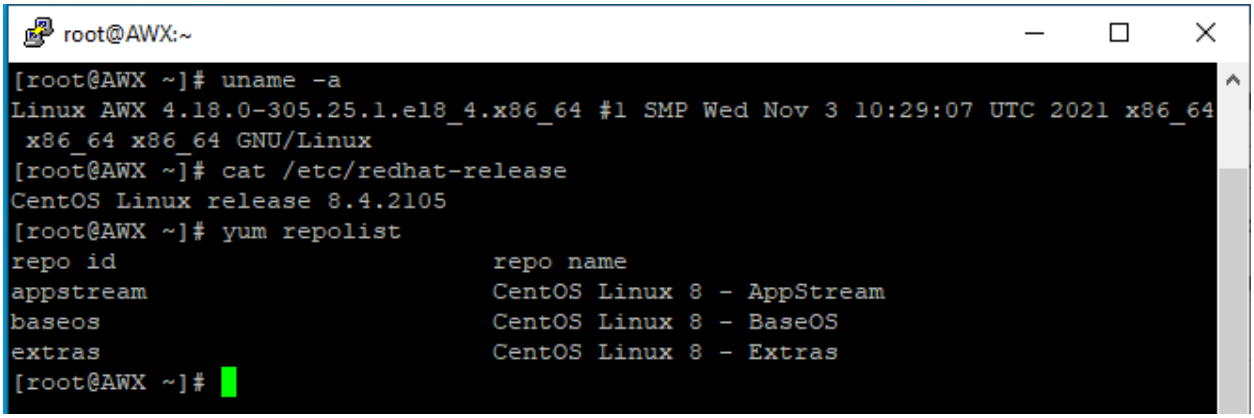
Configuration Manual - Automate Provisioning and Orchestration of Cloud Infrastructure using AWX

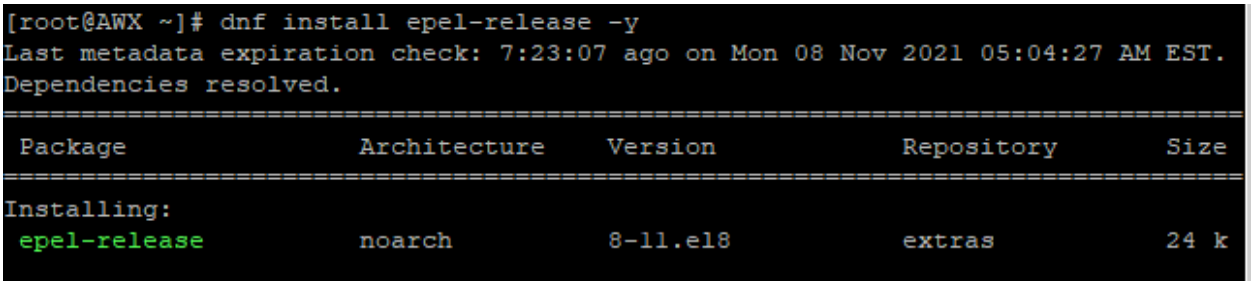
Setup the AWX host, specifications below

Type	Value
CPU Processor	Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz
No. of Cores	2
Memory	4G
Operating system	Centos 8.4
Ansible Version	2.9.18
AWX Version	17.1.0
Docker Version	20.10.10
Docker Compose	1.29.2

Step 1: Install Centos8 using the OracleVM image and install dependency packages.




```
# dnf install epel-release -y
```




```
# dnf install git gcc gcc-c++ ansible nodejs gettext device-mapper-
persistent-data lvm2 bzip2 python3-pip -y
```

```
[root@AWX ~]# dnf install git gcc gcc-c++ ansible nodejs gettext device-mapper-p
ersistent-data lvm2 bzip2 python3-pip -y
Extra Packages for Enterprise Linux Modular 8 - 532 kB/s | 955 kB     00:01
Extra Packages for Enterprise Linux 8 - x86_64  2.2 MB/s |  11 MB     00:04
```

```
# dnf config-manager --add-
repo=https://download.docker.com/linux/centos/docker-ce.repo
```

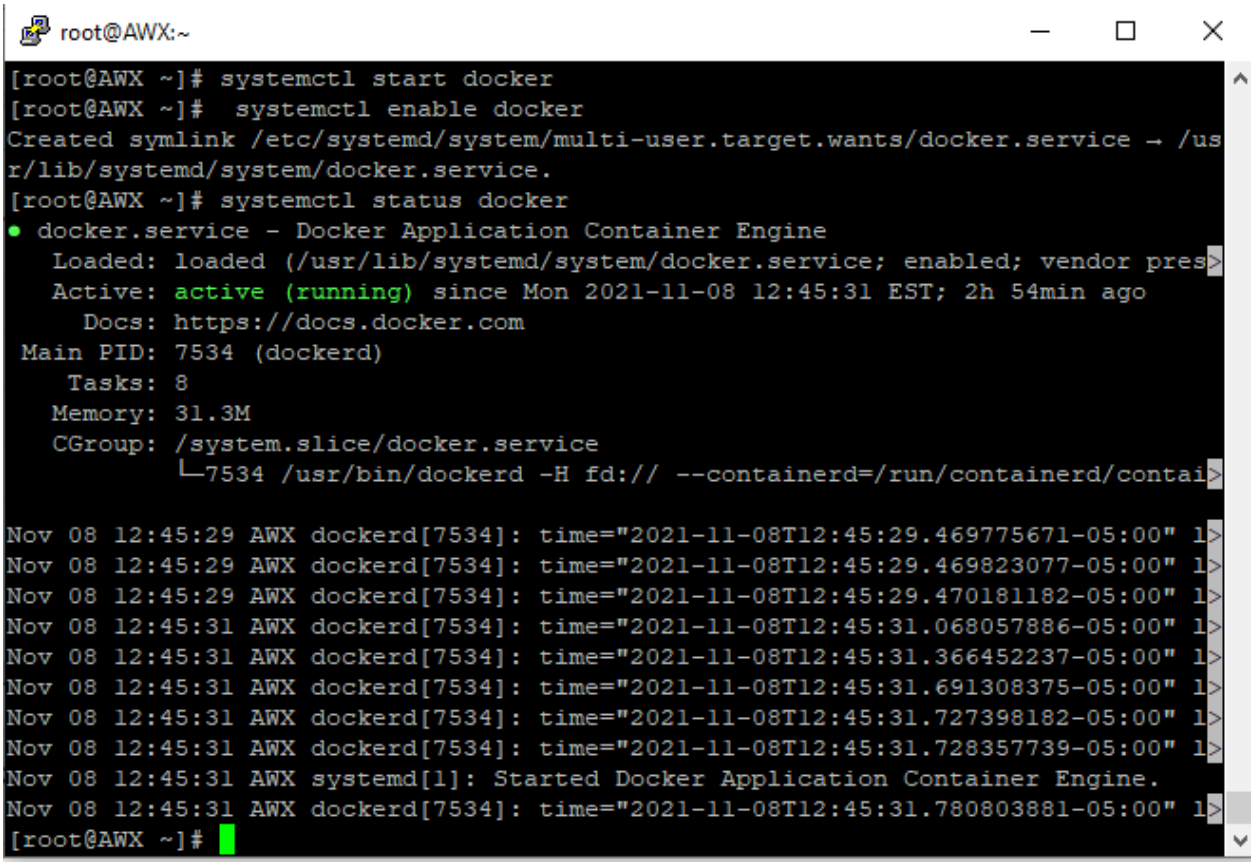
```
[root@AWX ~]# dnf config-manager --add-repo=https://download.docker.com/linux/ce
ntos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[root@AWX ~]#
```

```
# dnf install docker-ce.x86_64
```

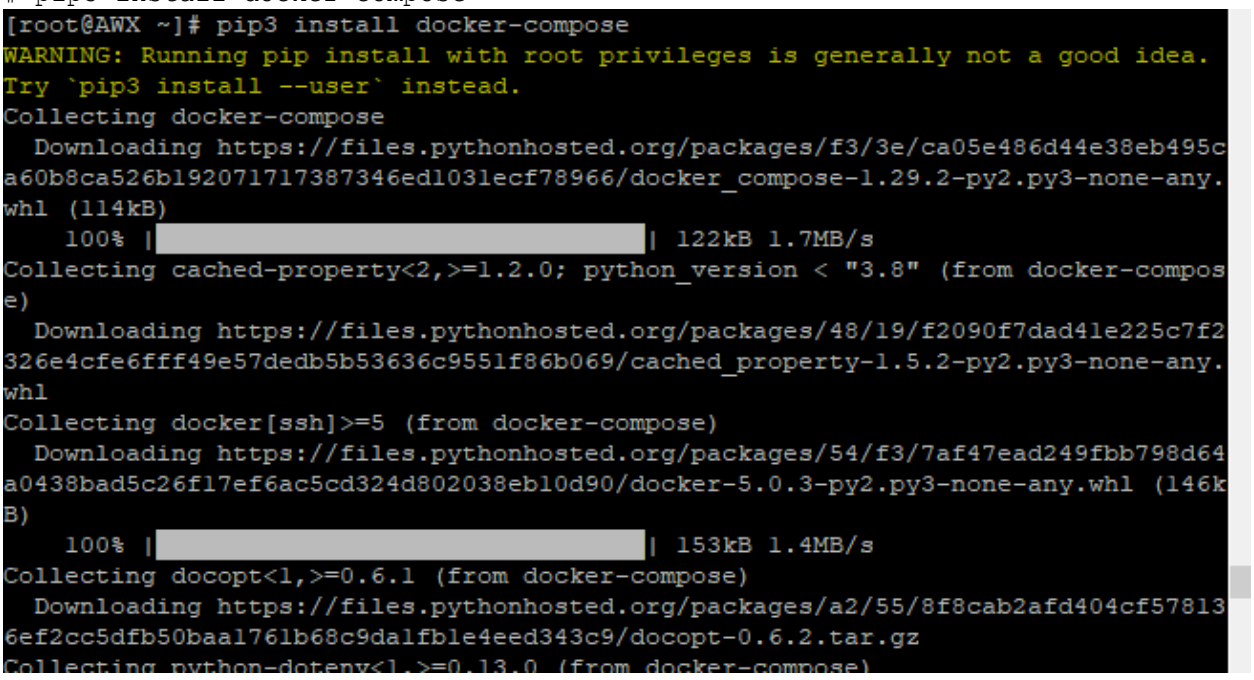
```
[root@AWX ~]# dnf install docker-ce.x86_64
Last metadata expiration check: 0:01:24 ago on Mon 08 Nov 2021 12:33:49 PM EST.
Dependencies resolved.
================================================================================
 Package                    Arch   Version              Repository        Size
================================================================================
Installing:
 docker-ce                  x86_64 3:20.10.10-3.el8     docker-ce-stable  22 M
Installing dependencies:
 checkpolicy                x86_64 2.9-1.el8            baseos           348 k
 container-selinux          noarch 2:2.167.0-1.module_el8.4.0+942+d25aada8
                                                        appstream         52 k
 containerd.io              x86_64 1.4.11-3.1.el8       docker-ce-stable  28 M
 docker-ce-cli              x86_64 1:20.10.10-3.el8     docker-ce-stable  29 M
 docker-ce-rootless-extras  x86_64 20.10.10-3.el8       docker-ce-stable 4.6 M
 docker-scan-plugin         x86_64 0.9.0-3.el8          docker-ce-stable 3.7 M
 fuse-common                x86_64 3.2.1-12.el8         baseos            21 k
 fuse-overlayfs             x86_64 1.6-1.module_el8.4.0+886+c9a8d9ad
                                                        appstream         73 k
```

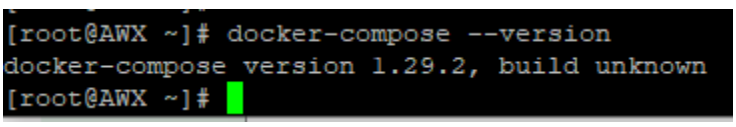
```
[root@AWX ~]# docker --version
Docker version 20.10.10, build b485636
[root@AWX ~]#
```

```
#systemctl start docker
#systemctl enable docker
#systemctl status docker
```



pip3 install docker-compose





Step 2: Import AWX packages from git and setup the application packages.

git clone https://github.com/ansible/awx.git

```
[root@AWX ~]# git clone -b 17.1.0 https://github.com/ansible/awx.git
Cloning into 'awx'...
remote: Enumerating objects: 292154, done.
remote: Counting objects: 100% (968/968), done.
remote: Compressing objects: 100% (444/444), done.
remote: Total 292154 (delta 530), reused 863 (delta 482), pack-reused 291186
Receiving objects: 100% (292154/292154), 248.70 MiB | 3.79 MiB/s, done.
Resolving deltas: 100% (225187/225187), done.
Note: switching to 'clab815c80cac96508d9779d92bc1280d0347627'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

[root@AWX ~]#
```

Update the installer file with (password, and user requirements) and install package using Ansible build playbook to deploy the AWX application.

vi /root/awx/installer/inventory

```
[root@AWX installer]# egrep -v ^'(#|$)' /root/awx/installer/inventory
localhost ansible_connection=local ansible_python_interpreter="/usr/bin/env pyth
on3"
[all:vars]
dockerhub_base=ansible
awx_task_hostname=awx
awx_web_hostname=awxweb
postgres_data_dir="/var/lib/awx/pgdocker"
host_port=80
host_port_ssl=443
docker_compose_dir="/var/lib/awx/awxcompose"
pg_username=awx
pg_password=postgrespass
pg_database=postgres
pg_port=5432
admin_user=admin
admin_password=Beauty123@
create_preload_data=True
secret_key=GP6XuvmaDbWg9P8wKClkTvhxclEOMMbPGIp4oNBi
awx_official=true
awx_alternate_dns_servers="8.8.8.8,8.8.4.4"
project_data_dir=/var/lib/awx/projects
[root@AWX installer]#
```

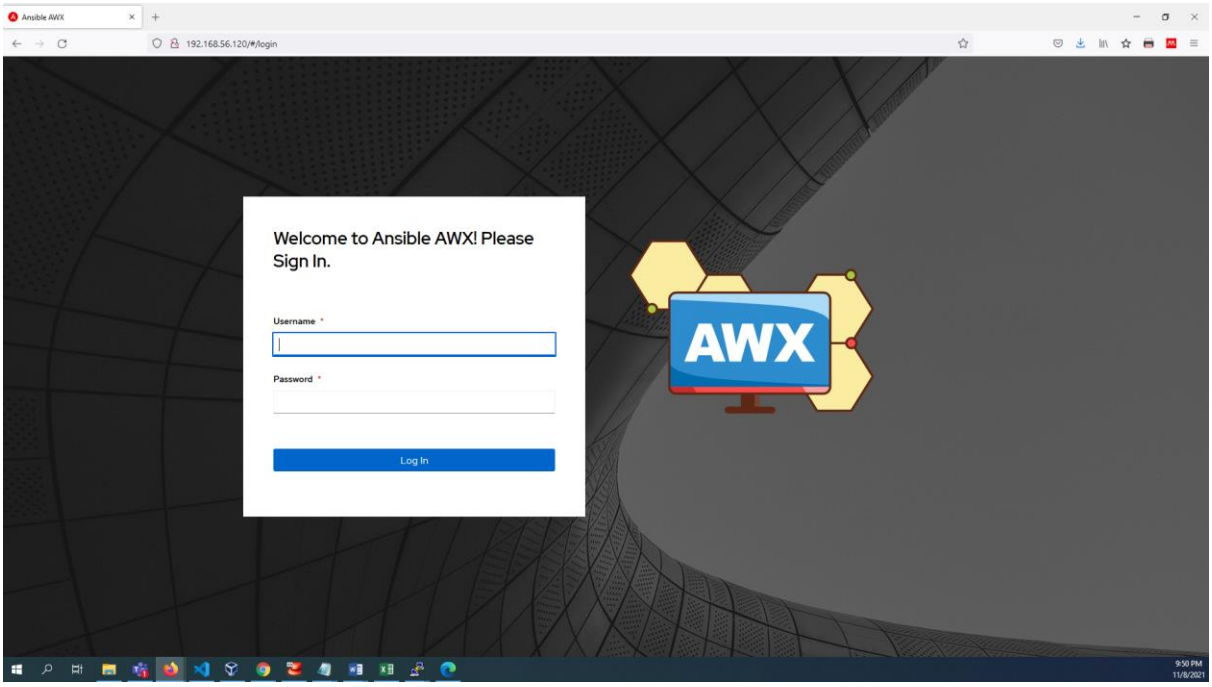
```
[root@AWX installer]# ansible-playbook -i ~/awx/installer/inventory ~/awx/instal
ler/install.yml -v
Using /etc/ansible/ansible.cfg as config file

PLAY [Build and deploy AWX] ****************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [check_vars : admin_password should be defined] ***************************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [check_vars : include_tasks] **********************************************
skipping: [localhost] => {"changed": false, "skip_reason": "Conditional result w
as False"}
```

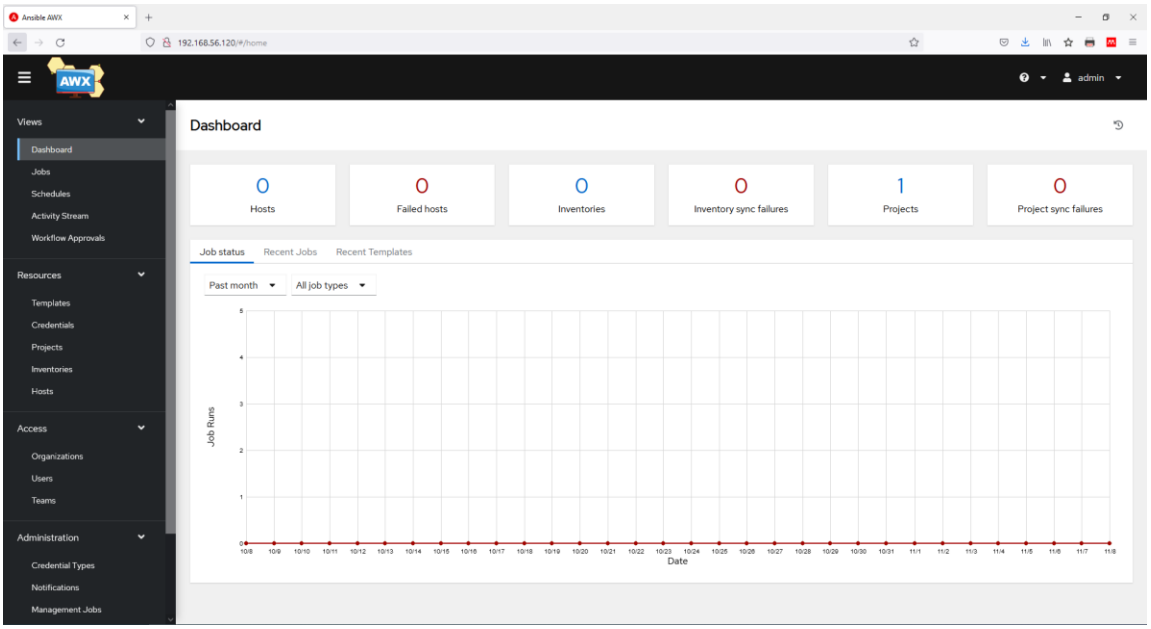
AWX application runs under docker compose, you could see four docker containers awx_web (handle web requests), aws_task(perform the ansible tasks), awx_postgres (maintain database), awx_redis(in-memory caching solution).

```
[root@AWX installer]# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS                                      NAMES
d3d09dc37544   ansible/awx:17.1.0   "/usr/bin/tini -- /u…"   3 minutes ago   Up 2 minutes   8052/tcp                                   awx_task
29c179abe214   ansible/awx:17.1.0   "/usr/bin/tini -- /b…"   9 minutes ago   Up 2 minutes   0.0.0.0:80->8052/tcp, :::80->8052/tcp      awx_web
a432e3bcecec   postgres:12          "docker-entrypoint.s…"   9 minutes ago   Up 2 minutes   5432/tcp                                   awx_postgres
f9ea3345f8c4   redis                "docker-entrypoint.s…"   9 minutes ago   Up 2 minutes   6379/tcp                                   awx_redis
[root@AWX installer]#
```

```
[root@AWX installer]# firewall-cmd --zone=public --add-masquerade --permanent
success
[root@AWX installer]# firewall-cmd --permanent --add-service=http
success
[root@AWX installer]# firewall-cmd --permanent --add-service=https
success
[root@AWX installer]# firewall-cmd --reload
success
[root@AWX installer]#
```

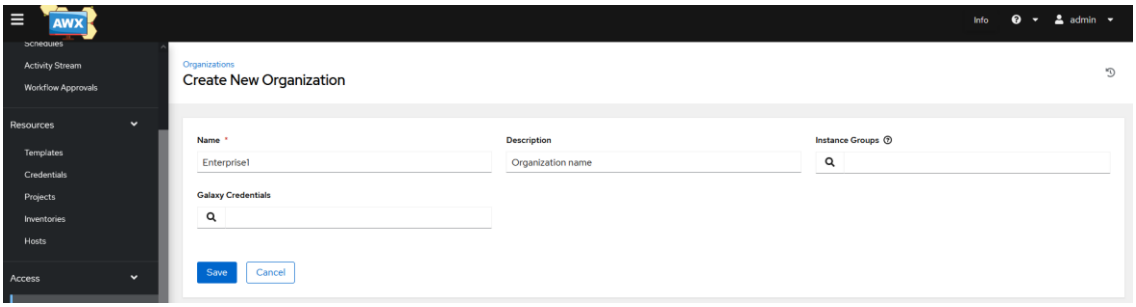
Login to the console using the password configured in above section:





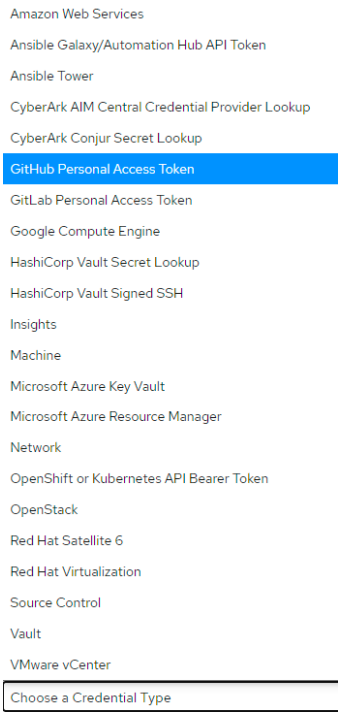
Create Organization: Navigate from left side options select Organization under Access.

I have created a Organization with name **Enterprise1.**



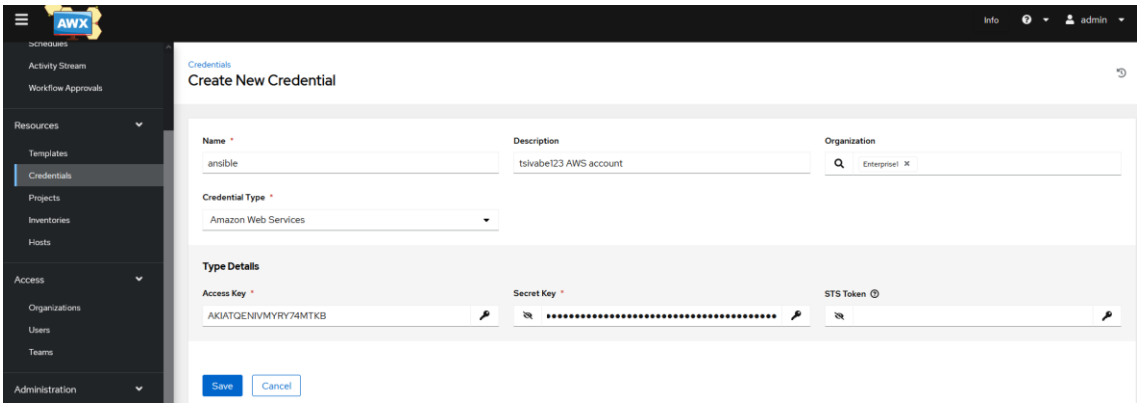
I Configure AWS credential / GCP credential / Machine credential (on-prem server password)

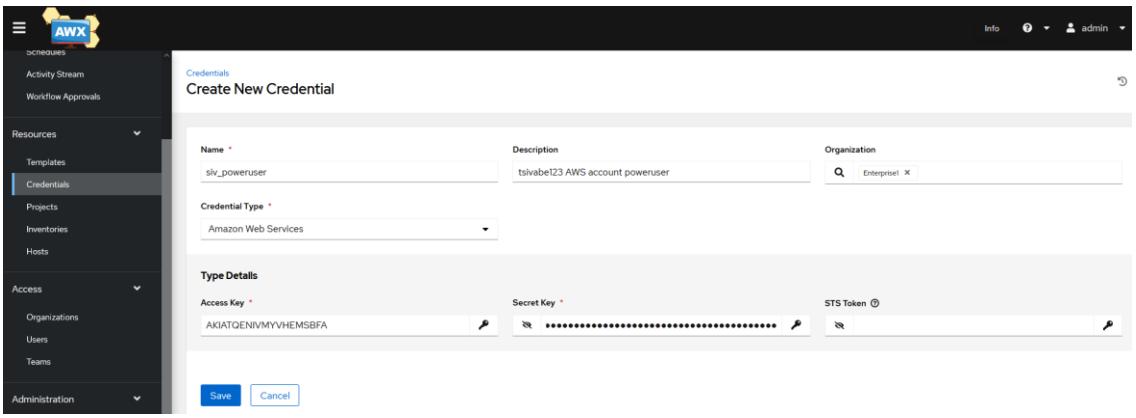
Below services can be integrated using this tool by configuring its credentials.



Create credentials: Navigate from left side options select credentials under Resources column.

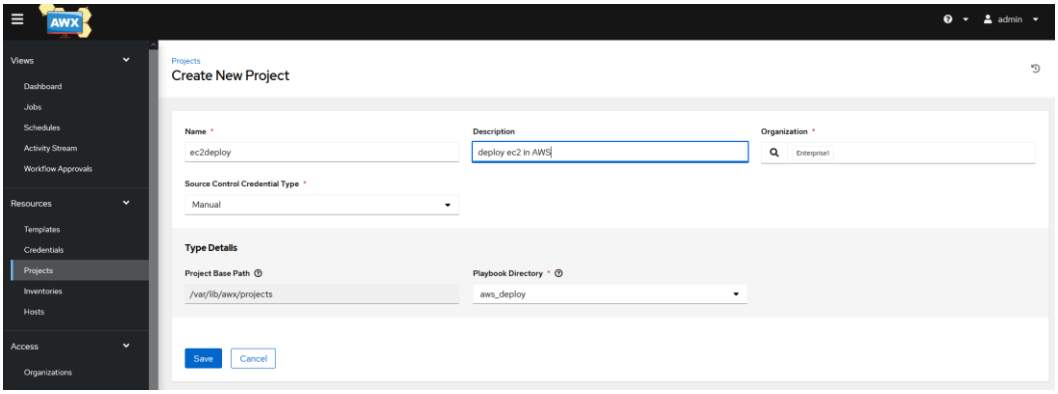
Provide Access key and secret access key. For Google cloud provide the JSON token file by uploading it in the credential column.



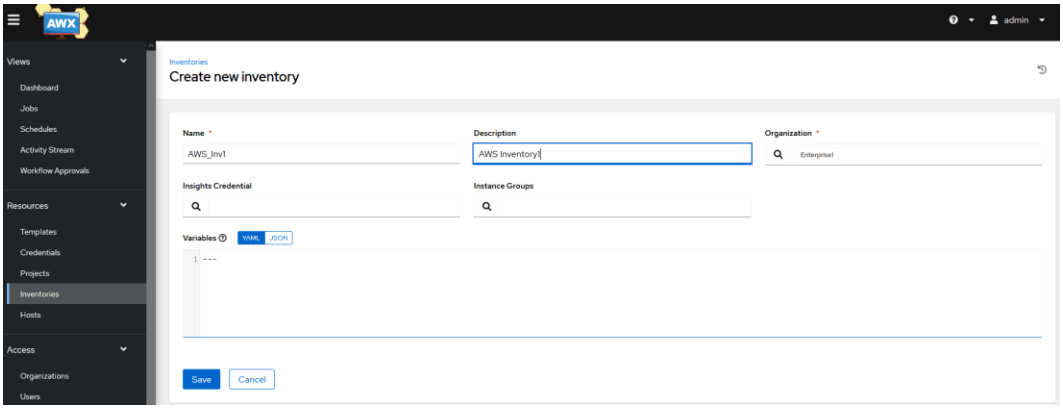


Create Project : Name project and map the playbook.

Default project location in server is under (/var/lib/awx/projects) folder, playbooks hosted as separate folder init.

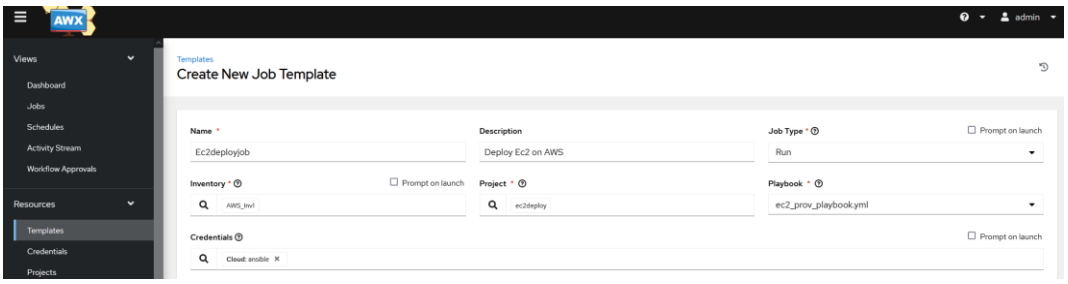


Create inventory: Under Resources >> inventories



Create playbooks for server and cluster deployment (playbook codes are attached as separate file to the project submission source code folder.

Create Job template: Map the master playbook from the project folder, tag the Inventory group and EC2 credential as this job is ti deploy EC2 instances

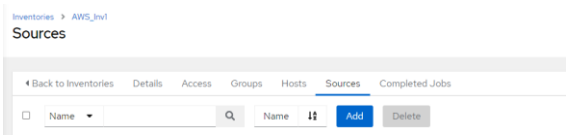


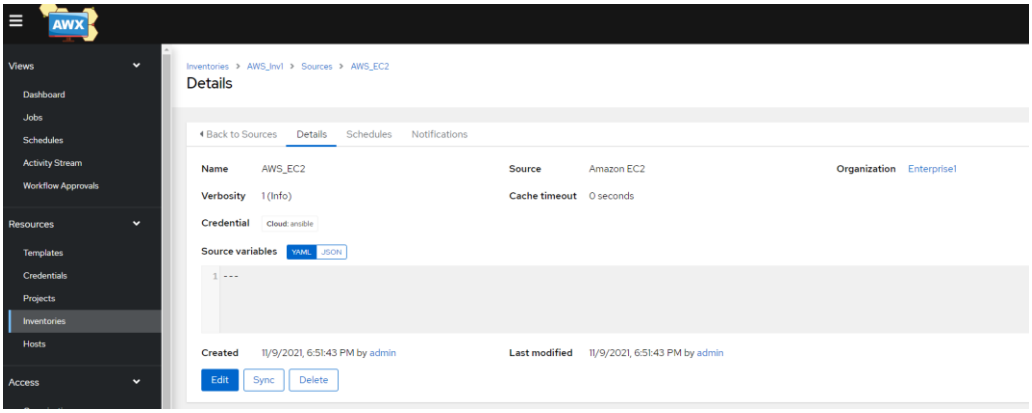
A) 1st Project (Deploy Ec2 instance)

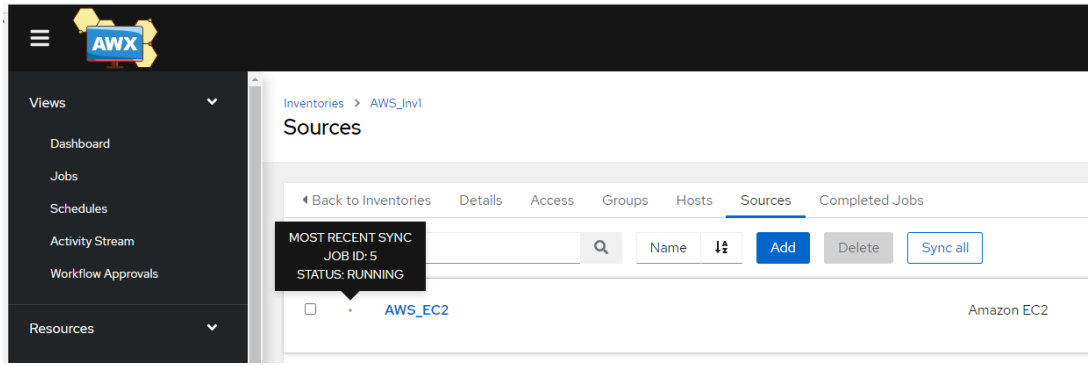
1) Create keypair - aws-awk-key-us-east-1 on the region.
2) Once the playbook ran it will deploy the ec2 and sg and vpc etc

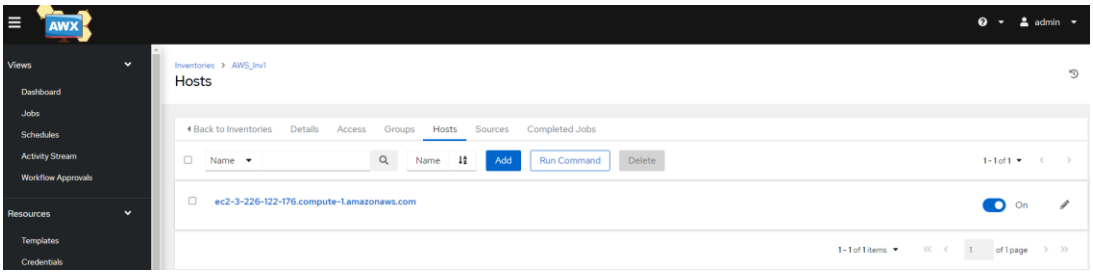
Create dynamic inventory: Under Resources >> inventories >> sources

Add AWS credential to enable sync, this help automatic inventory capture every time it syncs.



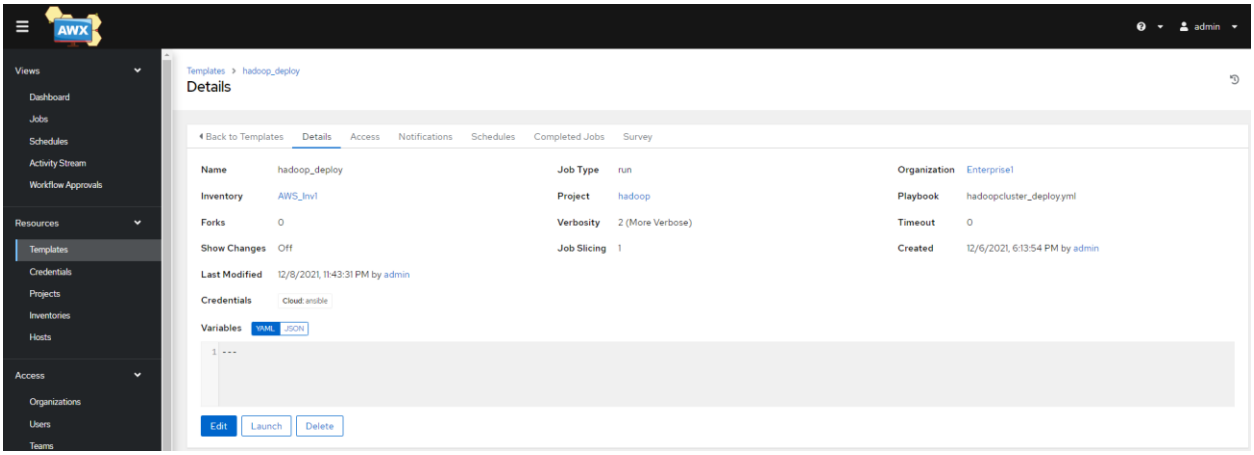


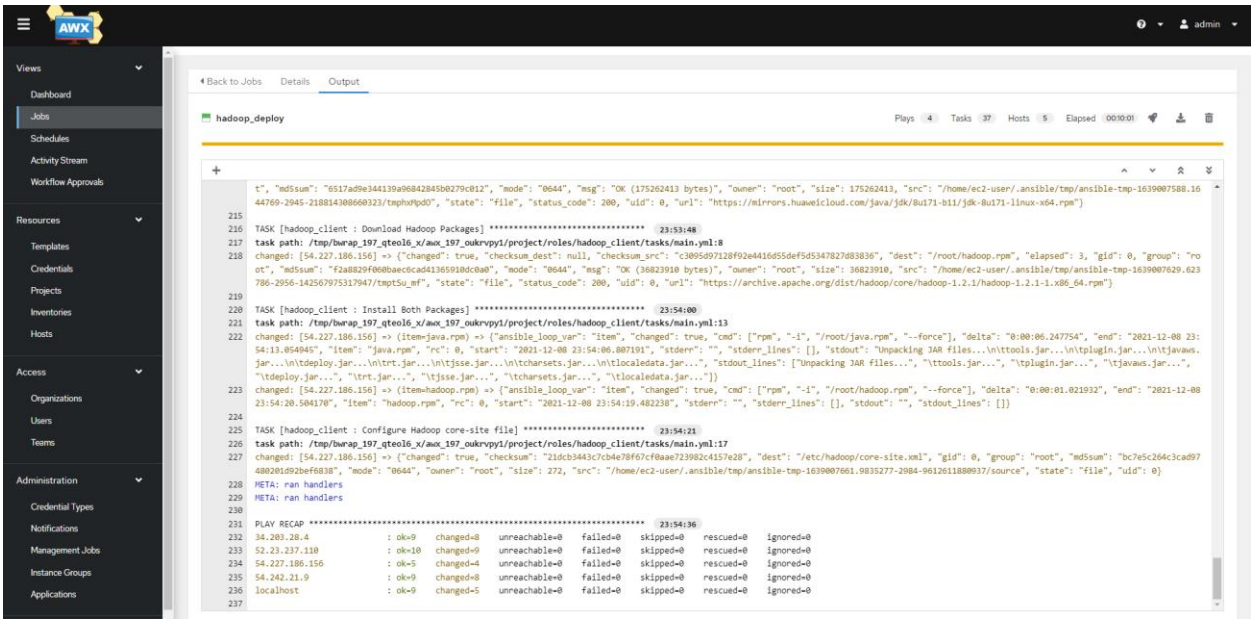


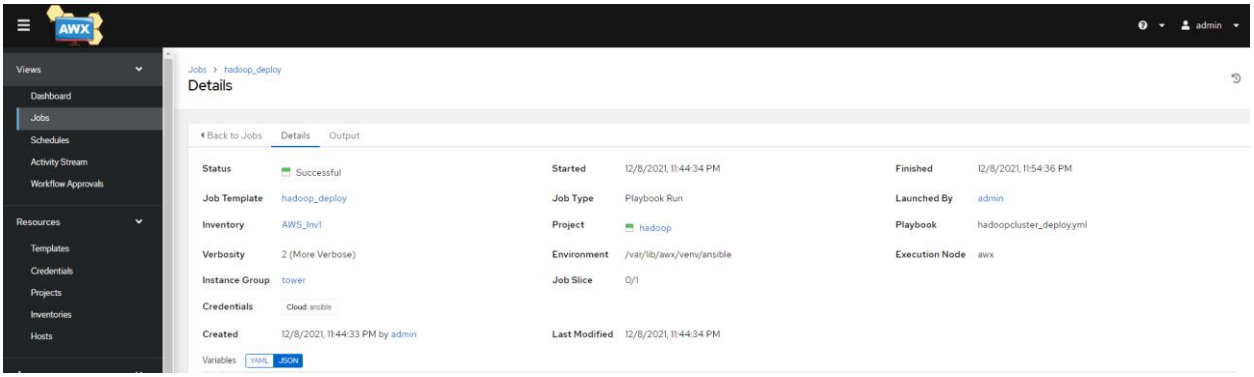


Hadoop Cluster Implementation:

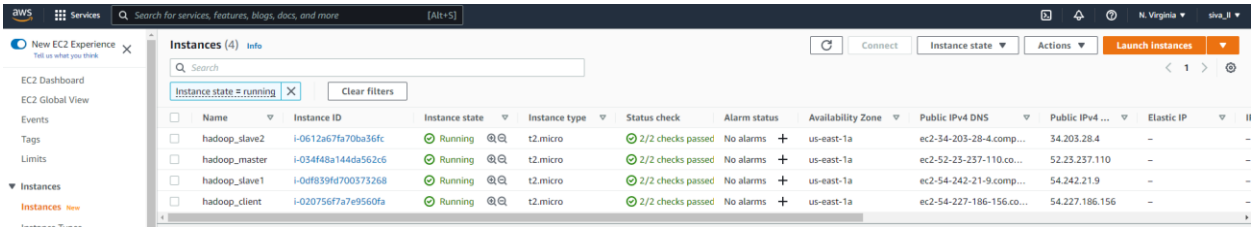
Create the Hadoop deploy Project and the Template.



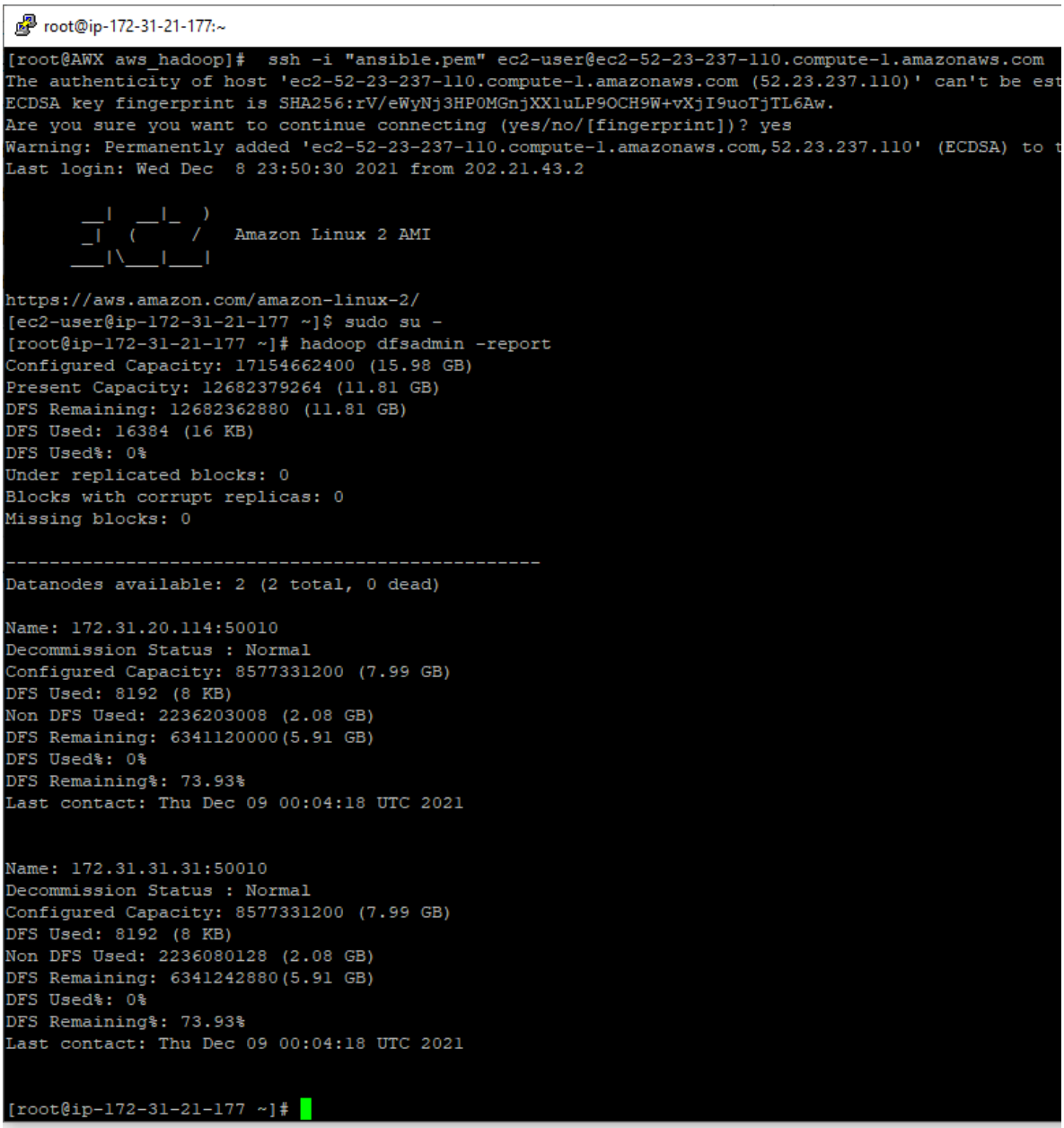




Validate the AWS console:

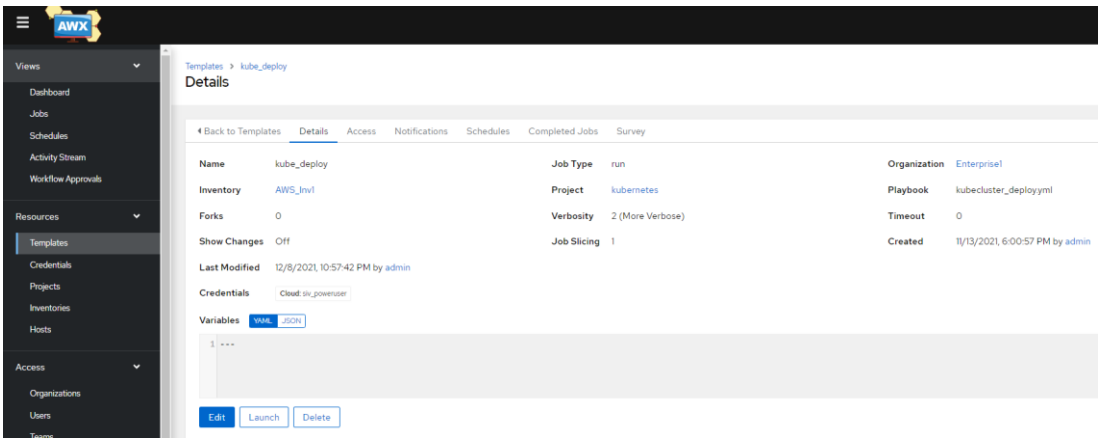


Validate the Cluster by login into the Master (Namenode) using command # hadoop dfsadmin –report

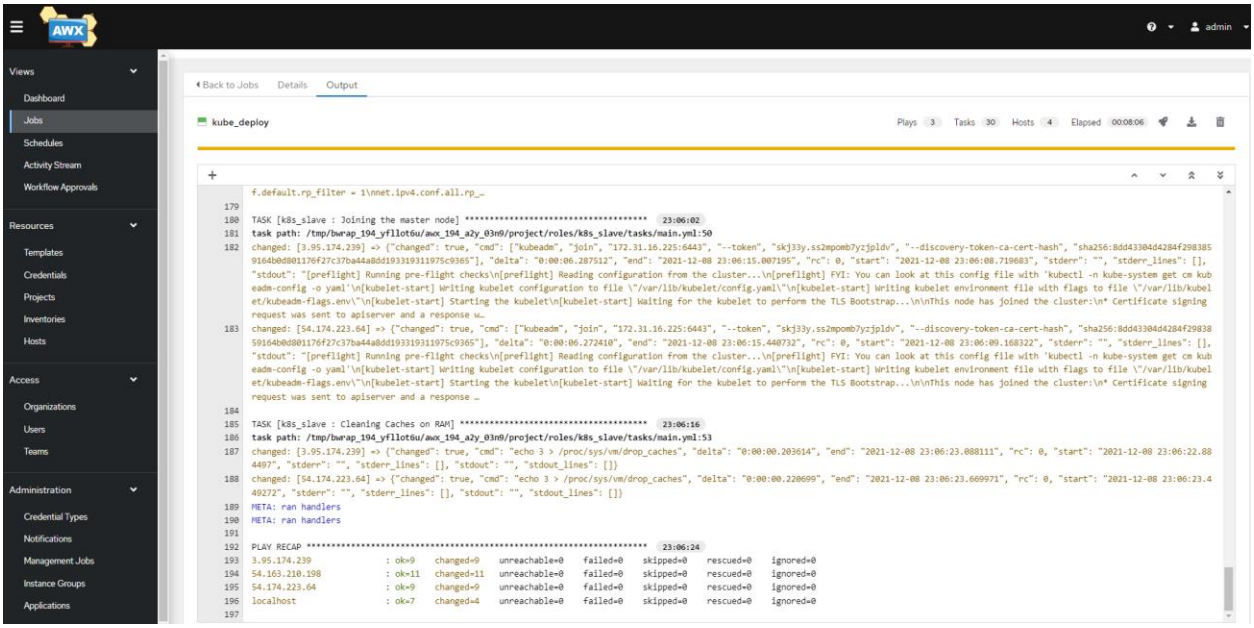


Kubernetes Cluster Implementation:

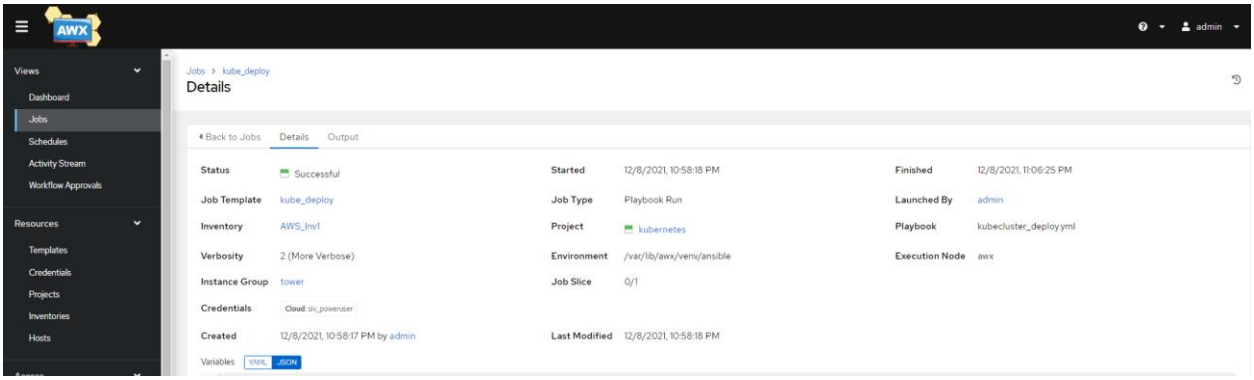
Create Kubernetes Projects, Project Template in awx console.



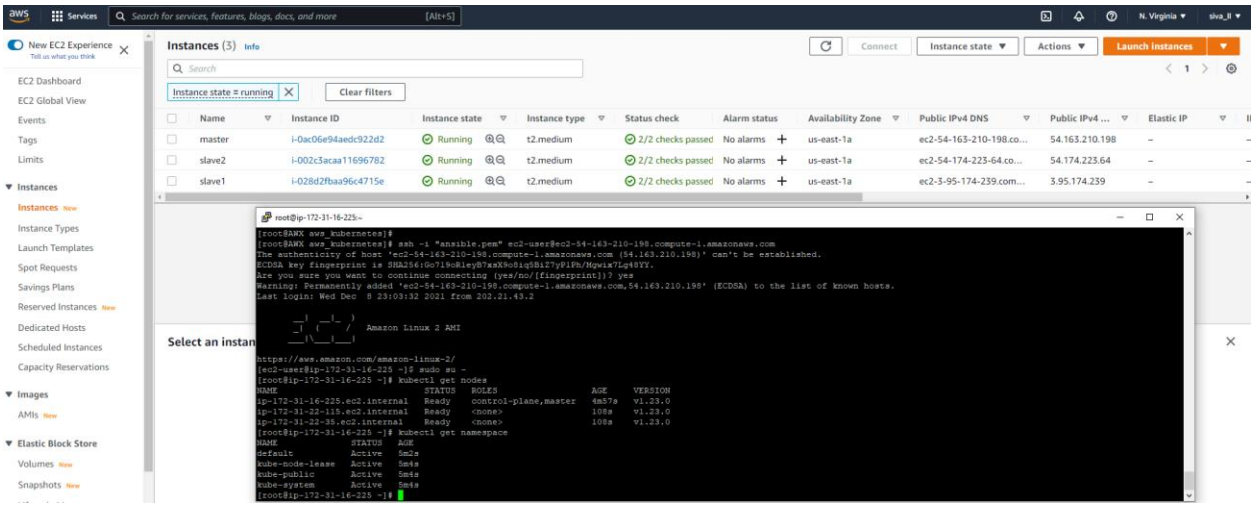
Executing Kubernetes deploy playbook from awx console.



Job Validation from AWX side:

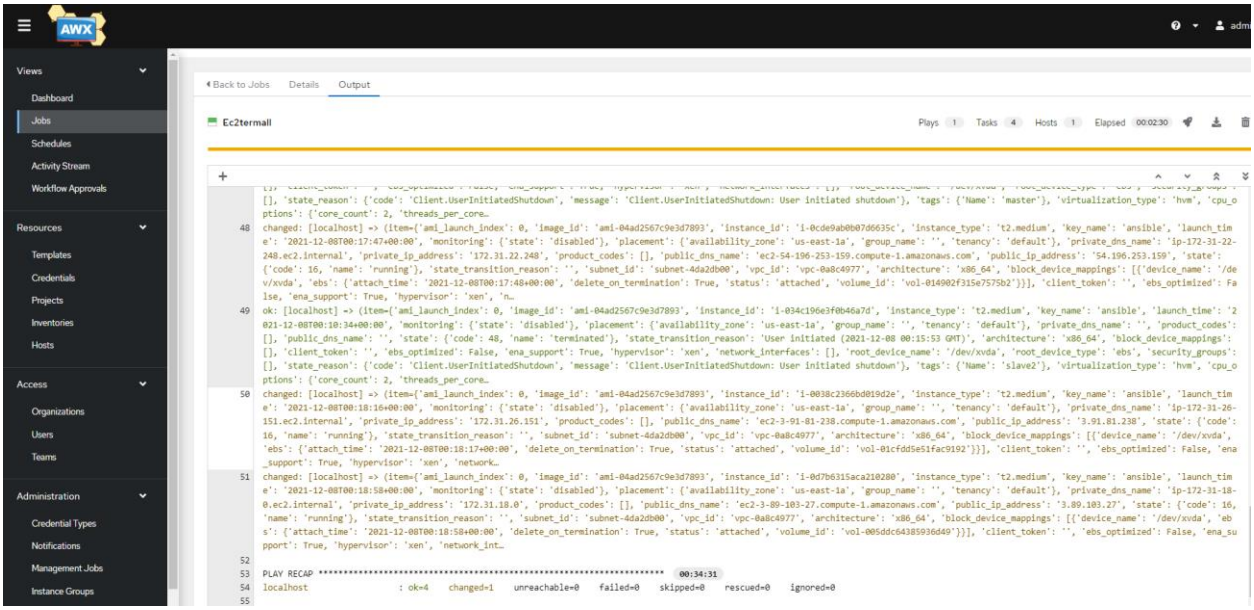


Job Validation from AWS console and Server side:

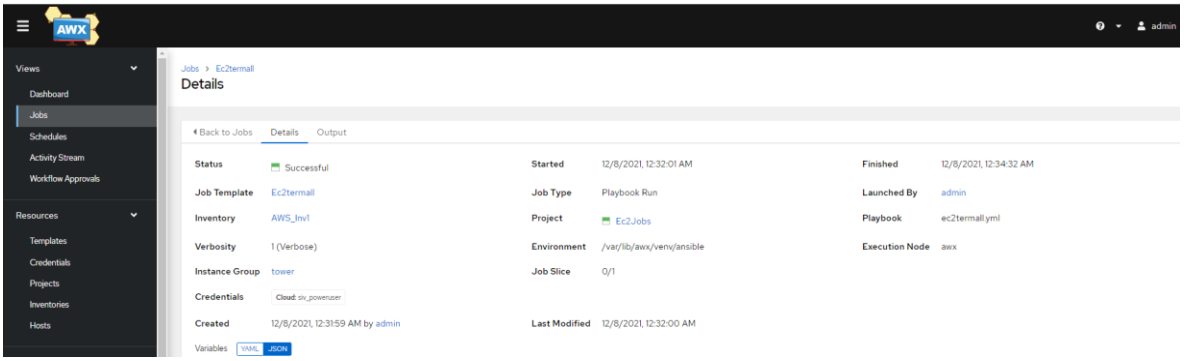


Terminate the AWS servers if not in use:

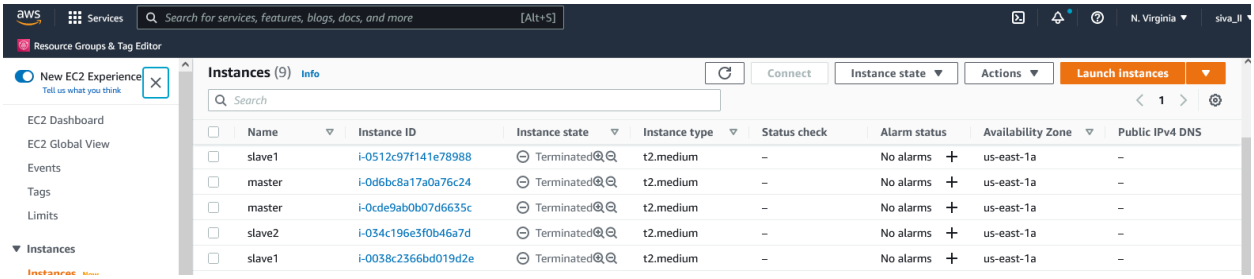
Job Execution from AWX Console:



Validate Job status from AWX Console:

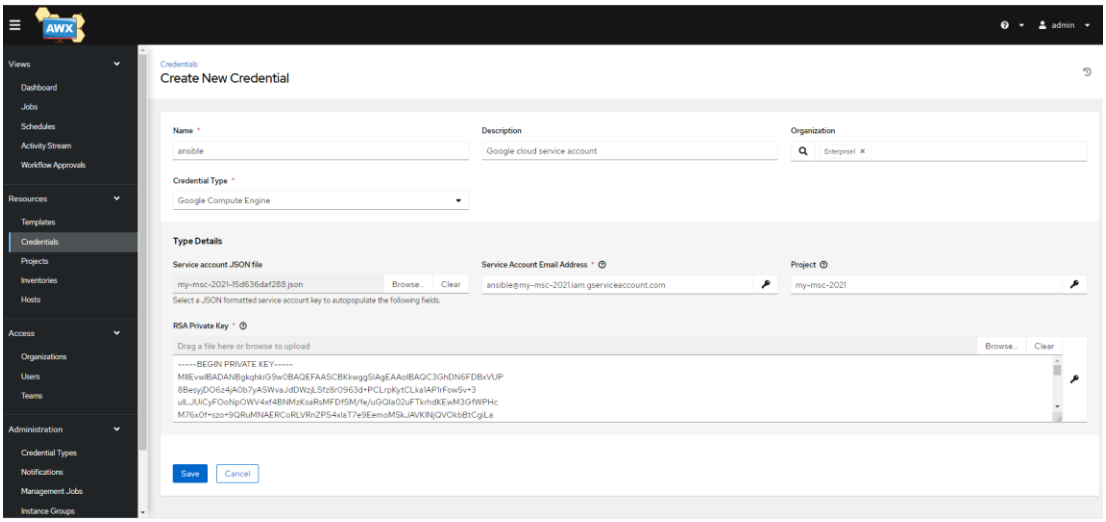


Validate the server removal status from AWS Cloud console:

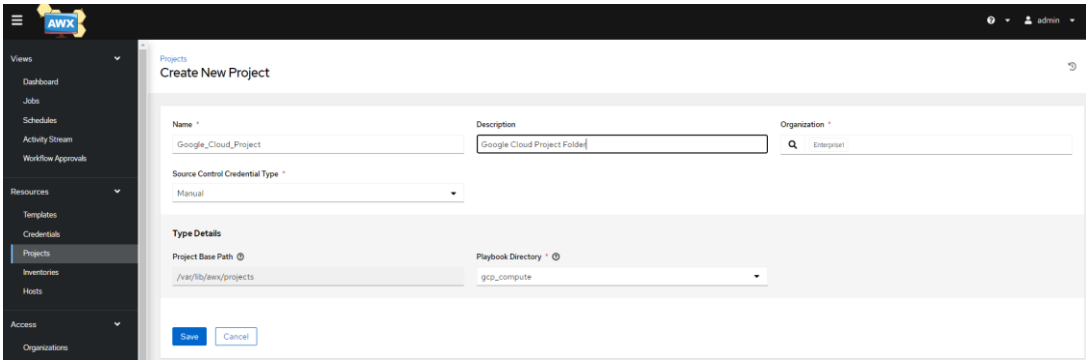


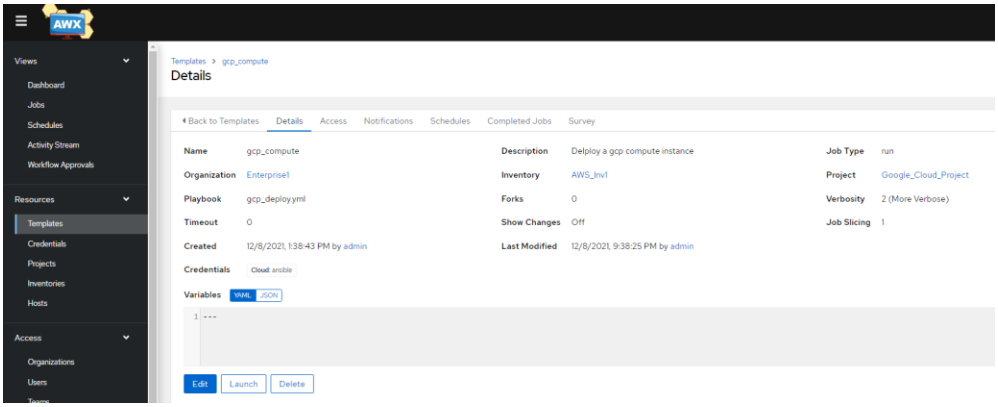
Create Google Cloud resources using Ansible and AWX:

Adding GCP service account to AWX-

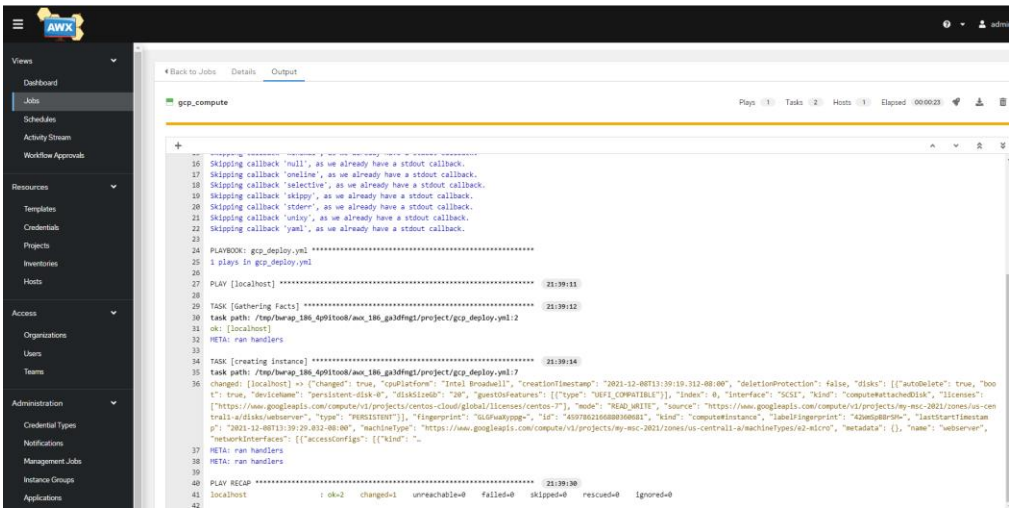


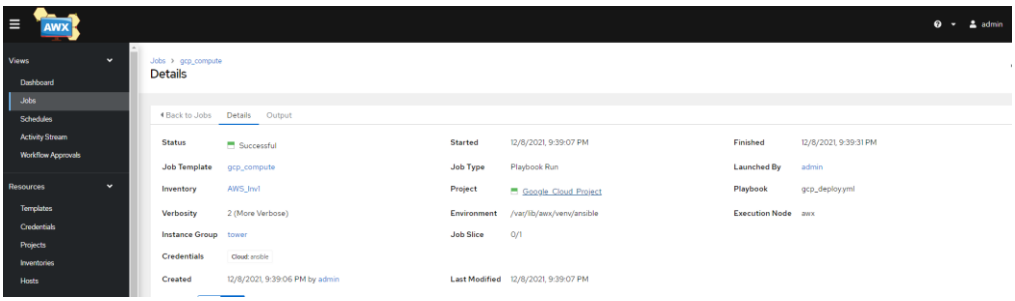
Create Project for GCP



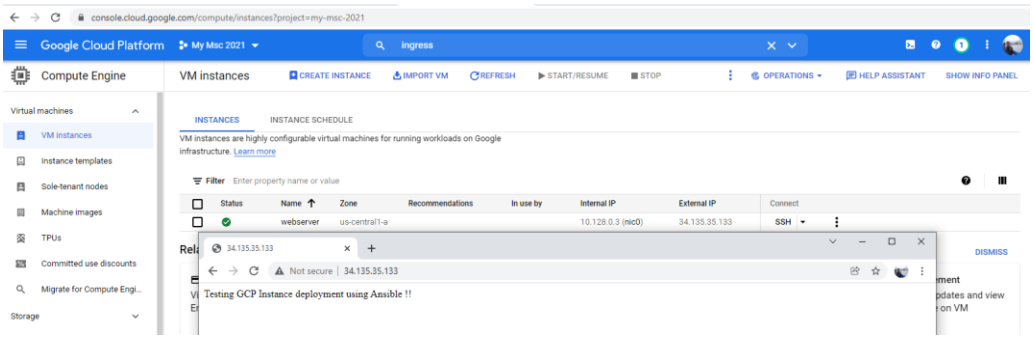


Deploy compute instance deploy job



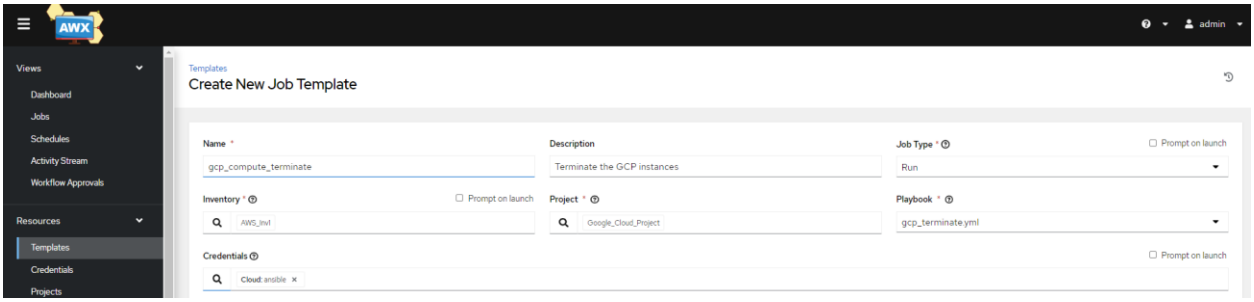


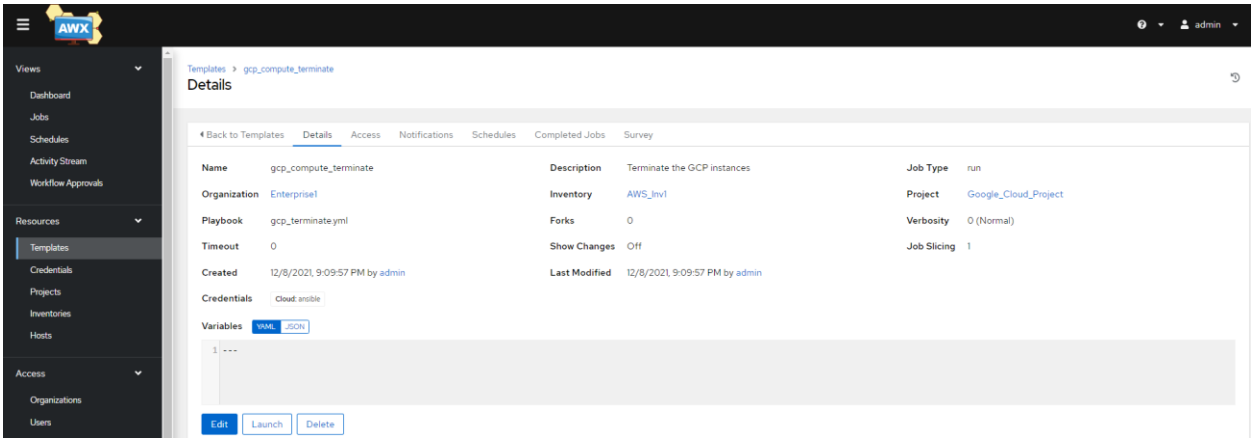
Validate the server status and validate weburl for webservice GCP console:

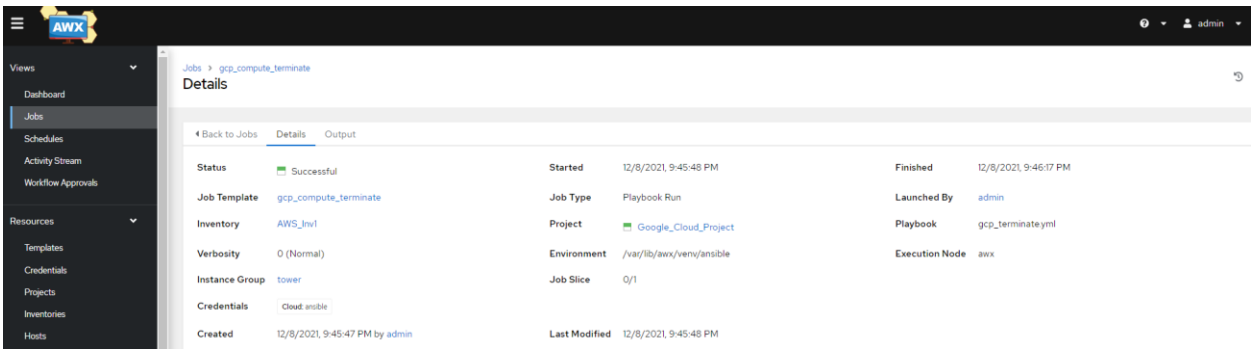


Terminate the GCP Instance:

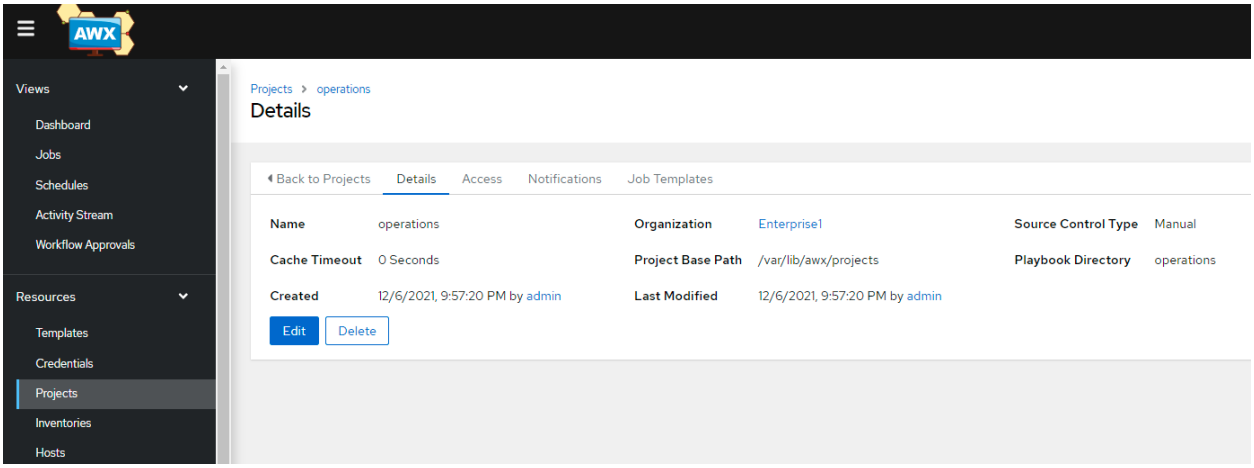
Update the Terminate yaml file on project folder and create Job template.

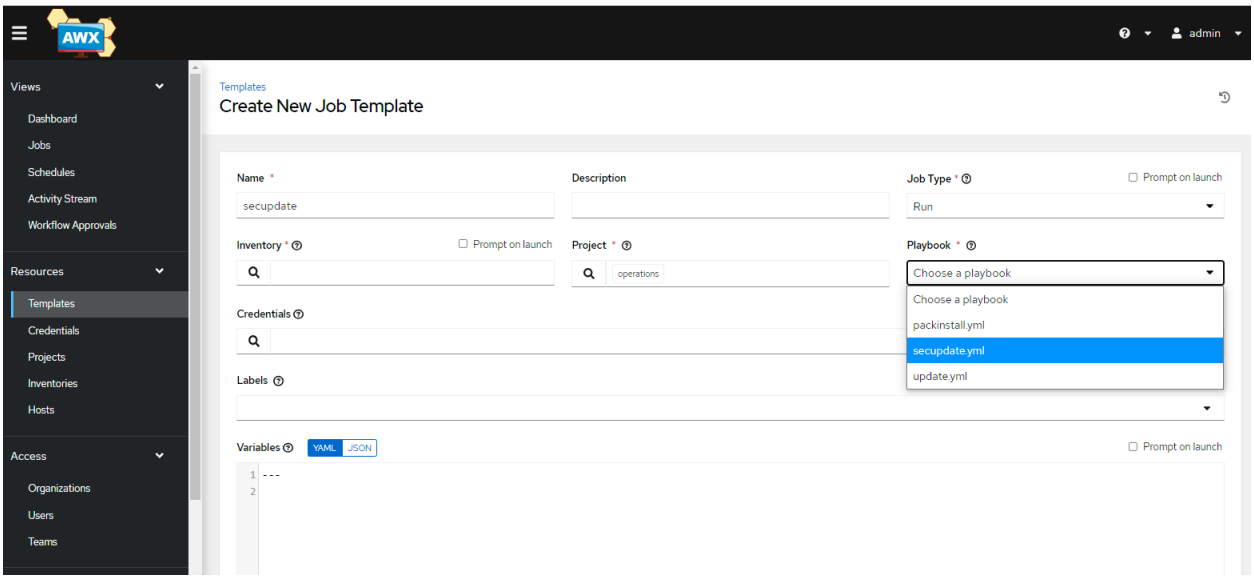






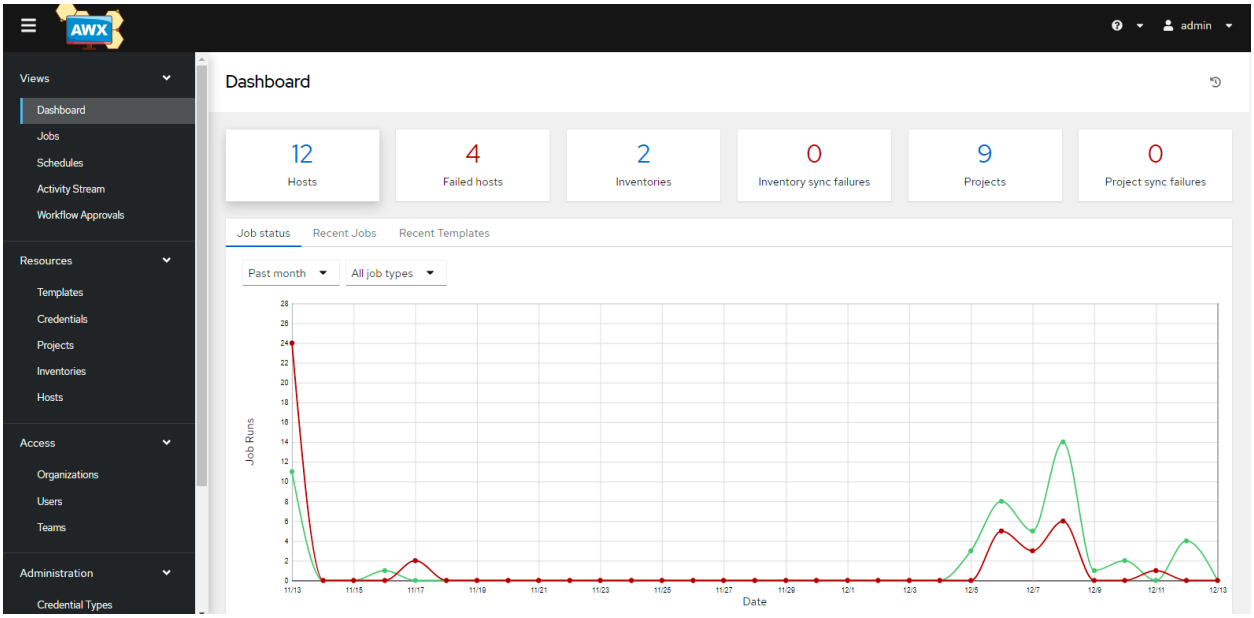
Additional system administration tasks playbook configured under operations folder:





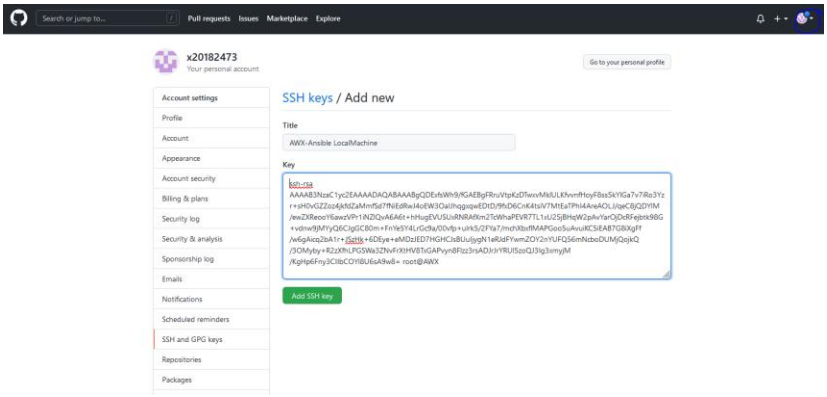
Monitor and control all the jobs using Centralized dashboard:

You can view Hosts, Failed Hosts, Inventories, Projects and Git project sync failures on Dashboard for better centralized control of overall tasks.

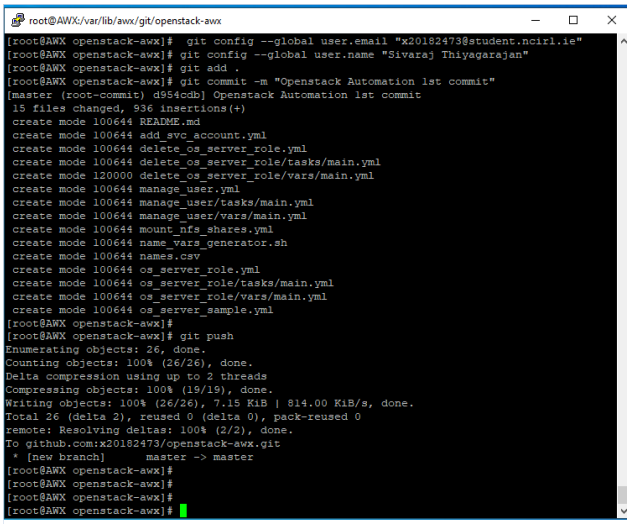


Additional work tried to enable GIT and openstack implementation but couldn't make it work. Kindly ignore below logs:

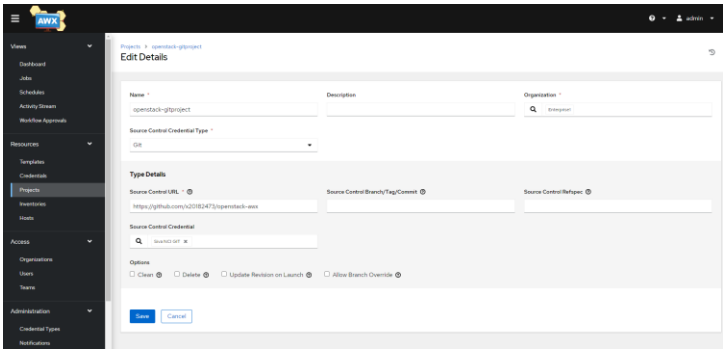
Enable GIT Passwordless Authentication:

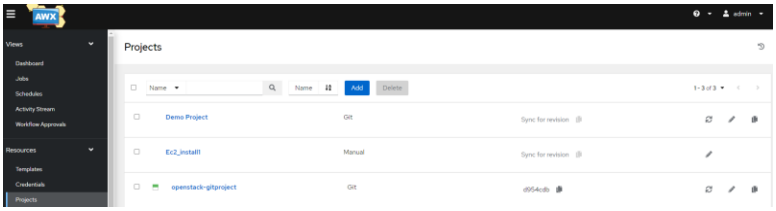


Git Code commit:

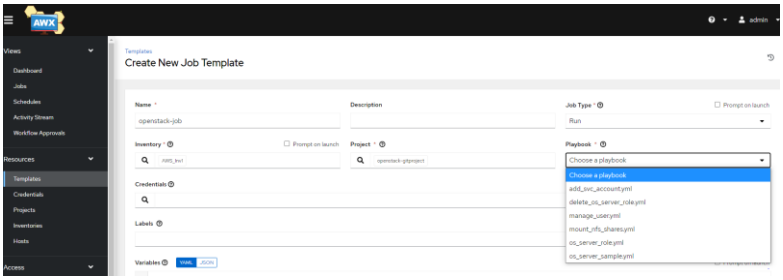


1) Credential
2) Project





3) Template



Git hub personal acces token:

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