Project Report NSSA-713

Automating IT infrastructure using Ansible and AWX

Group members:
Bashair Algarni - Zaki Bawade- Askar S Altalidi

Rochester Institute of Technology

2021

INTRODUCTION:

Automation in any infrastructure is when the organizations implement any process that could reduce human interaction with IT infrastructure by building and running some scripts and functions that allow the software to do specific jobs. These scripts can be repeatable and run by any other apps or by running some commands [1]. So, these tools will control the organization's infrastructure elements such as server, network, and storage. Thus, implementing IT automation would improve the organization's IT operations and services [1].

IT automation is the main reason behind efficiency and digital transformation, especially with the IT organizations growth. Reduced security attacks [1], increased employee productivity, and agility are significant benefits of IT automation. Infrastructure automation has many benefits, such as provisioning, cost reduction, capacity planning, and VM Sprawl [1].

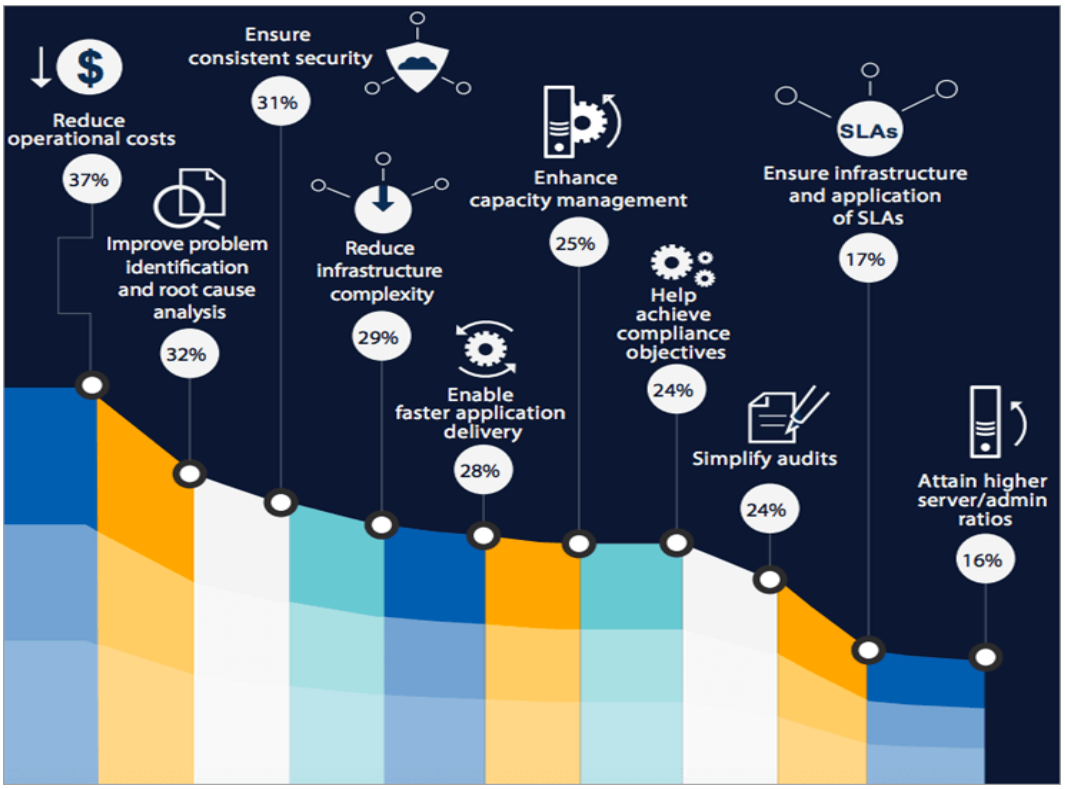


Fig 1: Business values delivered by IT Automation [3]

An infrastructure automation framework delivers the IT repeatability and predictability to manage IT workload [1]. This process helps the organization meet its customer's needs and service level agreements (SLAs) and reduces the complexity by focusing on the business value more than infrastructure management issues [1]. IT automation offers multi-cloud consistency across many cloud services; it also delivers infrastructure (IaC) as code. (IaC) enables developers to create virtual machine templates by using Git, YAML, and JSON [1]. We can automate many processes such as Self-Service Cloud, Kubernetes Multi-Cloud, Network, and DevOps for Infrastructure [1].

Ansible is a straightforward IT automation engine that automates cloud provisioning, configuration management, OS deployments, intra-service orchestration, and various other IT tasks. One of the most common complaints from Ansible users is the lack of GUI since it is entirely CLI-based. Ansible Tower, formerly known as the AWX project, is the solution to this issue. It is a robust web-based Ansible UI that includes the most significant Ansible features, especially those that function best as graphical rather than text-based output, such as real-time node monitoring. AWX is an open-source project with GUI tools of Ansible automation that run on Docker using Docker Python module [2]. Ansible AWX tools built by Red Hat OS allow the user to have all the features of Ansible Tower [2]. In this project, we are using AWX, the free and open-source version of Ansible Tower.

Though Ansible is an effective configuration management tool, it can be overwhelming to IT administrators unfamiliar with command-line tools. AWX attempts to solve this issue by including a graphical user interface while simultaneously expanding Ansible features with new management capabilities. Ansible AWX has many features, such as access management through the organization, users, and roles [2]. AWX has many other features such as job scheduling, user activity tracking, centralized logs, and inventory management [2].

This proposed project will show the features of using AWX and Ansible in automating IT infrastructure. In addition, we will make use of the AWX features discussed earlier to run a successful job. Here we have used Lab 3 infrastructure and automated it using Ansible AWX.

LITERATURE REVIEW:

AWX is considered a new tool, and these days, there are no studies on it yet. However, AWX is very useful in many cases. These are some cases where users can utilize the AWX.

First case: Using AWX with Gitlab

"AWX provides a provision of holding multiple playbooks by creating a project and storing playbooks with the help of Source Code Management tools (SCM) such as GitLab and GitHub. The projects are under the directory */var/lib/awx/projects* on the Ansible AWX server. Ansible AWX and SCM tools make lives easier by building out CI/CD workflows, and this continuous integration pipeline helps secure the master branch" [5]. By creating a new feature branch and performing the appropriate tests, and then merging it to the master, this methodology can be achieved.

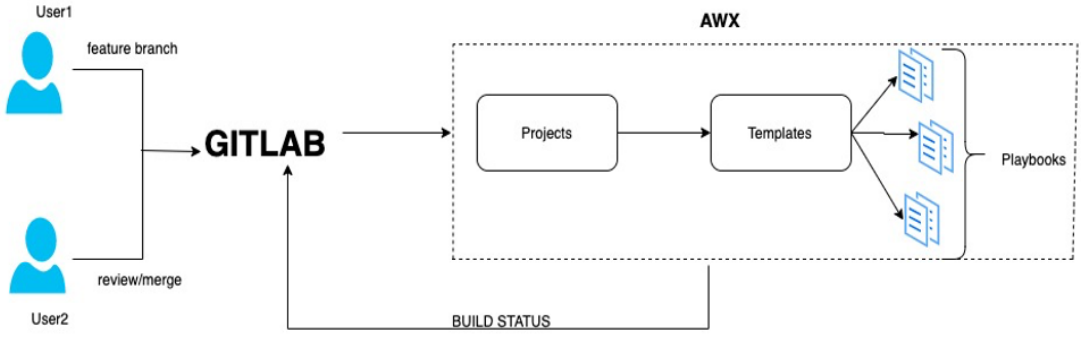


Fig2: Using AWX with Gitlab[5]

Second case: Using AWX and Spacewalk to perform Linux patching

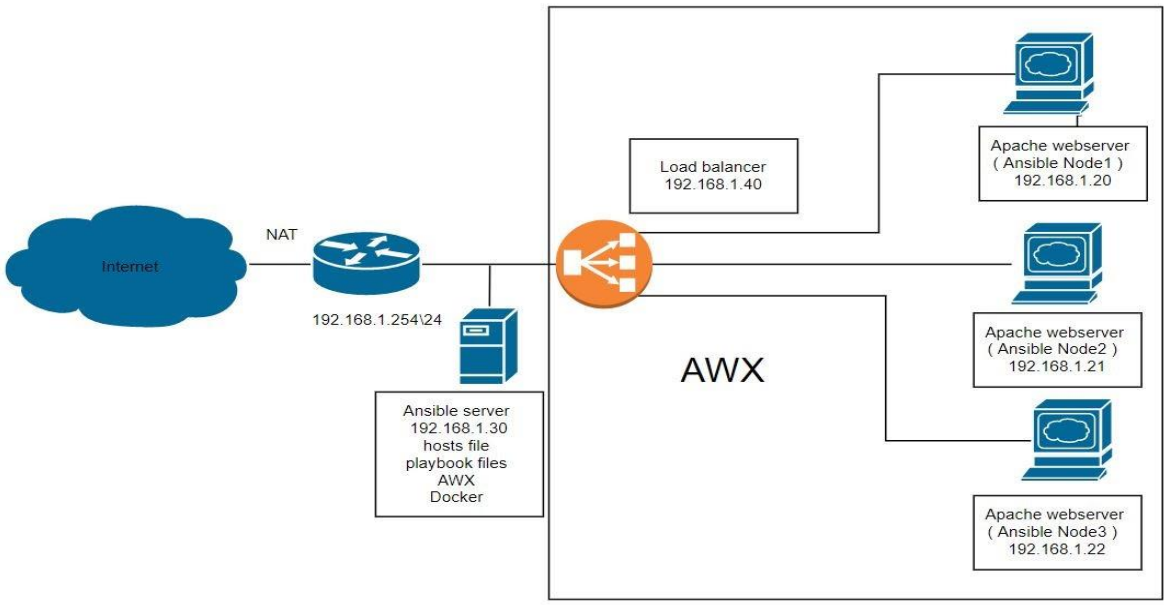
One of the crucial tasks in IT organizations and business offices is patching servers, automating maintaining Linux servers, and performing security patches to provide the best services to their customers. "Spacewalk helps find out the security loopholes left by outdated software or poorly written applications by checking any package update" [5]. Doing tasks such as maintaining inventory and ensuring that the software is running and updated will save a lot of time and effort.

Therefore, maintaining, installing, and updating software on the Linux/Windows systems can be done using an open-source solution like Spacewalk. "By connecting the Spacewalk plugin with the AWX, the playbooks can be scheduled according to the hosts which have been registered to the Spacewalk server for applying security patches without any human intervention"[5].

Third case: Performing API with Ansible AWX

To connect, generate outcomes, and obtain template information, REST API calls can be performed efficiently using Ansible AWX. Users can successfully run a job template and get the results by remotely connecting to the AWX REST API anywhere. "Ansible AWX provides an open-source framework for handling automation of Ansible applications, provisioning and managing servers with the help of web UI and RESTFUL API" [5]. Creating, destroying, and replacing servers at any time have become necessary to do without causing service disruptions.

METHODOLOGY:



An overview of the proposed project architecture

In this proposed project, the Architecture of lab three is used, and this project can be completed by implementing three phases:

Phase 1: Installation and configuration Ansible, AWX, and Docker on centos 8

- Installing Ansible and Docker dependencies and performing prerequisites.
  - Install EPEL on centos 8:

Command: # dnf install epel-release -y

- Install additional packages:

Command: # dnf install git gcc gcc-c++ nodejs gettext device-mapper-persistent-data lvm2 bzip2 python3-pip

```
[root@awx-ansible ~]# dnf install git gcc gcc-c++ nodejs gettext device-mapper-persistent-data lvm2 bzip2
python3-pip

Package gettext-0.19.8.1-14.el8.x86_64 is already installed.
Package python3-pip-9.0.3-13.el8.noarch is already installed.
Dependencies resolved.
================================================================================================================
 Package                        Arch    Version                                          Repository  Size
================================================================================================================
Installing:
 gcc                            x86_64  8.2.1-3.5.el8                                    AppStream   23 M
 gcc-c++                        x86_64  8.2.1-3.5.el8                                    AppStream   12 M
 git                            x86_64  2.18.1-3.el8                                     AppStream  186 k
 nodejs                         x86_64  1:10.16.3-2.module_el8.0.0+186+542b25fc          AppStream  9.0 M
 bzip2                          x86_64  1.0.6-26.el8                                     BaseOS      60 k
 device-mapper-persistent-data  x86_64  0.7.6-1.el8                                      BaseOS     456 k
 lvm2                           x86_64  8:2.03.02-6.el8                                  BaseOS     1.5 M
Installing dependencies:
 cpp                            x86_64  8.2.1-3.5.el8                                    AppStream   10 M
 git-core                       x86_64  2.18.1-3.el8                                     AppStream  4.1 M
 git-core-doc                   noarch  2.18.1-3.el8                                     AppStream  2.3 M
```

- **Installing Ansible**

  Commands: # pip3 install ansible --user
  # subscription-manager repos --enable ansible-2.8-for-rhel-8-x86_64-rpms
  # dnf -y install ansible
  # ansible --version
- To test that Ansible is working: $ sudo systemctl status sshd
- **Installing Docker**
  - Install Docker CE
    - Install: # dnf install docker-ce-3:18.09.1-3.el7
    - Get version: # docker –version
    - start and enable Docker: # systemctl start Docker
      # systemctl enable Docker.service
  - Install Docker – Compose
    # pip3 install docker-compose

```
[root@awx ~]#
[root@awx ~]# pip3 install docker-compose
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting docker-compose
  Downloading https://files.pythonhosted.org/packages/2e/93/b8fb6532487fcc40f5c607ac428a609e7f74bfb26a1c3c980a253c6
e5a14/docker_compose-1.25.0-py2.py3-none-any.whl (137kB)
    100% |████████████████████████████████| 143kB 4.3MB/s
Collecting jsonschema<4,>=2.5.1 (from docker-compose)
  Downloading https://files.pythonhosted.org/packages/c5/8f/51e89ce52a085483359217bc72cdbf6e75ee595d5b1d4b5ade40c7e
018b8/jsonschema-3.2.0-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 9.4MB/s
Collecting cached-property<2,>=1.2.0 (from docker-compose)
  Downloading https://files.pythonhosted.org/packages/3b/86/85c1be2e8db9e13ef9a350aecd6dea292bd612fa288c2f40d035bb7
50ded/cached_property-1.5.1-py2.py3-none-any.whl
```

- **Installing AWX from GitHub**
  - clone the Git repo: # git clone https://github.com/ansible/awx.git

```
[root@awx-ansible ~]#
[root@awx-ansible ~]# git clone https://github.com/ansible/awx.git
Cloning into 'awx'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 219229 (delta 0), reused 0 (delta 0), pack-reused 219228
Receiving objects: 100% (219229/219229), 216.71 MiB | 35.20 MiB/s, done.
Resolving deltas: 100% (169392/169392), done.
[root@awx-ansible ~]#
[root@awx-ansible ~]#
```

- Navigate to the awx/installer directory and locate the inventory file.
- Adjustments on the file :

```
# Common Docker parameters
awx_task_hostname=awx
awx_web_hostname=awxweb
# Local directory that is mounted in the awx_postgres docker container to place the db in
postgres_data_dir="/root/.awx/pgdocker"
host_port=80
host_port_ssl=443
#ssl_certificate=
# Optional key file
#ssl_certificate_key=
docker_compose_dir="/root/.awx/awxcompose"
```

```
# Alternate DNS servers
awx_alternate_dns_servers="8.8.8.8,8.8.4.4"
```

- Configure Admin and password :

```
# This will create or update a default admin (superuser) account in AWX, if not provided
# then these default values are used
admin_user=admin
admin_password=student
```

- generate a cryptographic key for encryption of the inventory file: # openssl rand -base64 30
- Copy the secret key and attached it to the secret_key entry in the inventory file:
- secret_key:

```
secret_key=D2VXKMe3y/OwoIkf9Up+Xkw7WdoYvJci4qTwxAFi
```

- confirm and print out the changes made:#grep -v '^#' inventory | grep -v '^$'

```
[root@awx-ansible installer]#
[root@awx-ansible installer]# grep -v '^#' inventory | grep -v '^$'
localhost ansible_connection=local ansible_python_interpreter="/usr/bin/env python"
[all:vars]
dockerhub_base=ansible
awx_task_hostname=awx
awx_web_hostname=awxweb
postgres_data_dir="~/.awx/pgdocker"
host_port=80
host_port_ssl=443
docker_compose_dir="~/.awx/awxcompose"
pg_username=awx
pg_password=awxpass
pg_database=awx
pg_port=5432
 pg_admin_password=postgrespass@123
rabbitmq_password=awxpass
rabbitmq_erlang_cookie=cookiemonster
admin_user=admin
admin_password=Linuxtechi@123
create_preload_data=True
secret_key=SGYsSWciI5yRDQeZuEm5wW98pQeJMG+ACABPsGfC
[root@awx-ansible installer]#
```

- Install AWX by run the Ansible command: #ansible-playbook -i inventory install.yml

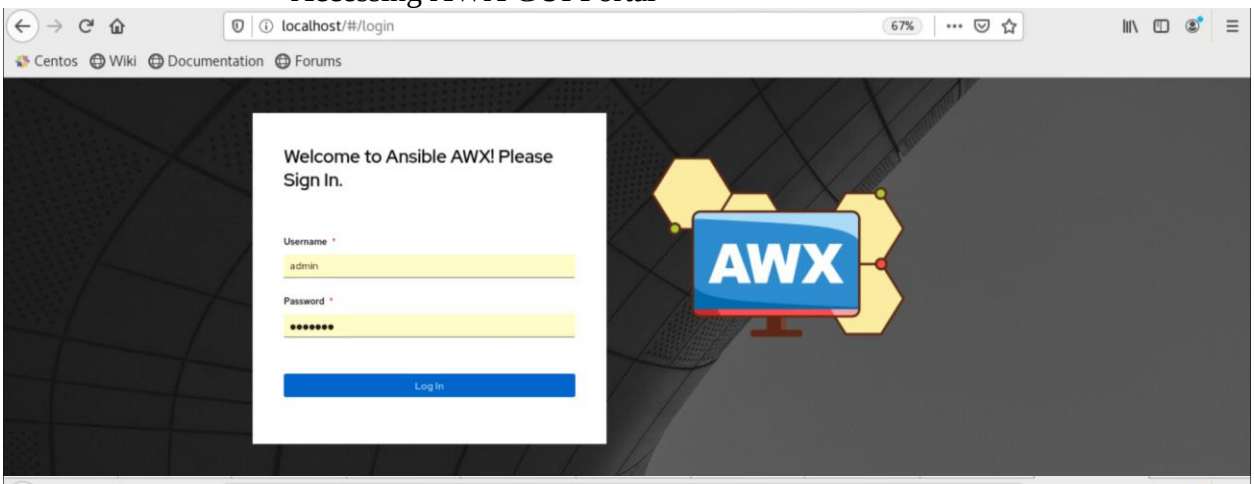
- **Run AWX on Docker container**
  - check the containers which are launched via docker-compose: # docker ps

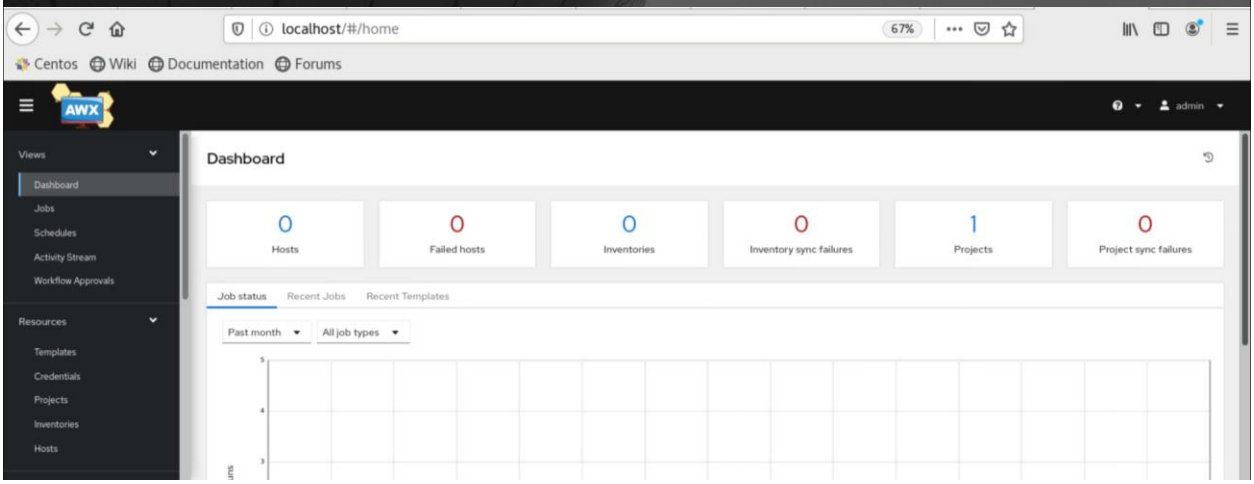
```
[root@localhost installer]# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED       STATUS       PORTS                                       NAMES
3a815475cbec   ansible/awx:17.1.0   "/usr/bin/tini -- /u…"   12 days ago   Up 12 days   8052/tcp                                    awx_task
60f70d0cc7ef   ansible/awx:17.1.0   "/usr/bin/tini -- /b…"   12 days ago   Up 12 days   0.0.0.0:80->8052/tcp, :::80->8052/tcp       awx_web
623e61d723fe   redis                "docker-entrypoint.s…"   12 days ago   Up 12 days   6379/tcp                                    awx_redis
f7a11f1dcb31   postgres:12          "docker-entrypoint.s…"   12 days ago   Up 12 days   5432/tcp                                    awx_postgres
[root@localhost installer]#
```

  - Allow the http and https ports :

```
# firewall-cmd --zone=public --add-masquerade --permanent
# firewall-cmd --permanent --add-service=http
# firewall-cmd --permanent --add-service=https
# firewall-cmd –reload
```

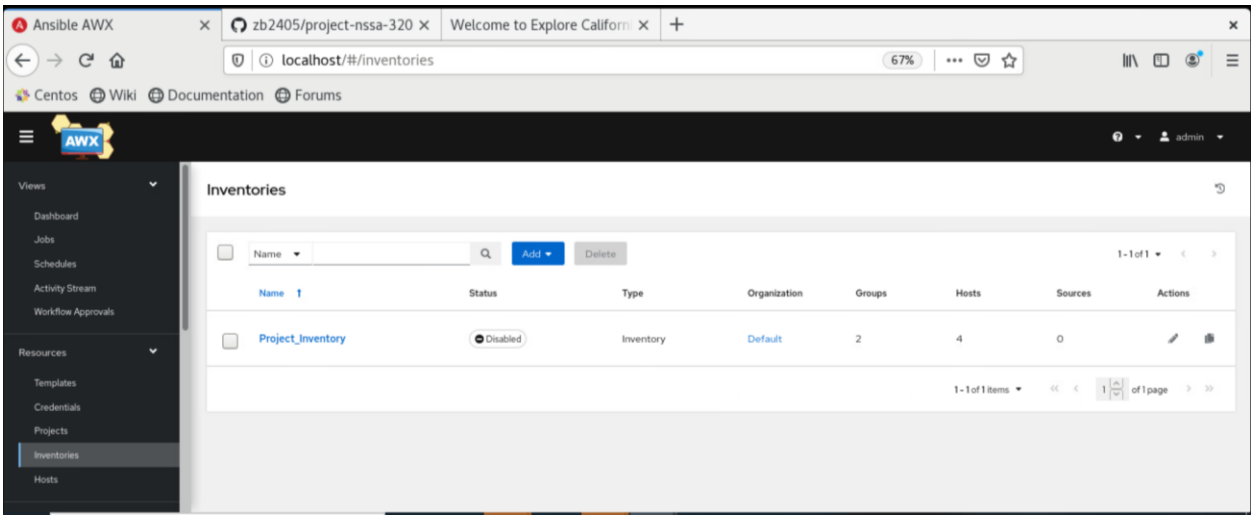
  - Accessing AWX GUI Portal



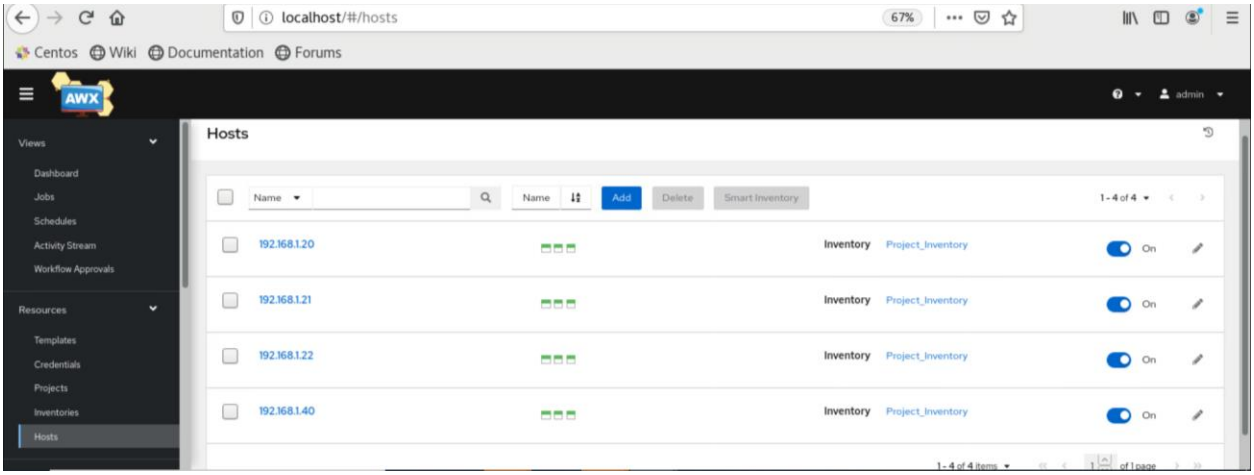


Phase 2: **Working on AWX**

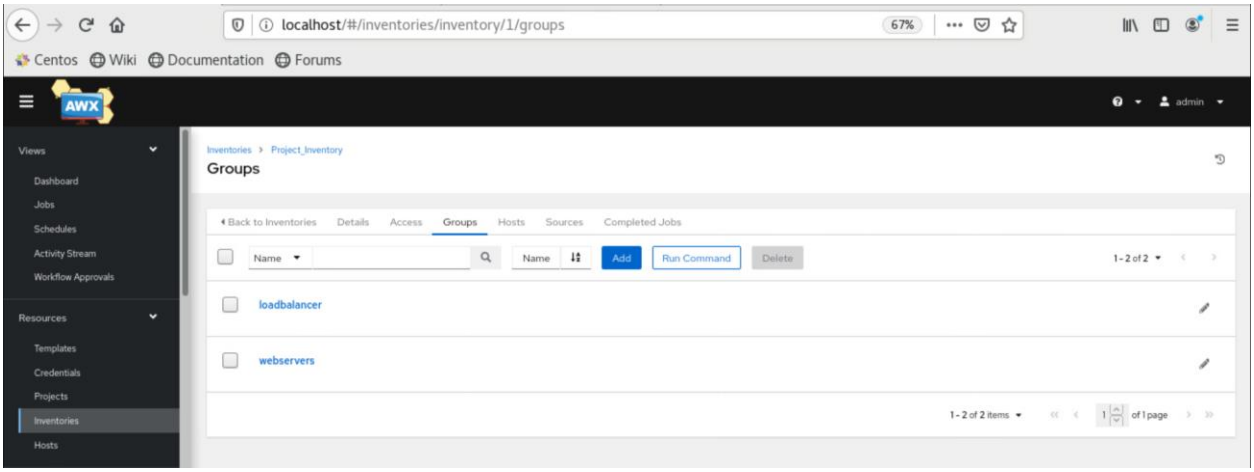
- **Add inventory**: A task to add hosts to the server.
  - When inventory is created, all the remote hosts will be under it. In this project, the inventory is divided into two groups: the load balancer group and the webservers group.
  - steps to add inventory:
  - On the left pane, click INVENTORIES ---> ADD INVENTORY. On this page, add a name for the inventory ---> click SAVE



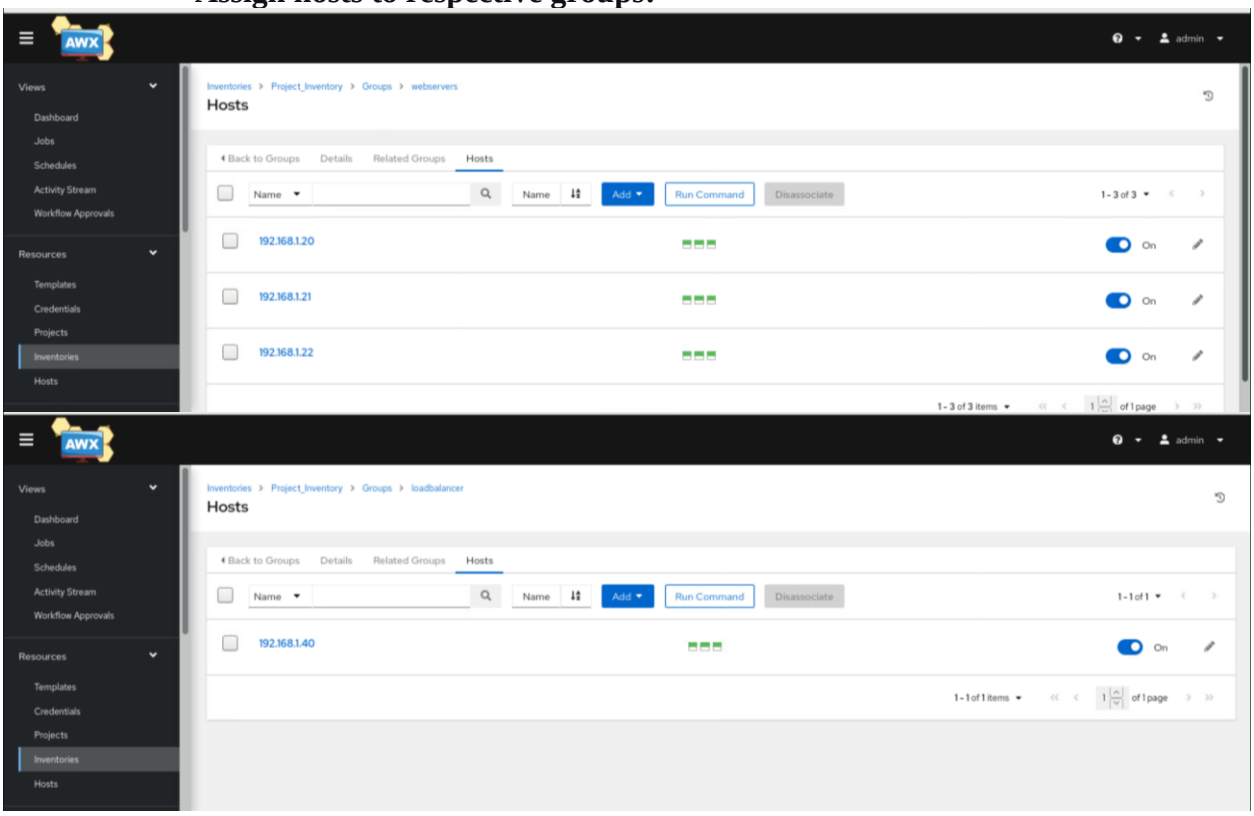
- **Add hosts to the inventory**: The host name can be a working IP address or a URL and users can any number of hosts to an inventory.
  - In the same inventory page previously created, click the HOSTS tab--> click ADD HOST-->Enter the host name of the webserver --> SAVE



- **Add groups in the inventory** (load balancer and webservers)
  - A group in AWX might represent a particular environment, in this case, a server type (web servers) and load balancer

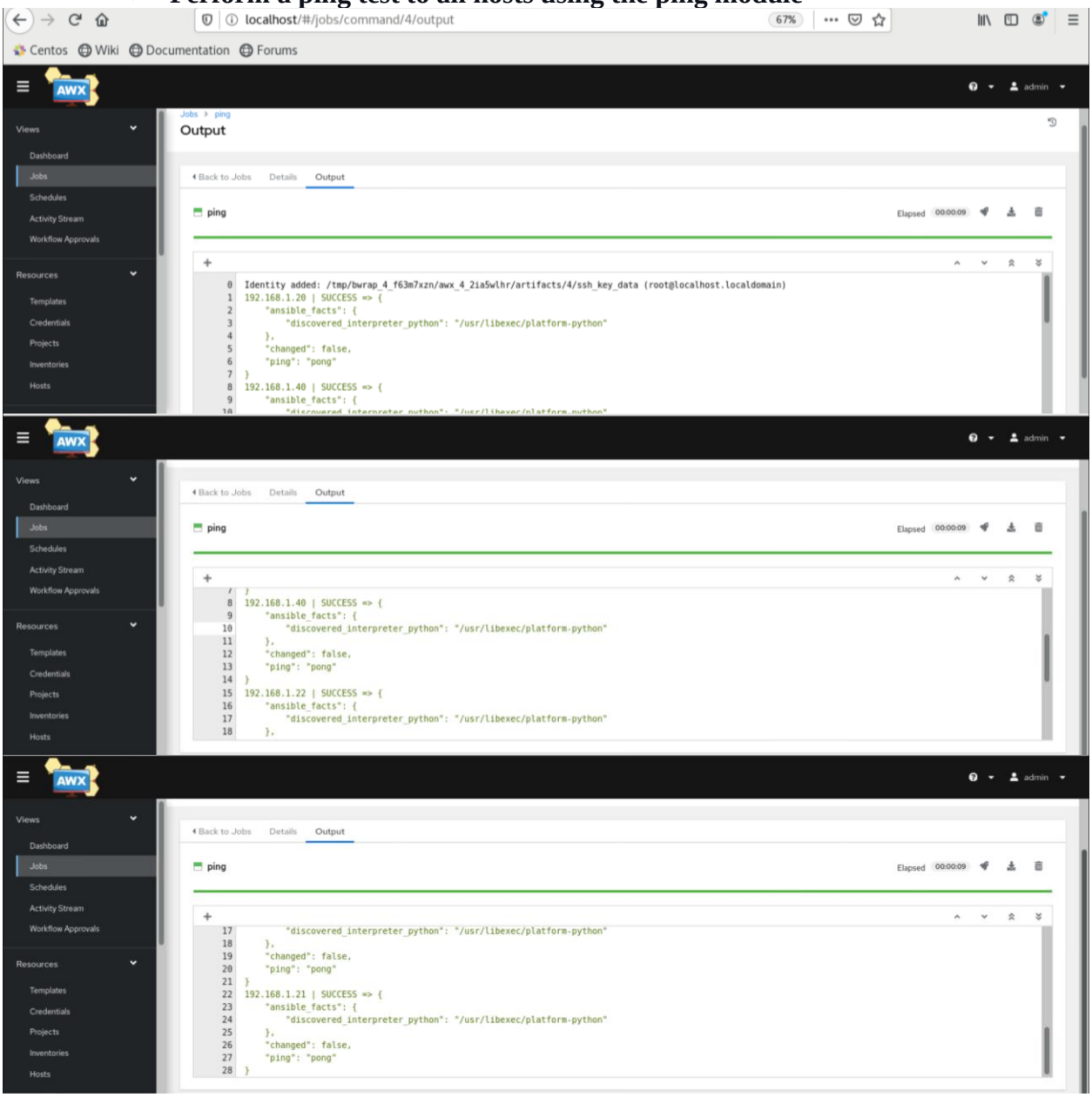


- **Assign hosts to respective groups:**

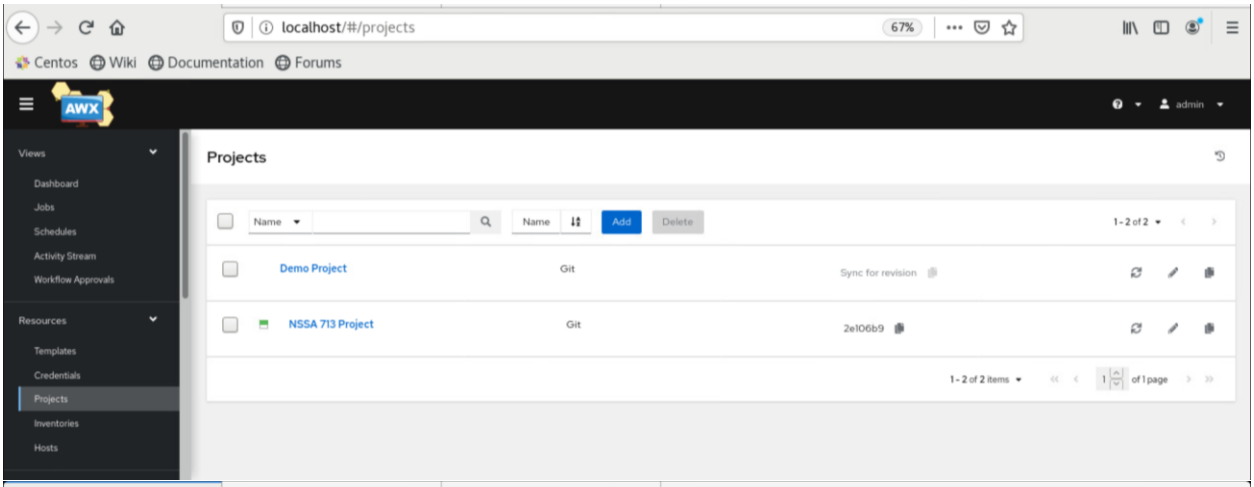


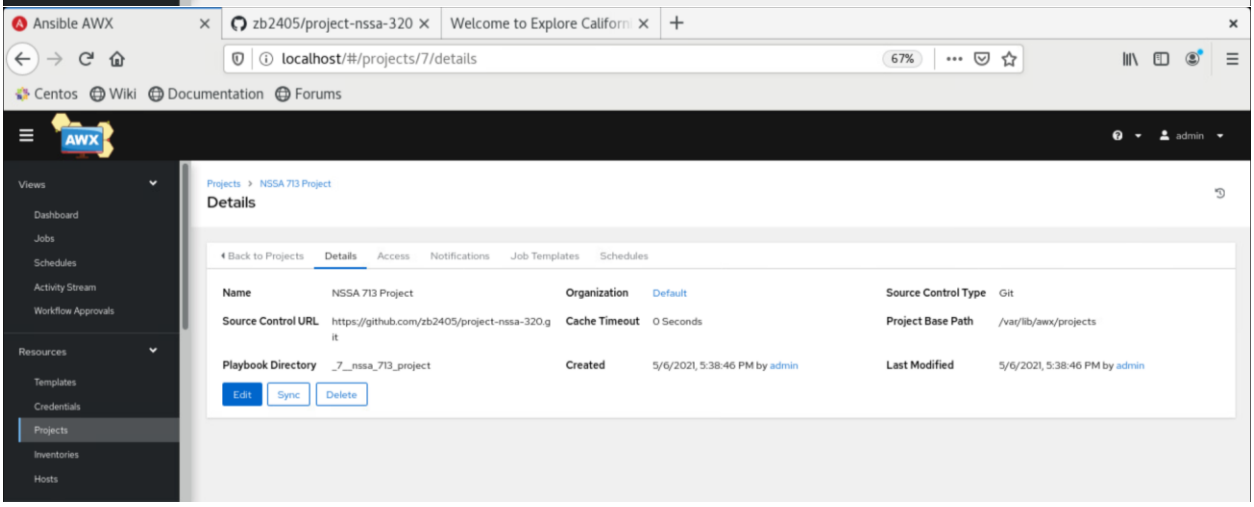
- **Add the (ssh) credentials:**
  - credentials in AWX are stored separately, which is very efficient because users can use a single credential to any number of hosts.
  - On the left pane, click CREDENTIALS --> ADD --> enter a name -->Select a credential type (Machine – similar to SSH login) -->Enter the user name and password of the remote machine→Click SAVE

- **Perform a ping test to all hosts using the ping module**



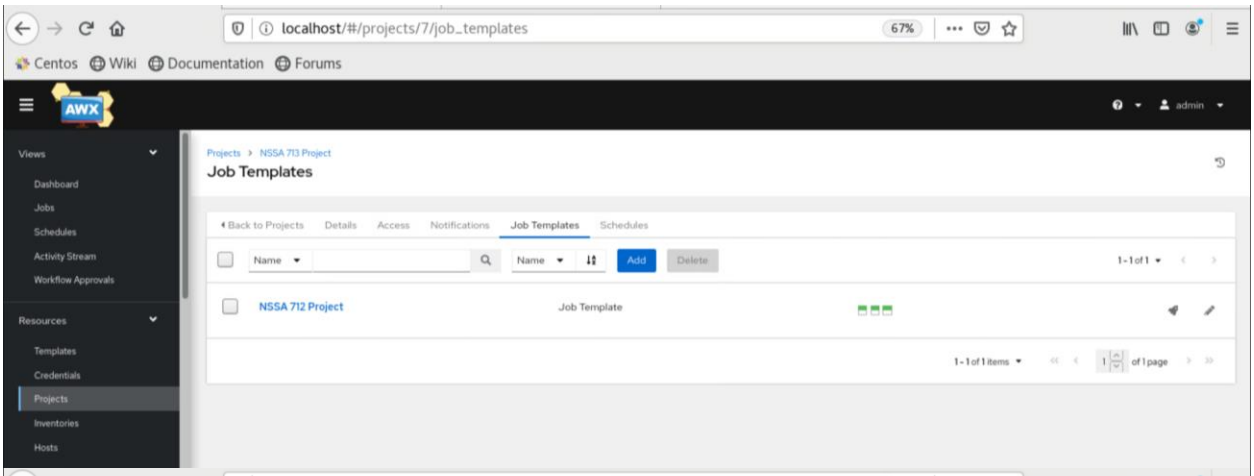
- **Add project:**
  - When users add a new project, the base path to the repository is given. A base path can be the link to your GitHub repository or the directory holding playbooks. In our case, we selected Git as our SCM Type because we will be using the GitHub repository.
  - On the left pane, click PROJECTS -->Click ADD and enter a name-->
  - Select an SCM type-->enter SCM URL/Playbook directory. We will add a public GitHub repository: (https://github.com/zb2405/project-nssa-320)-->Click SAVE to save changes.

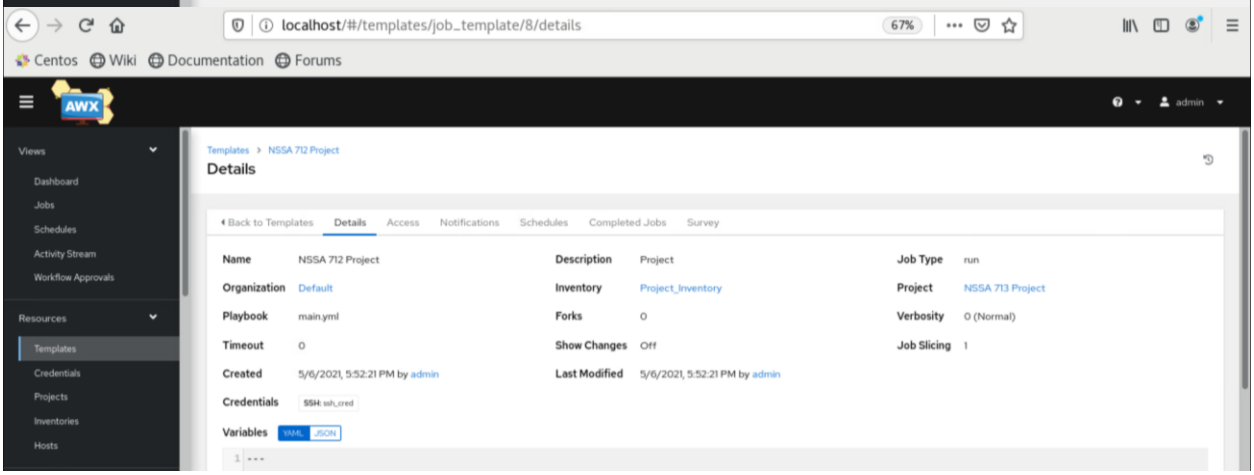




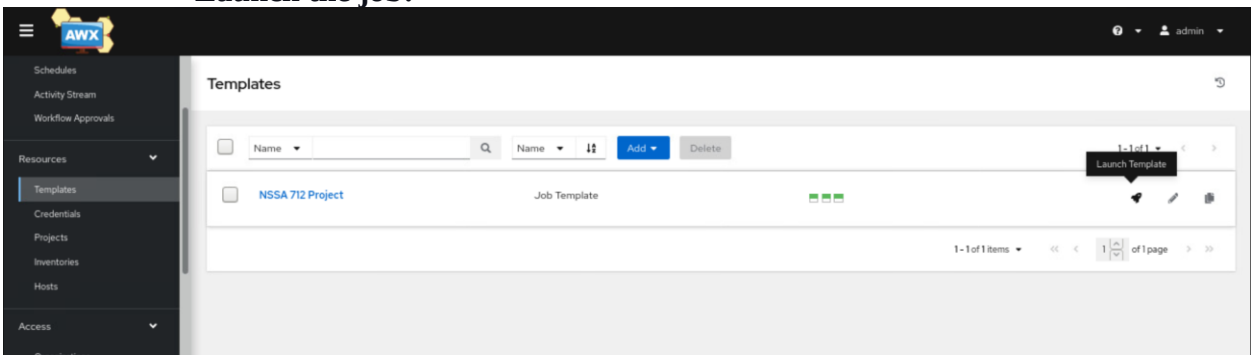
Phase 3: **Run the project**

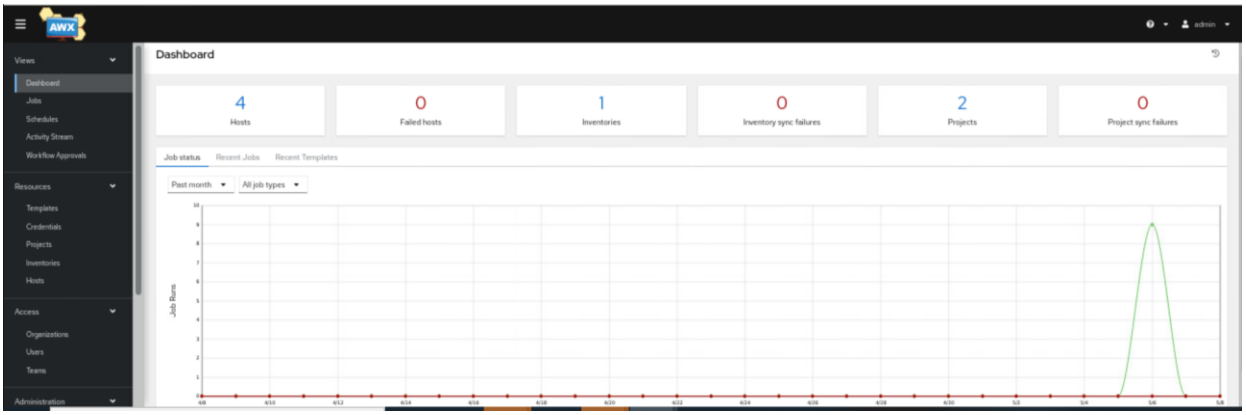
- **Create a job template:**
  - The specific playbook is selected to be executed from the project we added.
  - On the left pane, click TEMPLATES-->Enter a name-->Select the job type as Run or Check -->Select the inventory and the project in which playbooks are present-->Select playbook from the PLAYBOOK drop-down list-->Add credential for the particular inventory of hosts-->Click SAVE to save changes.





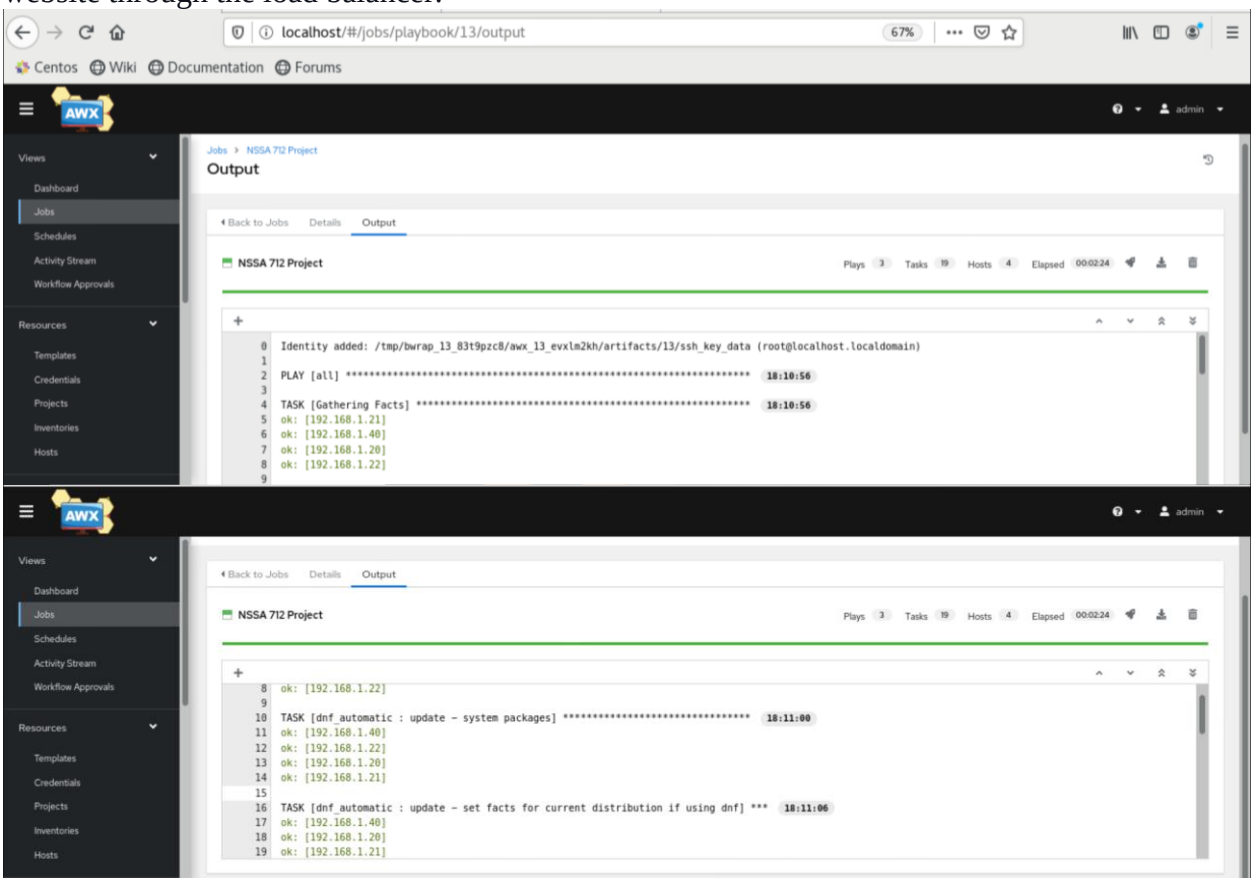
- **Launch the job:**

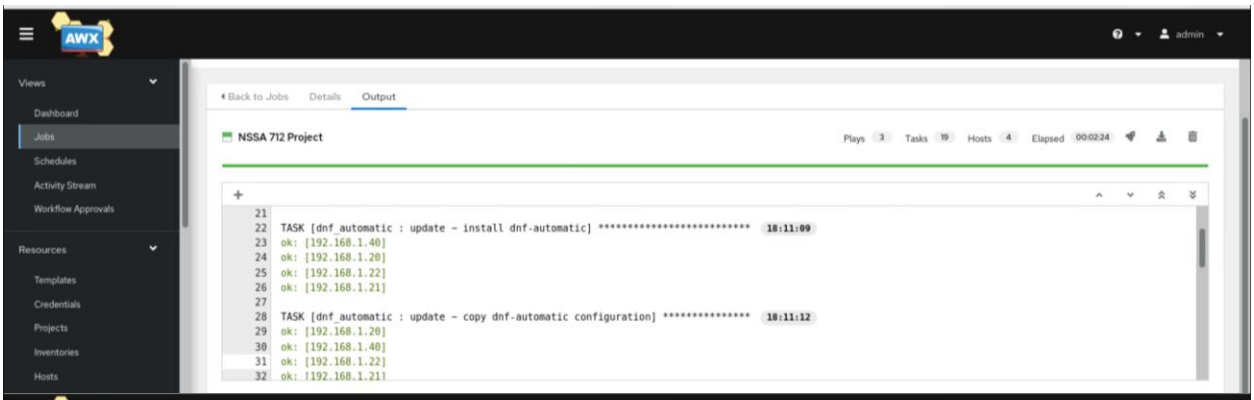


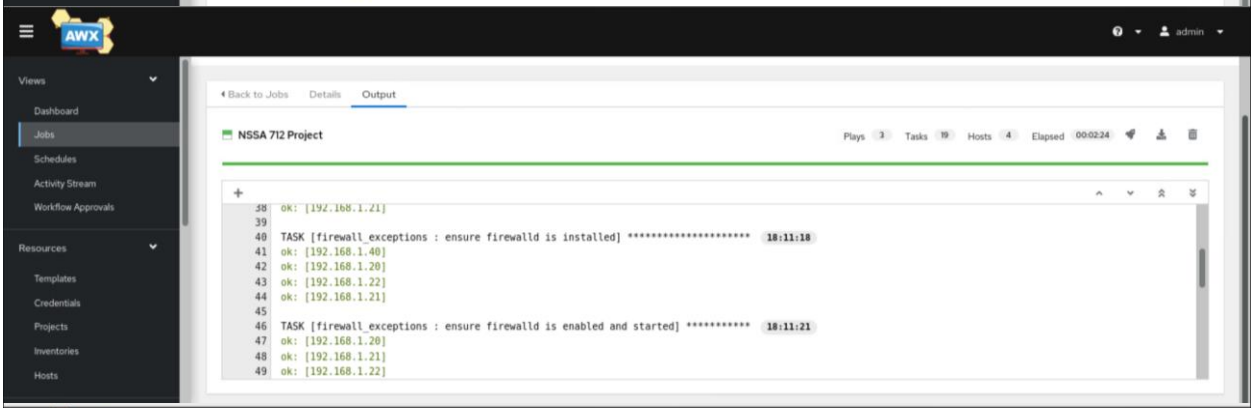


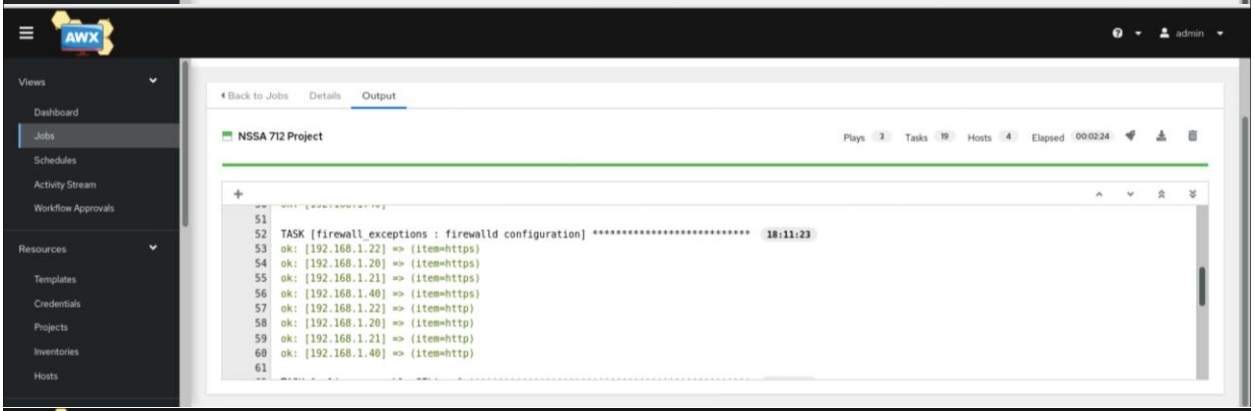
RESULTS:

As we can see below, our job ran successfully across all the web servers, and we can access the website through the load balancer.

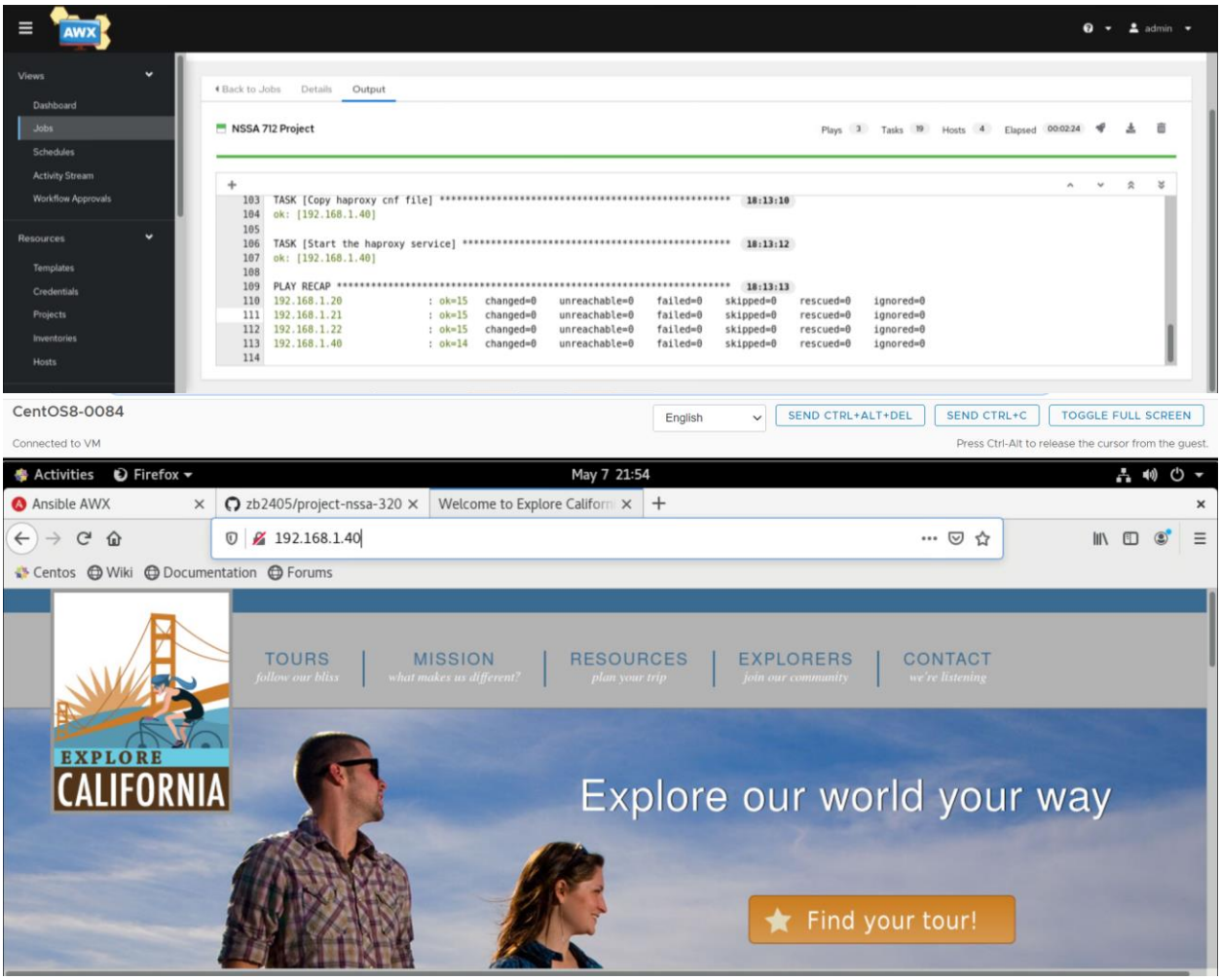



AWX
admin
Views
Dashboard
Jobs
Schedules
Activity Stream
Workflow Approvals
Resources
Templates
Credentials
Projects
Inventories
Hosts
Back to Jobs
Details
Output
NSSA 712 Project
Plays 3 Tasks 19 Hosts 4 Elapsed 00:02:24
21
22 TASK [dnf_automatic : update - install dnf-automatic] ********** 18:11:09
23 ok: [192.168.1.40]
24 ok: [192.168.1.20]
25 ok: [192.168.1.22]
26 ok: [192.168.1.21]
27
28 TASK [dnf_automatic : update - copy dnf-automatic configuration] ********** 18:11:12
29 ok: [192.168.1.20]
30 ok: [192.168.1.40]
31 ok: [192.168.1.22]
32 ok: [192.168.1.21]


AWX
admin
Views
Dashboard
Jobs
Schedules
Activity Stream
Workflow Approvals
Resources
Templates
Credentials
Projects
Inventories
Hosts
Back to Jobs
Details
Output
NSSA 712 Project
Plays 3 Tasks 19 Hosts 4 Elapsed 00:02:24
ok: [192.168.1.21]
39
40 TASK [firewall_exceptions : ensure firewalld is installed] ********** 18:11:18
41 ok: [192.168.1.40]
42 ok: [192.168.1.20]
43 ok: [192.168.1.22]
44 ok: [192.168.1.21]
45
46 TASK [firewall_exceptions : ensure firewalld is enabled and started] ********** 18:11:21
47 ok: [192.168.1.20]
48 ok: [192.168.1.21]
49 ok: [192.168.1.22]


AWX
admin
Views
Dashboard
Jobs
Schedules
Activity Stream
Workflow Approvals
Resources
Templates
Credentials
Projects
Inventories
Hosts
Back to Jobs
Details
Output
NSSA 712 Project
Plays 3 Tasks 19 Hosts 4 Elapsed 00:02:24
51
52 TASK [firewall_exceptions : firewalld configuration] ********** 18:11:23
53 ok: [192.168.1.22] => (item=https)
54 ok: [192.168.1.20] => (item=https)
55 ok: [192.168.1.21] => (item=https)
56 ok: [192.168.1.40] => (item=https)
57 ok: [192.168.1.22] => (item=http)
58 ok: [192.168.1.20] => (item=http)
59 ok: [192.168.1.21] => (item=http)
60 ok: [192.168.1.40] => (item=http)
61


AWX
admin
Views
Dashboard
Jobs
Schedules
Activity Stream
Workflow Approvals
Resources
Templates
Credentials
Projects
Inventories
Hosts
Back to Jobs
Details
Output
NSSA 712 Project
Plays 3 Tasks 19 Hosts 4 Elapsed 00:02:24
68 PLAY [webservers] ********** 18:11:31
69
70 TASK [Gathering Facts] ********** 18:11:31
71 ok: [192.168.1.20]
72 ok: [192.168.1.22]
73 ok: [192.168.1.21]
74
75 TASK [apache : Apache latest version installation] ********** 18:11:33
76 ok: [192.168.1.20]
77 ok: [192.168.1.22]
78 ok: [192.168.1.21]



FUTURE WORK:

Although we used Ansible AWX in this project, it is suggested to use RedHat Ansible Tower for future work. Ansible Tower was built on the open-source project AWX, and it has better documentation, support, and improved security, updates, and backup features.

REFERENCES:

[1] "What is Infrastructure Automation?: VMware Glossary," *VMware*, 05-May-2021. [Online]. Available: https://www.vmware.com/topics/glossary/content/infrastructure-automation.html. [Accessed: 08-May-2021].

[2] B. Lee, *Ansible Tower vs Ansible AWX for Automation*. [Online]. Available: https://4sysops.com/archives/ansible-tower-vs-ansible-awx-for-automation/#what-is-ansible-awx.

[3] https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/07/IT-automation.png

[4] "Use Cases of Ansible AWX," *ACTIVE IT Services*. [Online]. Available: https://it.activenetwork.com/blog/use-cases-of-ansible-awx. [Accessed: 08-May-2021].

[5] "Automating IT infrastructure using Ansible and AWX," *IBM Developer*. [Online]. Available: https://developer.ibm.com/articles/automation-using-ansible-awx-gui/#:~:text=Ansible%20is%20an%20open%20source,for%20all%20your%20automation%20tasks. [Accessed: 08-May-2021].

[6] "8. Create a new Inventory and add it to the organization," *8. Create a new Inventory and add it to the Organization - Ansible Tower Quick Setup Guide v3.8.3*. [Online]. Available: https://docs.ansible.com/ansible-tower/latest/html/quickstart/create_inventory.html. [Accessed: 08-May-2021].

[7] "How to Install and Setup Docker on Centos 8 {Quickstart}," *Knowledge Base by phoenixNAP*, 26-Apr-2021. [Online]. Available: https://phoenixnap.com/kb/how-to-install-docker-on-centos-8. [Accessed: 08-May-2021].

[8] "Comparing AWX and Red Hat Ansible Tower," *Red Hat - We make open source technologies for the enterprise*. [Online]. Available: https://www.redhat.com/en/resources/awx-and-ansible-tower-datasheet. [Accessed: 08-May-2021].

[9] C. Houseknecht, "5 Things You Can Do With AWX," *Red Hat Ansible*. [Online]. Available: https://www.ansible.com/blog/5-things-you-can-do-with-awx#:~:text=AWX%20adds%20a%20custom%20callback,of%20inventory%20successes%20and%20failures. [Accessed: 08-May-2021].

[10]https://www.doag.org/formes/pubfiles/10846538/2018-DO-Alexander_Hofstetter-Einfuehrung_in_Ansible_AWX_Project_der_quot_freie_quot_Ansible_Tower-Praesentation.pdf