1
curl-7.29.0-10.el7_2.1.1
curl-devel-7.29.0-10.el7_2.1.1
curl-devel-devel-7.29.0-10.el7_2.1.1
[root@centos6-7-disk1 ~]# curl --version
curl/7.29.0 (x86_64-redhat-linux-gnu) libcurl/7.29.0 OpenSSL/1.0.2f zlib/1.2.8 libidn2/2.0.0 libssh2/1.8.0 libnghttp2/1.32.0
[root@centos6-7-disk1 ~]# curl -V
curl/7.29.0 (x86_64-redhat-linux-gnu) libcurl/7.29.0 OpenSSL/1.0.2f zlib/1.2.8 libidn2/2.0.0 libssh2/1.8.0 libnghttp2/1.32.0
[root@centos6-7-disk1 ~]# curl -V | grep libcurl
curl/7.29.0 (x86_64-redhat-linux-gnu) libcurl/7.29.0 OpenSSL/1.0.2f zlib/1.2.8 libidn2/2.0.0 libssh2/1.8.0 libnghttp2/1.32.0
[root@centos6-7-disk1 ~]# curl -V | grep libcurl | awk '{print $1}' | sed 's/ /-/g'
curl/7.29.0
```

Timezone

This playbook uses the `timezone` module for setting the timezone.

```
[root@centos-7 ~]# curl -X POST http://127.0.0.1:8080/api/v1/hosts -H "Content-Type: application/json" -d '{"host_ip":"192.168.1.100","host_name":"host1"}'
```

Jira

This playbook creates a task in Jira.

Notifications

Slack

```
- name: Sending message to Slack Channel
slack:
  token: '{{ slack_token }}'
  channel: "#companynameAnsible"
  domain: "companyname.slack.com"
  parse: "full"
  color: "good"
  msg: 'The changes is completed on {{ inventory_hostname }}.'
```

Twilio

```
- name: Send an SMS to multiple phone numbers when the change is
completed
twilio:
  msg: The configuration change is completed!
  account_sid: XXXXXXXXXXXX
  auth_token: XXXXXXXXXXXX
  from_number: +34XXXXXXXXX
  to_number:
    - +34XXXXXXXXX1
    - +34XXXXXXXXX2
  delegate_to: localhost
```

NOTE: The delegate_to: localhost is often required when interworking with APIs.

connection: local runs your entire playbook play locally.

delegate_to: localhost runs a specific task on the localhost.

Custom Modules: Writing your own module

Folder Structure

Ansible requires that the module is stored at /library/custom_module_name.py at the same level where the playbook is located.

In the following example, the playbook is called "main.yaml" and the customer module "multiply.py" is located in the /library/ folder.

```
laptop:~/Desktop/ansible-modules-test$ tree
.
+- __init__.py
+- __main__.py
+- __tests__.py
laptop:~/Desktop/ansible-modules-test$ ls -al
total 12
drwxr-xr-x  2 root root 4096 May 17 14:27 .
drwxr-xr-x 10 root root 4096 May 17 14:27 ..
-rw-r--r--  1 root root  220 May 17 14:27 __init__.py
-rw-r--r--  1 root root  220 May 17 14:27 __main__.py
-rw-r--r--  1 root root  220 May 17 14:27 __tests__.py
laptop:~/Desktop/ansible-modules-test$
```

Ansible recommends to include strings for DOCUMENTATION and EXAMPLES at the top of our module code.

```
DOCUMENTATION = ""
---
module: multiply
short_description: Multiply two given numbers
""
```

```
EXAMPLES = ""
tasks:
- multiply:
  a: 200
  b: 10
  register: result
- debug: var=result
""
```

The key part is to import the boilerplate code from `ansible.module_utils.basic` like this:

```
from ansible.module_utils.basic import AnsibleModule
if __name__ == '__main__':
    main()
```

The `AnsibleModule` provides lots of common code for handling returns, parses your arguments for you, and allows you to check inputs.

Let's run the playbook and check the result:

Roles

Roles are nothing but directories laid out in a specific manner. Roles follow predefined directory layout conventions and expect each component to be in the path meant for it. A role directory structure contains the directories below. Each directory must contain a main.yml which is the default file.

```
[root@centos-7-base unxnn.mysql]# ls -lrt
total 12
-rw-rw-r--. 1 root root 7657 May 8 11:03 README.md
-rw-rw-r--. 1 root root 1061 May 8 11:03 LICENSE
drwxr-xr-x. 2 root root 22 Jun 14 12:04 handlers
drwxr-xr-x. 2 root root 22 Jun 14 12:04 defaults
drwxr-xr-x. 3 root root 21 Jun 14 12:04 molecule
drwxr-xr-x. 2 root root 67 Jun 14 12:04 templates
drwxr-xr-x. 2 root root 143 Jun 14 12:04 tasks
drwxr-xr-x. 2 root root 64 Jun 14 12:04 vars
drwxr-xr-x. 2 root root 39 Jun 14 12:04 tests
drwxr-xr-x. 2 root root 50 Jun 14 12:04 meta
```

Defaults: default variables for the role

Vars: variables for the role

Tasks: the main list of steps to be executed by the role

Files: contains files which we want to be transferred to the host

Templates: file template which supports modifications from the role

Meta: contains metadata of role such as dependencies

Handlers: handlers which can be invoked by notify directives

```
-- unxnn.mysql
|-- defaults
| '-- main.yml
|-- handlers
| '-- main.yml
|-- LICENSE
|-- meta
| '-- main.yml
|-- molecule
| '-- default
|   |-- molecule.yml
|   |-- playbook.yml
|   |-- tests
|     | '-- test_default.py
|   '-- yaml-lint.yml
|-- README.md
|-- tasks
| '-- configure.yml
| '-- main.yml
| '-- secure-installation.yml
| '-- setup-Debian.yml
| '-- setup-RedHat.yml
| '-- variables.yml
|-- templates
| '-- my.cnf.j2
| '-- root-my.cnf.j2
| '-- user-my.cnf.j2
|-- tests
| '-- inventory
| '-- test.yml
```

Let's take a glance of these files:

/defaults/main.yml

```
# MySQL connection settings.
mysql_port: "3306"
mysql_bind_address: "127.0.0.1"
mysql_skip_name_resolve: false
mysql.sock_dir: "/var/lib/mysql"
mysql.sock_mode: "1"
# The following variables have a default value depending on operating system.
# mysql.pid_file: /var/run/mysqld/mysqld.pid
# mysql_socket: /var/lib/mysql/mysql.sock

# Log file settings.
mysql_log_file_group: mysql

# Slow query log settings.
mysql_slow_query_log_enabled: false
mysql_slow_query_time: "2"
# The following variable has a default value depending on operating system.
# mysql_slow_query_log_file: /var/log/mysql-slow.log

# Memory settings (default values optimized >512MB RAM).
mysql_key_buffer_size: "256M"
mysql_max_allowed_packet: "16M"
mysql_table_open_cache: "256"
mysql_sort_buffer_size: "1M"
mysql_read_buffer_size: "1M"
mysql_read_rnd_buffer_size: "4M"
mysql_myisam_sort_buffer_size: "64M"
```

/handlers/mail.yml

```
[root@centos-7-base handlers]# pwd
/rvtr_usable/roles/www_mysql/handlers
[root@centos-7-base handlers]# ls
mail.yml
[root@centos-7-base handlers]# cat mail.yml
...
- name: restart mysql
  service: "name={{ mysql_daemon }} state=restarted sleep=5"
[root@centos-7-base handlers]#
```

/meta/main.yml

```
[root@centos-7-base meta]# pwd
/rwt/centos-7-base/meta
[root@centos-7-base meta]# ls
meta.yaml
[root@centos-7-base meta]# cat main.yml
...
dependencies: []
galaxy_info:
  author: unxon
  role_name: mysql
  description: MySQL server for RHEL/CentOS and Debian/Ubuntu.
  license: "Apache License (MIT)"
  min_ansible_version: 2.4
  platforms:
    - name: EL
      versions:
        - 6
        - 7
    - name: Ubuntu
      versions:
        - all
    - name: Debian
      versions:
        - all
  galaxy_tags:
    - database
    - mysql
    - mariadb
    - db
    - sql
[root@centos-7-base meta]#
```

/tasks/main.yml

```
[root@centos-7-base unxon.mysql]# cd tasks/
[root@centos-7-base tasks]# ls
configure.yml  incln.yml  secure-installation.yml  setup-Debian.yml  setup-Redhat.yml  variables.yml
[root@centos-7-base tasks]# more main.yml
...
# Variable configuration.
- include_tasks: variables.yml

# Setup/install tasks.
- include_tasks: setup-Redhat.yml
  when: ansible_os_family == 'Redhat'

- include_tasks: setup-Debian.yml
  when: ansible_os_family == 'Debian'

- name: Check if MySQL packages were installed.
  set_fact:
    mysql_install_packages: "{{ (rhmysql_install_packages is defined and rhmysql_install_packages.changed) or (deb_mysql_install_packages is defined and deb_mysql_install_packages.changed) }}"

# Configure MySQL.
- include_tasks: configure.yml
- include_tasks: secure-installation.yml

- name: Ensure MySQL databases are present.
  mysql_db:
    name: "{{ item.name }}"
    collation: "{{ item.collation | default('utf8_general_ci') }}"
    encoding: "{{ item.encoding | default('utf8') }}"
    state: '{{ item.state | default('present') }}'
    with_items: "{{ mysql_databases }}"

- name: Ensure MySQL users are present.
```

/templates/

```
(root@centos-7-base:~mysql]# cd templates/
[root@centos-7-base templates]# ls
my.cnf.j2  root-my.cnf.j2  user-my.cnf.j2
[root@centos-7-base templates]# cat my.cnf.j2
# {{ ansible_managed }}

[client]
password = your password
port = {{ mysql_port }}
socket = {{ mysql_socket }}

[mysqld]
port = {{ mysql_port }}
bind-address = {{ mysql_bind_address }}
datadir = {{ mysql_datadir }}
socket = {{ mysql_socket }}
pid-file = {{ mysql_pid_file }}
#( if mysql_skip_name_resolve %)
skip-name-resolve
#( if mysql %)
#( if mysql_local_mode %)
sql-mode = {{ mysql_sql_mode }}
#( and if %)

# Logging configuration.
#( if mysql_log_error == 'syslog' or mysql_log == 'syslog' %)
syslog
syslog-tag = {{ mysql_syslog_tag }}
#( else %)
#( if mysql_log %)
log = {{ mysql_log }}
#( endif %)
log-error = {{ mysql_log_error }};
```

/vars/

```
(root@centos-7-base:~mysql]# cd vars/
[root@centos-7-base vars]# ls
Debian.yml  Redhat-4.yml  Redhat-7.yml
[root@centos-7-base vars]# cat Redhat-7.yml
---
mysql_daemon: mariadb
mysql_packages:
- mariadb
- mariadb-server
- mariadb-libs
  MySQL-python
- perl-DBD-MYSQL
mysql_slow_query_log_file: /var/log/mysql-slow.log
mysql_log_error: /var/log/mariadb/mariadb.log
mysql_syslog_tag: mariadb
mysql_pid_file: /var/run/mariadb/mariadb.pid
mysql_config_file: /etc/my.cnf.d
mysql_config_include_dir: /etc/my.cnf.d
mysql_socket: /var/lib/mysql/mysql.sock
mysql_supports_innodb_large_prefix: true
[root@centos-7-base vars]#
```

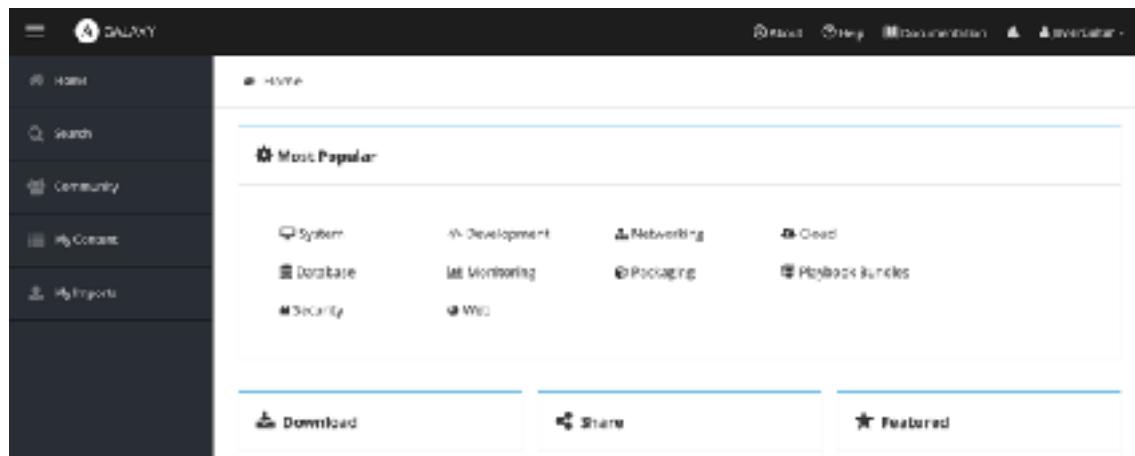
Molecule

Please refer to the following link: <https://molecule.readthedocs.io/en/stable/>

Ansible Galaxy

Ansible Galaxy refers to the Ansible website where users can share roles, and to a command line tool for installing, creating and managing roles.

<https://galaxy.ansible.com/home>



The `ansible-galaxy` command comes bundled with Ansible, and you can use it to install roles from Galaxy or directly from a git based SCM. You can also use it to create a new role, remove roles, or perform tasks on the Galaxy website.

The command line tool by default communicates with the Galaxy website API using the server address <https://galaxy.ansible.com>. Since the Galaxy project is an open source project, you may be running your own internal Galaxy server and wish to override the default server address. You can do this using the `--server` option or by setting the Galaxy server value in your `ansible.cfg` file. For information on setting the value in `ansible.cfg` visit Galaxy Settings.

Search

Search the Galaxy database by tags, platforms, author and multiple keywords. For example:

```
$ ansible-galaxy search elasticsearch
```

List installed roles

Use `list` to show the name and version of each role installed in the `roles_path`.

```
$ ansible-galaxy list
```

- `chouseknecht.role-install_mongod`, master
- `chouseknecht.test-role-1`, v1.0.2
- `chrismeyersfsu.role-iptables`, master
- `chrismeyersfsu.role-required_vars`, master

Remove installed roles

Use `remove` to delete a role from `roles_path`:

Import a role

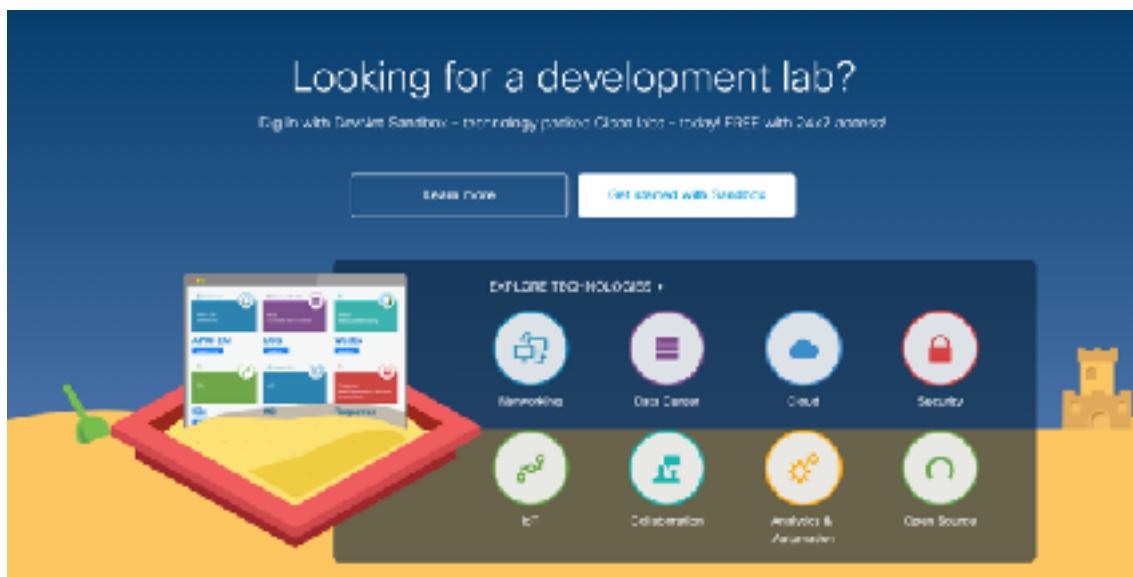
The import command requires that you first authenticate using the login command. Once authenticated you can import any GitHub repository that you own or have been granted access.

Use the following to import to role:

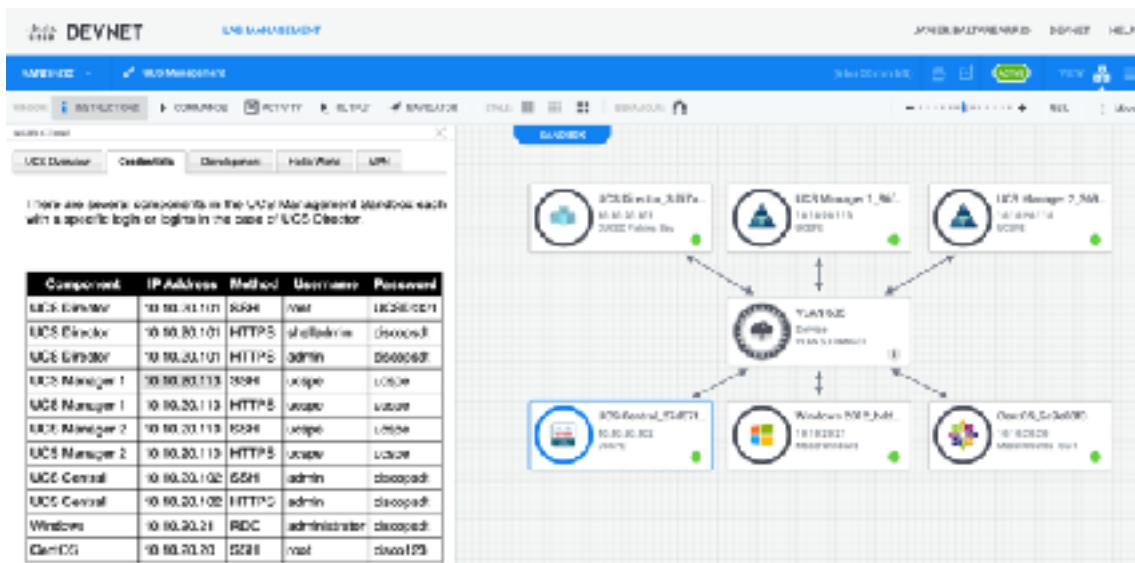
```
$ ansible-galaxy import github_user github_repo
```

UCS Manager Management

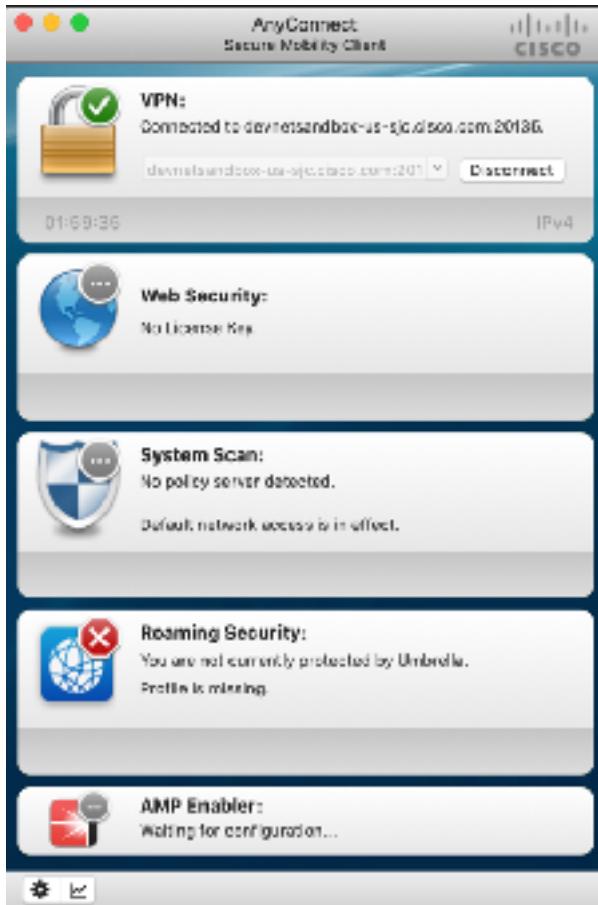
Some of the examples below are executed upon Cisco Devnet sandboxes: <https://developer.cisco.com/site/sandbox/>



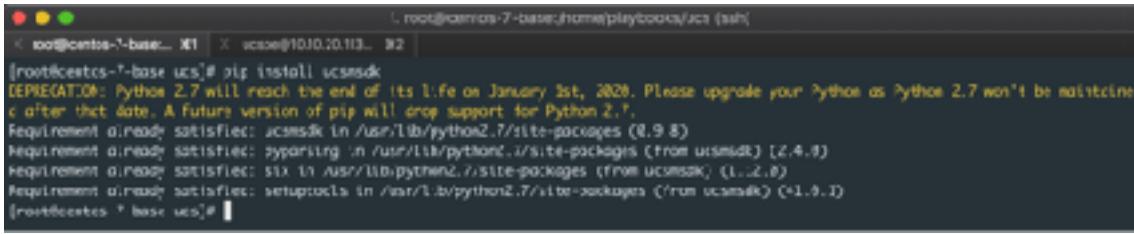
Once your lab is reserved, the VPN credentials are sent by email as shown below:



You have to set up the VPN connection:

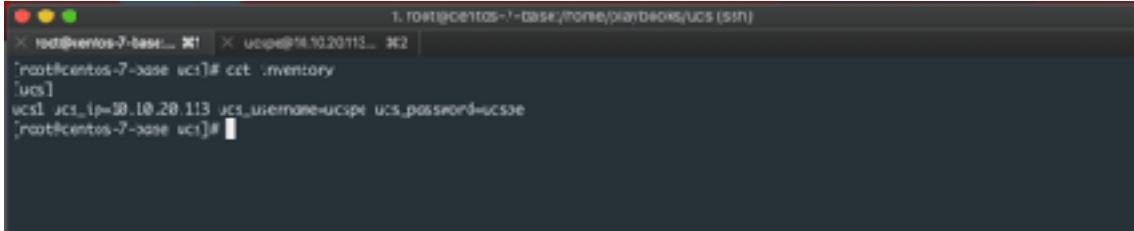


In order to interwork with UCS Manager, the "ucsmsdk" must be installed:



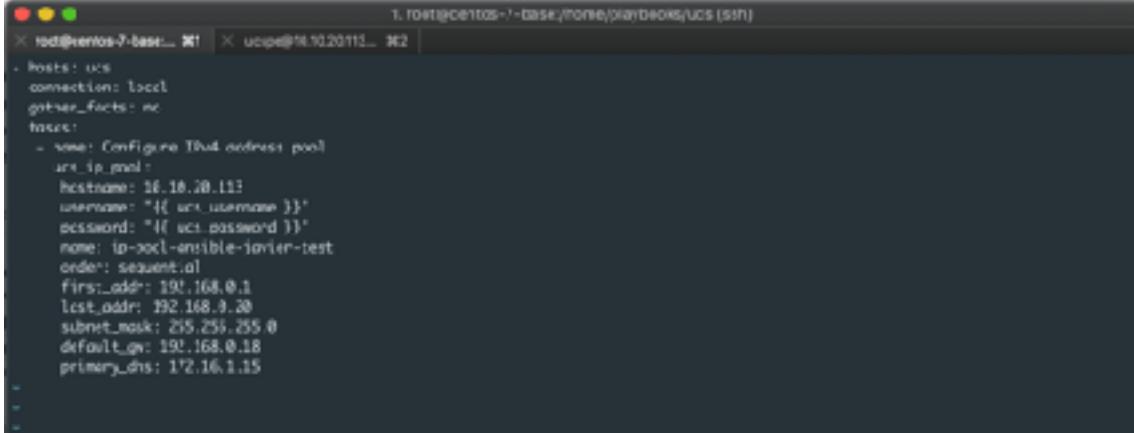
```
[root@centos-7-base ucs]# pip install ucsdk
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python or Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.
Requirement already satisfied: ucsdk in /usr/lib/python2.7/site-packages (0.9.8)
Requirement already satisfied: pyasn1 in /usr/lib/python2.7/site-packages (from ucsdk) (0.4.9)
Requirement already satisfied: six in /usr/lib/python2.7/site-packages (from ucsdk) (1.12.0)
Requirement already satisfied: setuptools in /usr/lib/python2.7/site-packages (from ucsdk) (41.9.1)
[root@centos-7-base ucs]#
```

For testing purposes, I have created the inventory file below:



```
[root@centos-7-base ucs]# cat inventory
[ucs]
ucs1.ucs_ip=10.10.20.113 ucs_username=ucspe ucs_password=ucspe
[root@centos-7-base ucs]#
```

The following playbook configures an address pool in UCS.



```
- hosts: ucs
  connection: local
  gather_facts: no
  tasks:
    - name: Configure Test address pool
      ucs_ip_pool:
        hostname: 10.10.20.113
        username: "{{ ucs_username }}"
        password: "{{ ucs_password }}"
        name: ip-pool-enable-ionview-test
        order: sequential
        first_addr: 191.168.0.1
        last_addr: 192.168.0.30
        subnet_mask: 255.255.255.0
        default_gw: 191.168.0.1
        primary_dns: 172.16.1.15
```

Running the playbook:

```
# ansible-playbook create-pool-yaml -i inventory -vvv
```

```

root@centos-7-base... 301] x ucspe@192.168.0.113... 302 ] 1. root@centos-7-base/home/playbooks/ucs (sish)

ucs1> EXEC /bin/sh -c 'rm -f -r /root/.ansible/tmp/ansible-tmp-1568551799.79-249281772156968/ > /dev/null 2>&1 && sleep 8'
changed: [ucs1] => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "invocation": {
        "module_args": {
            "default_gw": "192.168.0.39",
            "descr": "",
            "first_ip": "192.168.0.2",
            "hostnames": "192.168.0.113",
            "ip_ranges": null,
            "ipv6_first_ip": null,
            "ipv6_ipnetmask": null,
            "ipv6_prefix": "64",
            "ipv6_primary_dra": null,
            "ipv6_secondary_dra": null,
            "last_ip": "192.168.0.23",
            "name": "ip-pool-ansible-test-javier",
            "order": "sequential",
            "org_name": null,
            "password": "VALUE_SPECIFIED_IN_NO_LOG_PARAMETER",
            "port": null,
            "primary_dra": "172.16.1.25",
            "proxy": null,
            "secondary_dra": "16.0.0.0",
            "state": "present",
            "subnet_mask": "255.255.255.0",
            "use_proxy": true,
            "use_ssl": true,
            "username": "VALUE_SPECIFIED_IN_NO_LOG_PARAMETER"
        }
    }
}
META: ran handlers
META: ran handlers

PLAY RECAP ****
ucs1 : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

You can establish an SSH connection towards UCS Manager and check that the pool is created:

```

root@centos-7-base... 301] x ucspe@192.168.0.113... 302 ] 1. ssh ucspe@192.168.0.113 (sish)

ucspe# show configuration | b ip-pool-ansible-test-javier
  enter ip-pool ip-pool-ansible-test-javier
    enter ip-block 192.168.0.2 192.168.0.21 192.168.0.19 255.255.255.0
      set primary-dra 172.16.1.15 secondary-dra 0.0.0.0
    exit
  set assignment-order sequential

```

Similarly, the following playbook creates a VLAN.

```

root@centos-7-base... 301] x ucspe@192.168.0.113... 302 ] 1. root@centos-7-base/home/playbooks/ucs (sish)

[root@centos-7-base ucs]# cat configure-vlan.yaml
- hosts: ucs
  connection: local
  gather_facts: no
  tasks:
    - name: Configure VLAN
      uci_vlans:
        id: 10
        hostnames: 192.168.0.113
        username: "{{ ucs_username }}"
        password: "{{ ucs_password }}"
        name: vlan-javier-dmc-available
        id: 9
        native: "yes"
[root@centos-7-base ucs]# 

```

Running the playbook:

```
1. root@centos-7-base:~ [root@centos-7-base:~]# ./run playbook/ucs1 (s1h)
x root@centos-7-base:~ [root@centos-7-base:~]# ./run playbook/ucs1 (s1h)
[root@centos-7-base ucs1]# ansible-playbook configure-vlan.yaml -i inventory

PLAY [ucs1] ****

TASK [Configure VLAN]
changed: [ucs1]

PLAY RECAP
ucs1 : ok=1    changed=1    unreachable=0    failed=0    skipped=0    resoled=0    ignored=0

[root@centos-7-base ucs1]#
```

Validating that the VLAN is created:

```
[root@centos-7-vm ~]# ./vlan-javier-dmz-ansible
[root@centos-7-vm ~]# exit
```

Cisco NXOS Management

Please refer to: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html?highlight=nxos

Documentation		ANSIBLE TEST	PRODUCTS	COMMUNITY	VIDEOS & TRAINING	EDUC
Ansible 2.6	Ansible 2.5					
For previous versions, see the documentation archive.						
Version 2.6.0-rc1						
DOCUMENTATION LIBRARIES & CONFIGURATION						
Installation Guide						
Ansible Porting Guide						
Coding Standards						
User Guide						
Ansible Controller						
Getting Started						
Working with Command Line (CLI)						
Introduction to Basic Linux Commands						
Nodes						
ansible-jboss-server - Manage JBoss AMQ server global configuration						
ansible-nexxus-server-host - Manages NEXXUS server host specific configuration						
ansible-lldp - Manages lldp entries for ATCs						
ansible-interlock - Manages aclosing ACI to Interlock						
ansible-juniper - Manages and configures Juniper MXOS devices						
ansible-l3prv - Manages BGP configuration						
ansible-l3bgp - Manages BGP Address-family configuration						
ansible-l3bgp-holder - Manages BGP Hold timer configurations						
ansible-l3bgp-neighbor - Manages BGP address-family's neighbors configuration						
ansible-vrrp - Run arbitrary command on Cisco MXOS devices						
ansible-junos - Manage Cisco JUNOS configuration machine						
ansible-vpn-global - Handles the VPN control plane for VOLAN						
ansible-vxlans - Manages Cisco IPVPN VXLAN Network Identifier (VNI)						
ansible-facts - Gets facts about NEOS switches						
ansible-fabrics - Manage fabrics in HPE 3PAR switches						
ansible-fibrearray - Create a fabric or remove NEOS device						
ansible-gb - Trigger a graceful manual reconnection (GR) of the switch						

Playbooks

Create VLANs

This is the initial NXOS device configuration:

```

root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)
* root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)

sbs-nxos# show running-config | grep vlan
    limit-resource vlan minimum 16 maximum 4094
feature interface-vlan
snmp-server enable traps vtp vlancreate
snmp-server enable traps vtp vlandelete
vlan 1,20,30,40,100-306
vlan 20
vlan 30
vlan 40
vlan 100
vlan 301
vlan 302
vlan 303
vlan 304
vlan 305
vlan 306
vlan 307
vlan 308
vlan 309
vlan 310
vlan 311
vlan 312
vlan 313
vlan 314
vlan 315
vlan 316
vlan 317
vlan 318
vlan 319
vlan 320
vlan 321
vlan 322
vlan 323
vlan 324
vlan 325
vlan 326
vlan 327
vlan 328
vlan 329
vlan 330
vlan 331
vlan 332
vlan 333
vlan 334
vlan 335
vlan 336
vlan 337
vlan 338
vlan 339
vlan 340
vlan 341
vlan 342
vlan 343
vlan 344
vlan 345
vlan 346
vlan 347
vlan 348
vlan 349
vlan 350
vlan 351
vlan 352
vlan 353
vlan 354
vlan 355
vlan 356
vlan 357
vlan 358
vlan 359
vlan 360
vlan 361
vlan 362
vlan 363
vlan 364
vlan 365
vlan 366
vlan 367
vlan 368
vlan 369
vlan 370
vlan 371
vlan 372
vlan 373
vlan 374
vlan 375
vlan 376
vlan 377
vlan 378
vlan 379
vlan 380
vlan 381
vlan 382
vlan 383
vlan 384
vlan 385

```

The playbook below creates a range of VLANs:

```

root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)
* root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)

hosts: nexus7000
connection: local
gather_facts: no

vars:
  nxos_provider:
    username: "{{ username }}"
    password: "{{ password }}"
    transport: https
    port: 443
    timeout: 90
    host: "{{ inventory_hostname }}"

tasks:
- name: Create VLANs
  nxos_vlan:
    vlan_range: "30-55"
    state: present
    provider: "{{ nxos_provider }}"

```

Running the playbook:

```

root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)
* root@centos-7-base...:~| admin@sbs-nxos-m...:~| t. ssh admin@sbs-nxos-mgmt.cisco.com -p 8101 (ssh)

ansible-tmp=1568568466.34-282568797466722/Ansible12_nxos_vlans.py & sleep 0'
<sbx-exos-agm> cisco.com> EXEC /bin/sh -c '/usr/bin/python /root/.ansible/tmp/ansible-tmp-1568568466.34-282568797466722/Ansible12_nxos_vlans.py & sleep 0'
<sbx-exos-agm> cisco.com> EXEC /bin/sh -c 'rm -f -r /root/.ansible/tmp/ansible-tmp-1568568466.34-282568797466722/ > /dev/null 2>&1 & sleep 0'
changed: [sbx-nxos-agm.cisco.com] => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "commands": [
        "vlan 30",
        "vlan 31",
        "vlan 32",
        "vlan 33",
        "vlan 34",
        "vlan 35",
        "vlan 36",
        "vlan 37",
        "vlan 38",
        "vlan 39",
        "vlan 40",
        "vlan 41",
        "vlan 42",
        "vlan 43",
        "vlan 44",
        "vlan 45",
        "vlan 46",
        "vlan 47",
        "vlan 48",
        "vlan 49",
        "vlan 50",
        "vlan 51",
        "vlan 52",
        "vlan 53",
        "vlan 54",
        "vlan 55"
    ],
    "invocation": {
        "module_args": {
            "admin_state": "up"
        }
    }
}

```

This is the list of VLANs once the playbook is executed:

```

1. ssh admin@10.0.0.1 -p 8101 (ssh)
< root@cenis-7-ba... 31 | x admin@10.0.0.1... 32
dbx-n9kv-ad# show running-config | grep vlon
  limit-resource vlan minimum 15 maximum 4094
feature interface-vlan
snmp-server enable traps vtp vlancreate
snmp-server enable traps vtp vlandelete
vlon 1,20,30,40,50-55,100-106
vlon 20
vlon 30
vlon 40
vlon 50
vlon 55
vlon 100
vlon 101
vlon 102
vlon 103
vlon 104
vlon 105

```

Configure Portchannels

tasks:

```

- name: Create portchannel 200
  nxos_portchannel:
    group: 200
    members: ['Ethernet1/1','Ethernet1/2']
    mode: 'active'
    host: "{{ inventory_hostname }}"
    state: present

```

The screenshot shows the Ansible AWX interface with a workflow titled "Create portchannel 200". The workflow consists of two tasks:

- Task 1: "Create portchannel 200" (id: 1). It uses the "nxos_portchannel" module with the following parameters:
 - group: 200
 - members: ['Ethernet1/1', 'Ethernet1/2']
 - mode: 'active'
 - host: "{{ inventory_hostname }}"
 - state: present
- Task 2: "Ansible ping" (id: 2). It uses the "ping" module with the following parameters:
 - connection: local
 - gather_facts: no
 - gather_subset: none
 - username: "EE username 1"
 - password: "EE password 1"
 - transport: ssh
 - ports: 22
 - timeout: 30
 - block: true
 - check_mode: false

Feature Enablement

tasks:

```
- name: Ensure DHCP is enabled
  nxos_feature:
    feature: dhcp
    state: enabled
  host: "{{ inventory_hostname }}"
```

This is the list of features enabled:

```
t ssh admin@sbx-nxos-mgmt.cisco.com -p 6101 (ssh)
* root@centos-7-base... R1 | x admin@sbx-nxos-m... R2 |
The copyrights to certain works contained herein are owned by other
third parties and are used and distributed under license. Some parts
of this software may be covered under the GNU Public License or the
GNU Lesser General Public License. A copy of each such license is
available at
  https://www.gnu.org/licenses/gpl.html and
  https://www.gnu.org/licenses/lgpl.html
-----
* Nexus 9000v is strictly limited to use for evaluation, demonstration, *
* and NDA-protected education. Any use or disclosure, in whole or in part of *
* the Nexus 9000v Software or Documentation to any third party for any *
* purposes is expressly prohibited except as otherwise authorized by *
* Cisco in writing.
-----
sbx-nxos-m# show running-config | grep feature
feature nxcapi
feature bash-shell
feature scp-server
feature lftp
feature netconf
feature restconf
feature gryc
feature interface-vlan
feature API, explore features, and test scripts. Please
The following programmability features are already enabled:
  nra-server enable traps feature-control FeatureOpStatusChange
  nra-server enable traps feature-control CiscoFeatureStatusChange
  devinfo add |
```

The inventory file includes the Nexus 9K hostname, username and password:

```
t root@centos-7-base:/home/playbooks/nxos (ssh)
* root@centos-7-base R1 | x admin@sbx-nxos-m... R2 |
[nexus7000] ibx-nxos-mgmt.cisco.com username=admin password=admin_1234!
[nxthelixos-7-base nexus]#
```

The playbook enables two new features: DHCP and OSPF:

```

root@centos-7-base... K1 | x admin@sbx-nxos-m... K2 |
[root@centos-7-base nxos]# cat enable-feature-yaml
---
hosts: nexus7000
connection: local
gather_facts: no

vars:
  nxos_provider:
    username: "{{ username }}"
    password: "{{ password }}"
    transport: cli
    port: 8881
    timeout: 30
    host: "{{ inventory_hostname }}"

tasks:
  - name: Enable features
    nxos_feature:
      feature: "[{{ item }}]"
      state: enabled
      provider: '{{ nxos_provider }}'
    with_items:
      - dhcp
      - vspf
[root@centos-7-base nxos]#

```

Running the playbook:

```

root@centos-7-base... K1 | x admin@sbx-nxos-m... K2 |
[root@centos-7-base nxos]# ansible-playbook enable-feature-yaml -i inventory -vv
ansible-playbook 2.8.0
config file = None
configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python2.7/site-packages/ansible
executable location = /bin/ansible-playbook
python version = 2.7.5 (Default: Apr  9 2019, 14:30:58) [GCC 4.8.5 20150623 (Red Hat 4.8.5-36)]
No config file found; using defaults

PLAYBOOK: enable-feature-yaml -----
  1 plays in enable-feature-yaml

PLAY [nexus7000] -----
META: ran handlers

TASK [Enable Features] *****
task path: /home/admin/nxos/enable-feature-yaml:3
changed: [sbx-nxos-mgt.cisco.com] => (item=dhcp) => {"ansible_features": {"discoverd_interpreter_python": "/usr/bin/python"}, "ansible_liaop_ip": "item", "changed": true, "commands": ["Terminal don't-ask", "Feature dhcp"], "item": "dhcp"}
changed: [sbx-nxos-mgt.cisco.com] => (item=vs pf) => {"ansible_liaop_ip": "item", "changed": true, "commands": ["Terminal don't-ask", "Feature vspf"], "item": "vs pf"}
META: ran handlers
META: ran handlers

PLAY RECAP *****
sbx-nxos-mgt.cisco.com : ok=1    changed=1    unreachable=0    failed=0   skipped=0    rescued=0    ignored=0
[root@centos-7-base nxos]#

```

The new features are enabled:

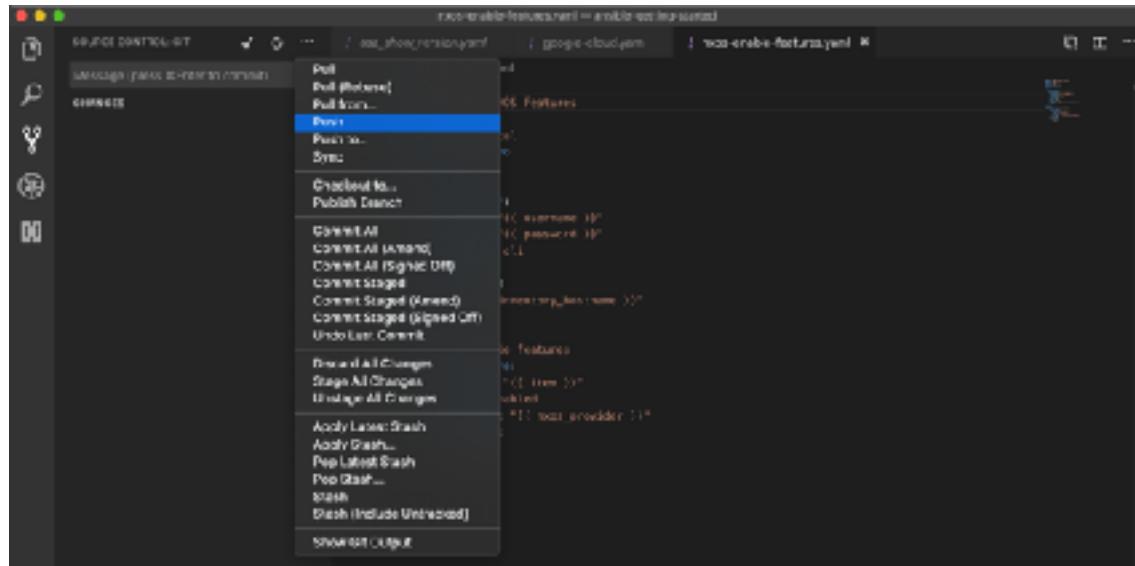
```

root@centos-7-base... K1 | x admin@sbx-nxos-m... K2 |
sbx-nxos# show running-config | grep feature
feature nxapi
feature batch-shell
feature scap-server
feature ospf
feature bgp
feature netconf
feature restconf
feature grpc
feature interface-vlan
feature snmp
test APIs, execute features, and test scripts. Please
the following programmability features are already enabled:
netconf-server enable traps feature-control testbed-status-change
snmp-server enable traps feature-control testbed-status-change
sbx-nxos# 

```

It is time to get into AWX and play around with all these possibilities.

You have to push your code to the Git repository:



A screenshot of a GitHub commit page for the 'ansible-getting-started' repository. The commit message is 'Tuning timeout variable for Urinet connection' and it was made by 'jwilder' 2 minutes ago. The commit details show a file named 'ansible-features.yml' with 25 lines changed, 127 bytes added, and 0 bytes removed. The commit content shows the YAML configuration for enabling features on an N9K switch using the 'nxos_provider' module.

```

1  ---
2  - name: Enable 100S Features
3    nxos:
4      connection: local
5      gather_facts: no
6
7    vars:
8      nxos_provider:
9        username: "{{ username }}"
10       password: "{{ password }}"
11       transport: cli
12       ports: 22
13       timeouts: 10
14       hosts: "{{ inventory_hostname }}"
15
16    tasks:
17      - name: enable features
18        nxos_features:
19          features: "{{ item }}"
20          state: enabled
21          provider: "{{ nxos_provider }}"
22          with_items:
23            - 100S
24            - 100T

```

Ensure that AWX project is updated by clicking on Sync:

PROJECTS	
PRJ0001 - 项目	
NAME	PRJ0001
CREATOR	John
LAST MODIFIED	2023-01-15 10:15:24
LAST USED	2023-01-15 10:15:24
Available training [3]	
VERSION	5.2.0.0.14
DESCRIPTION	Project A
LAST MODIFIED	2023-01-15 10:15:24
LAST USED	2023-01-15 10:15:24
PRJ0002 - 项目 [6]	
VERSION	6.0.0.0.15
DESCRIPTION	Project B
LAST MODIFIED	2023-01-15 10:15:26
LAST USED	2023-01-15 10:15:26

The revision matches with Github:

The screenshot shows the AWX web interface. On the left is a sidebar with navigation links: Home, Projects, Inventories, Inventory Details, Access, and Help. The main area is titled 'PROJECTS' and lists two projects: 'PP000015 - 001' and 'PP000015 - 002'. Each project card includes a search bar, a 'REVISION' section (set to 'dev'), a 'DESCRIPTION' section ('DevEnv'), and a timestamp ('2018-09-11 10:15:26'). Below the cards is a section titled 'PP000015 - 001' containing a table with columns 'NAME', 'TYPE', and 'LAST UPDATED'. The table has three rows: 'PP000015 - 001', 'Inventory', and '2018-09-11 10:15:26'. At the bottom right of the page are three small icons.

Before creating the template, you have to provision the credentials.

AWX

admin ⚙️ 🌐 🏷️

TEMPLATES → NCDF Enable Features

NCDF Enable Features

DETAILS 🔍 **PERMISSIONS** 🔍 **NOTIFICATIONS** 🔍 **COMPUTE JOB** 🔍 **SCHEDULE** 🔍 **JOB HISTORY** 🔍

NAME	DESCRIPTION	JOB TYPE	<input type="checkbox"/> ADD ON UNDER
NCDF Enabled Features		Run	<input type="button" value="▼"/>
INVENTORY 🔍	<input type="radio"/> Has from user	PROJECT 🔍	<input type="radio"/> Has from user
<input type="button" value="Q"/> NCDF	<input type="button" value="Q"/> Available Training	<input type="radio"/> Job ID	<input type="checkbox"/> ADD ON UNDER
PARENTALITY 🔍	<input type="radio"/> Has from user	NAME 🔍	<input type="radio"/> Has from user
<input type="button" value="Q"/> NCDF	<input type="button" value="DEPOLL"/>	<input type="radio"/> ID	<input type="button" value="▼"/>
VISIBILITY 🔍	<input type="radio"/> Has from user	JOB TYPES 🔍	<input type="radio"/> Has from user
2 (Non-Webcast)	<input type="button" value="Q"/>	<input type="radio"/> Has from user	<input type="radio"/> Skipped
LADDS 🔍	INSTANCE GROUPS 🔍	JOBSUCH 🔍	<input type="checkbox"/> ADD ON UNDER
<input type="button" value="Q"/>	<input type="button" value="Q"/>	<input type="radio"/> ID	<input type="button" value="▼"/>
SHOW CHANGES 🔍	<input type="radio"/> PENDING APPROVAL	OPTIONS	
<input type="button" value="Q"/>		<input type="checkbox"/> Enable Privilege Escalation	
		<input type="checkbox"/> Allow Provisioning (Collective)	
		<input type="checkbox"/> Enable Concurrent Jobs	
		<input type="checkbox"/> Use Fast Cache	

Click on Credentials Types

Input Configuration:

```
fields:
- type: string
  id: username
  label: NXOS username
  - secret: true
  type: string
  id: password
  label: NXOS password
  required:
  - username
  - password
```

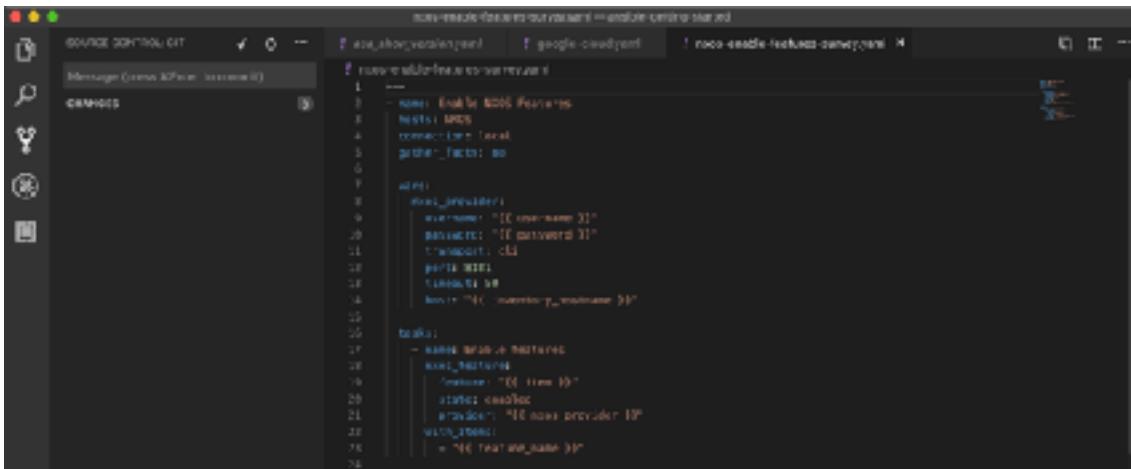
Click on Inventory and add the NXOS host:

Create the corresponding group:

Let's run the playbook:

You can also add a survey, which passes a variable value to the playbook execution.

I am adding a new variable called "feature_name".



```

# Ansible Survey Example
# https://github.com/ansible/awx/tree/devel/roles/surveys/tasks/main.yml

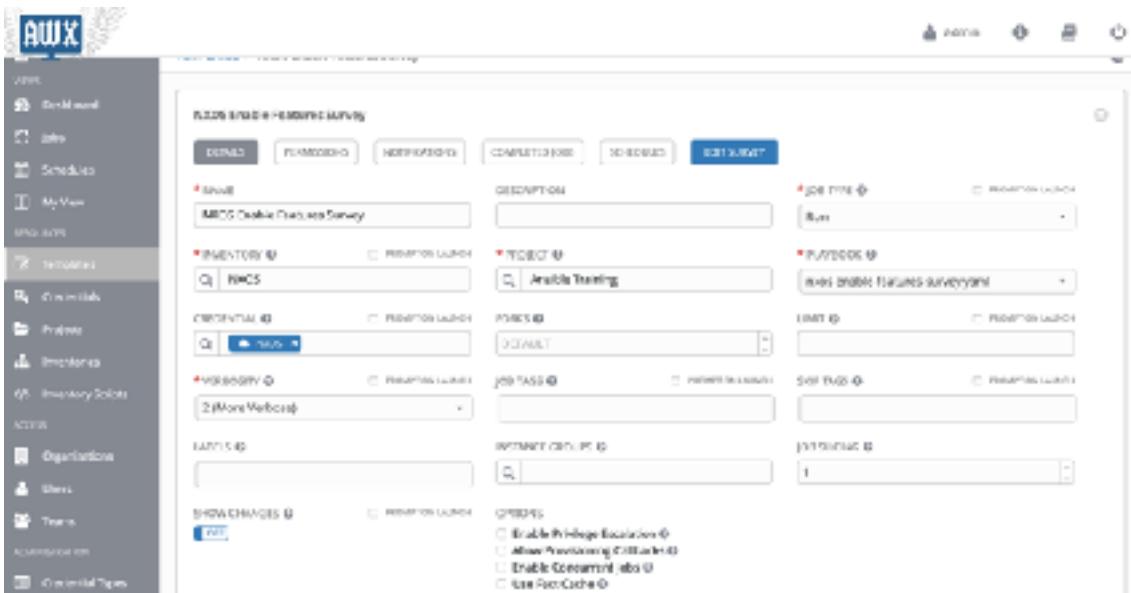
# vars
- name: Enable RDP Features
  ntpss: RDP
  connection: local
  gather_facts: no

# vars
mail:
  - username: "TCI\user@TCI"
  - password: "TCI@12345"
  - transport: ssh
  - port: 3321
  - timeout: 300
  - keyfile: "/etc/ssh/ssh_host_rsa_key"
  - host: "TCI\user@TCI"

# vars
- name: RDP
  - address: 192.168.1.100
  - port: 3389
  - timeout: 10
  - user: user
  - provider: RDP
  - width: 1280
  - height: 800
  - job_type: Survey

```

Click on your template and Edit Survey:



AWX

Surveys

AWX Onboarding Survey

DIMINISHING RETURNS SURVEY

DETAILS **PERMISSIONS** **NOTIFICATIONS** **COMPLETED FORMS** **RECENT** **EDIT SURVEY**

NAME AWX Onboarding Survey

DESCRIPTION

JOB TYPE Survey

INVENTORY RDP

PLAYBOOKS Ansible Training

LIMITS RDP

ORGANIZATIONS

PROMPT Job Type: RDP

SHOW CHANGES **PROMPT**

OPTIONS

- Enable Privilege Escalation
- Allow Provisioning Canceled
- Enable Comment tab
- Use Pre-Cache

Add a prompt, variable name and answer type:

In this case, you have to specify the feature before running the playbook as shown below:

Cisco ASA Management

Please refer to: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html?highlight=asa

Playbooks

Create Network object

```
---
- hosts: asa
  gather_facts: no
  connection: network_cli

vars:
  object_hosts:
    - 10.45.16.79
    - 10.45.16.72

  object_group: TEST

tasks:
  - name: "CREATE HOST AND GROUPS ON {{ ansible_ssh_host }}"
    asa_config:
      lines:
        - network-object host {{ item }}
          parents: object-group network {{ object_group }}
          with_items: "{{ object_hosts }}"
```

Cisco IOS Management

Please refer to: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html?highlight=ios

<p>Documentation</p> <p>Ansible 1.8</p> <p>For previous versions, use the documentation archive.</p> <p>SEARCH</p> <p>ANSIBLE DOCUMENTATION</p> <p>Installation Guide</p> <p>Ansible Porting Guide</p> <p>Using Ansible</p> <p>File User Guide</p> <p>Ansible Collection</p> <p>Getting Started</p> <p>Working with Command Line (v1.8)</p> <p>Introduction to Ansible: Commands</p> <p>Working with Variables</p>	<p>Ios</p> <ul style="list-style-type: none">• ios_banner - Manage multiline banners on Cisco IOS devices• ios_config - Configure global IOS2 privileged settings on Cisco IOS• ios_command - Run commands on remote devices running Cisco IOS• ios_config_file - Manage Cisco IOS configuration sections• ios_facts - Collect facts from remote devices running Cisco IOS• ios_interface - Manage interfaces on Cisco IOS network devices• ios_l2_interface - Manage Layer 2 interfaces on Cisco IOS network devices• ios_l3_interface - Manage Layer 3 interfaces on Cisco IOS network devices• ios_lldp - Manage LLDP management groups on Cisco IOS network devices• ios_logging - Manage logging on network devices• ios_ping - Manages over N ICMP ping sessions• ios_ping - Test reachability using ping from Cisco IOS network devices• ios_route_width - Manage static IP routes on Cisco IOS network devices• ios_system - Manage the system attributes on Cisco IOS devices• ios_vrf - Manage the aggregate of virtual routes on Cisco IOS devices• ios_vrf - Manage VRFs on IOS network devices• ios_vrf - Manage the aggregated set of VRFs attributes on Cisco IOS devices
--	--

Deploying to Amazon Web Services

Inventory

You can configure a dynamic inventory syncing with your AWS resources.

First, configure your AWS account credentials:

NAME	TYPE	OWNERS	ACTIONS
AWS Personal User	AWS	admin, default	
Amazon VPC	Network	admin	
Amazon VPC (Compute Engine)	Compute Engine	admin, default	
RDS	RDS	admin, Default	
RDS Multi AZ	RDS	admin, Default	

Add your access and secrets keys.

Click on Inventories:

NAME	TYPE	OBSERVATION	ACTIONS
AWS	Inventory	Default	
AWS Inventory	Inventory	Default	
Customized AWS	Inventory	Default	

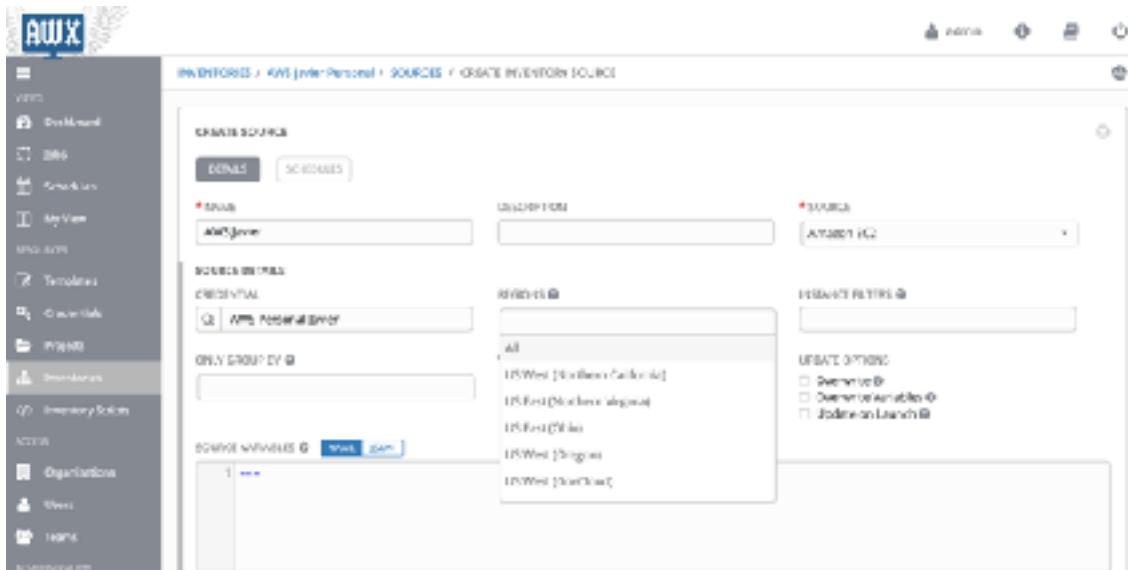
Click on Sources.

Choose the Amazon EC2 source.

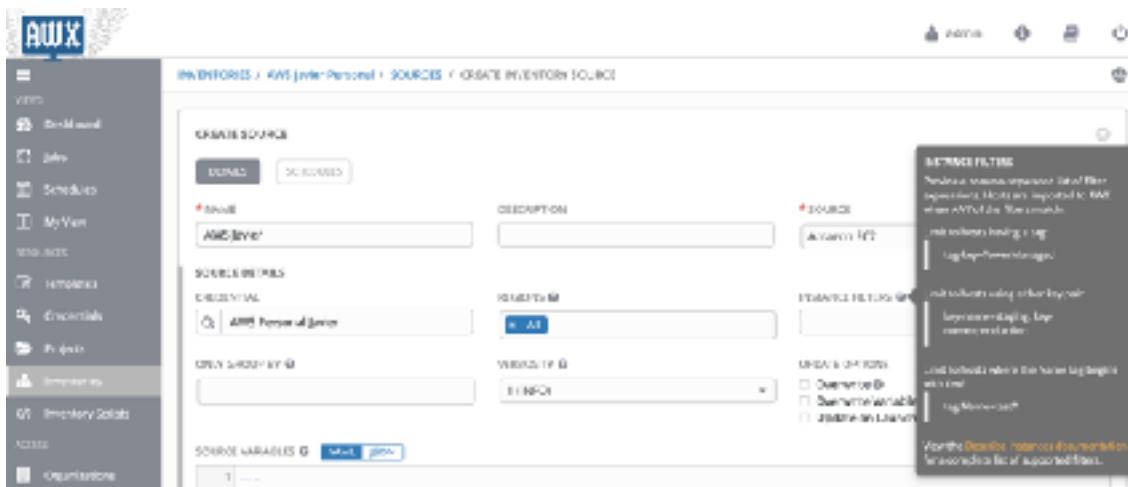
NOTE: VMware vCenter is also supported.

Choose your AWS credentials:

Select your AWS regions:



You can also apply filters:



Start the sync process:

Your inventory is synced:

Click on hosts and check that the hosts are imported:

You can check the latest sync details.

In addition, you can run ad-hoc commands as shown below:

Playbooks

In the following example, an AWS subnet is created.

This is the list of subnets before running my playbook:

The screenshot shows the AWS VPC Dashboard with a list of subnets. The table has the following data:

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv4 C
subnet-0a1ab1c2d3e4f	subnet-0a1ab1c2d3e4f	available	vpc-07cd831	172.31.12.0/28	4096	-
subnet-0b2cd3e4f5a6	subnet-0b2cd3e4f5a6	available	vpc-07cd831	172.31.0.0/30	4096	-
subnet-0c3de4f5a6b7	subnet-0c3de4f5a6b7	available	vpc-0823981a81030	172.16.48.0/11	16384	-
subnet-0d4ef6a7b8c9d	subnet-0d4ef6a7b8c9d	available	vpc-0922291a961030	172.16.128.0/18	7672	-
subnet-0e5f7c9d8e0f9	subnet-0e5f7c9d8e0f9	available	vpc-0922291a961030	172.16.120.0/18	7672	-

For testing purposes, I am just including the values hardcoded.

```

$ tree aws-create-subnet
.
└── aws-create-subnet.yml

1 file, 0 directories

```

The content of the selected file is:

```

# Create AWS Subnet for Testing Purposes
# ID: vpc-0a1ab1c2d3e4f
# Region: us-east-1
# State: present
# VPC: vpc-0a1ab1c2d3e4f
# CIDR: 172.31.12.0/28
# Name: test-vpc-subnet
# NetworkInterfaceName: test-interface
# AssociatePublicIpAddress: true
# SubnetId: subnet-0a1ab1c2d3e4f
# TagSet:
  - Name: Name
    Value: test-vpc-subnet
  - Name: NetworkInterfaceName
    Value: test-interface
  - Name: AssociatePublicIpAddress
    Value: true
  - Name: SubnetId
    Value: subnet-0a1ab1c2d3e4f
  - Name: TagSet
    Value: null

```

Lets create the template in AWX:

The screenshot shows the AWX UI for creating a new Ansible role. The role is named "aws-create-subnet". The configuration includes:

- NAME:** aws-create-subnet
- DESCRIPTION:** AWS Create Subnet
- ROLE TYPE:** Ansible Role
- INVENTORY:** AWS over VPC
- CREDENTIAL:** Ansible AWX
- PRIVILEGE:** Ansible Testing
- FORCES:** DEFAULT
- LIMIT:** None
- SUB TAGS:** None
- JOBTAGS:** None
- INTIMACY (W/ PING):** 1
- SHOW CHANGES:** On
- OPTIONS:**
 - Enable Privilege Escalation (Off)
 - Allow Provisioning of Unlocked (Off)
 - Enable Environment Jobs (Off)
 - Use Red Cache (Off)

Running the playbook:

NOTE: The output can be downloaded and shared:

The screenshot shows the Ansible AWX web interface. On the left is a sidebar with navigation links like 'Playbooks', 'Jobs', 'Schedules', etc. The main area has a title 'PLAY / 150 - AWS Create Subnet'. Below it, a table lists tasks: 'INFO' (Successful), 'INFO' (154/299 14:04:55), 'INFO' (154/299 14:07:52), and 'INFO' (154/299 14:07:52) with the message 'Ansible AWX: AWS Create Subnet'. A 'PLAYBOOK' section shows 'aws-create-subnet.yml' and a 'CREDENTIALS' section shows 'AWS Personal access'. The bottom part of the screen displays the log output for the play, which includes several Ansible tasks and their results.

```

PLAY [AWS Create Subnet] *****
[...] 1 taking AWS/available/available.cfg as config file
[...] 2 PLAY [Create AWS Subnet] *****
[...] 3    1: INFO [AWS Create Subnet] ...
[...] 4    2: INFO [AWS Create Subnet] ...
[...] 5    3: INFO [AWS Create Subnet] ...
[...] 6    4: INFO [AWS Create Subnet] ...
[...] 7    5: INFO [AWS Create Subnet] ...
[...] 8    6: INFO [AWS Create Subnet] ...
[...] 9    7: INFO [AWS Create Subnet] ...
[...] 10   8: INFO [AWS Create Subnet] ...
[...] 11   9: INFO [AWS Create Subnet] ...
[...] 12   10: INFO [AWS Create Subnet] ...
[...] 13   11: INFO [AWS Create Subnet] ...
[...] 14   12: INFO [AWS Create Subnet] ...
[...] 15   13: INFO [AWS Create Subnet] ...
[...] 16   14: INFO [AWS Create Subnet] ...
[...] 17   15: INFO [AWS Create Subnet] ...
[...] 18   16: INFO [AWS Create Subnet] ...
[...] 19   17: INFO [AWS Create Subnet] ...
[...] 20   18: INFO [AWS Create Subnet] ...
[...] 21   19: INFO [AWS Create Subnet] ...
[...] 22   20: INFO [AWS Create Subnet] ...
[...] 23   21: INFO [AWS Create Subnet] ...
[...] 24   22: INFO [AWS Create Subnet] ...
[...] 25   23: INFO [AWS Create Subnet] ...
[...] 26   24: INFO [AWS Create Subnet] ...
[...] 27   25: INFO [AWS Create Subnet] ...
[...] 28   26: INFO [AWS Create Subnet] ...
[...] 29   27: INFO [AWS Create Subnet] ...
[...] 30   28: INFO [AWS Create Subnet] ...
[...] 31   29: INFO [AWS Create Subnet] ...
[...] 32   30: INFO [AWS Create Subnet] ...
[...] 33   31: INFO [AWS Create Subnet] ...
[...] 34   32: INFO [AWS Create Subnet] ...
[...] 35   33: INFO [AWS Create Subnet] ...
[...] 36   34: INFO [AWS Create Subnet] ...
[...] 37   35: INFO [AWS Create Subnet] ...
[...] 38   36: INFO [AWS Create Subnet] ...
[...] 39   37: INFO [AWS Create Subnet] ...
[...] 40   38: INFO [AWS Create Subnet] ...
[...] 41   39: INFO [AWS Create Subnet] ...
[...] 42   40: INFO [AWS Create Subnet] ...
[...] 43   41: INFO [AWS Create Subnet] ...
[...] 44   42: INFO [AWS Create Subnet] ...
[...] 45   43: INFO [AWS Create Subnet] ...
[...] 46   44: INFO [AWS Create Subnet] ...
[...] 47   45: INFO [AWS Create Subnet] ...
[...] 48   46: INFO [AWS Create Subnet] ...
[...] 49   47: INFO [AWS Create Subnet] ...
[...] 50   48: INFO [AWS Create Subnet] ...
[...] 51   49: INFO [AWS Create Subnet] ...
[...] 52   50: INFO [AWS Create Subnet] ...
[...] 53   51: INFO [AWS Create Subnet] ...
[...] 54   52: INFO [AWS Create Subnet] ...
[...] 55   53: INFO [AWS Create Subnet] ...
[...] 56   54: INFO [AWS Create Subnet] ...
[...] 57   55: INFO [AWS Create Subnet] ...
[...] 58   56: INFO [AWS Create Subnet] ...
[...] 59   57: INFO [AWS Create Subnet] ...
[...] 60   58: INFO [AWS Create Subnet] ...
[...] 61   59: INFO [AWS Create Subnet] ...
[...] 62   60: INFO [AWS Create Subnet] ...
[...] 63   61: INFO [AWS Create Subnet] ...
[...] 64   62: INFO [AWS Create Subnet] ...
[...] 65   63: INFO [AWS Create Subnet] ...
[...] 66   64: INFO [AWS Create Subnet] ...
[...] 67   65: INFO [AWS Create Subnet] ...
[...] 68   66: INFO [AWS Create Subnet] ...
[...] 69   67: INFO [AWS Create Subnet] ...
[...] 70   68: INFO [AWS Create Subnet] ...
[...] 71   69: INFO [AWS Create Subnet] ...
[...] 72   70: INFO [AWS Create Subnet] ...
[...] 73   71: INFO [AWS Create Subnet] ...
[...] 74   72: INFO [AWS Create Subnet] ...
[...] 75   73: INFO [AWS Create Subnet] ...
[...] 76   74: INFO [AWS Create Subnet] ...
[...] 77   75: INFO [AWS Create Subnet] ...
[...] 78   76: INFO [AWS Create Subnet] ...
[...] 79   77: INFO [AWS Create Subnet] ...
[...] 80   78: INFO [AWS Create Subnet] ...
[...] 81   79: INFO [AWS Create Subnet] ...
[...] 82   80: INFO [AWS Create Subnet] ...
[...] 83   81: INFO [AWS Create Subnet] ...
[...] 84   82: INFO [AWS Create Subnet] ...
[...] 85   83: INFO [AWS Create Subnet] ...
[...] 86   84: INFO [AWS Create Subnet] ...
[...] 87   85: INFO [AWS Create Subnet] ...
[...] 88   86: INFO [AWS Create Subnet] ...
[...] 89   87: INFO [AWS Create Subnet] ...
[...] 90   88: INFO [AWS Create Subnet] ...
[...] 91   89: INFO [AWS Create Subnet] ...
[...] 92   90: INFO [AWS Create Subnet] ...
[...] 93   91: INFO [AWS Create Subnet] ...
[...] 94   92: INFO [AWS Create Subnet] ...
[...] 95   93: INFO [AWS Create Subnet] ...
[...] 96   94: INFO [AWS Create Subnet] ...
[...] 97   95: INFO [AWS Create Subnet] ...
[...] 98   96: INFO [AWS Create Subnet] ...
[...] 99   97: INFO [AWS Create Subnet] ...
[...] 100  98: INFO [AWS Create Subnet] ...
[...] 101  99: INFO [AWS Create Subnet] ...
[...] 102  100: INFO [AWS Create Subnet] ...
[...] 103  101: INFO [AWS Create Subnet] ...
[...] 104  102: INFO [AWS Create Subnet] ...
[...] 105  103: INFO [AWS Create Subnet] ...
[...] 106  104: INFO [AWS Create Subnet] ...
[...] 107  105: INFO [AWS Create Subnet] ...
[...] 108  106: INFO [AWS Create Subnet] ...
[...] 109  107: INFO [AWS Create Subnet] ...
[...] 110  108: INFO [AWS Create Subnet] ...
[...] 111  109: INFO [AWS Create Subnet] ...
[...] 112  110: INFO [AWS Create Subnet] ...
[...] 113  111: INFO [AWS Create Subnet] ...
[...] 114  112: INFO [AWS Create Subnet] ...
[...] 115  113: INFO [AWS Create Subnet] ...
[...] 116  114: INFO [AWS Create Subnet] ...
[...] 117  115: INFO [AWS Create Subnet] ...
[...] 118  116: INFO [AWS Create Subnet] ...
[...] 119  117: INFO [AWS Create Subnet] ...
[...] 120  118: INFO [AWS Create Subnet] ...
[...] 121  119: INFO [AWS Create Subnet] ...
[...] 122  120: INFO [AWS Create Subnet] ...
[...] 123  121: INFO [AWS Create Subnet] ...
[...] 124  122: INFO [AWS Create Subnet] ...
[...] 125  123: INFO [AWS Create Subnet] ...
[...] 126  124: INFO [AWS Create Subnet] ...
[...] 127  125: INFO [AWS Create Subnet] ...
[...] 128  126: INFO [AWS Create Subnet] ...
[...] 129  127: INFO [AWS Create Subnet] ...
[...] 130  128: INFO [AWS Create Subnet] ...
[...] 131  129: INFO [AWS Create Subnet] ...
[...] 132  130: INFO [AWS Create Subnet] ...
[...] 133  131: INFO [AWS Create Subnet] ...
[...] 134  132: INFO [AWS Create Subnet] ...
[...] 135  133: INFO [AWS Create Subnet] ...
[...] 136  134: INFO [AWS Create Subnet] ...
[...] 137  135: INFO [AWS Create Subnet] ...
[...] 138  136: INFO [AWS Create Subnet] ...
[...] 139  137: INFO [AWS Create Subnet] ...
[...] 140  138: INFO [AWS Create Subnet] ...
[...] 141  139: INFO [AWS Create Subnet] ...
[...] 142  140: INFO [AWS Create Subnet] ...
[...] 143  141: INFO [AWS Create Subnet] ...
[...] 144  142: INFO [AWS Create Subnet] ...
[...] 145  143: INFO [AWS Create Subnet] ...
[...] 146  144: INFO [AWS Create Subnet] ...
[...] 147  145: INFO [AWS Create Subnet] ...
[...] 148  146: INFO [AWS Create Subnet] ...
[...] 149  147: INFO [AWS Create Subnet] ...
[...] 150  148: INFO [AWS Create Subnet] ...

```

A text file with the output is generated:

This screenshot shows the Ansible AWX interface again. The left sidebar is identical. The main area shows a play run titled 'PLAY / 150 - AWS Create Subnet'. The log output is identical to the previous screenshot, showing the execution of the 'aws-create-subnet.yml' playbook. The right side of the screen now shows a 'Download Output' button next to the log viewer, indicating that the generated output file is available for download.

The new AWS subnet is created:

The screenshot shows the AWS VPC Dashboard. On the left, there's a sidebar with various VPC-related options like Virtual Private Cloud, New VPCs, Subnets, Route Tables, Internet Gateways, Express Gateways, DHCP Options Sets, Elastic IP, Endpoint Services, NAT Gateways, Peering Connections, Security, Network ACLs, and Security Groups. The main area displays a table of subnets. One subnet, 'subnet-01234567890abcdef0', is selected and highlighted in blue. The table columns include Subnet ID, State, VPC, IPv4 Range, and IPv6 Range. Below the table, there's a detailed view for the selected subnet, showing its Availability Zone (us-east-1a), Network ACL (acl-12345678), Auto Assign Public IP Address (No), and Owner (AWS Account 123456789012345678). There are tabs for Description, Firewall, Health Checks, Network ACL, Tags, and Sharing.

Let's take a glance of workflow templates and discuss how to inherit variables.

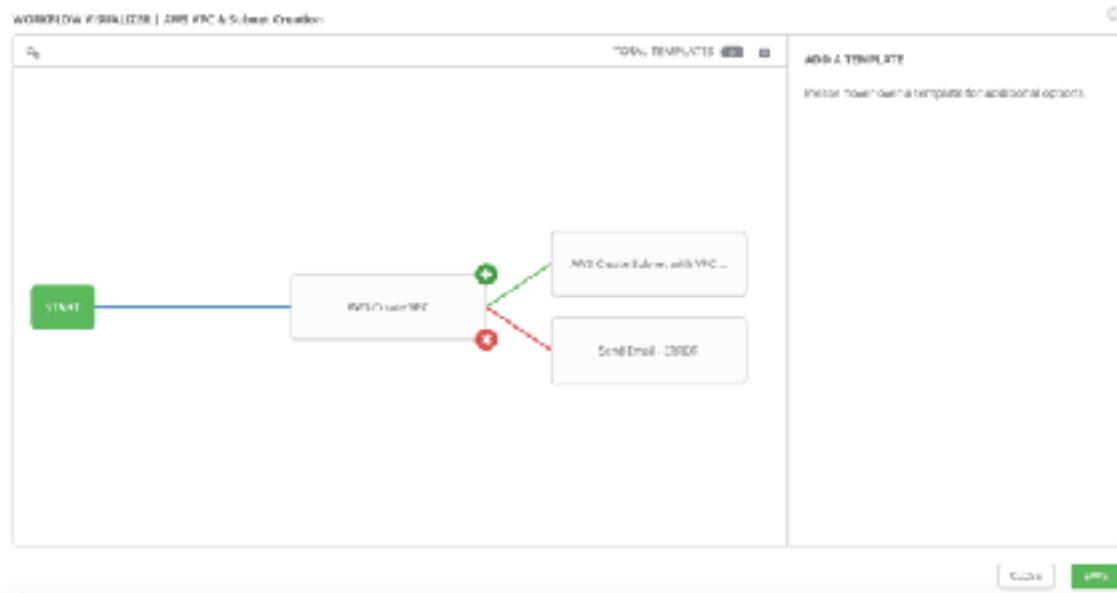
Click on Templates and add a new Workflow Template:

The screenshot shows the AWX interface. The left sidebar includes 'vars', 'Blueprints', 'Jobs', 'Variables', 'My View', 'Workflow', 'Templates' (which is selected), 'Credentials', 'Projects', and 'Organizations'. The main area is titled 'TEMPLATES' and shows a list of existing templates. A new template, 'AWS Create Subnet in VPC 12', is being created. Its details are shown in a modal: NAME is 'AWS Create Subnet in VPC 12', INVOCATION is 'Ansible Personal', OBJECT is 'Ansible Tasking', and DESCRIPTION is 'Ansible personal job'. It has two tags: 'Ansible' and 'Ansible Job'. The modal also includes tabs for 'Job Template' and 'Inventory Template'. At the bottom right of the modal are 'Cancel' and 'Save' buttons.

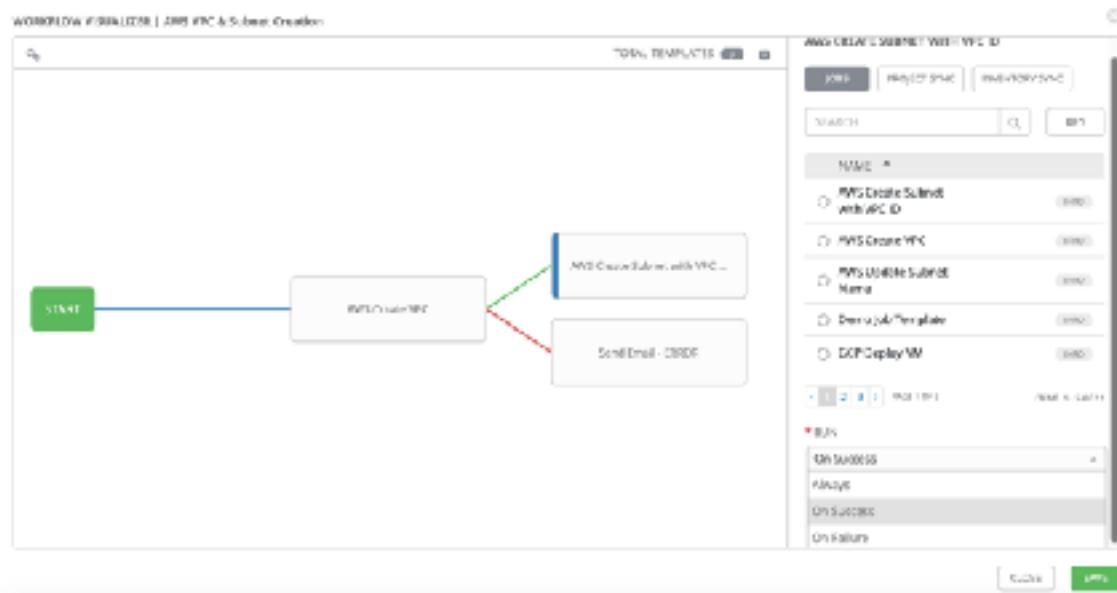
Once the name is saved, click on Workflow Visualizer

The screenshot shows the AWX interface again. The left sidebar is identical to the previous one. The main area is titled 'TEMPLATES / Ansible VPC Subnet Creation'. It shows a workflow titled 'Ansible VPC Subnet Creation'. The workflow has several steps: 'NAME' (Ansible VPC Subnet Creation), 'DESCRIPTION' (Ansible VPC Subnet Creation), 'ORGANIZATION' (None), 'TAGS' (Ansible, Ansible Job), and 'OPTIONS' (Enable Inheritance, Allow Inheritance). Below the workflow, there's a 'EDIT VARIABLES' section with tabs for 'YAML' and 'ENV'. At the bottom right are 'CANCEL' and 'SAVE' buttons. A tooltip 'Click here to open the workflow editor' is visible near the top right of the workflow area.

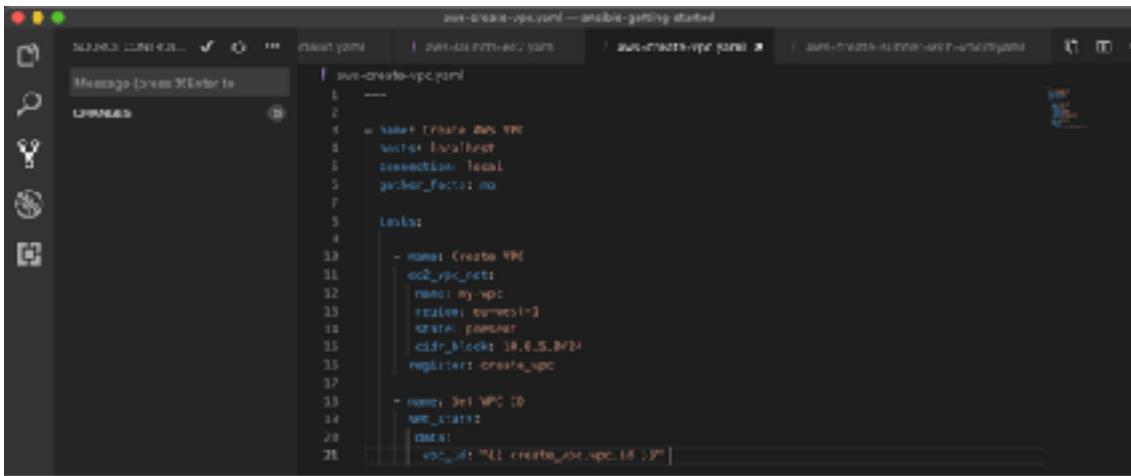
I have created the workflow below:



In order to add templates, click over a template and select a job, project sync or inventory sync.



In order to pass the "vpc_id" from the AWS Create VPC playbook to "AWS Create Subnet with VPC ID", you can use the "set_stats" module:

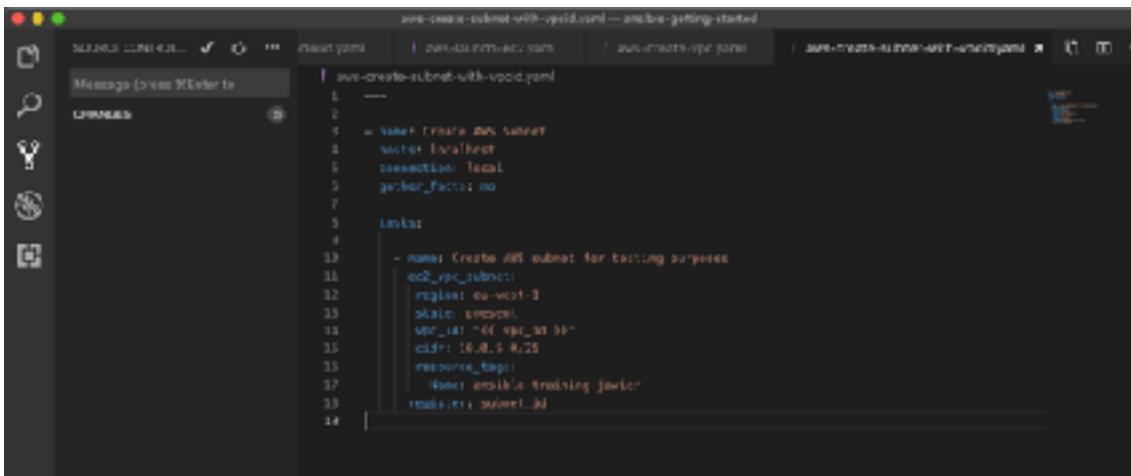


```

playbook: aws-create-vpc.yaml
version: 2
- name: Create VPC
  ec2_vpc:
    name: my-vpc
    routes:
      - route_id: rtr-00000000
        destination_cidr: 10.0.0.0/16
        gateway_id: gw-00000000
    register: vpc
  register: vpc
  result: <redacted>

```

The create subnet playbook inherits that value.



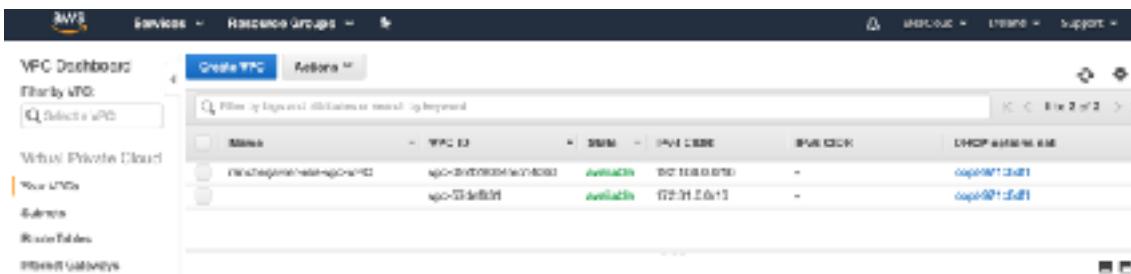
```

playbook: aws-create-subnet-with-vpc.yaml
version: 2
- name: Create VPC
  ec2_vpc:
    name: my-vpc
    routes:
      - route_id: rtr-00000000
        destination_cidr: 10.0.0.0/16
        gateway_id: gw-00000000
    register: vpc
  register: vpc
  result: <redacted>
- name: Create subnet for testing purposes
  ec2_vpc_subnet:
    vpc_id: vpc-00000000
    cidr: 10.0.1.0/25
    map_public_ip_on_launch: true
    subnet_name: subnet-00000000
  register: subnet
  result: <redacted>

```

Let's run the workflow.

This is the list of VPCs and subnets created.



Name	VPC ID	STATE	AVAILABILITY ZONE	SUBNETS	ENHANCED NETWORKING
aws-awx-aws-00000000	vpc-00000000	available	us-east-1a	0	disabled
aws-awx-aws-00000001	vpc-00000001	available	us-east-1a	0	disabled

Name	Subnet ID	Status	IP Range	Available IP's
subnet-Subnet01	subnet-Subnet01	available	172.31.12.6/30	4096
subnet-Subnet02	subnet-Subnet02	available	172.31.16.6/30	4096
subnet-Subnet03	subnet-Subnet03	available	172.31.32.32/30	4096
subnet-Subnet04	subnet-Subnet04	available	192.168.64.3/18	10240
subnet-Subnet05	subnet-Subnet05	available	192.168.136.8/26	16384
subnet-Subnet06	subnet-Subnet06	available	192.168.162.8/26	16384

Running the workflow:

First job is successfully completed:

Now, the second job is also executed:

The new VPC called "my-vpc" is created.

Name	VPC ID	Status	IPv4 CIDR	IPv6 CIDR	DHCP Options Set
my-vpc	vpc-052e123456789012	available	10.0.0.0/16	-	Ansible-v6f1
Ansible-Subnet-DRBD-VPC	vpc-052f906491d45678	available	10.0.0.0/16	-	Ansible-v6f1
Ansible-Subnet-DRBD	vpc-052f906491d45678	available	10.0.1.0/16	10.0.1.0/16	Ansible-v6f1

A new subnet called "ansible-training-javier" is also provisioned in the VPC created in the same workflow.

Name	Subnet ID	Status	VPC	IPv4 CIDR	Available IPv4
subnet-0d441a78ac21d	subnet-0d441a78ac21d	available	vpc-052e123456789012	172.31.12.0/20	4096
subnet-072db4f4	subnet-072db4f4	available	vpc-052f906491d45678	172.31.16.0/20	4096
subnet-0c953bb1	subnet-0c953bb1	available	vpc-052f906491d45678	172.31.2.0/20	4096
ansible-training-javier	subnet-0d441a78ac21d	available	vpc-052e123456789012	10.0.1.0/16	128
Ansible-Subnet-DRBD-VPC	subnet-072db4f491d45678	available	vpc-052f906491d45678	10.0.0.0/16	15360
Ansible-Subnet-DRBD	subnet-0c953bb191d45678	available	vpc-052f906491d45678	10.0.1.0/16	15360

Deploying to Google Cloud Platform

Inventory

You will need to install additional packages in order to gather information about GCP-based hosts:

```
pip install requests google-auth apache-libcloud
```

First, you have to create a new service account for Ansible. Open the Google Cloud dashboard and select your project from the top header.

Select IAM & admin then go to the Service Accounts section. You can create a new service account by clicking on the Create Service Account button at the top.

From the pop-up, we can create a service account and download the JSON credentials file as shown below.

Email	Name	Description	Key ID	Key type	Created
gcp-sa-google-cloud@...com	gcp-sa-google-cloud	Google Cloud Service Account	test-21012019-0000000000000000	JSON	21 Jan 2019
gcp-sa-test@...com	gcp-sa-test	Google Cloud Service Account	gcp-sa-test-0000000000000000	JSON	

Considering Ansible Engine, once we have the credentials we should put them in roles/gce/vars/secrets.yml:

```
---
gs_access_key: XXXXXXXXXXXXXXXXXXXXXXX
gs_secret_key: XXXXXXXXXXXXXXXXXXXXXXX
```

Now, we encrypt them:

```
$ ansible-vault encrypt roles/gce/vars/secrets.yml
```

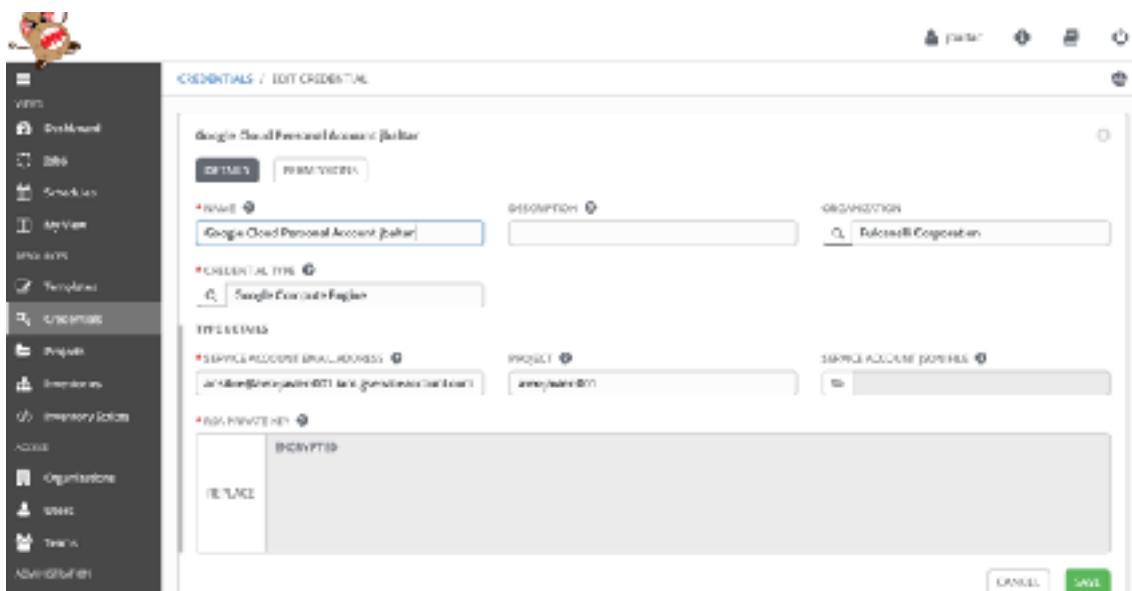
From AWX perspective, the configuration is similar to AWS.

Click on credentials and choose Google Compute Engine.



Add the service account details retrieved from the file downloaded:

Once the credential is saved, it is automatically encrypted.



You can generate a ssh key for Ansible and uploaded it to the Google Cloud console:

Click on Compute Engine > Metadata > SSH Keys and add the new key.

The inventory configuration is similar to the one we did for AWS.

Click on Inventories > Sources and add a Google Compute Engine source type.

As an example, the following playbook launched a new VM in GCP.

```

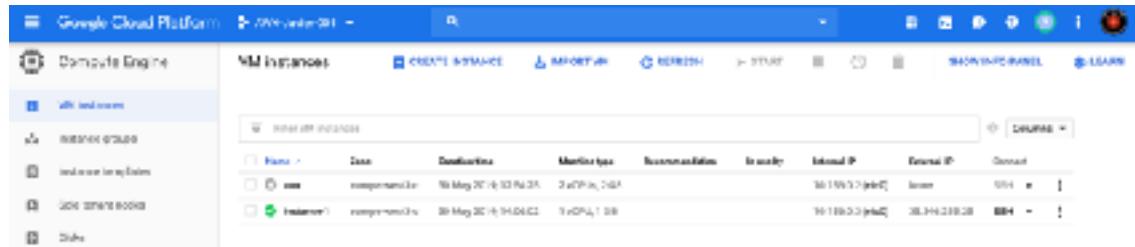
- name: Create Google Cloud VM
  hosts: localhost
  gather_facts: no
  connection: local

  vars:
    machine_type: n1-standard-1
    image: centos7
    service_account_email: ansible@awx-
javier-001.iam.gserviceaccount.com
    credentials_file: /home/googlecloud/playbooks/key.json
    project_id: AWX-Javier-001

  tasks:
    - name: Launch VM
      gce:
        instance_names: instance-1
        machine_type: "{{ machine_type }}"
        image: "{{ image }}"
        service_account_email: "{{ service_account_email }}"
        credentials_file: "{{ credentials_file }}"
        project_id: "{{ project_id }}"

```

Running the playbook, the new virtual machine is created:



The screenshot shows the Google Cloud Platform Compute Engine VM Instances page. The left sidebar lists projects: 'Compute Engine' (selected), 'VM instances', 'Machine types', 'Resource Manager', 'Cloud Build', and 'Cloud Functions'. The main area displays a table of VM instances with the following data:

Name	Date	Condition	Machine type	Resource manager	Is ready	Internal IP	External IP	Connect
vm	20 May 2016 10:12 PM (UK)	2 vCPUs, 2GB	1 vCPU, 1 GB		Ready	10.100.0.2 [edit]	None	VM4
instance1	20 May 2016 10:14:02	2 vCPUs, 2GB	1 vCPU, 1 GB		Ready	10.100.0.3 [edit]	38.34.218.28	VM4