AWXv21 Server Setup

Contents

[1. Prerequisites: 2](#_Toc113607673)

[2. AWX Installation Procedures 3](#_Toc113607674)

[2.1 Obtain Local IP Address 3](#_Toc113607675)

[2.2 Install AWX Dependencies 4](#_Toc113607676)

[2.3 Setup and Configure Kubernetes 5](#_Toc113607677)

[2.4 Install AWX using AWX-Operator 6](#_Toc113607678)

[2.5 Install and Configure NGINX Reverse Proxy 8](#_Toc113607679)

[3. Ansible CLI Tutorial 10](#_Toc113607680)

[3.1 Install and Configure Ansible 10](#_Toc113607681)

[3.2 Test Switch Connectivity 12](#_Toc113607682)

[3.3 Run Tasks with *ansible* Command 13](#_Toc113607683)

[3.4 Run Sample Playbooks with *ansible-playbook* Command 14](#_Toc113607684)

[4. AWX Basic Tutorial 16](#_Toc113607685)

[4.1 Manual Project Directory Setup 16](#_Toc113607686)

[4.2 Initial AWX Login 17](#_Toc113607687)

[4.3 Projects 18](#_Toc113607688)

[4.4 Credentials 20](#_Toc113607689)

[4.5 Inventories 22](#_Toc113607690)

[4.6 Job Templates 25](#_Toc113607691)

[5. Vim Basics 30](#_Toc113607692)

[6. References 31](#_Toc113607693)

[7. Additional Resources 31](#_Toc113607694)

# 1. Prerequisites:

* Fresh Ubuntu 22.05 Server install w/ OpenSSH server
* RAM: 8+ GB recommended (4GB minimum for lightweight install)
* Disk Space: 60+ GB
* Processor Cores: 4+
* Bridged network adapter (if using a VM)
* Disable Dynamic Memory (if using a VM)

# 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: eth0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP group default qlen 1000  link/ether 00:15:5d:20:01:0b brd ff:ff:ff:ff:ff:ff  inet 192.168.1.154/24 metric 100 brd 192.168.1.255 scope global dynamic eth0  valid\_lft 51864sec preferred\_lft 51864sec  inet6 fe80::215:5dff:fe20:10b/64 scope link  valid\_lft forever preferred\_lft forever  3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc mq state UP group default qlen 1000  link/ether 00:15:5d:20:01:0c brd ff:ff:ff:ff:ff:ff  inet6 fe80::215:5dff:fe20:10c/64 scope link  valid\_lft forever preferred\_lft forever |

## 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 to update the apt package list and reboot the server:

|  |
| --- |
| **sudo apt update -y &&**  **sudo reboot** |

When the server has finished rebooting, log back into the server and copy and paste the following command into the PuTTY window to install the initial dependencies:

|  |
| --- |
| **sudo apt install make docker.io -y** |

Create and **cd** into the “Documents” directory:

|  |
| --- |
| **mkdir ~/Documents &&**  **cd ~/Documents** |

Download and install **minikube**:

|  |
| --- |
| **curl -LO https://github.com/kubernetes/minikube/releases/download/v1.25.2/minikube\_1.25.2-0\_amd64.deb &&**  **sudo dpkg -i minikube\_1.25.2-0\_amd64.deb &&**  **rm minikube\_1.25.2-0\_amd64.deb** |

Download and execute the install script for **kustomize**:

|  |
| --- |
| **curl -s "https://raw.githubusercontent.com/kubernetes-sigs/kustomize/master/hack/install\_kustomize.sh" | bash** |

## 2.3 Setup and Configure Kubernetes

Use the **usermod** command to add the user to the “docker” group using the syntax below:

|  |
| --- |
| sudo usermod -aG docker <username> |

Apply the group change with this command:

|  |
| --- |
| **newgrp docker** |

Issue the following commands to configure the container driver: and deploy a minikube cluster:

|  |
| --- |
| **minikube config set driver docker** |

If installing AWX with the minimum amount of RAM (8GB or more), issue the following command:

|  |
| --- |
| **minikube start --cpus=4 --memory=6g --addons=ingress** |

For a lightweight AWX install on system with less RAM (minimum of 4GB), issue the following command:

|  |
| --- |
| minikube start --cpus=4 --addons=ingress |

It is unnecessary to install **kubectl** as it comes wrapped inside of minikube. To execute **kubectl** commands, prefix them with “minikube kubectl --" or configure an alias:

|  |
| --- |
| **alias kubectl="minikube kubectl --"** |

Note: The alias will only remain active for the login session and will need to be reconfigured after logging in again.

The following commands can be used to ensure everything is working as expected:

|  |
| --- |
| kubectl get nodes  kubectl get pods -A |

The output should look similar to the following:

|  |
| --- |
| awxuser@awxserver:~$ kubectl get nodes  kubectl get pods -A  > kubectl.sha256: 64 B / 64 B [--------------------------] 100.00% ? p/s 0s  > kubectl: 44.43 MiB / 44.43 MiB [---------------] 100.00% 2.58 MiB p/s 17s  NAME STATUS ROLES AGE VERSION  minikube Ready control-plane,master 73m v1.23.3  NAMESPACE NAME READY STATUS RESTARTS AGE  ingress-nginx ingress-nginx-admission-create-fklql 0/1 Completed 0 73m  ingress-nginx ingress-nginx-admission-patch-5cbdv 0/1 Completed 0 73m  ingress-nginx ingress-nginx-controller-cc8496874-4k27t 1/1 Running 0 73m  kube-system coredns-64897985d-twmc5 1/1 Running 0 73m  kube-system etcd-minikube 1/1 Running 0 73m  kube-system kube-apiserver-minikube 1/1 Running 0 73m  kube-system kube-controller-manager-minikube 1/1 Running 0 73m  kube-system kube-proxy-8p6vt 1/1 Running 0 73m  kube-system kube-scheduler-minikube 1/1 Running 0 73m  kube-system storage-provisioner 1/1 Running 0 73m |

## 2.4 Install AWX using AWX-Operator

Create a manifest to install AWX-Operator using vim:

|  |
| --- |
| **vim kustomization.yaml** |

Copy and paste the following:

|  |
| --- |
| apiVersion: kustomize.config.k8s.io/v1beta1  kind: Kustomization  resources:  - github.com/ansible/awx-operator/config/default?ref=0.21.0  images:  - name: quay.io/ansible/awx-operator  newTag: 0.21.0  namespace: awx |

Install the manifest with the following command:

|  |
| --- |
| **./kustomize build . | kubectl apply -f -** |

The containers will take a few minutes to build. Check the container progress with the following command:

|  |
| --- |
| **kubectl get pods -n awx** |

When the containers have been built, the command output will show the following:

|  |
| --- |
| awxuser@awxserver:~/Documents$ kubectl get pods -n awx  NAME READY STATUS RESTARTS AGE  awx-operator-controller-manager-675865446d-xldn8 2/2 Running 0 6s |

Note the status and number of ready containers.

Execute the following to keep from having to specify the namespace for commands with “-n awx”:

|  |
| --- |
| **kubectl config set-context --current --namespace=awx** |

Note: this command does not persist after logging off the server.

Create a manifest that will be used to set the password for the default AWX account:

|  |
| --- |
| **vim awx-secret.yaml** |

Copy and paste the following:

|  |
| --- |
| ---  apiVersion: v1  kind: Secret  metadata:  name: ansible-awx-admin-password  namespace: awx-namespace  stringData:  password: awxpassword |

Note that the password “awxpassword” will be used for the default AWX account in this example. The password can be changed here before installation or it can be changed after installation in AWX.

Create a new file, “awx-demo.yaml”:

|  |
| --- |
| **vim awx-demo.yaml** |

Copy and paste the following into awx-demo:

|  |
| --- |
| ---  apiVersion: awx.ansible.com/v1beta1  kind: AWX  metadata:  name: ansible-awx  spec:  projects\_persistence: true  nodeport\_port: 30080  service\_type: nodeport  admin\_user: awxuser  admin\_password\_secret: ansible-awx-admin-password |

Add the following lines to kustomization.yaml, under “resources”:

- awx-secret.yaml

- awx-demo.yaml

The kustomization.yaml file should now look like the following:

|  |
| --- |
| ---  apiVersion: kustomize.config.k8s.io/v1beta1  kind: Kustomization  resources:  - github.com/ansible/awx-operator/config/default?ref=0.21.0  - awx-secret.yaml  - awx-demo.yaml  images:  - name: quay.io/ansible/awx-operator  newTag: 0.21.0  namespace: awx |

Run the kustomize command to install AWX into the cluster:

|  |
| --- |
| **./kustomize build . | kubectl apply -f -** |

Installation progress can be monitored by viewing the logs with the following command:

|  |
| --- |
| **kubectl logs -f deployments/awx-operator-controller-manager \**  **-c awx-manager -n awx** |

The container status can be viewed with the following command:

|  |
| --- |
| **kubectl get pods -l "app.kubernetes.io/managed-by=awx-operator"** |

Results similar to the following should be displayed:

|  |
| --- |
| awxuser@awxserver:~/Documents$ kubectl get pods -l "app.kubernetes.io/managed-by=awx-operator"  NAME READY STATUS RESTARTS AGE  ansible-awx-7575547798-rknsh 4/4 Running 0 21h  ansible-awx-postgres-0 1/1 Running 0 21h |

## 2.5 Install and Configure NGINX Reverse Proxy

A separate container running NGINX provides the web server component of the AWX installation. As the container is only accessible from the host system, a reverse proxy running on the host system will allow external web access to AWX.

The IP address and port of the AWX web container can be obtained with the following command:

|  |
| --- |
| **minikube service ansible-awx-service --url -n awx** |

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** |

Copy and paste the following. Replace the first two highlighted IP addresses with the server IP and replace the third IP address with the IP and port obtained with the **minikube service** command:

|  |
| --- |
| server {  listen 80;  server\_name 192.168.1.154;  add\_header Strict-Transport-Security max-age=2592000;  rewrite ^ https://$server\_name$request\_uri? permanent;  }  server {  listen 443 ssl http2;  server\_name 192.168.1.154;  access\_log /var/log/nginx/awx.access.log;  error\_log /var/log/nginx/awx.error.log;  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://192.168.49.2:30080/;  }  } |

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) []:192.168.1.154  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** |

# 3. Ansible CLI Tutorial

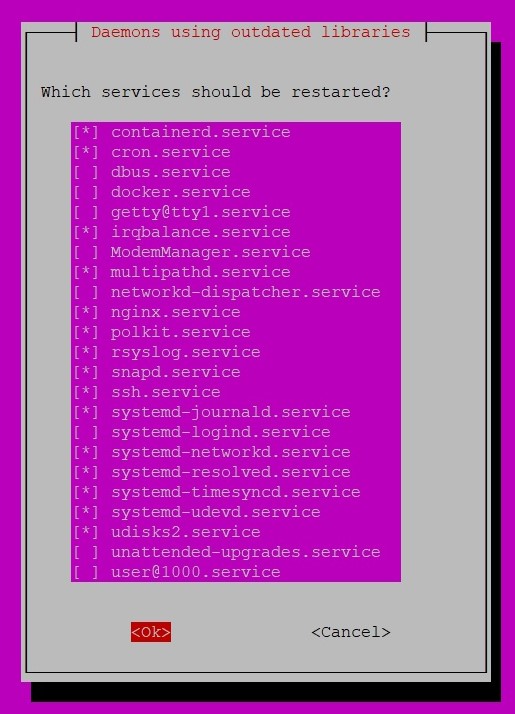
The steps in this tutorial are optional and are included as a guide for learning how to create and deploy Ansible playbooks.

## 3.1 Install and Configure Ansible

Install Pip for Python3 with the following command:

|  |
| --- |
| **sudo apt install python3-pip -y** |

When the installation has completed, a screen will appear that will ask to restart services. Press the **Tab** key until “<OK>” is highlighted, then press **Enter**:



Issue the following commands to install Ansible and create the *~/.ansible* folder with the proper permissions:

|  |
| --- |
| **sudo pip3 install ansible==5.7.0 &&**  **sudo pip3 install paramiko &&**  **ansible --version** |

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. To configure this, 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:

|  |
| --- |
| **ansible --version** |

The following results should be displayed:

|  |
| --- |
| awxuser@awxserver:~$ ansible --version  ansible [core 2.12.8]  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/python3.10/dist-packages/ansible  ansible collection location = /home/awxuser/.ansible/collections:/usr/share/ansible/collections  executable location = /usr/local/bin/ansible  python version = 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0]  jinja version = 3.0.3  libyaml = True |

Note: The minikube cluster will not start automatically after a system restart. To start the cluster, issue the following command:

|  |
| --- |
| **minikube start** |

## 3.2 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 -o HostKeyAlgorithms=+ssh-rsa |

The following output should be displayed:

|  |
| --- |
| awxuser@awxserver:~$ ssh awxuser@10.10.10.100 -o KexAlgorithms=+diffie-hellman-group1-sha1 -o HostKeyAlgorithms=+ssh-rsa  The authenticity of host '10.10.10.100 (10.10.10.100)' can't be established.  RSA key fingerprint is SHA256:uqB0UxJIa3XDgfGHx52abxReL08Wb8abnJtVPujw9dM.  This key is not known by any other names  Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  Warning: Permanently added '10.10.10.100' (RSA) to the list of known hosts.  (awxuser@10.10.10.100) 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 |

## 3.3 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 <switch\_ip>, \  -c network\_cli \  -k -K -b \  --become-method enable \  -e ansible\_network\_os=icx \  -m icx\_facts -a gather\_subset=all \  -u <username> |

Command breakdown:

* Host group we are connecting to (all)
* -i specifies an inventory followed by either target device with a trailing comma.
* -k ask for the ssh password
* -K ask you for the become (privileged user) password
* -b run operations with become
* --become-method privilege 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

## 3.4 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 ~/Documents/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, using the syntax below:

|  |
| --- |
| ansible-playbook -i <switch\_ip>, -k -K -u <username> ~/Documents/pbfacts.yml |

As an example, to configure VLAN 999 on a switch, the following commands can 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 |

Alternatively, a playbook can be created to accomplish the same task.

First create the playbook file:

|  |
| --- |
| **vim ~/Documents/pbvlan.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: 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 <switch\_ip>, -k -K \  -u <username> \  -e trunk\_ports="'e1/2/1 e1/2/2'" \  -e vlan\_name=U\_USER\_999 \  -e vlan\_number=999 \  ~/Documents/pbvlan.yml |

# 4. AWX Basic Tutorial

The steps in the following tutorial are optional. This tutorial will cover setting up a manual project in AWXv21.

## 4.1 Manual Project Directory Setup

The playbooks created in the previous section will need to be placed in the proper directory in order to be usable with AWX. As AWXv21 does not include the ICX Python libraries, they will need to be included in the playbook directory along with their dependencies.

Copy and paste the following command to stage the *Ruckus* project directory with the required dependencies:

|  |
| --- |
| **cd ~/Documents &&**  **mkdir -p Ruckus/ansible\_collections/ansible &&**  **mkdir -p Ruckus/ansible\_collections/community &&**  **mv pbfacts.yml pbvlan.yml Ruckus/ &&**  **cd Ruckus/ansible\_collections &&**  **git clone https://github.com/ansible-collections/ansible.netcommon \**  **ansible/netcommon &&**  **git clone https://github.com/ansible-collections/community.network \**  **community/network** |

The project directory will need to be placed in the */var/lib/awx/projects*directory in one of the containers. To shell into the container, the name of the pod that contains them is need. Issue the following command to get a list of running pods:

|  |
| --- |
| **kubectl get pods -n awx** |

An output similar to the following will be displayed:

|  |
| --- |
| awxuser@awxserver:~/Documents$ minikube kubectl -- get pods -n awx  NAME READY STATUS RESTARTS AGE  ansible-awx-7575547798-rknsh 4/4 Running 0 37m  ansible-awx-postgres-0 1/1 Running 2 (50m ago) 14d  awx-operator-controller-manager-675865446d-xldn8 2/2 Running 4 (2d1h ago) 14d |

Note the name of the pod.

Next, shell into the *ansible-awx-task* container with the following command syntax:

|  |
| --- |
| kubectl exec -it <pod\_name> -c ansible-awx-task -n awx -- /bin/bash |

Next copy the entire *Ruckus* project folder using **scp:**

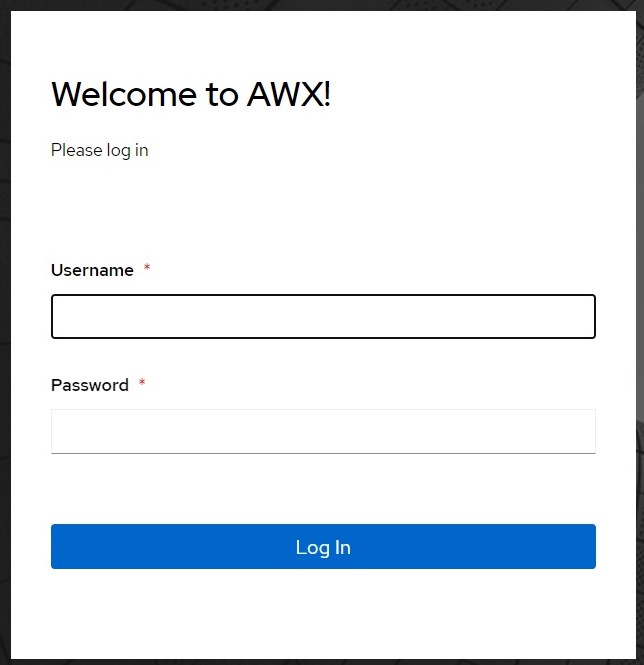
|  |
| --- |
| scp -r <username>@<awx\_ip>:/home/<username>/Documents/Ruckus \  /var/lib/awx/projects/ |

Exit the container with the following command:

|  |
| --- |
| **exit** |

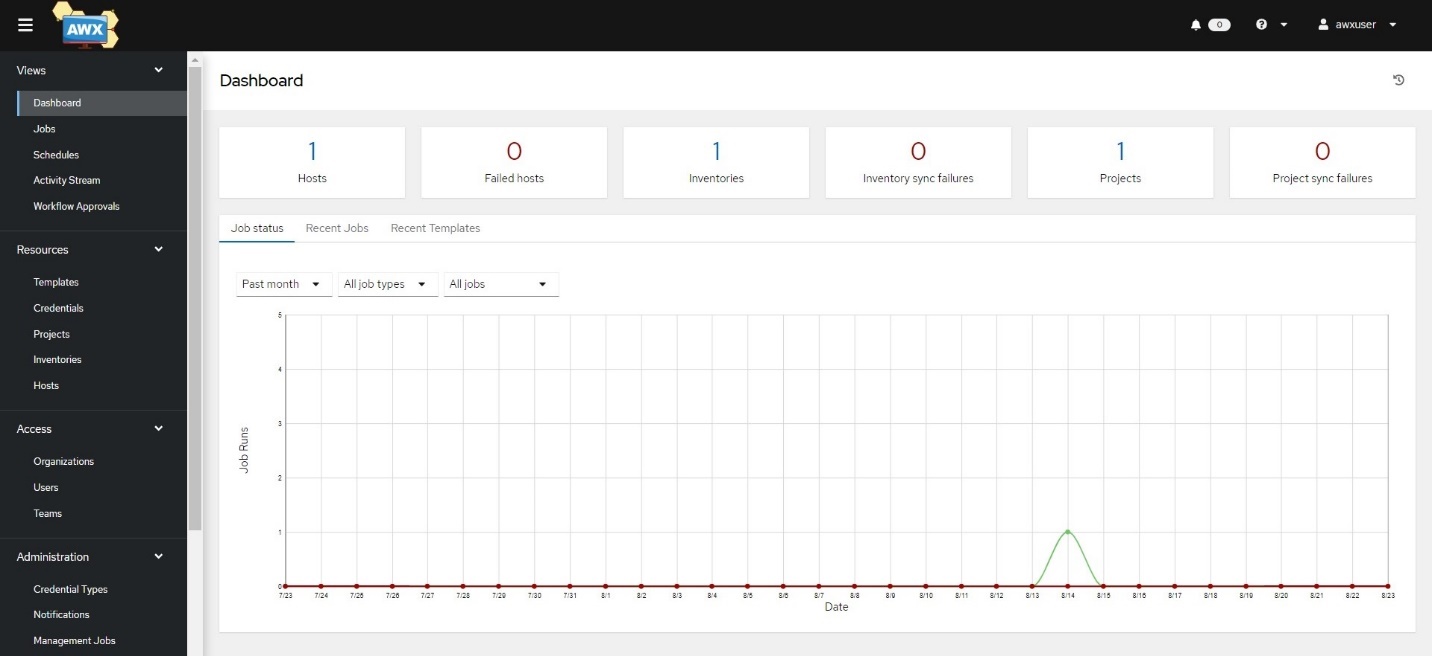
## 4.2 Initial AWX Login

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 that was set in *awx-demo.yaml* (under *admin\_user*) and the password that was set in *awx-secret.yaml* (under *password*).

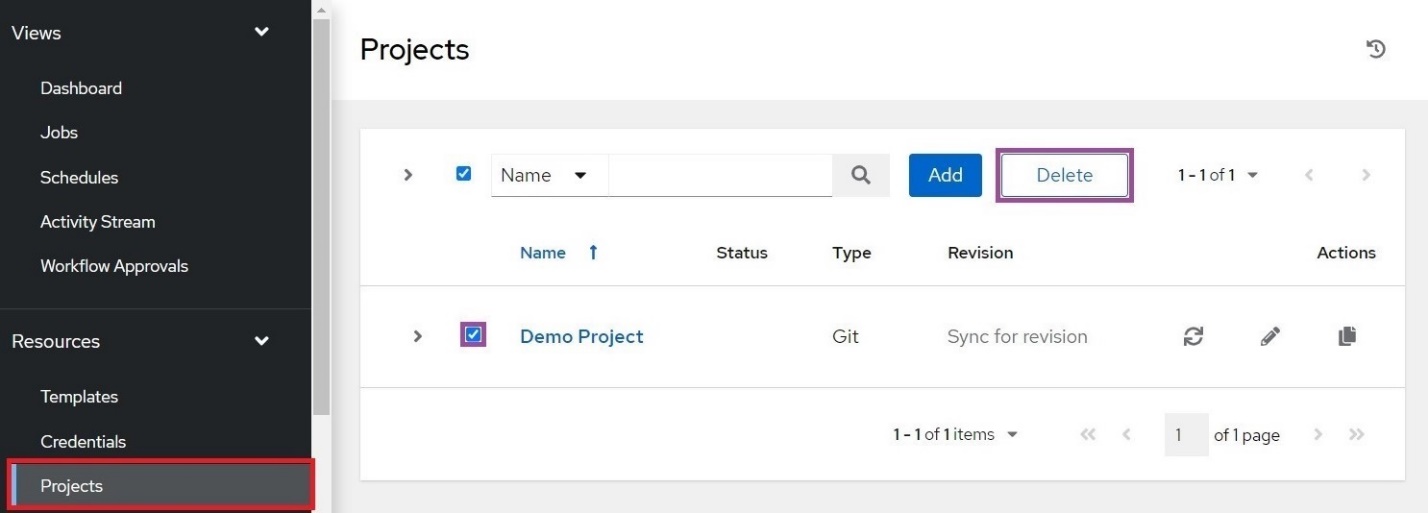
Upon login, the dashboard will display with the navigation bar on the left:



## 4.3 Projects

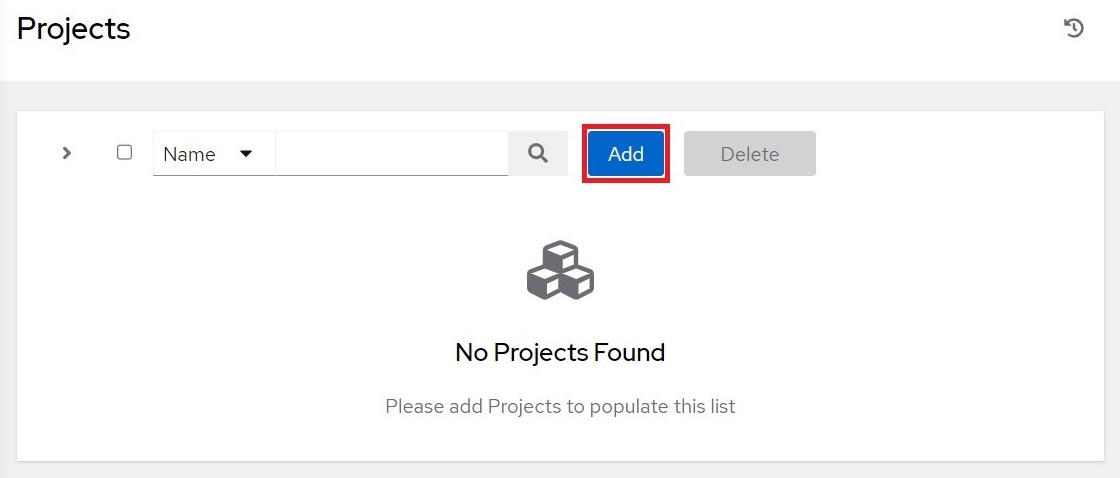
A project is a logical collection of playbooks. The following will guide you through the creation of a project whose playbooks have been manually placed in the project base path of the AWX server.

Click **Projects** from the left navigation menu. *Demo Project* canbe optionally removed by checking the box next to it and clicking **Delete**:



A window will appear warning that the project is in use by other resources. Click **Delete**.

Click the **Add** button to add a new project:

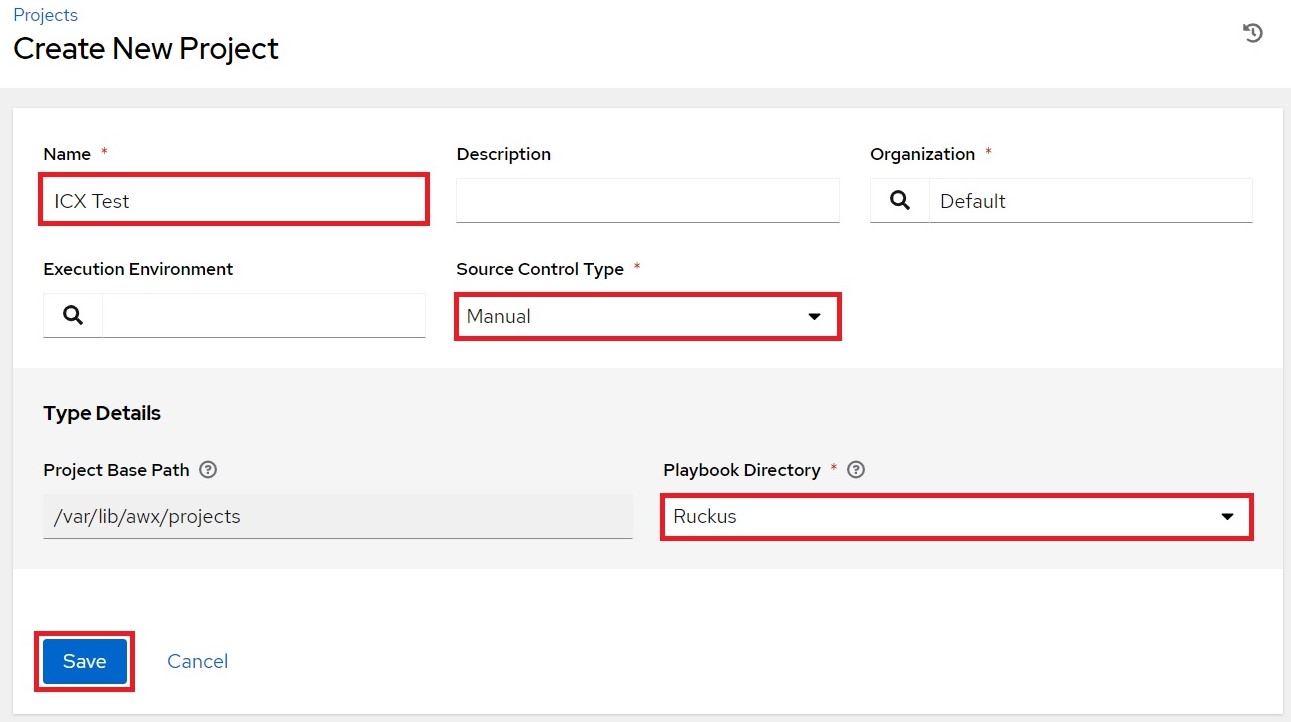
****

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

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

**Source Control Type**:Select **Manual**.

**Playbook Directory**: Select the name of the folder you copied the playbooks into.

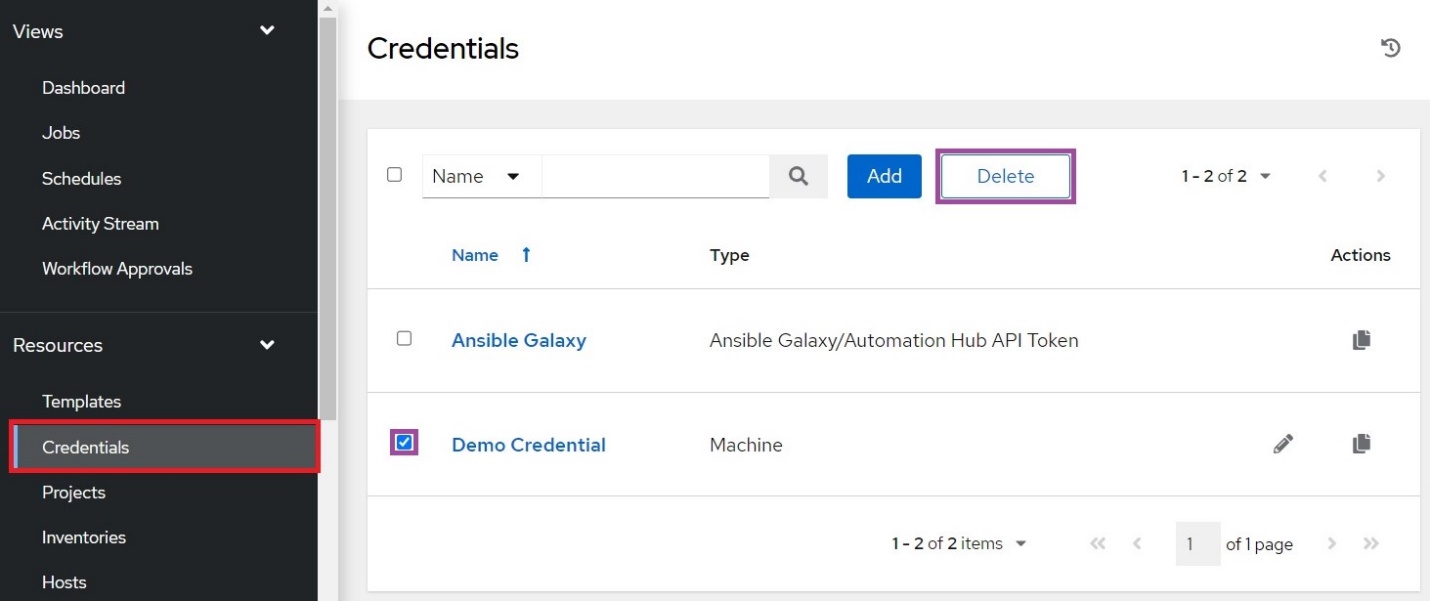


When finished, click the **Save** button.

## 4.4 Credentials

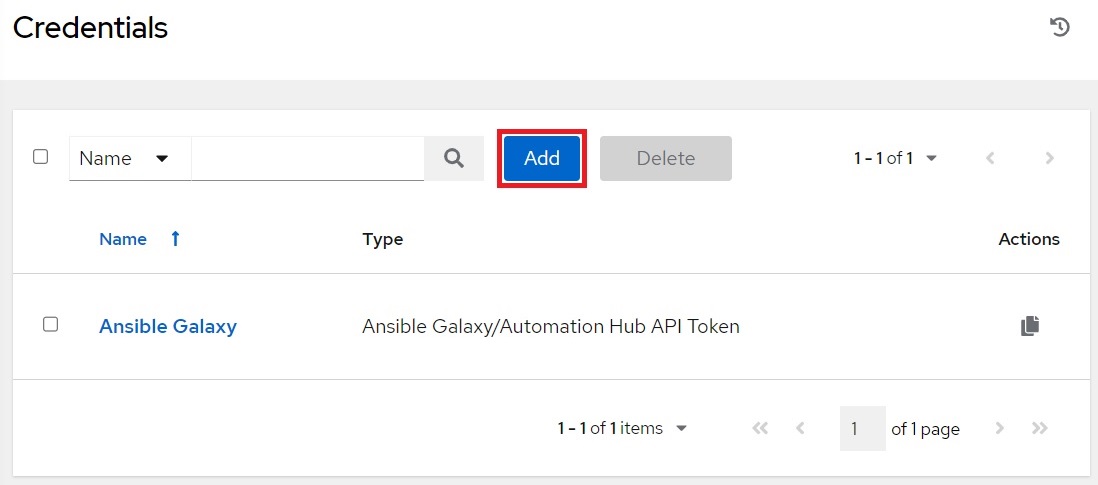
For AWX to log into a device, you will need to create a set of credentials that can be used in a playbook.

Click **Credentials** from the left navigation menu. *Demo Credential* canbe optionally removed by checking the box next to it and clicking **Delete**:



A window will appear warning that the credential is in use by other resources. Click **Delete**.

Then click the **Add** button to add a credential set:

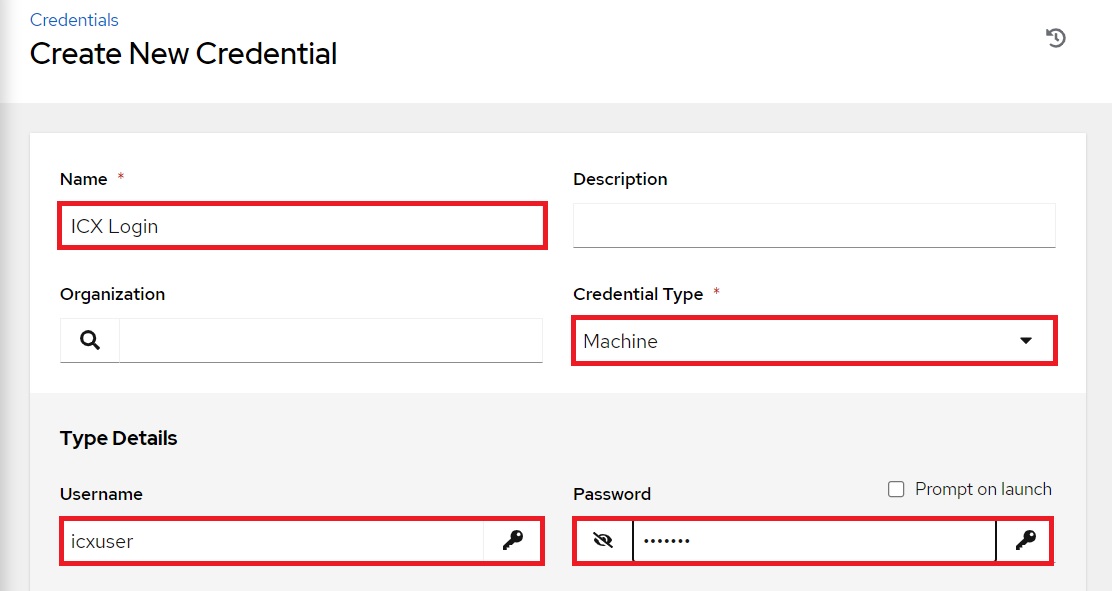


The **Create New Credential** page will display. Complete the following fields:

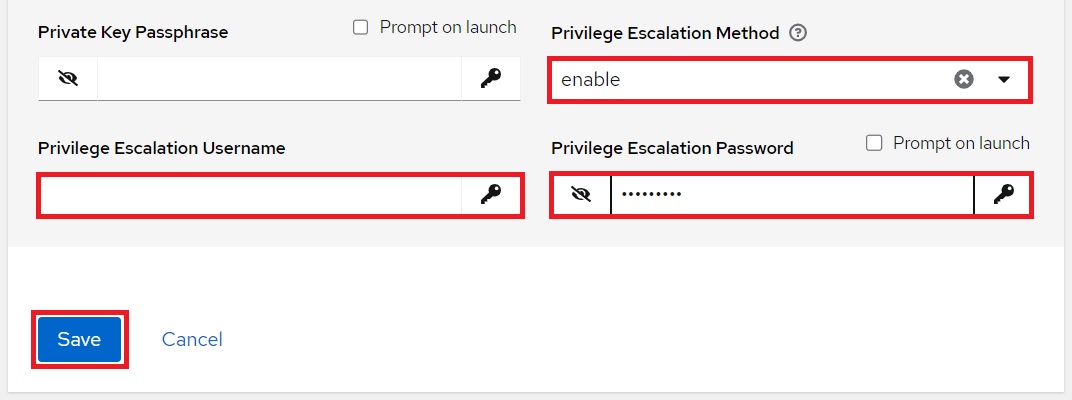
**Name**: Create a name for the credential set. Here it is named "*ICX Login*"

**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. Set the **Privilege Escalation Method** to **enable**, then set the enable username and/or password:

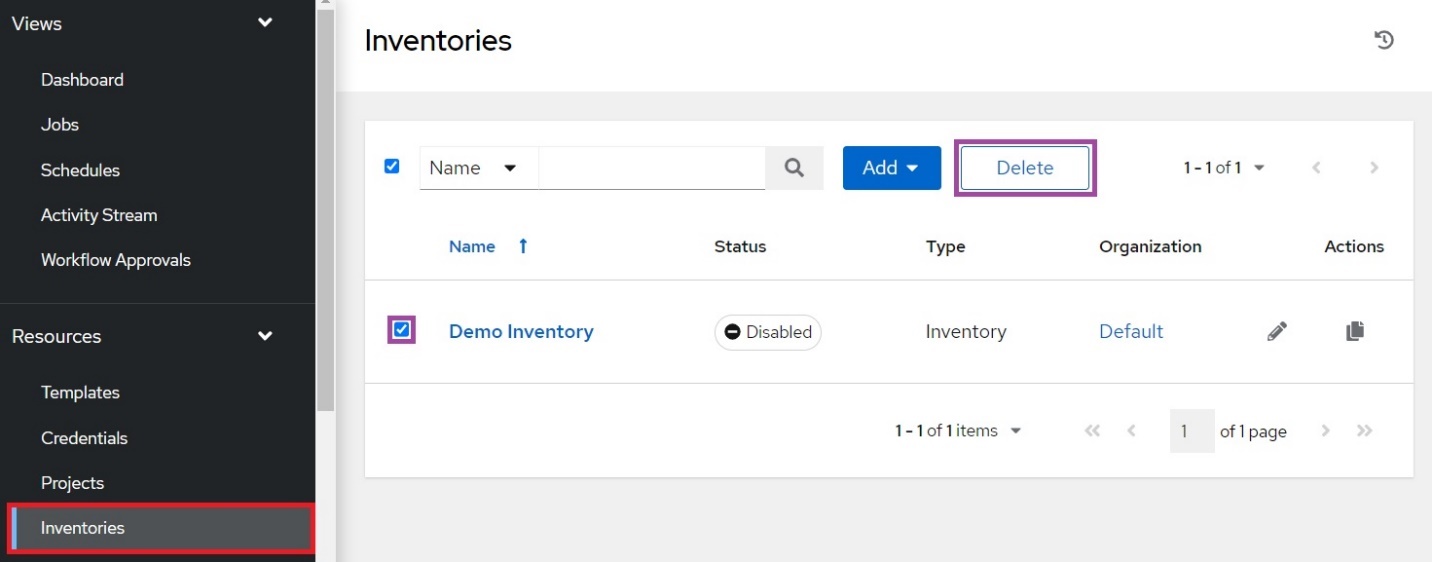


When finished, click **Save**.

## 4.5 Inventories

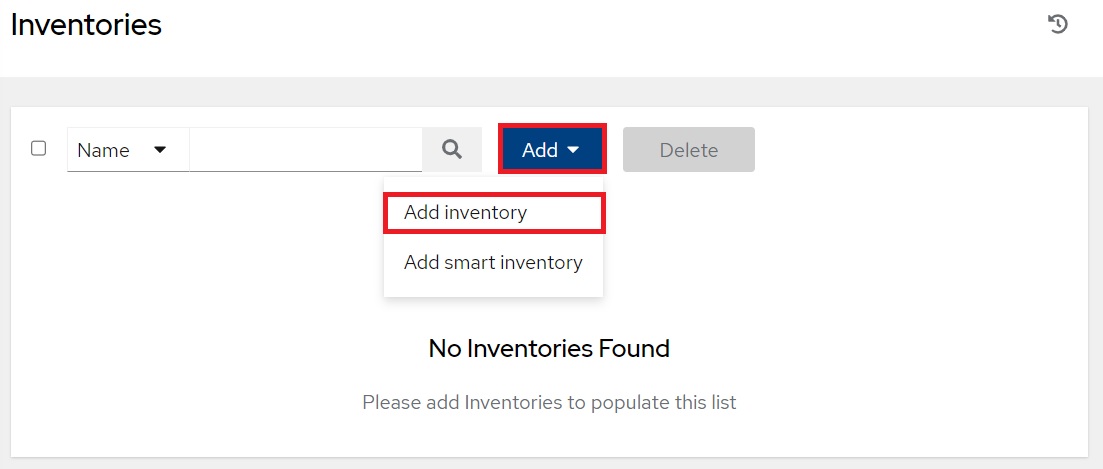
To add a switch for AWX to manage, you will first need to create an inventory.

Click **Inventories** from the left navigation menu. *Demo Inventory* canbe optionally removed by checking the box next to it and clicking **Delete**:

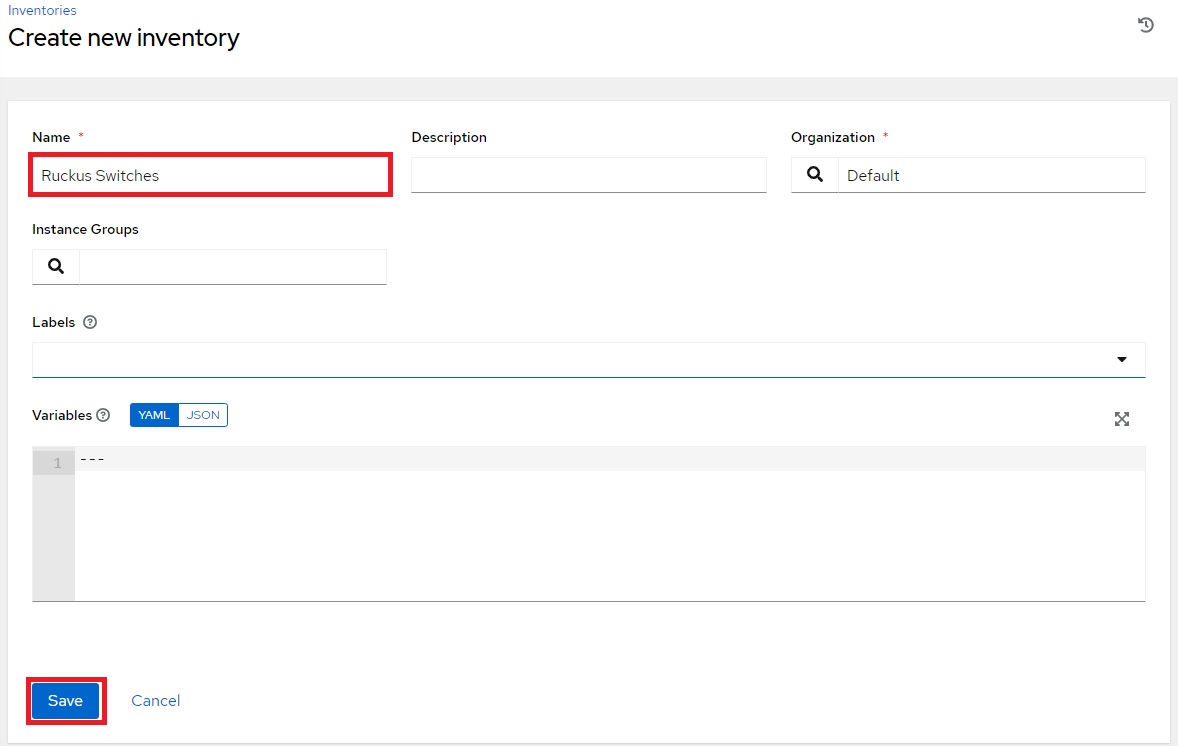


A window will appear warning that the inventory is in use by other resources. Click **Delete**.

Click the **Add** button, then select **Add inventory** from the drop-down menu:

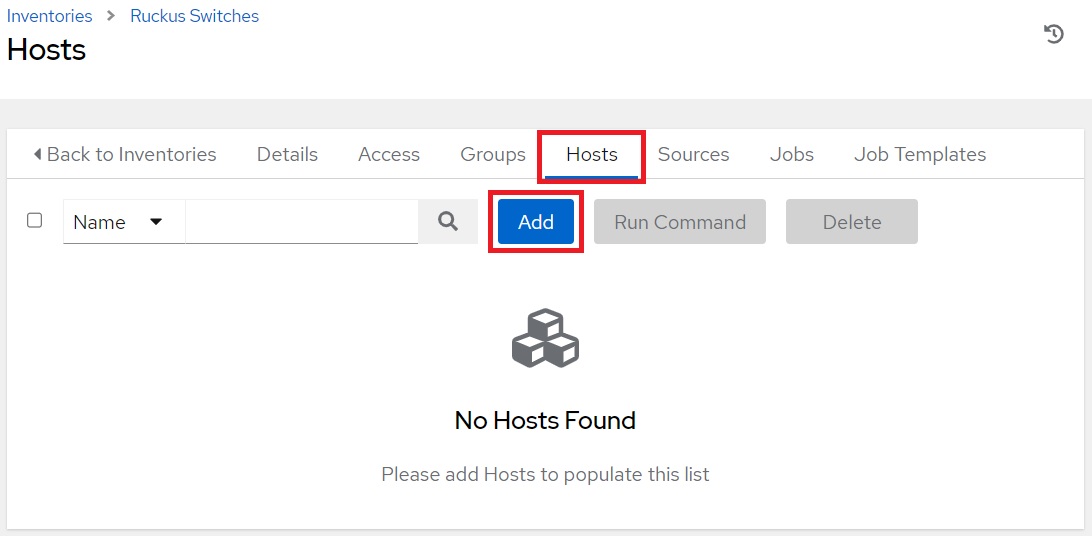


The **Create new inventory** page will display. Create a name for your inventory, then click **Save**:



Switches can now be added to the newly created inventory.

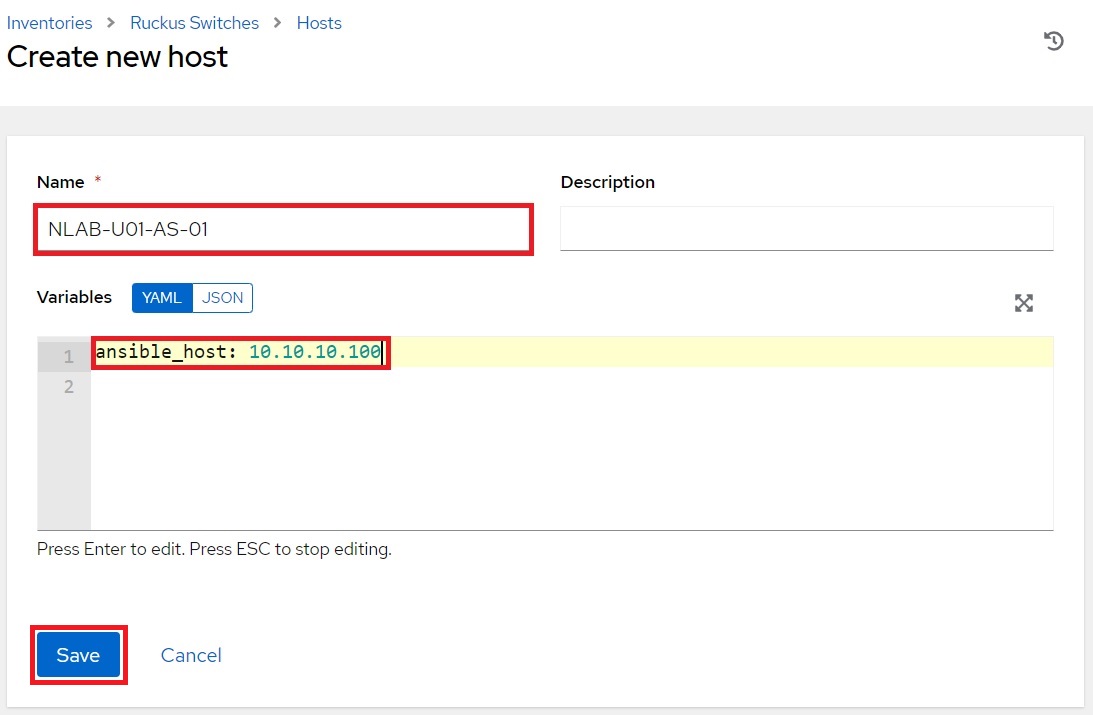
Click on the **Hosts** tab, then click the **Add** button:



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

**Name:** Enter the hostname of the switch.

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

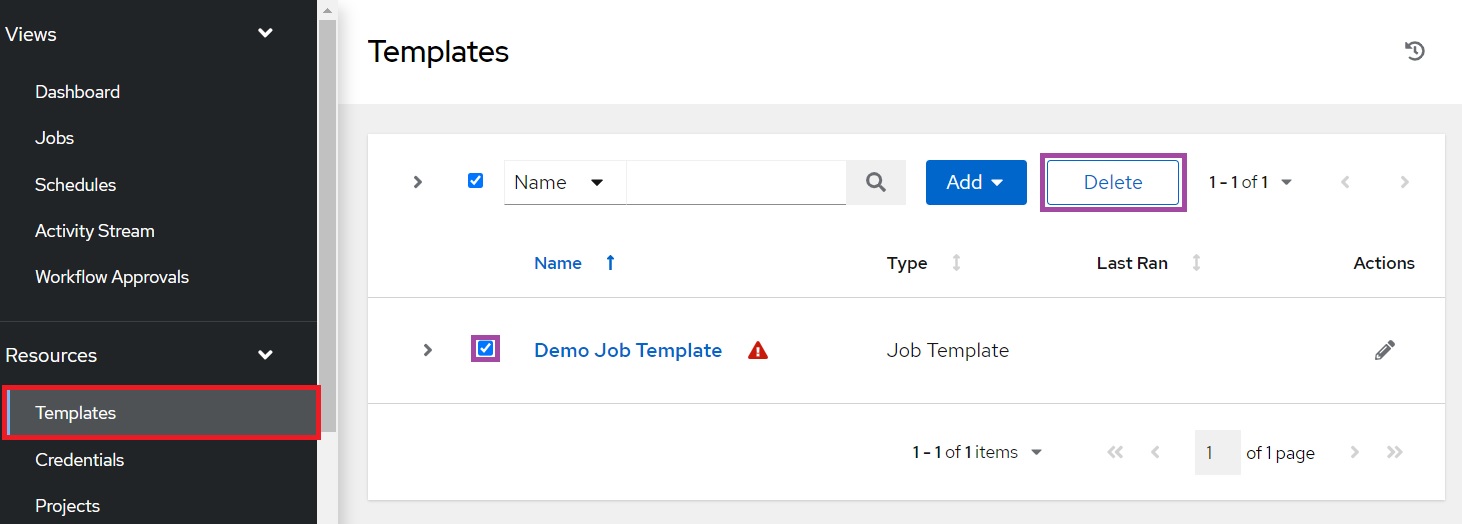


When finished, click **Save.**

## 4.6 Job Templates

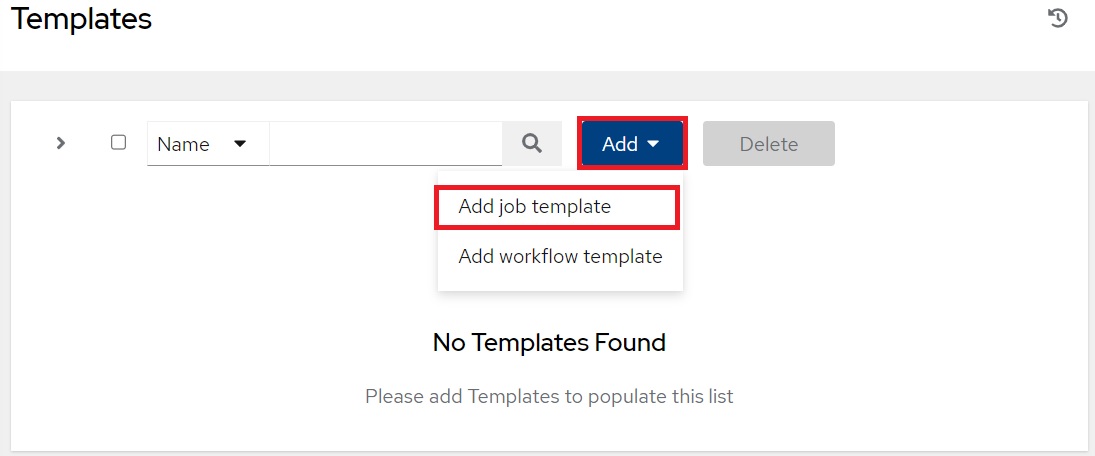
Now that the other pieces are in place, job templates can be created to run against host devices. The following guide will walk through the creation of a template that uses the '*pbfacts.yml*' playbook created earlier.

Click **Templates** from the left navigation menu. *Demo Job Template* canbe optionally removed by checking the box next to it and clicking **Delete**:



A window will appear warning that the action will delete *Demo Job Template*. Click **Delete**.

Click the **Add** button, then select **Add job template** from the drop-down menu:



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

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

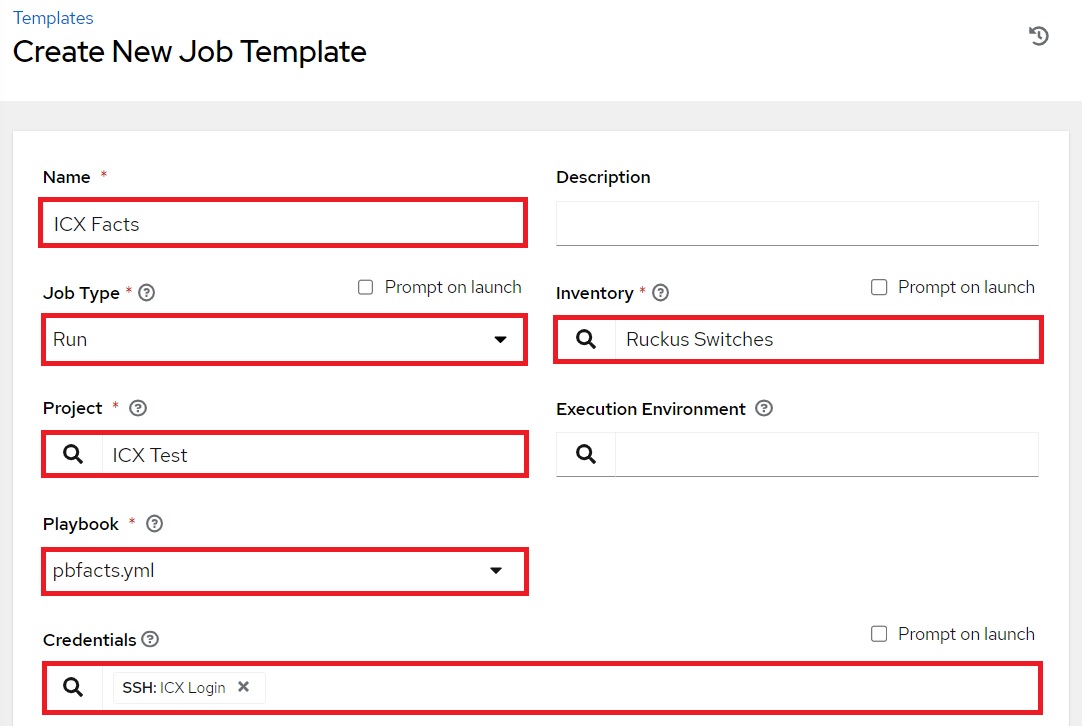
**Job Type**: Select **Run**.

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

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

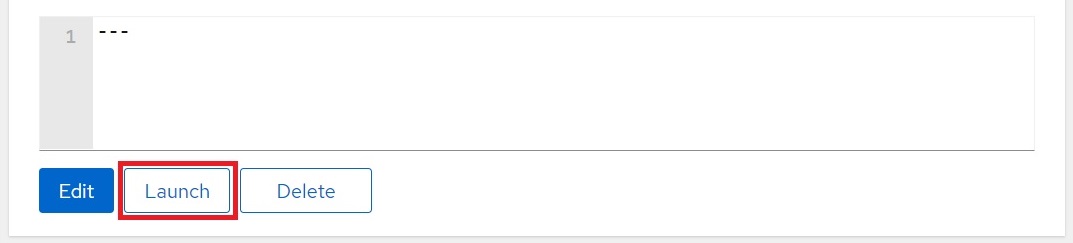
**Playbook**:Select **pbfacts.yml**.

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

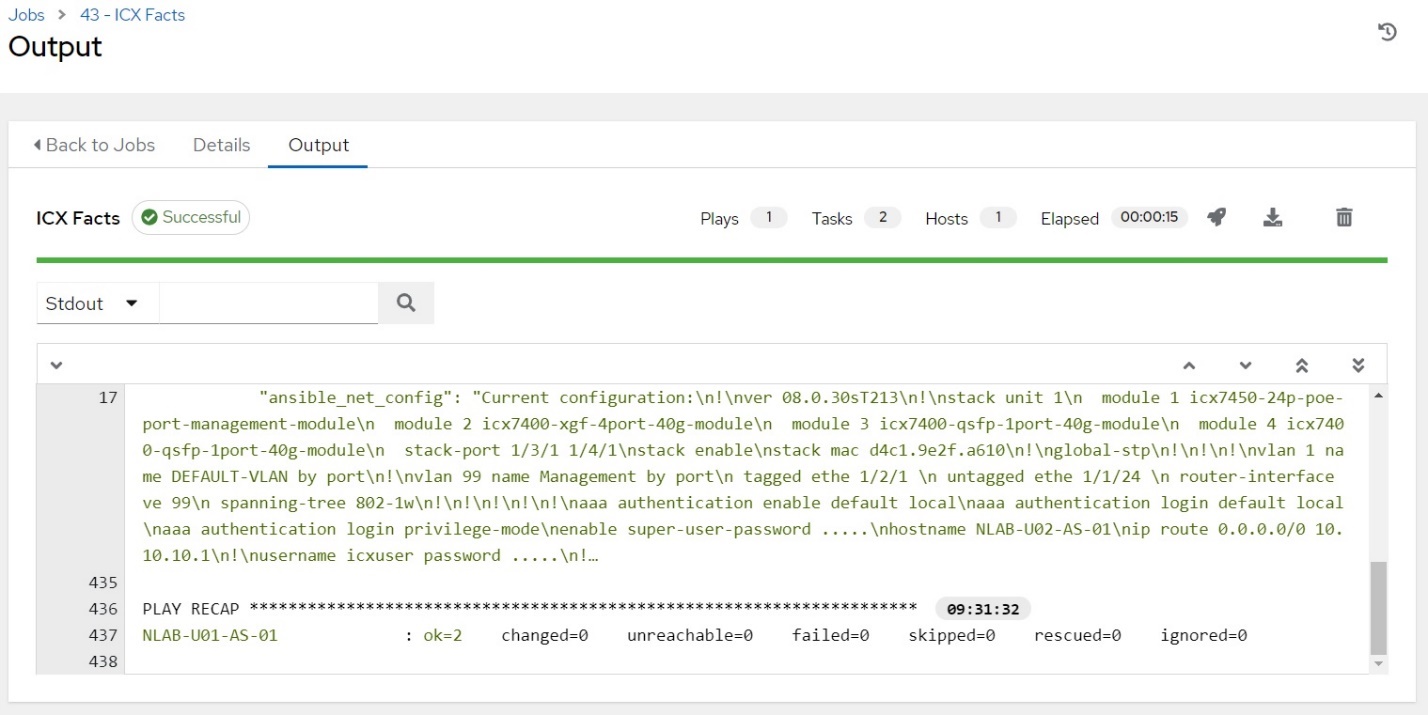


When finished, click **Save**.

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

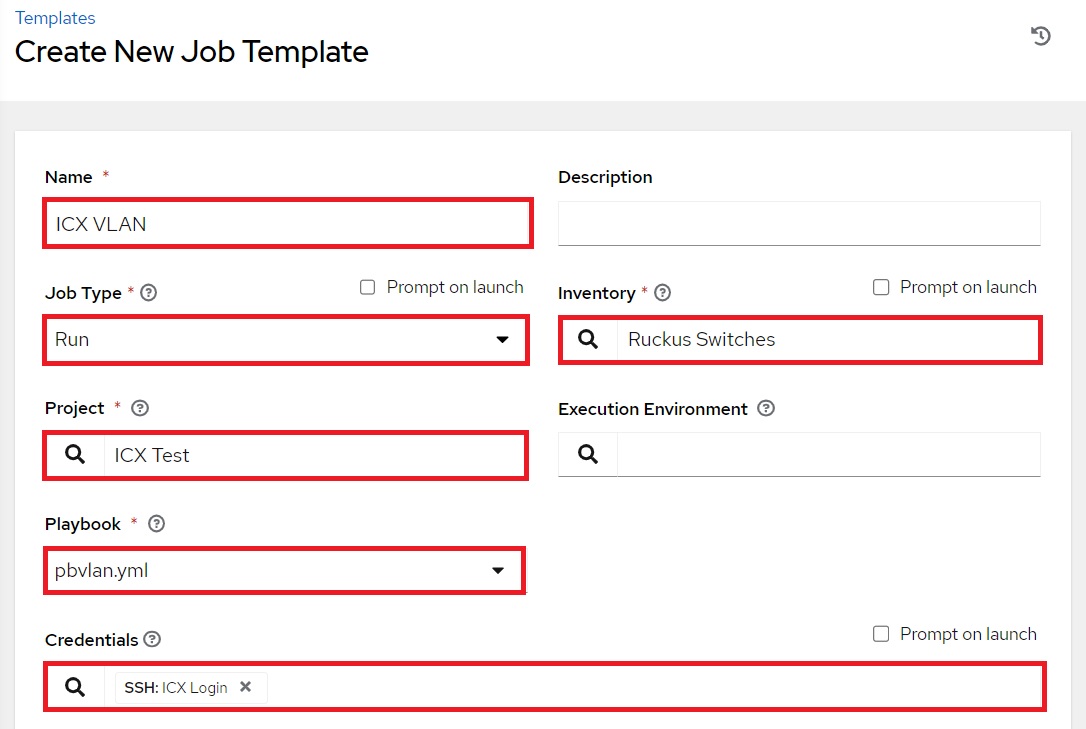


This will bring up the **Output** page with a status window:

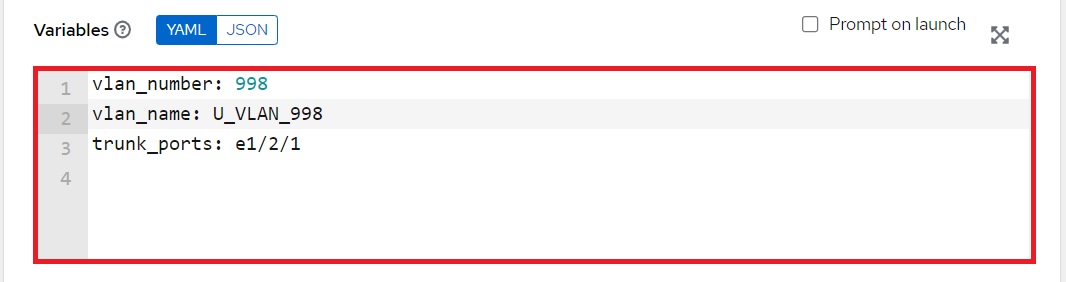


The status window will display an abbreviated output when the template has finished executing. You can click on the output to show it in full. Also note the color of the status text. Green means the template executed successfully whereas red would indicate a failure.

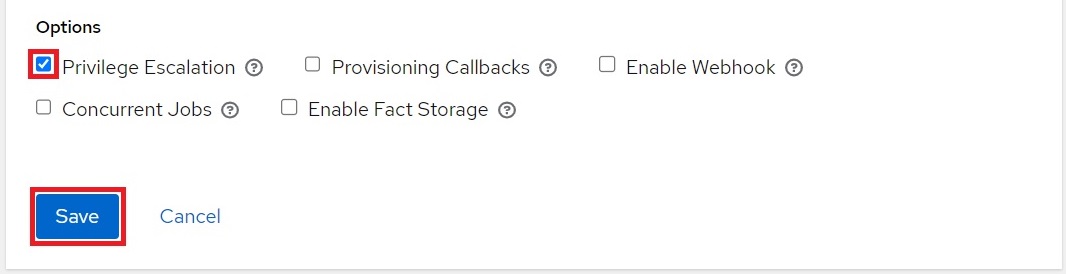
The next guide will walk through the creation of a template that uses variables. Go back to the **Templates** page and add a new job template. Configure the new template to use the '*pbvlan.yml*' playbook:



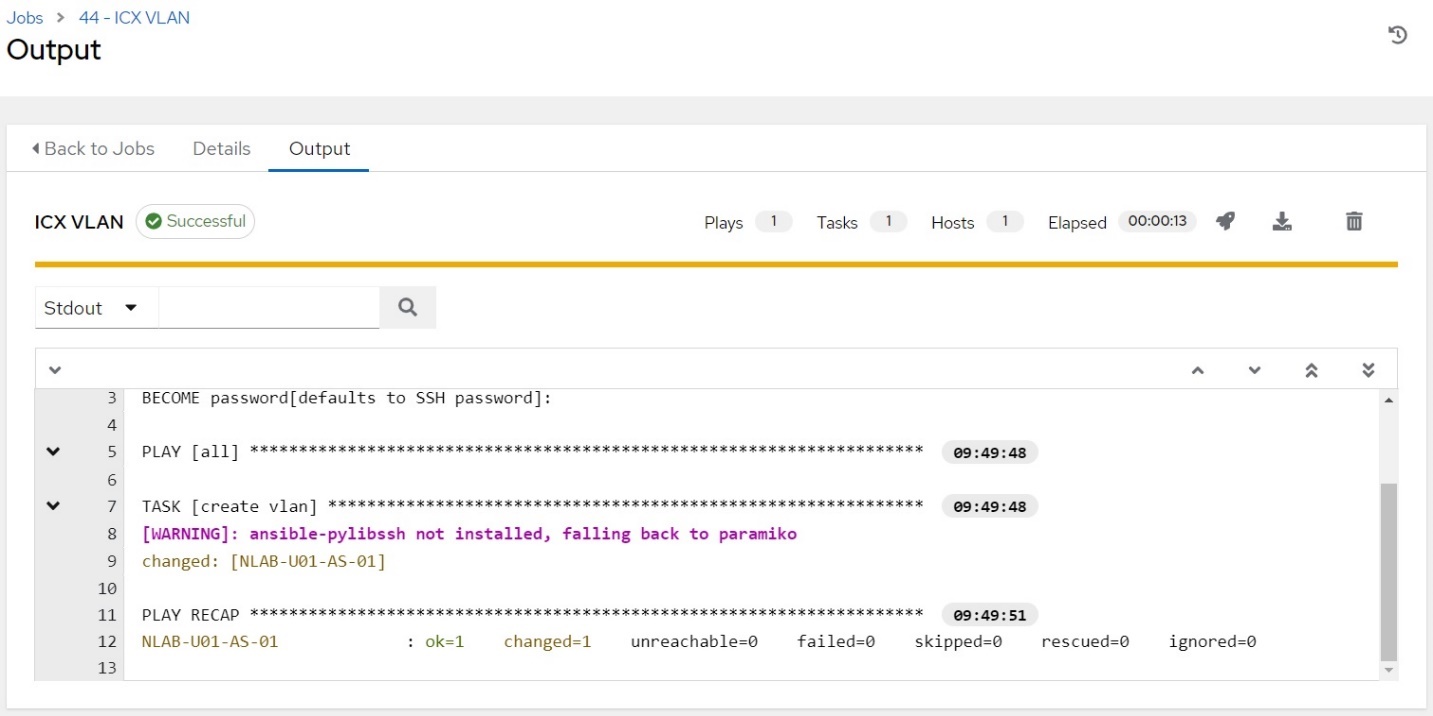
Scroll down to the **Variables** section. The three playbook variables can be defined as shown here:



Scroll down to the bottom and check the box next to **Privilege Escalation**, then 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:



# 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://github.com/ansible/awx-operator>

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/>