AWX Server Setup

Contents

[1. Prerequisites: 2](#_Toc45278624)

[2. AWX Installation Procedures 3](#_Toc45278625)

[2.1 Obtain Local IP Address 3](#_Toc45278626)

[2.2 Install AWX Dependencies 4](#_Toc45278627)

[2.3 Download AWX Source 5](#_Toc45278628)

[2.4 AWX Pre-Install Configuration 6](#_Toc45278629)

[2.5 Install AWX 7](#_Toc45278630)

[2.6 Verify AWX Installation 8](#_Toc45278631)

[2.7 Configure and Install Web Server 9](#_Toc45278632)

[2.8 Additional Configurations 11](#_Toc45278633)

[2.8.1 Ansible SSH Configuration 11](#_Toc45278634)

[2.8.2 Docker Service Configuration 11](#_Toc45278635)

[3. Ansible CLI Tutorial 12](#_Toc45278636)

[3.1 Test Switch Connectivity 12](#_Toc45278637)

[3.2 Run Tasks with *ansible* Command 13](#_Toc45278638)

[3.3 Run Sample Playbooks with *ansible-playbook* Command 14](#_Toc45278639)

[4. AWX Basic Setup 16](#_Toc45278640)

[4.1 Initial GUI Login 16](#_Toc45278641)

[4.2 Create Projects 17](#_Toc45278642)

[4.3 Add Credentials 18](#_Toc45278643)

[4.4 Create Inventories 20](#_Toc45278644)

[4.5 Creating Job Templates 22](#_Toc45278645)

[5. Vim Basics 31](#_Toc45278646)

[6. References 32](#_Toc45278647)

[7. Additional Resources 32](#_Toc45278648)

# 1. Prerequisites:

* Fresh Ubuntu 18.04 Server install
* RAM: 3+ GB
* Disk Space: 40+ GB
* Bridged network adapter (if using a VM)

The following tutorial will walk through a setup of Ubuntu Server 18.04:

<https://www.fosslinux.com/6406/how-to-install-ubuntu-server-18-04-lts.htm>

# 2. AWX Installation Procedures

## 2.1 Obtain Local IP Address

Use the following command to obtain the IP address:

|  |
| --- |
| **ip addr** |

Make a note of the server IP address shown in the output. For example:

|  |
| --- |
| awxuser@awxserver:~$ ip addr  1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  inet 127.0.0.1/8 scope host lo  valid\_lft forever preferred\_lft forever  inet6 ::1/128 scope host  valid\_lft forever preferred\_lft forever  2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000  link/ether 00:0c:29:67:b2:b7 brd ff:ff:ff:ff:ff:ff  inet 192.168.41.66/23 brd 192.168.41.255 scope global dynamic ens33  valid\_lft 85833sec preferred\_lft 85833sec  inet6 fe80::20c:29ff:fe67:b2b7/64 scope link  valid\_lft forever preferred\_lft forever  awxuser@awxserver:~$ |

## 2.2 Install AWX Dependencies

SSH into the server using PuTTY.

When logged into the server using PuTTY, copy and paste the following commands into the PuTTY window:

|  |
| --- |
| sudo apt update -y &&  sudo apt install docker.io -y &&  sudo apt install python3-pip -y &&  sudo pip3 install docker-compose==1.26.0 &&  sudo pip3 install ansible &&  ansible --version &&  sudo docker version |

This will install the initial dependencies for the AWX server and check the version of Ansible and Docker.

The following results should be displayed:

|  |
| --- |
| ansible 2.9.10  config file = None  configured module search path = ['/home/awxuser/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']  ansible python module location = /usr/local/lib/python3.6/dist-packages/ansible  executable location = /usr/local/bin/ansible  python version = 3.6.9 (default, Apr 18 2020, 01:56:04) [GCC 8.4.0]  Client:  Version: 19.03.6  API version: 1.40  Go version: go1.12.17  Git commit: 369ce74a3c  Built: Fri Feb 28 23:45:43 2020  OS/Arch: linux/amd64  Experimental: false  Server:  Engine:  Version: 19.03.6  API version: 1.40 (minimum version 1.12)  Go version: go1.12.17  Git commit: 369ce74a3c  Built: Wed Feb 19 01:06:16 2020  OS/Arch: linux/amd64  Experimental: false  containerd:  Version: 1.3.3-0ubuntu1~18.04.2  GitCommit:  runc:  Version: spec: 1.0.1-dev  GitCommit:  docker-init:  Version: 0.18.0  GitCommit: |

Note: The first time the **ansible** command is run, the '~/.ansible' directory is created. It is important that this directory is created with the local user as owner as a root owner can break certain functions. The **ansible** command above accomplishes this task as well as showing the Ansible version.

## 2.3 Download AWX Source

Download the AWX v13.0.0 server source code and go to the installation directory with the following commands:

|  |
| --- |
| **git clone https://github.com/ansible/awx -b 13.0.0**  **cd awx/installer** |

## 2.4 AWX Pre-Install Configuration

Generate a new secret key for AWX using the following command:

|  |
| --- |
| **openssl rand -hex 32** |

Copy the generated secret key to the clipboard.

Edit the 'inventory' configuration file:

|  |
| --- |
| **vim inventory** |

Change the value of 'secret\_key' from to the output of the 'openssl' command above.

Change the 'postgres\_data\_dir' to the *"/var/lib/pgdocker"* directory:

|  |
| --- |
| postgres\_data\_dir="/var/lib/pgdocker" |

Change the 'host\_port' to *8080*:

|  |
| --- |
| host\_port=8080 |

Change the 'docker\_compose\_dir' to the "*/var/lib/awx*" directory:

|  |
| --- |
| docker\_compose\_dir="/var/lib/awx" |

Uncomment the 'project\_data\_dir' and leave the default value:

|  |
| --- |
| project\_data\_dir=/var/lib/awx/projects |

Change the credentials for the 'pg\_password', 'admin\_user' and 'admin\_password' with your own password credentials. Do not include special characters in the password as this can break the install.

To see the active config entries, issue the following command:

|  |
| --- |
| **grep -v "^#" inventory | grep -v "^$"** |

Results similar to the following should be displayed:

|  |
| --- |
| localhost ansible\_connection=local ansible\_python\_interpreter="/usr/bin/env python3"  [all:vars]  dockerhub\_base=ansible  awx\_task\_hostname=awx  awx\_web\_hostname=awxweb  postgres\_data\_dir="/var/lib/pgdocker"  host\_port=8080  host\_port\_ssl=443  docker\_compose\_dir="/var/lib/awx"  pg\_username=awx  pg\_password=CallM3Snak3  pg\_database=awx  pg\_port=5432  admin\_user=awxuser  admin\_password=awxpassword  create\_preload\_data=True  secret\_key=aa27d3da1b2bb4192265939e082f8e7ae1e30cd8dce1cfe943f564ca7ca3fbe8  project\_data\_dir=/var/lib/awx/projects |

## 2.5 Install AWX

Install the AWX server with the following command:

|  |
| --- |
| **sudo ansible-playbook -i inventory install.yml** |

This is a snippet of what the result should look like if the install completed successfully:

|  |
| --- |
| TASK [local\_docker : Copy task image to docker execution] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  skipping: [localhost]  TASK [local\_docker : Load web image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  skipping: [localhost]  TASK [local\_docker : Load task image] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  skipping: [localhost]  TASK [local\_docker : Set full image path for local install] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  skipping: [localhost]  TASK [local\_docker : Set DockerHub Image Paths] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ok: [localhost]  TASK [local\_docker : Create /var/lib/awx directory] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost]  TASK [local\_docker : Create Docker Compose Configuration] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost] => (item=environment.sh)  changed: [localhost] => (item=credentials.py)  changed: [localhost] => (item=docker-compose.yml)  changed: [localhost] => (item=nginx.conf)  TASK [local\_docker : Render SECRET\_KEY file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost]  TASK [local\_docker : Start the containers] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost]  TASK [local\_docker : Update CA trust in awx\_web container] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost]  TASK [local\_docker : Update CA trust in awx\_task container] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  changed: [localhost]  PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  localhost : ok=14 changed=6 unreachable=0 failed=0 skipped=100 r escued=0 ignored=0  awxuser@awxserver:~/awx/installer$ |

## 2.6 Verify AWX Installation

With the AWX server installed, this is a good time to check for problems before proceeding. Issue the following command to display the AWX log:

|  |
| --- |
| **sudo docker logs -f awx\_task** |

The logs will continuously display until Ctrl-c is pressed. Something similar to the following would normally be displayed:

|  |
| --- |
| 2020-02-11 14:00:41,345 DEBUG awx.main.dispatch publish awx.main.tasks.awx\_periodic\_scheduler(46570e74-f9c5-4181-a6cd-45cbf317ca78, queue=awx\_private\_queue)  [2020-02-11 14:00:41,345: DEBUG/Process-1] publish awx.main.tasks.awx\_periodic\_scheduler(46570e74-f9c5-4181-a6cd-45cbf317ca78, queue=awx\_private\_queue)  2020-02-11 14:00:41,379 DEBUG awx.main.dispatch delivered 46570e74-f9c5-4181-a6cd-45cbf317ca78 to worker[190] qsize 0  2020-02-11 14:00:41,384 DEBUG awx.main.dispatch task 46570e74-f9c5-4181-a6cd-45cbf317ca78 starting awx.main.tasks.awx\_periodic\_scheduler(\*[])  2020-02-11 14:00:41,399 DEBUG awx.main.tasks Starting periodic scheduler  2020-02-11 14:00:41,402 DEBUG awx.main.tasks Last scheduler run was: 2020-02-11 14:00:11.530984+00:00  2020-02-11 14:00:41,419 DEBUG awx.main.dispatch task 46570e74-f9c5-4181-a6cd-45cbf317ca78 is finished  2020-02-11 14:00:51,640 DEBUG awx.main.dispatch publish awx.main.scheduler.tasks.run\_task\_manager(5544bef6-bdfe-4d59-b084-cbea735c4104, queue=awx\_private\_queue)  [2020-02-11 14:00:51,640: DEBUG/Process-1] publish awx.main.scheduler.tasks.run\_task\_manager(5544bef6-bdfe-4d59-b084-cbea735c4104, queue=awx\_private\_queue)  2020-02-11 14:00:51,677 DEBUG awx.main.dispatch delivered 5544bef6-bdfe-4d59-b084-cbea735c4104 to worker[193] qsize 0  2020-02-11 14:00:51,680 DEBUG awx.main.dispatch task 5544bef6-bdfe-4d59-b084-cbea735c4104 starting awx.main.scheduler.tasks.run\_task\_manager(\*[])  2020-02-11 14:00:51,684 DEBUG awx.main.scheduler Running Tower task manager.  2020-02-11 14:00:51,698 DEBUG awx.main.scheduler Starting Scheduler  2020-02-11 14:00:51,725 DEBUG awx.main.dispatch task 5544bef6-bdfe-4d59-b084-cbea735c4104 is finished  RESULT 2  OKREADY  2020-02-11 14:01:10,907 DEBUG awx.main.dispatch publish awx.main.tasks.cluster\_node\_heartbeat(bae84654-267c-47d5-840c-8cb42fc1f48e, queue=awx)  [2020-02-11 14:01:10,907: DEBUG/Process-1] publish awx.main.tasks.cluster\_node\_heartbeat(bae84654-267c-47d5-840c-8cb42fc1f48e, queue=awx)  2020-02-11 14:01:10,951 DEBUG awx.main.dispatch delivered bae84654-267c-47d5-840c-8cb42fc1f48e to worker[191] qsize 0  2020-02-11 14:01:10,957 DEBUG awx.main.dispatch task bae84654-267c-47d5-840c-8cb42fc1f48e starting awx.main.tasks.cluster\_node\_heartbeat(\*[])  2020-02-11 14:01:10,961 DEBUG awx.main.tasks Cluster node heartbeat task.  2020-02-11 14:01:10,981 DEBUG awx.main.dispatch task bae84654-267c-47d5-840c-8cb42fc1f48e is finished |

An indication of a bad install would be a large amount of repeating ERROR logs and stack traces.

## 2.7 Configure and Install Web Server

Install nginx and navigate to the nginx config directory with the following commands:

|  |
| --- |
| **sudo apt install nginx -y**  **cd /etc/nginx/sites-available/** |

Create a new virtual host configuration called "*awx*" using vim:

|  |
| --- |
| **sudo vim awx** |

Substitute the highlighted IP addresses before pasting the following configuration:

|  |
| --- |
| server {    listen 80;    server\_name 192.168.41.66;    add\_header Strict-Transport-Security max-age=2592000;    rewrite ^ https://$server\_name$request\_uri? permanent; }  server {     listen 443 ssl http2;     server\_name 192.168.41.66;      access\_log /var/log/nginx/awx.access.log;     error\_log /var/log/nginx/awx.error.log;      ssl on;     ssl\_certificate /etc/nginx/ssl/fullchain.pem;     ssl\_certificate\_key /etc/nginx/ssl/privkey.pem;     ssl\_session\_timeout 5m;     ssl\_ciphers EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES:!MD5;     ssl\_protocols TLSv1.2;     ssl\_prefer\_server\_ciphers on;  location / {     proxy\_http\_version 1.1;     proxy\_set\_header Host $host;     proxy\_set\_header X-Real-IP $remote\_addr;     proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;     proxy\_set\_header X-Forwarded-Proto $scheme;     proxy\_set\_header Upgrade $http\_upgrade;     proxy\_set\_header Connection "upgrade";     proxy\_pass http://192.168.41.66:8080/;     } } |

Activate the "*awx*" virtual host with the following command:

|  |
| --- |
| **sudo ln -s /etc/nginx/sites-available/awx /etc/nginx/sites-enabled/** |

Create the */etc/nginx/ssl* directory and generate the self-signed SSL certificate and key with the following commands:

|  |
| --- |
| **sudo mkdir /etc/nginx/ssl**  **sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \**  **-keyout /etc/nginx/ssl/privkey.pem \**  **-out /etc/nginx/ssl/fullchain.pem** |

A series of questions will be asked during the creation of the certificate and key:

|  |
| --- |
| awxuser@awxserver:/etc/nginx/sites-available$ sudo mkdir /etc/nginx/ssl  awxuser@awxserver:/etc/nginx/sites-available$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \  > -keyout /etc/nginx/ssl/privkey.pem \  > -out /etc/nginx/ssl/fullchain.pem  Can't load /home/awxuser/.rnd into RNG  140217327120832:error:2406F079:random number generator:RAND\_load\_file:Cannot open file:../crypto/rand/randfile.c:88:Filename=/home/awxuser/.rnd  Generating a RSA private key  ...................................+++++  .............................................................................................+++++  writing new private key to '/etc/nginx/ssl/privkey.pem'  -----  You are about to be asked to enter information that will be incorporated  into your certificate request.  What you are about to enter is what is called a Distinguished Name or a DN.  There are quite a few fields but you can leave some blank  For some fields there will be a default value,  If you enter '.', the field will be left blank.  -----  Country Name (2 letter code) [AU]:US  State or Province Name (full name) [Some-State]:Virginia  Locality Name (eg, city) []:Norfolk  Organization Name (eg, company) [Internet Widgits Pty Ltd]:  Organizational Unit Name (eg, section) []:  Common Name (e.g. server FQDN or YOUR name) []:192.168.41.66  Email Address []:  awxuser@awxserver:/etc/nginx/sites-available$ |

The nginx setup can be tested with the following command:

|  |
| --- |
| **sudo nginx -t** |

If the test was successful, the results will look like this:

|  |
| --- |
| awxuser@awxserver:/etc/nginx/sites-available$ sudo nginx -t  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  awxuser@awxserver:/etc/nginx/sites-available$ |

If there are no errors, restart the nginx service:

|  |
| --- |
| **sudo systemctl restart nginx** |

## 2.8 Additional Configurations

### 2.8.1 Ansible SSH Configuration

When running playbooks from the CLI against a host, it is desirable to disable the interactive prompt that will cause the process to fail when the SSH client connects to a new host.

First create a new Ansible config file:

|  |
| --- |
| **vim ~/.ansible.cfg** |

Then paste the following into the file:

|  |
| --- |
| [defaults]  host\_key\_checking = False |

To ensure Ansible is using this config, issue the following command:

|  |
| --- |
| **sudo ansible –version** |

The following results should be displayed:

|  |
| --- |
| ansible 2.9.10  config file = /home/awxuser/.ansible.cfg  configured module search path = ['/home/awxuser/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']  ansible python module location = /usr/local/lib/python3.6/dist-packages/ansible  executable location = /usr/local/bin/ansible  python version = 3.6.9 (default, Apr 18 2020, 01:56:04) [GCC 8.4.0] |

### 2.8.2 Docker Service Configuration

By default, Docker must be started manually. To make Docker start on boot, issue the following command:

|  |
| --- |
| **sudo systemctl enable docker** |

# 3. Ansible CLI Tutorial

## 3.1 Test Switch Connectivity

To test credentials and connectivity to the switch, use the **ssh** command with this syntax:

|  |
| --- |
| ssh <user>@<hostip> -o KexAlgorithms=+diffie-hellman-group1-sha1 |

The following output should be displayed:

|  |
| --- |
| awxuser@awxserver:~$ ssh awxuser@192.168.20.72 -o KexAlgorithms=+diffie-hellman-group1-sha1  The authenticity of host '192.168.20.72 (192.168.20.72)' can't be established.  RSA key fingerprint is SHA256:4Vytdyb7L3DiRc+3CpcvqUGbeq1vCSoyFkg91n7htNM.  Are you sure you want to continue connecting (yes/no)? yes  Warning: Permanently added '192.168.20.72' (RSA) to the list of known hosts.  Password:  SSH@NLAB-U01-AS-01>enable  Password:  SSH@NLAB-U01-AS-01# |

While in the switch display the config, then exit with the following commands:

|  |
| --- |
| skip  show running-config  exit  exit |

## 3.2 Run Tasks with *ansible* Command

A single **ansible** command can be run from the command line as an alternative to gathering information from switches manually:

|  |
| --- |
| **ansible all \**  **-i 192.168.20.72, \**  **-c network\_cli \**  **-k -K -b \**  **--become-method enable \**  **-e ansible\_network\_os=icx \**  **-m icx\_facts -a gather\_subset=all \**  **-u awxuser** |

Command breakdown:

* Host group we are connecting to (all)
* -i specifies an inventory followed by either target device with a trailing comma.
* -k ask you for the ssh password
* -K ask you for the become (privileged user) password
* -b run operations with become
* --become-method priviledge escalation method
* -e sets an extra variable used by ansible, in this case setting the ansible\_network\_os to 'icx'
* -m name of the python module to execute
* -a sets an argument to pass to the python module
* -u login username for the switch

## 3.3 Run Sample Playbooks with *ansible-playbook* Command

Tasks run with the **ansible** command can be written into a YAML playbook that can be run with the **ansible-playbook** command and the AWX GUI.

First create a new playbook:

|  |
| --- |
| **vim ~/pbfacts.yml** |

Then copy and paste the following:

|  |
| --- |
| ---  - hosts: all  connection: network\_cli  gather\_facts: no  vars:  ansible\_network\_os: icx  ansible\_become: true  ansible\_become\_method: enable  tasks:  - name: Gather Switch Info  icx\_facts:  gather\_subset: all  register: output  - name: Show Info  debug:  msg: "{{ output }}" |

In this playbook you can see the command 'register' defines a variable named 'output' as the output of the task above it. Variables can be dereferenced using '{{ }}'.

To run this playbook, enter the following:

|  |
| --- |
| **ansible-playbook -i 192.168.20.72, -k -K -u awxuser pbfacts.yml** |

Let's say VLAN 999 is to be configured on a switch. To create the VLAN manually, the following commands might be entered directly on the switch:

|  |
| --- |
| vlan 999 name U\_USER\_999 by port  tagged ethe 1/2/1 to 1/2/2  spanning-tree  !  write memory |

As an alternative to manually running the previous commands on a switch, a playbook can be created that can accomplish the same task.

First create the playbook file:

|  |
| --- |
| **vim ~/pbvlan.yml** |

Then copy and pasted the following:

|  |
| --- |
| ---  - hosts: all  connection: network\_cli  gather\_facts: no  vars:  ansible\_network\_os: icx  ansible\_become: true  ansible\_become\_method: enable  tasks:  - name: create vlan  icx\_config:  lines: |  vlan {{ vlan\_number }} name {{ vlan\_name }}  tagged {{ trunk\_ports }}  spanning-tree  write memory |

This playbook dereferences three variables that are not defined in the playbook. These variables can be defined with '-e' on the command line when running a playbook:

|  |
| --- |
| **ansible-playbook -i 192.168.20.72, -k -K \**  **-u awxuser \**  **-e trunk\_ports="'e1/2/1 e1/2/2'" \**  **-e vlan\_name=U\_USER\_999 \**  **-e vlan\_number=999 \**  **pbvlan.yml** |

In order to run the playbooks from the AWX GUI, they will need to be in a projects folder. First create a new project directory:

|  |
| --- |
| **sudo mkdir /var/lib/awx/projects/ruckus** |

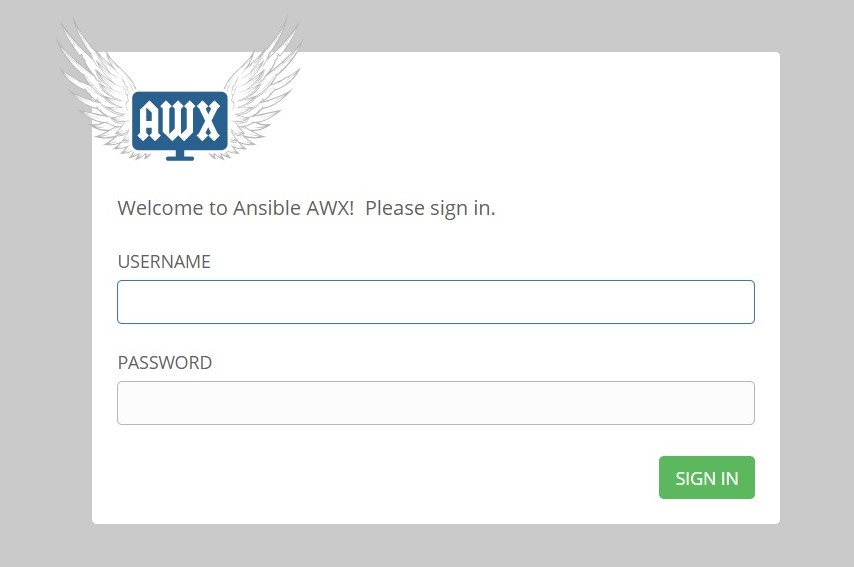
Then copy the playbooks into the project directory:

|  |
| --- |
| **sudo cp ~/pbfacts.yml ~/pbvlan.yml /var/lib/awx/projects/ruckus** |

# 4. AWX Basic Setup

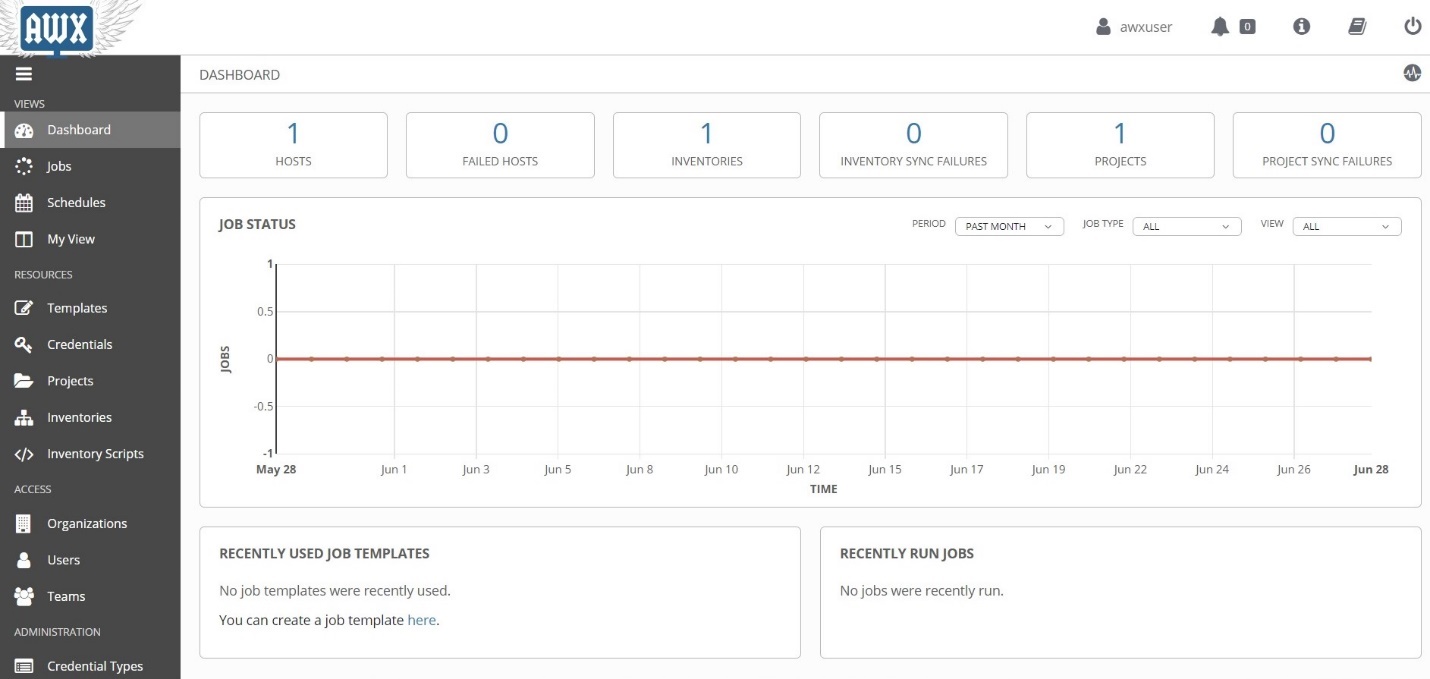
## 4.1 Initial GUI Login

At this point the AWX server setup should be finished and the AWX web GUI can be accessed from a browser by entering the AWX IP into the address bar. The AWX login page should display:



Login with the username and password that was set in the Inventory file (admin\_user & admin\_password).

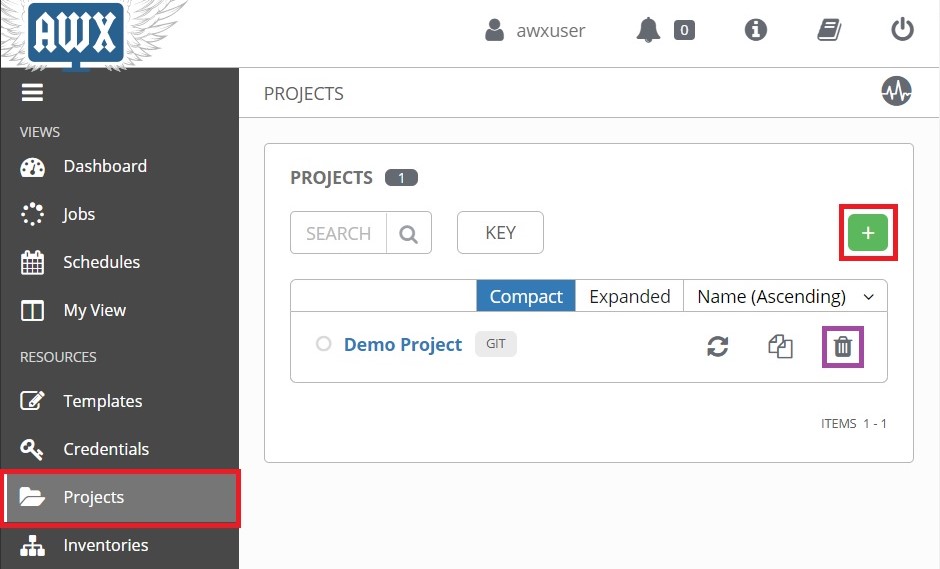
Upon login, the dashboard will display with the navigation bar on the left:



## 4.2 Create Projects

A project is a logical collection of playbooks. The following will guide you through the creation of a project whose playbooks have been manually placed in the project base path of the AWX server.

Click **Projects** from the left navigation menu. Click the delete () button to optionally remove the **Demo Project**. Then click the **+** button on the right to add a new project:

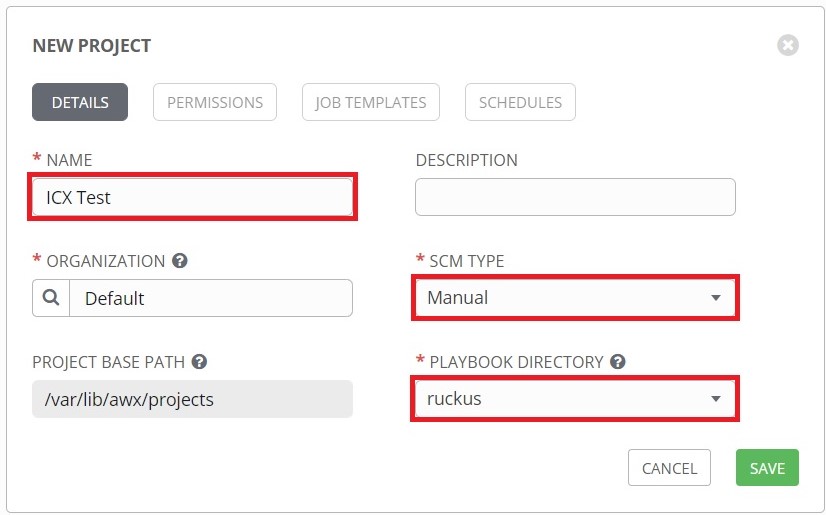


The **New Project** page will display. Complete the following fields:

**NAME:** Create a name for the project. Here it is named "*ICX Test*".

**SCM TYPE:** Select **Manual**.

**PLAYBOOK DIRECTORY:** Select the name of the folder you copied the playbooks into.

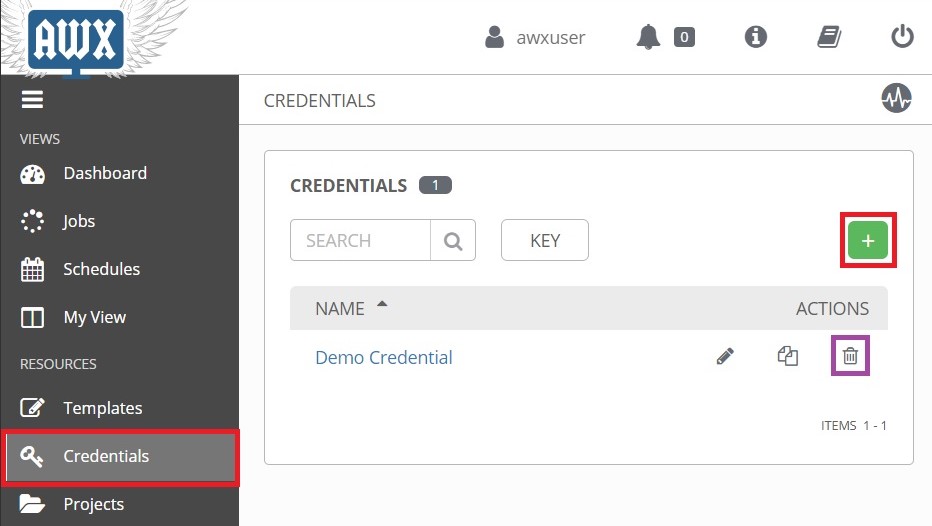


When finished, click the **Save** button.

## 4.3 Add Credentials

For AWX to log into a device, you will need to create a set of credentials that can be used in a playbook.

Click **Credentials** from the left navigation menu. Click the delete () button to optionally delete the **Demo Credential**. Then click the **+** button to add a credential set:

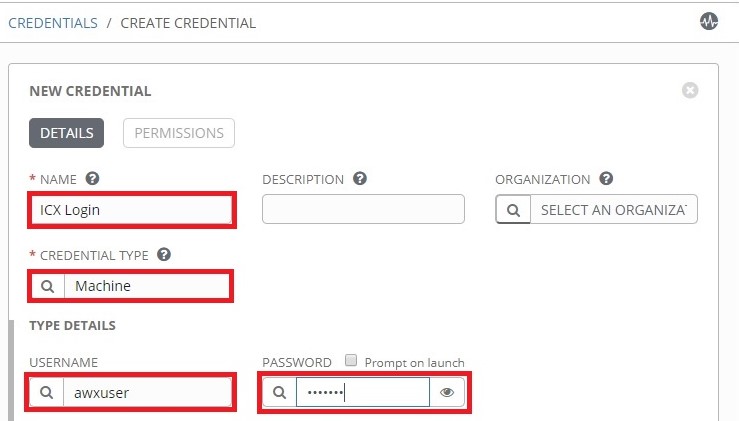


The **New Credential** page will display. Complete the following fields:

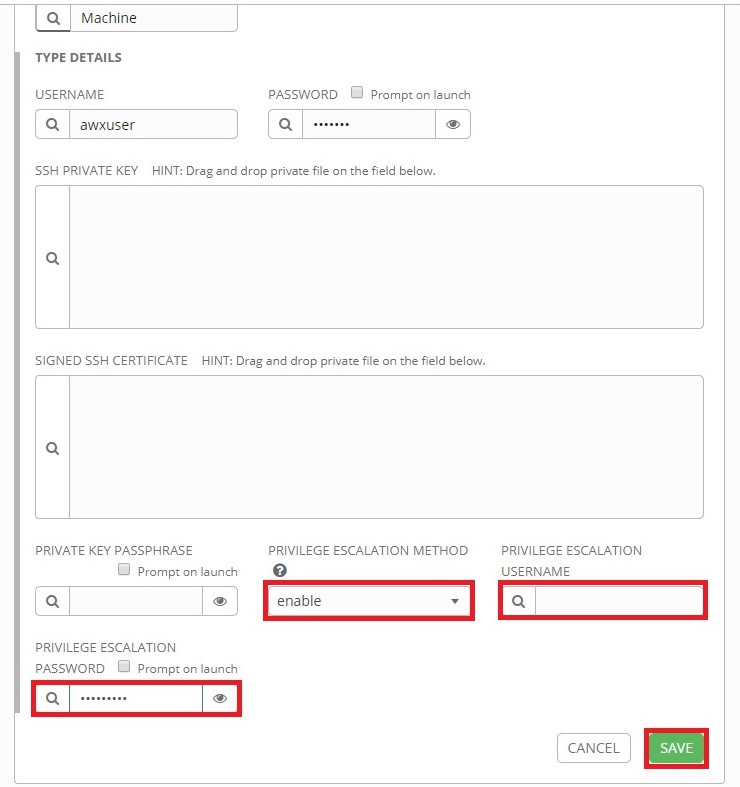
**NAME**: Create a name for the credential set. Here it is named "*ICX Login*"

**CREDENTIAL TYPE:** Select **Machine**.

**USERNAME** & **PASSWORD:** Enter the username and password used to login to the switch.



Scroll to the bottom of the **New Credential** page. Here you can set the enable username and/or password:

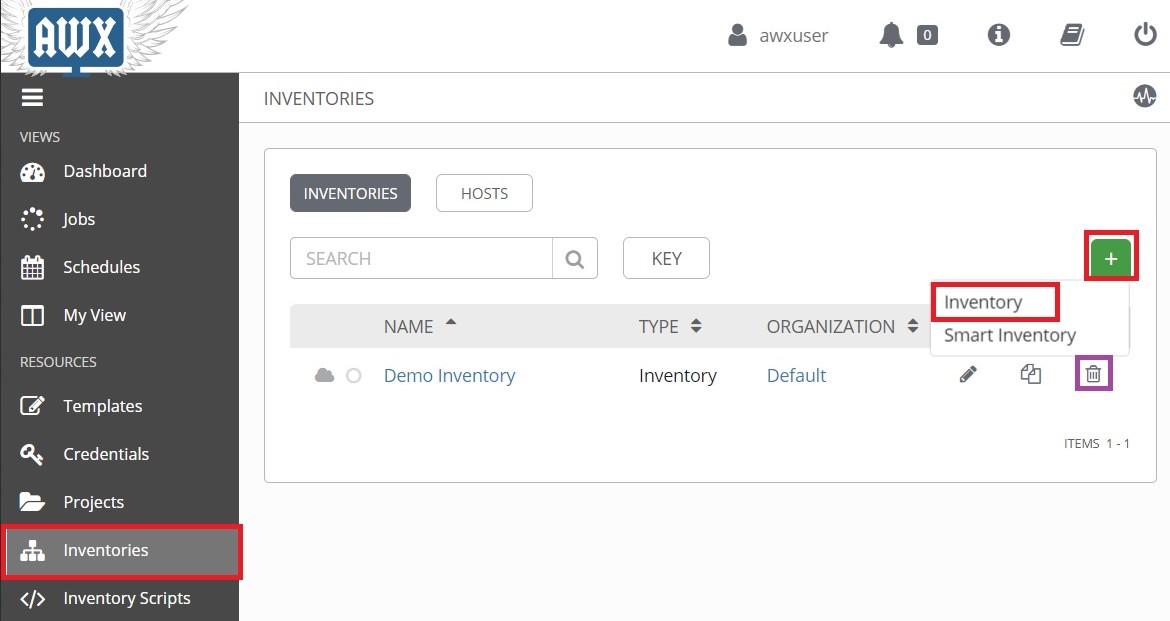


When finished, click **Save**.

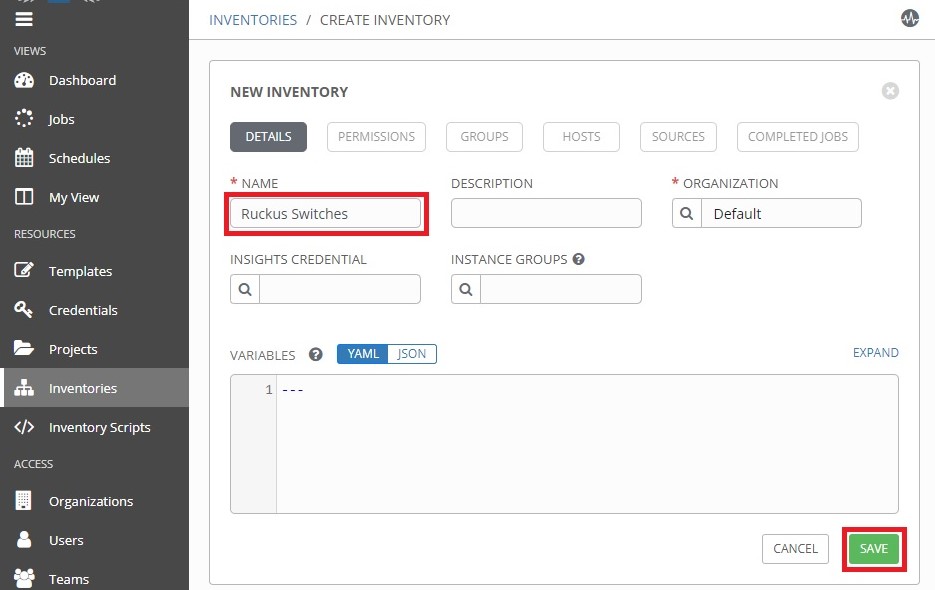
## 4.4 Create Inventories

To add a switch for AWX to manage, you will first need to create an inventory.

Click **Inventories** from the left navigation menu. Click the delete () button to optionally delete the **Demo Inventory**. Click the **+** button, then click **Inventory**:

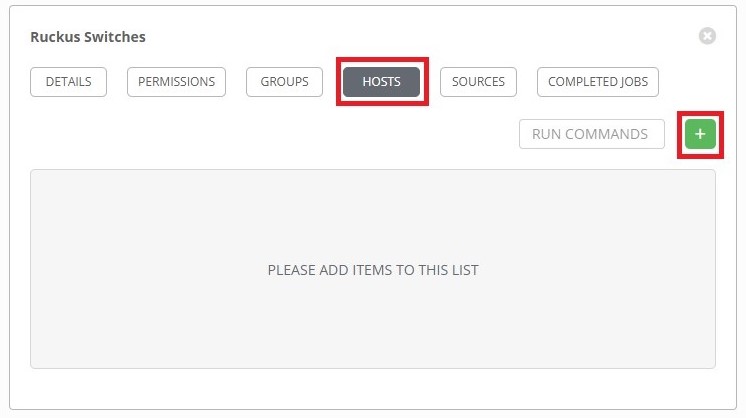


The **New Inventory** page will display. Create a name for your inventory, then click **Save**:



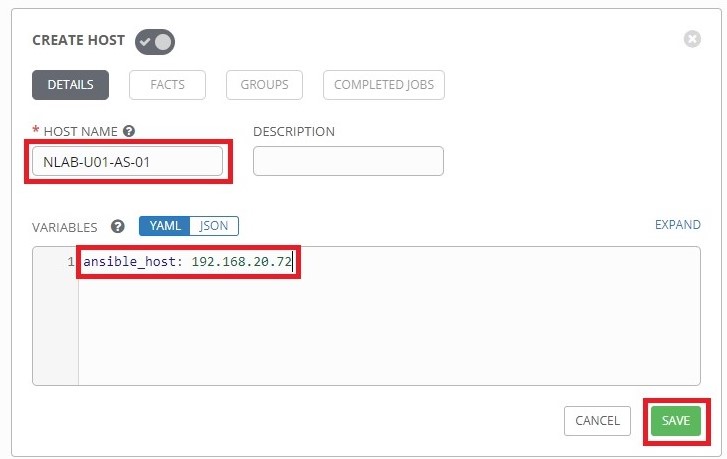
Switches can now be added to the newly created inventory.

Click on the **Hosts** tab, then click the **+** button:



The **Create Host** page will display. Complete the following fields:

**HOST NAME:** Enter the hostname of the switch.

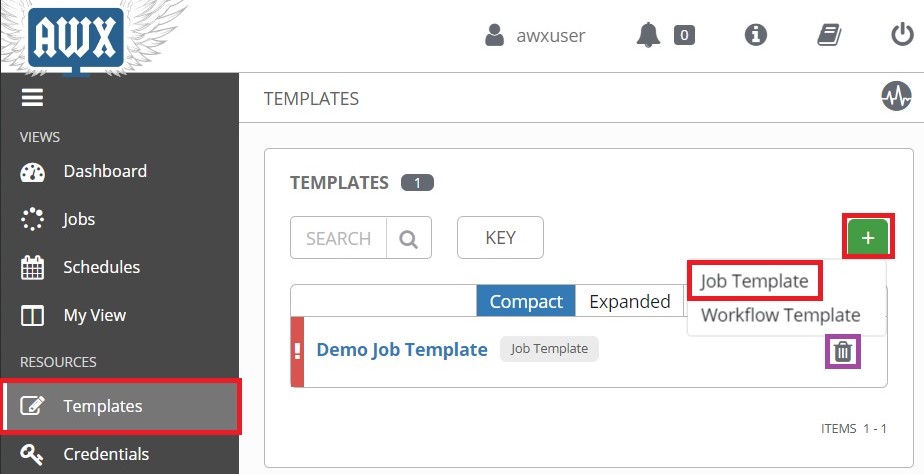
**VARIABLES:** Type "**ansible\_host:** " followed by the IP address of the switch.

When finished, click **Save.**

## 4.5 Creating Job Templates

Now that the other pieces are in place, job templates can be created to run against host devices. The following guide will walk through the creation of a template that uses the '*pbfacts.yml*' playbook created earlier.

Click **Templates** from the left navigation menu. Click the delete () button to optionally remove the **Demo Job Template**. Click the **+** button, then click **Job Template**:



The **New Job Template** page will display. Complete the following fields:

**NAME:** Create a name for the template. Here it is named "*ICX Facts*".

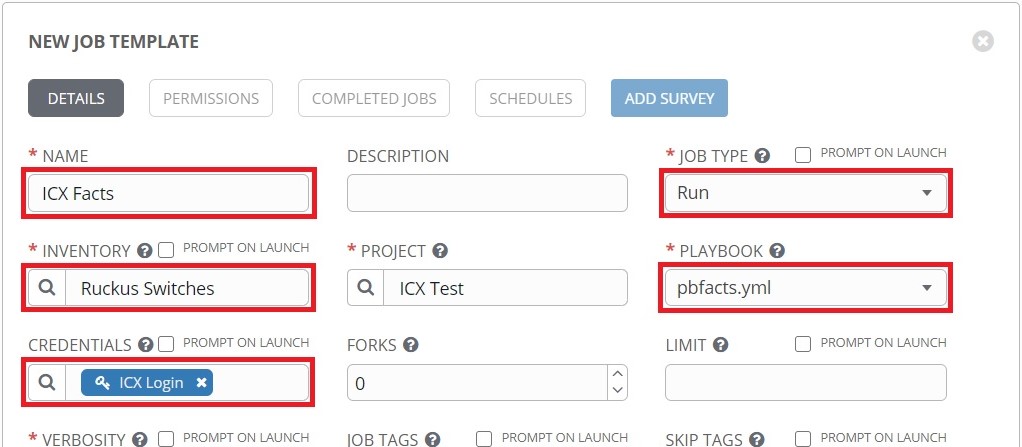
**JOB TYPE:** Select **Run**.

**INVENTORY:** Select the inventory created for the switch.

**PROJECT:** Select the project that contains the '*pbfacts.yml*' playbook.

**PLAYBOOK:** Select **pbfacts.yml**.

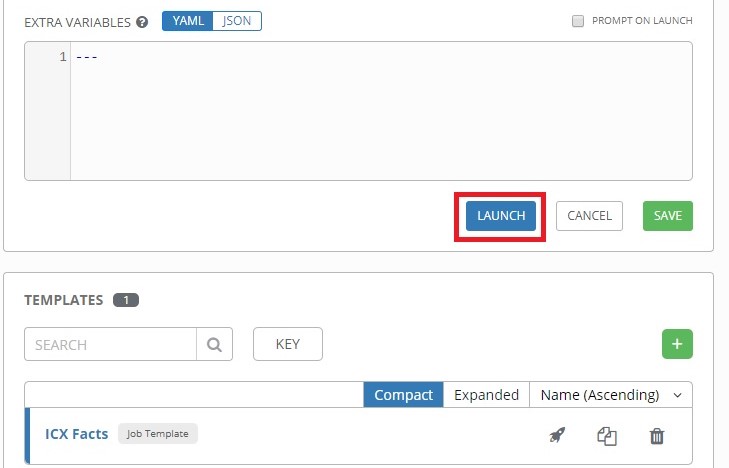
**CREDENTIALS:** Select the credential set that holds the switch login credentials.



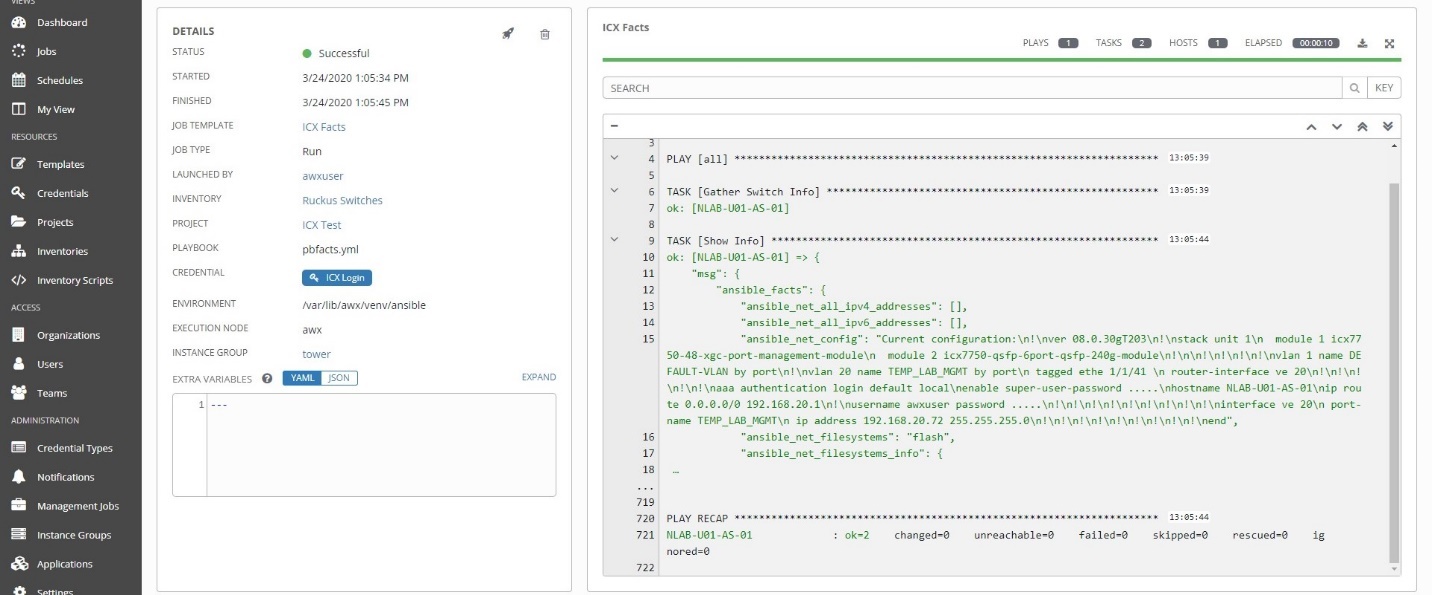
When finished, click **Save**.

Once the template is saved, the **Launch** button will no longer be grayed out.

Click the **Launch** button to execute the template:

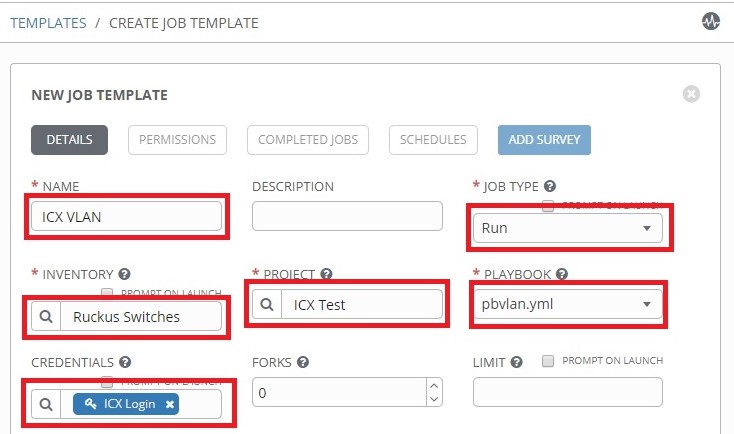


This will bring up the **Details** page with a status window:

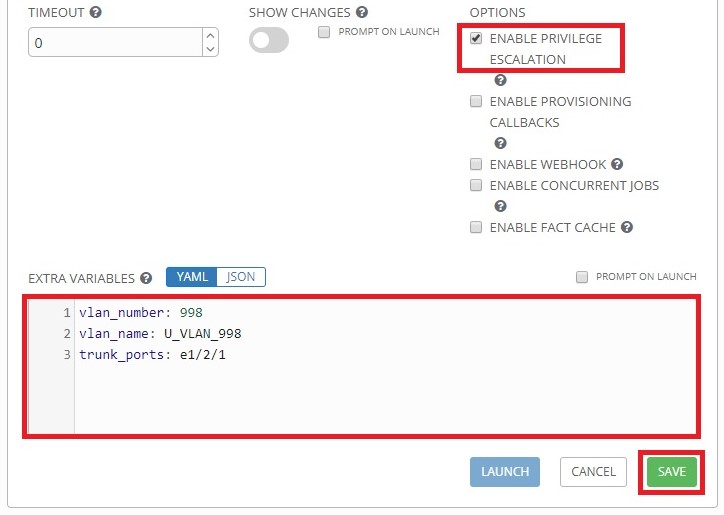


The status window will display an abbreviated output when the template has finished executing. You can click on the output to show it in full. Also note the color of the status text. Green means the template executed successfully whereas red would indicate a failure.

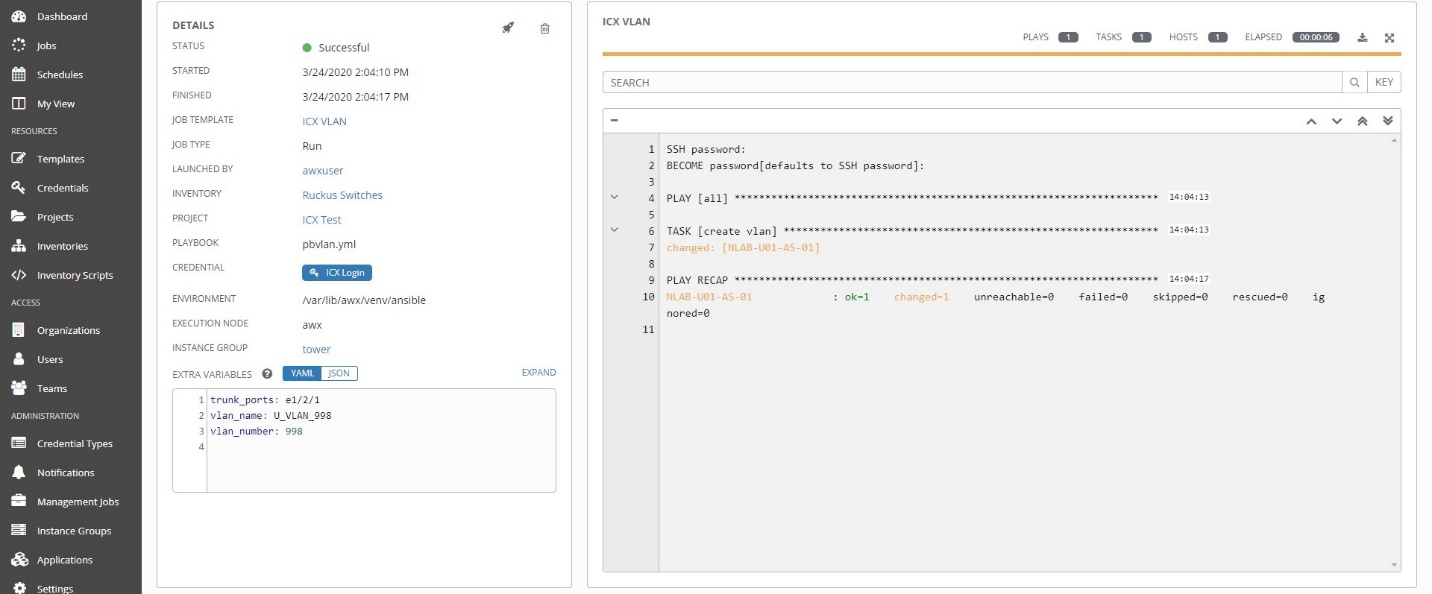
The next guide will walk through the creation of a template that uses variables. Go back to the **Templates** page and add a new job template. Configure the new template to use the '*pbvlan.yml*' playbook:



Scroll to the bottom. Select **Enable Privilege Escalation**. Fill out the variable section as seen in the picture below. When you are finished, click **Save**:

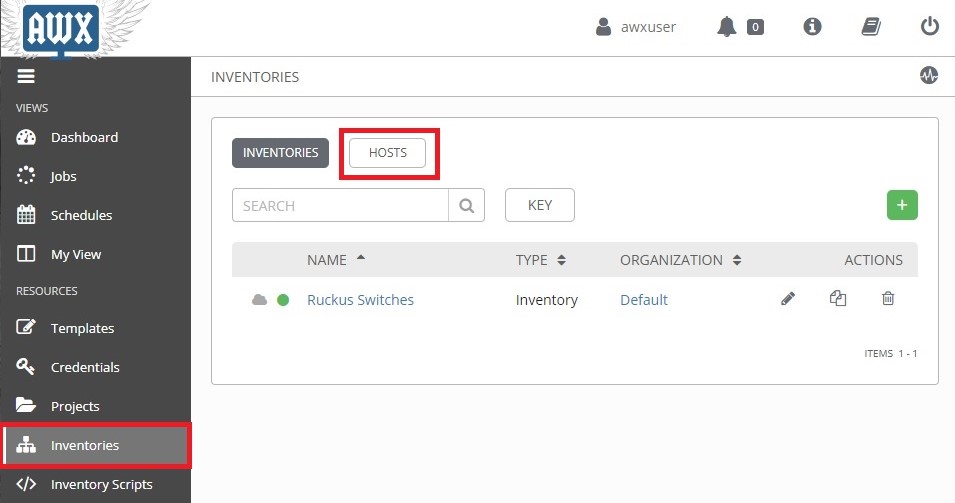


When the template has saved, click **Launch**. When the template has finished executing, you should see yellow status text if the job was successful. This indicates a change has occurred:

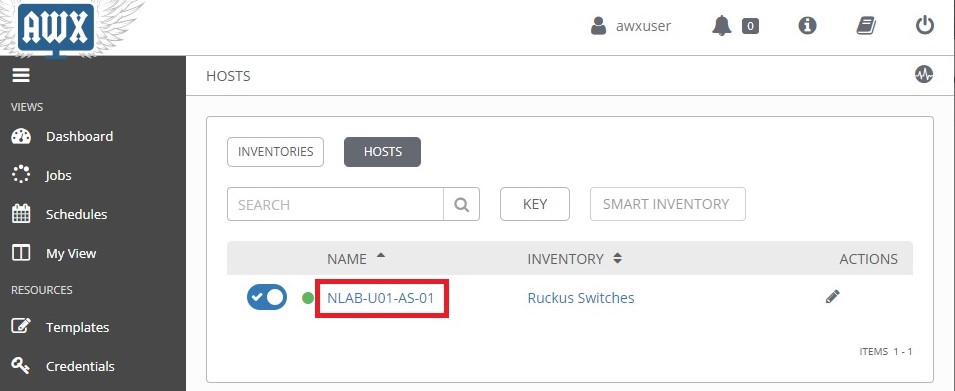


For the '*ICX VLAN*' template that was created, the '*trunk\_ports*' variable was defined within the template itself. Suppose there was a group of switches that had a different trunk than the one that was defined in the template. For those switches the template would be tagging the wrong port for the VLAN you are creating. One solution to this would be to define the trunk ports on the **Hosts** page for the switch.

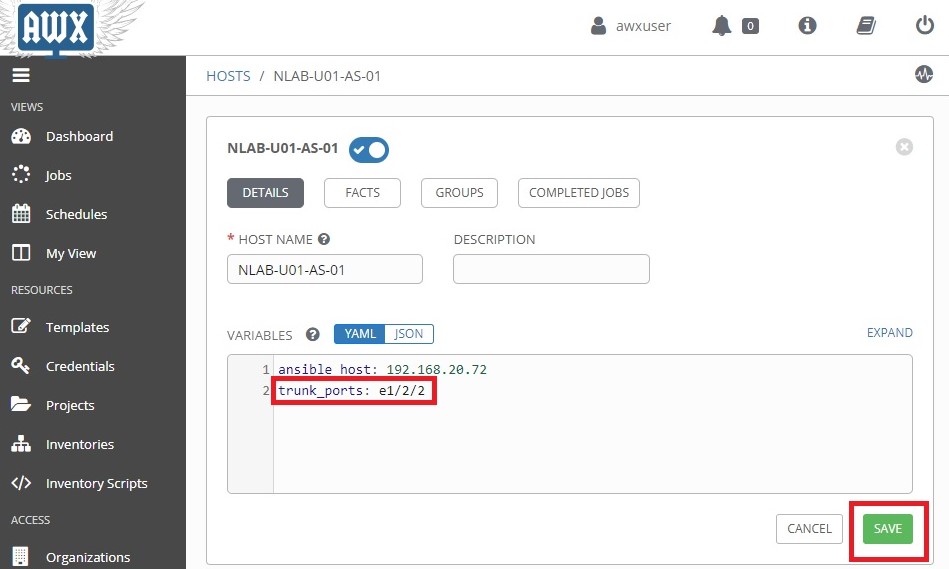
Click the **Inventories** button from the left navigation menu, then click on the **Hosts** tab:



You will see a list of defined hosts. Select the switch that was defined:

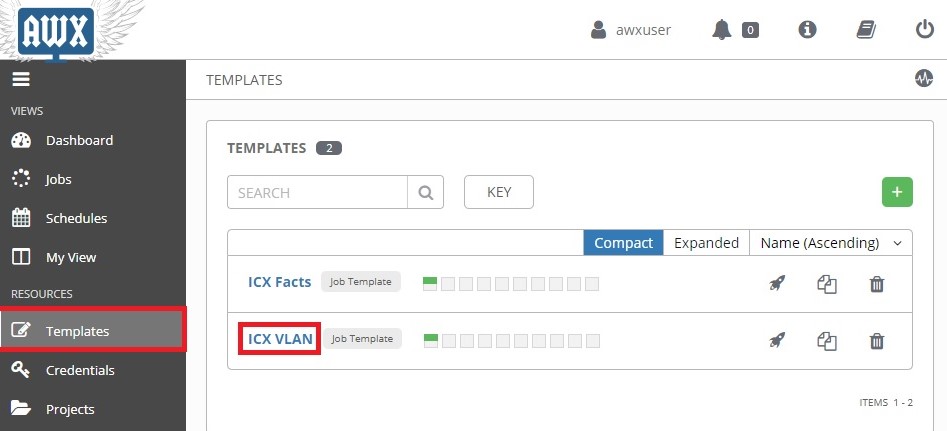


Add the '*trunk\_ports*' variable as seen below, then click **Save**:

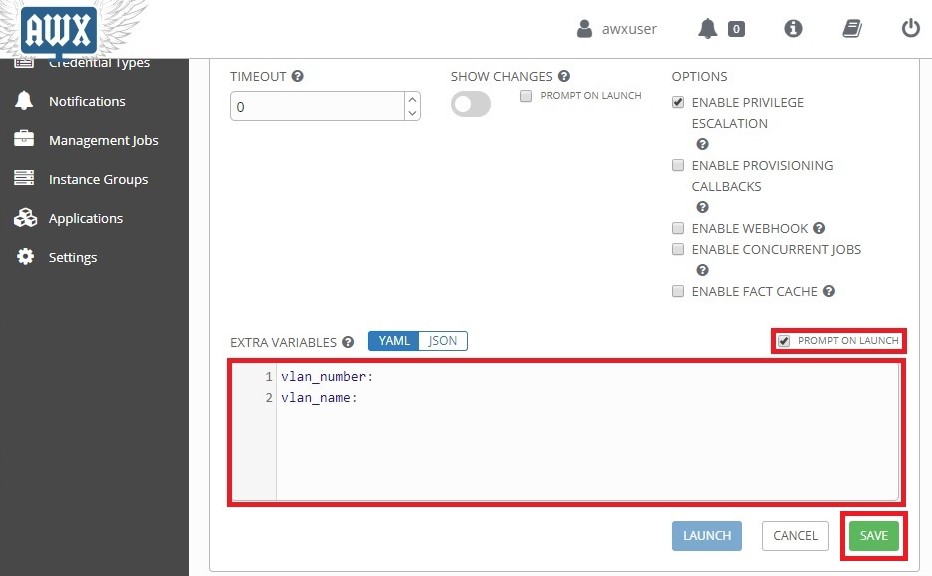


Some templates might be run multiple times with different variables defined for each run. In this case you might not want to have the variable definitions saved in the template. Templates can be configured to prompt for variables on launch.

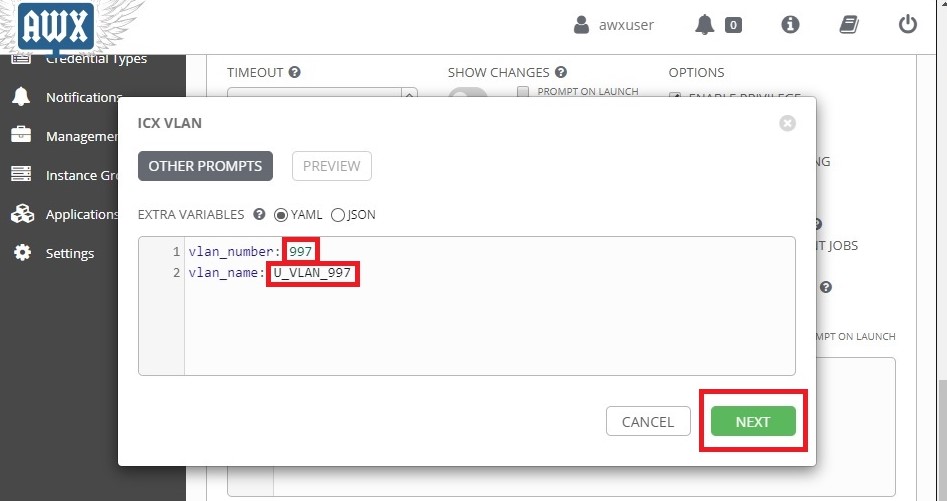
Click the **Templates** button from the left navigation menu, then click on the **ICX VLAN** template:



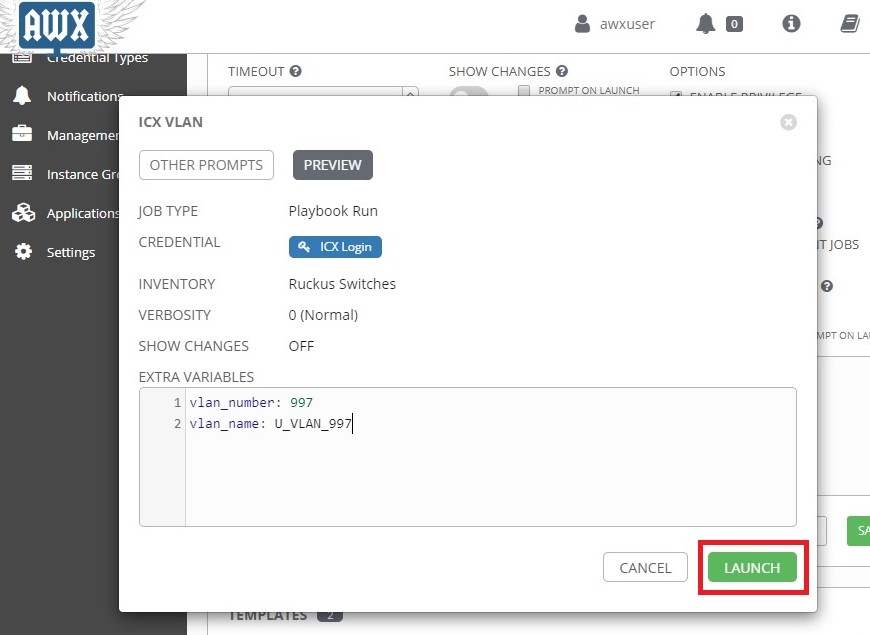
Scroll down to **Extra Variables**. Click the checkbox next to **Prompt on Launch**. Remove the variable for '*trunk\_ports*' and make the '*vlan\_number*' and '*vlan\_name*' variables undefined like the picture below, then click **Save**:



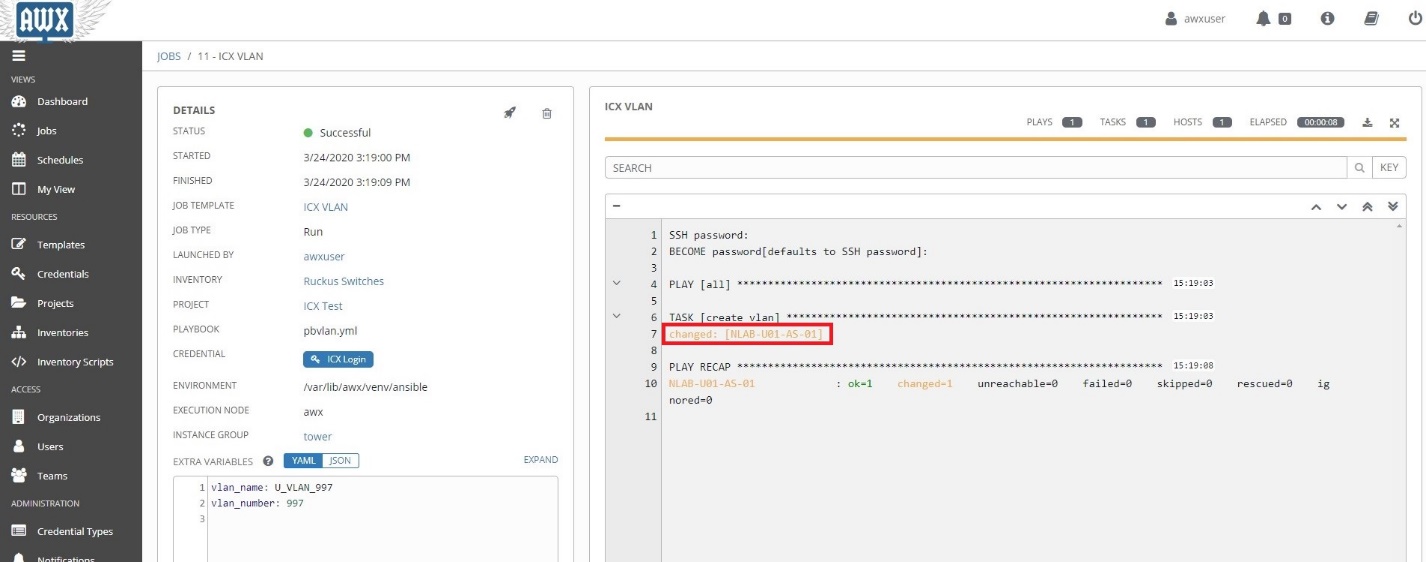
Click the **Launch** button. Define the variables as seen below, then click **Next**:



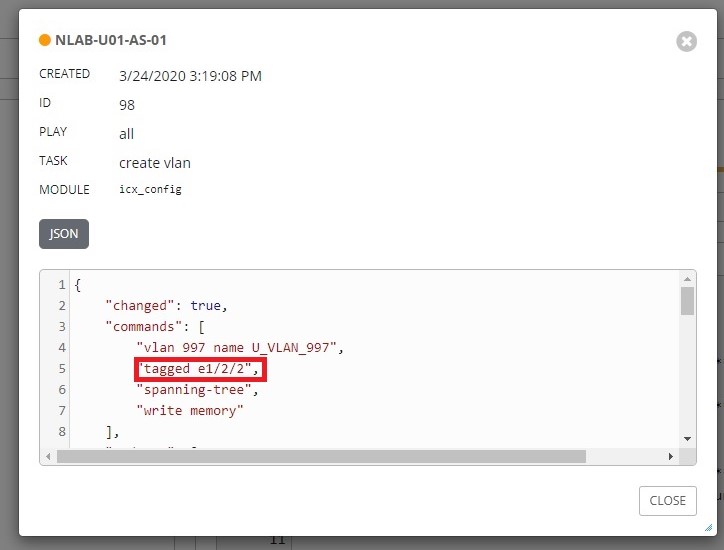
You will see the **Preview** window. Click the **Launch** button:



The status text will be yellow if the job execution was successful. Click on the task output:



Here you can see the template was executed with the '*trunk\_port*' variable defined under the host:



# 5. Vim Basics

The following can be entered from global mode:

|  |  |
| --- | --- |
| :w | Save |
| :q | Quit |
| :q! | Force Quit |
| a | Enter Insert (edit) Mode |
| v | Enter Visual Mode |

Insert mode allows you to edit a document.

Visual mode allows you to select text.

Hitting the Escape key returns to global mode.

# 6. References

AWX server install instructions are derived from this site:

<https://www.howtoforge.com/how-to-install-ansible-awx-with-nginx-reverse-proxy-on-ubuntu-1804/>

SSL certificate and key creation tutorial:

<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-16-04>

Ansible Playbook Tutorial:

<https://docs.ansible.com/ansible/latest/network/getting_started/first_playbook.html>

# 7. Additional Resources

AWX Documentation: <https://docs.ansible.com/ansible-tower/latest/html/userguide/index.html>

VIM Cheat Sheet: <https://vim.rtorr.com/>