

PROJECT REPORT

CI/CD Pipeline using GitHub, Jenkins, Ansible and Docker

By

Sahil Anande

Zaki Bawade

Table of Contents

I.	INTRODUCTION	3
II.	LITERATURE REVIEW	4
III.	METHODOLOGY	6
IV.	RESULTS	12
V.	FUTURE WORK	14
VI.	REFERENCES	15

1. Introduction

The target of the project is to build a continuous integration and continuous deployment (CI/CD) pipeline using DevOps tools such as Git, Jenkins, Ansible, DockerHub, and Docker. A CI/CD pipeline is a quick and effective way to automate the software delivery process right from building code, testing, and deployment to the end-user. Automated pipelines eliminate human errors, provides standardized development feedback loops and enables fast product developments.

CI/CD has been a key pillar for faster software delivery across various software development firms out there, this fast-paced software development powered by automation has been the key motivation for our CI/CD pipeline project.

The first stage of a CI/CD project starts with committing small chunks of code to a repository. GitHub is a web-based version control and collaborative platform where software developers working on the same part of a project can merge code multiple times in a day. GitHub is the most popular code repository platform and fortune 100 companies such as Netflix, Twitter, Adobe, IBM, Microsoft, Yelp, etc. have been using it over the years owing to its versatility and user-friendliness.

Once the code is committed to a git repository it needs to be build and tested. Jenkins is an open-source code building and testing platform which is written in Java. Jenkins is easily configurable and has a variety of plugins for building and deploying application code. Jenkins is a huge time and cost saver that keeps the errors in check and allows agile software delivery.

Once our code passes through the testing phase it needs to be deployed to an environment. Ansible is a popular and powerful tool that provision environments for app deployments. It is simple, free to use, flexible, agentless, efficient and hence it is the best deployment tool out there.

The IT world has switched to containers for the deployment of applications rather than the age-old virtual machines. The heart of the IT industry is Docker, a framework for easily deploying and managing the software in containers. Additionally, Docker provides ubiquity and auditability which makes managing applications a piece of cake.

A plethora of tools are available for the CI/CD pipeline process but owing to the popularity and the advantages of the above-mentioned tools (Git, Jenkins, Ansible, and Docker), we have decided that these tools would be ideal for our project.

Our CI/CD pipeline project involves committing the code to a version control repository such as GitHub. Once the code is committed the Jenkins CI server performs a “build and test” on the code. After a successful build and test the code is pushed to an environment where it can be deployed for the end user. Ansible is responsible to generate and manage this environment and create docker containers for deployment of our application/code. The docker hub is a container image sharing platform, Ansible pushes the created image into docker hub and at the same time initializes a docker container. The docker uses this image to create a container.

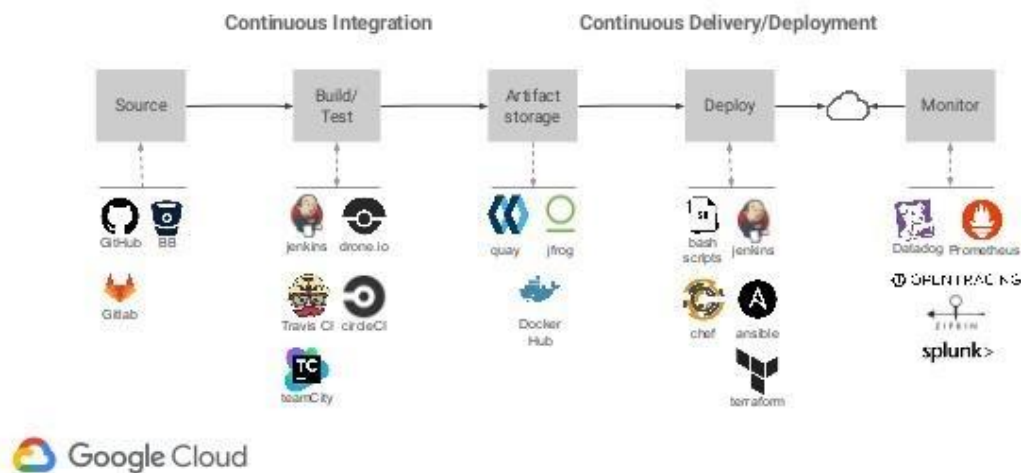
2. Literature Review

Although ours is a simple and zero investment project there are various projects out there that use different tools altogether. The following is the CI/CD pipeline architecture that we are implementing:



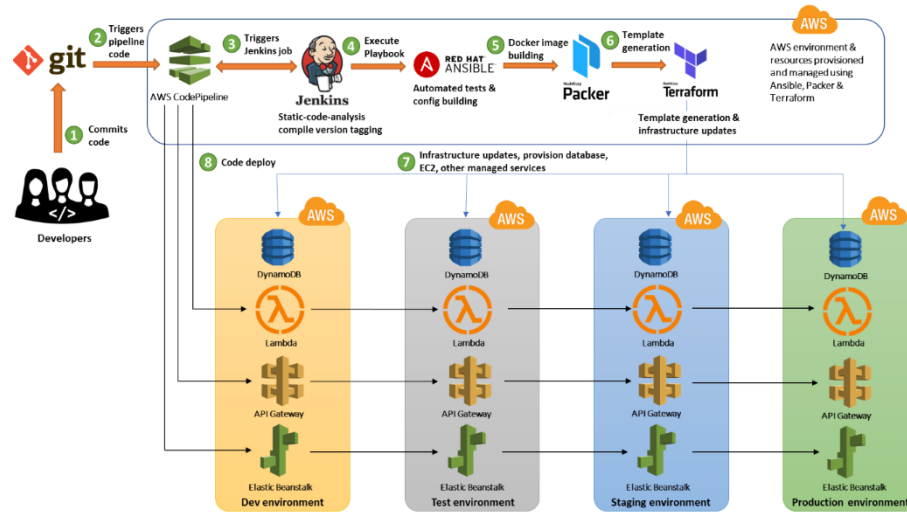
A CI/CD pipeline architecture of Google uses a different set of tools:

CI/CD has many ways ...



As seen above the Google Cloud CI/CD pipeline allows the use of Gitlab/Bitbucket as a version control tool. Jenkins can be substituted by platforms such as circleCI, teamCity, TravisCI, etc. Quay and jfrog are available as an alternative to Docker Hub. Similarly, terraform and bash scripts can replace ansible.

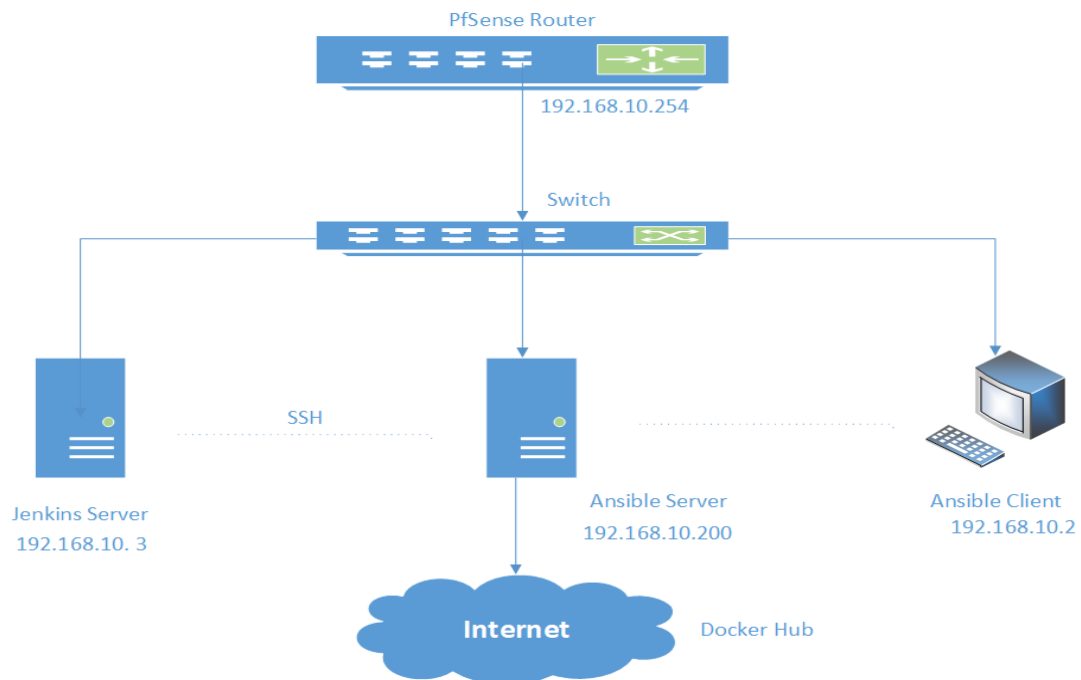
Likewise, when we take the CI/CD pipeline of AWS we can see that Packer is used as a replacement for Docker to create images for virtual machines and terraform is used as an infrastructure orchestration tool to deploy images in VMware.



Though the tools may be from different vendors the functionality of that tool at a CI/CD stage is not different. It is entirely possible to do this project using a different set of tools and still get the same result.

3. Methodology

1. Deploy 3 Virtual Machines (VMs) on RLES for Ansible Server node, Ansible Client node, and Jenkins Server node. Also, deploy a PfSense router to obtain private address space.



```
8) Shell
Enter an option:

FreeBSD/amd64 (pfSense.localdomain) (ttyv0)
VMware Virtual Machine - Netgate Device ID: ecf64a5d0741f9bdc1cf
*** Welcome to pfSense 2.4.4-RELEASE-p3 (amd64) on pfSense ***

WAN (wan)      -> em0      -> v4/DHCP4: 192.168.204.156/21
LAN (lan)      -> em1      -> v4: 192.168.10.254/24

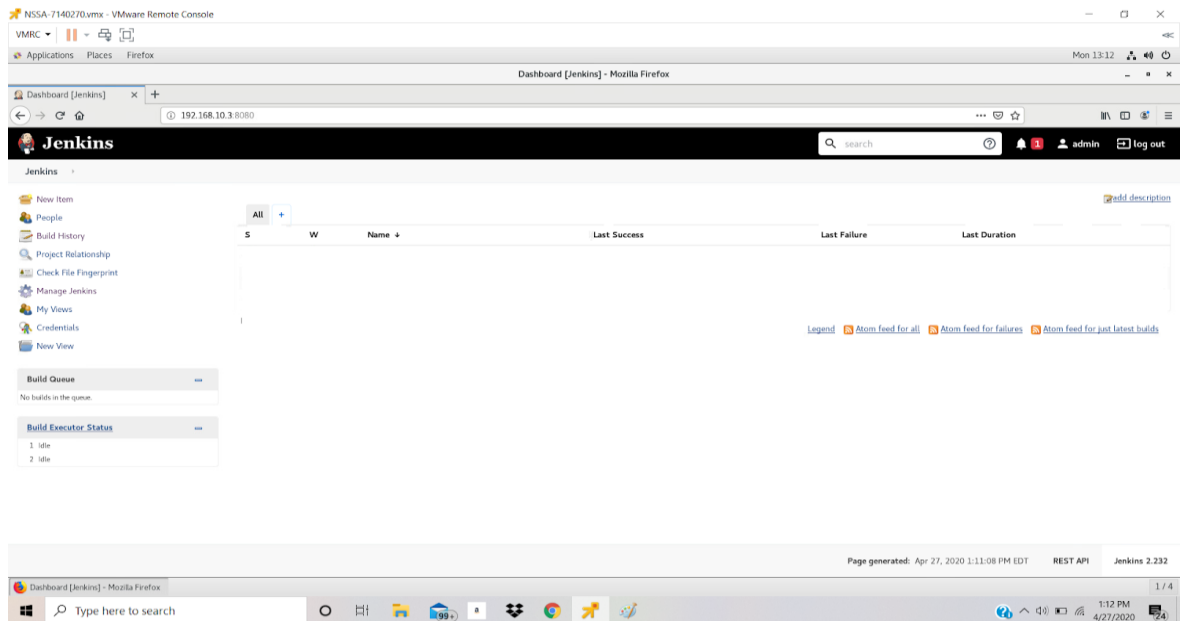
0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults    13) Update from console
5) Reboot system              14) Enable Secure Shell (sshd)
6) Halt system                15) Restore recent configuration
7) Ping host                  16) Restart PHP-FPM
8) Shell

Enter an option: █
```

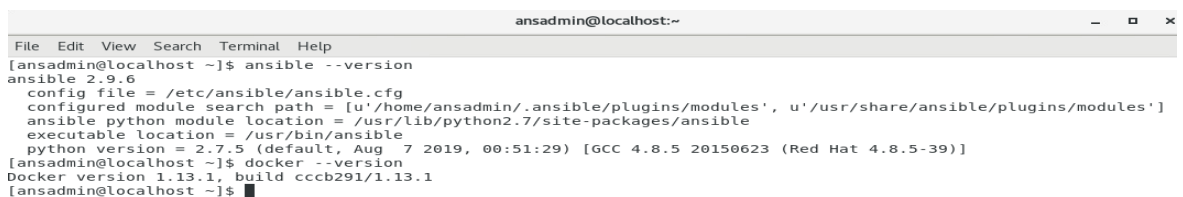
2. Git was installed, and a local repository was created for staging and committing the code.

```
MINGW64:/c:/Users/Sahil Anande
Sahil Anande@LAPTOP-HVH6R2NO MINGW64 ~
$ git --version
git version 2.26.0.windows.1
Sahil Anande@LAPTOP-HVH6R2NO MINGW64 ~
$
```

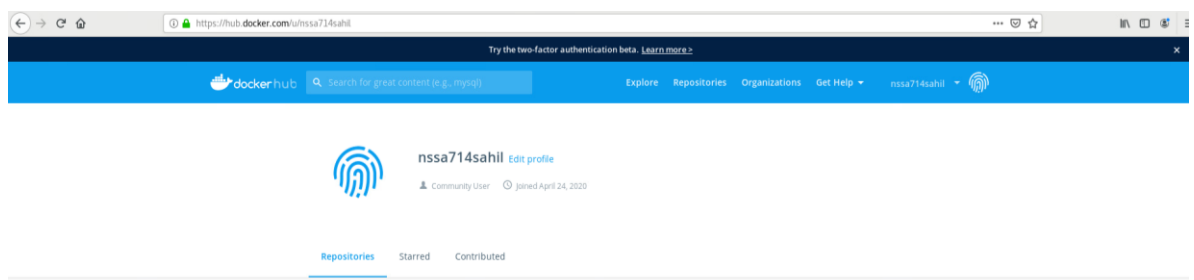
3. JAVA path was added in the Jenkins configuration for successful implementation when new job is built, and the Maven plugin was installed through the global configuration in the Jenkins Server.



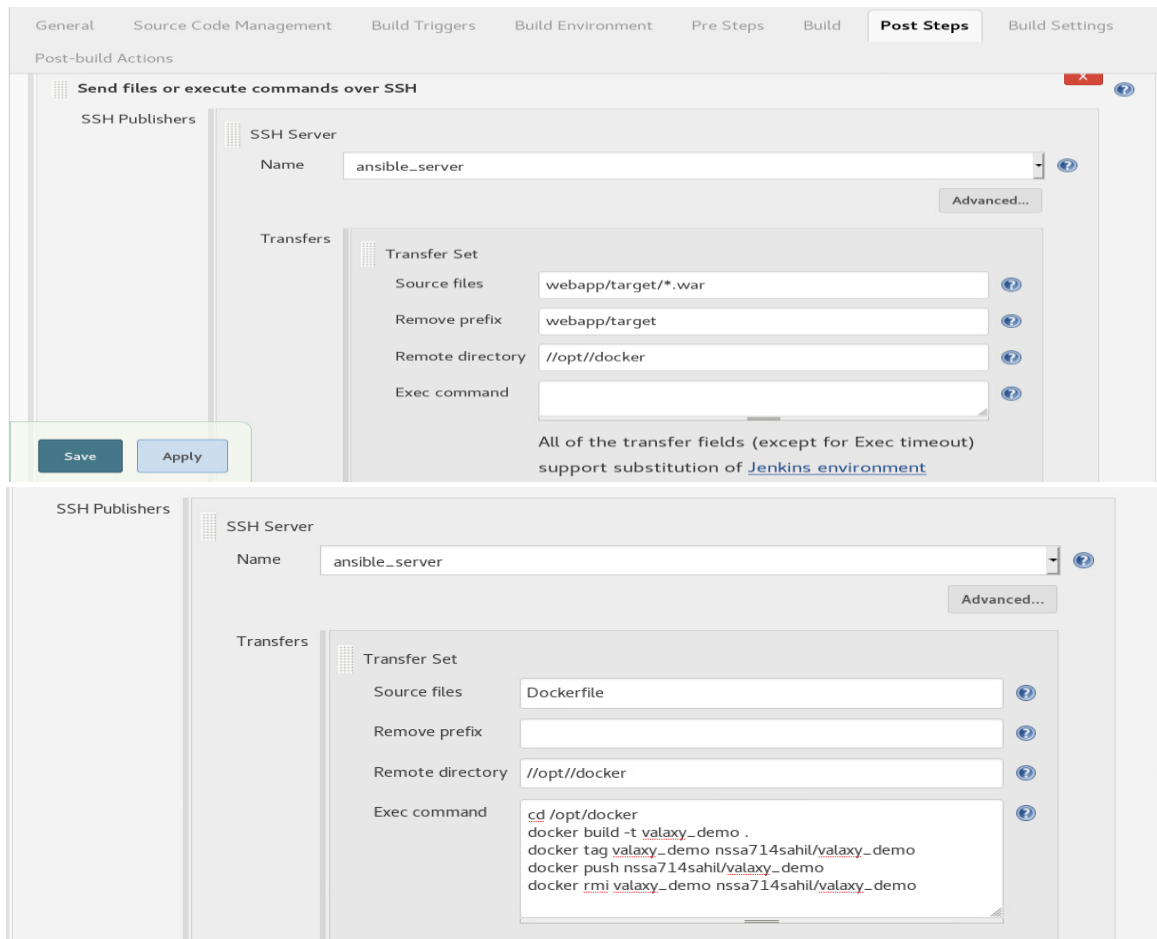
4. Installed Ansible server and Docker on the Ansible server node. Also, Docker was installed on the Ansible client node.



5. A docker hub account was created and with these credentials were used at the Ansible server and client to login into the docker



6. After successfully installing all the required tools, we transferred the .war and the Dockerfile from Github to the docker directory i.e. /opt/docker. We sent these files and executed the commands via a Secure Shell (SSH). We created a Jenkins job for this purpose and built it.



This job is created using a Maven Project. Web applications can be packaged into a Web Archive format (WAR) file using tools included in the standard JAVA archive. These files consist of different web services, HTML pages, servlets, etc. For efficiently managing multiple containers from the same image, a Dockerfile is used. A Dockerfile builds a docker image by executing the commands included in it. It is like a text file that consists of all the instructions to build an image successfully. Each command in the Dockerfile creates a layer

Here, the Dockerfile states,

```
# Pull base image
From tomcat:8-jre8
COPY ./webapp.war /usr/local/tomcat/webapps
```

After the Dockerfile is transferred to the Ansible Server machine, we will execute it to build an image and push it in the Docker Hub.

“cd /opt/docker” -> changes the current directory to the directory which contains the Dockerfile.

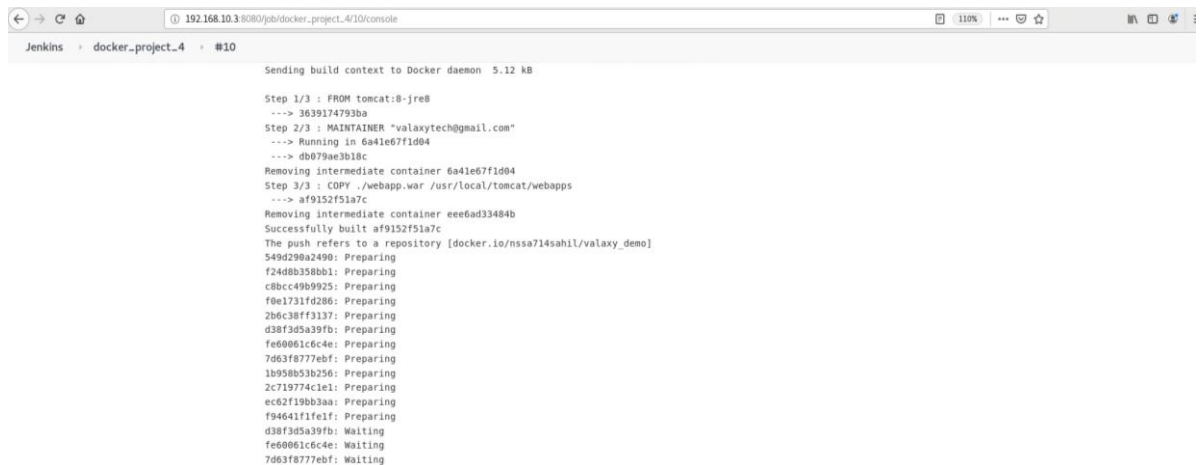
“docker build -t valaxy_demo” -> builds the dockerfile with the image name as valaxy_demo.

“docker tag valaxy_demo valaxy/valaxy_demo” -> Creates a tag Target image which points to the source image.

“docker push valaxy/valaxy_demo” -> Pushes the image to the docker hub.

“docker rmi valaxy_demo valaxy/valaxy_demo” -> Removes the image tab after it is pushed successfully to the docker hub.

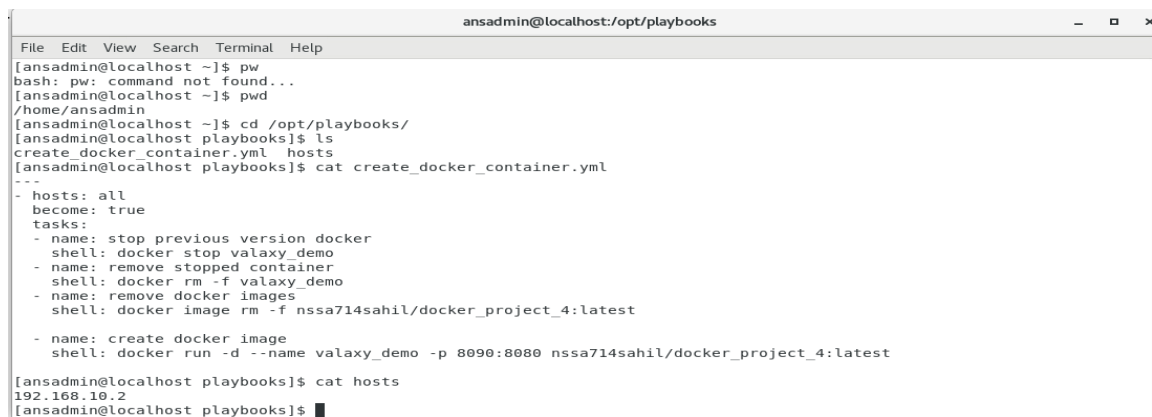
7. After the job is successfully built, we got a docker image in the docker hub, which hosts a simple index.html



```
Jenkins > docker_project_4 > #10
Sending build context to Docker daemon 5.12 kB

Step 1/3 : FROM tomcat:8-jre8
--> 3639174793ba
Step 2/3 : MAINTAINER "valaxytech@gmail.com"
--> Running in 6a41e67f1d94
--> db979ae3b18c
Removing intermediate container 6a41e67f1d94
Step 3/3 : COPY ./webapp.war /usr/local/tomcat/webapps
--> af9152f51a7c
Removing intermediate container eee6ad33484b
Successfully built af9152f51a7c
The push refers to a repository [docker.io/nssa714sahil/valaxy_demo]
549d290a2490: Preparing
f24d8b358bb1: Preparing
c8bcc4909925: Preparing
f9e1731f4286: Preparing
2b6c38ff1137: Preparing
d38f3d5a39fb: Preparing
fef0061c6c4e: Preparing
7d63f8777ebf: Preparing
1b958b53b256: Preparing
2c719774c1e1: Preparing
ec62f19bb3aa: Preparing
f94641f1f1ef: Preparing
d38f3d5a39fb: Waiting
fef0061c6c4e: Waiting
7d63f8777ebf: Waiting
```

8. To this point, we were able to get a docker image in the docker hub. Now, using Ansible we tried to automate and deploy a container from this image.
9. A playbook, .yml file, is created on the Ansible server which acts as a script for Ansible tool to automate configuration on multiple nodes



```
ansadmin@localhost:/opt/playbooks
File Edit View Search Terminal Help
[ansadmin@localhost ~]$ pw
bash: pw: command not found...
[ansadmin@localhost ~]$ pwd
/home/ansadmin
[ansadmin@localhost ~]$ cd /opt/playbooks/
[ansadmin@localhost playbooks]$ ls
create_docker_container.yml hosts
[ansadmin@localhost playbooks]$ cat create_docker_container.yml
--
- hosts: all
  become: true
  tasks:
    - name: stop previous version docker
      shell: docker stop valaxy_demo
    - name: remove stopped container
      shell: docker rm -f valaxy_demo
    - name: remove docker images
      shell: docker image rm -f nssa714sahil/docker_project_4:latest

    - name: create docker image
      shell: docker run -d --name valaxy_demo -p 8090:8080 nssa714sahil/docker_project_4:latest
[ansadmin@localhost playbooks]$ cat hosts
192.168.10.2
[ansadmin@localhost playbooks]$
```

hosts – target machine where the ansible playbook will be implemented,

‘all’ signifies all the IP addresses available

become – gives root privileges.

10. A host file is also created which includes all the IP addresses of the ansible client nodes

11. Another Jenkins job is created to execute this playbook

Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: ansible_server

Advanced...

Transfers

Transfer Set

Source files: create_docker_container.yml

Remove prefix:

Remote directory: /opt/docker

Exec command: cd /opt/playbooks; ansible-playbook -i /opt/playbooks/hosts create_docker_container.yml

All of the transfer fields (except for Exec timeout)

Save Apply

12. Till this point, we have deployed a docker container using a docker image but the container is always deployed using the latest image, thus we can't go back to a specific version if needed

```
TT3eae03ac3a: Pushed
latest: digest: sha256:4501081cf5d4de021cf5da88a4f27ef5f82f6713a3bc90dea4f811cbcf2ffe12 size: 2834
Untagged: valaxy_demo:latest
Untagged: nssa714sahil/valaxy_demo@sha256:4501081cf5d4de021cf5da88a4f27ef5f82f6713a3bc90dea4f811cbcf2ffe12
Untagged: nssa714sahil/valaxy_demo:latest
Deleted: sha256:8e3beba4d550d6433931becf3835996ef243f8fd2246523ff5bc0372c287d75
Deleted: sha256:1fff4e93de3359b8e86da241cca66996942f144bf3dd9abb415f46f3c2594c6
Deleted: sha256:e61d7ec842a9a9db05545a2a241bcab0204661607c9f959298f96c17d2defec5
SSH: EXEC: completed after 4,403 ms
SSH: Disconnecting configuration [ansible_server] ...
SSH: Transferred 1 file(s)
SSH: Connecting from host [localhost.localdomain]
SSH: Connecting with configuration [ansible_server] ...
SSH: EXEC: STDOUT/STDERR from command [cd /opt/playbooks; ansible-playbook -i /opt/playbooks/hosts create_docker_container.yml] ...

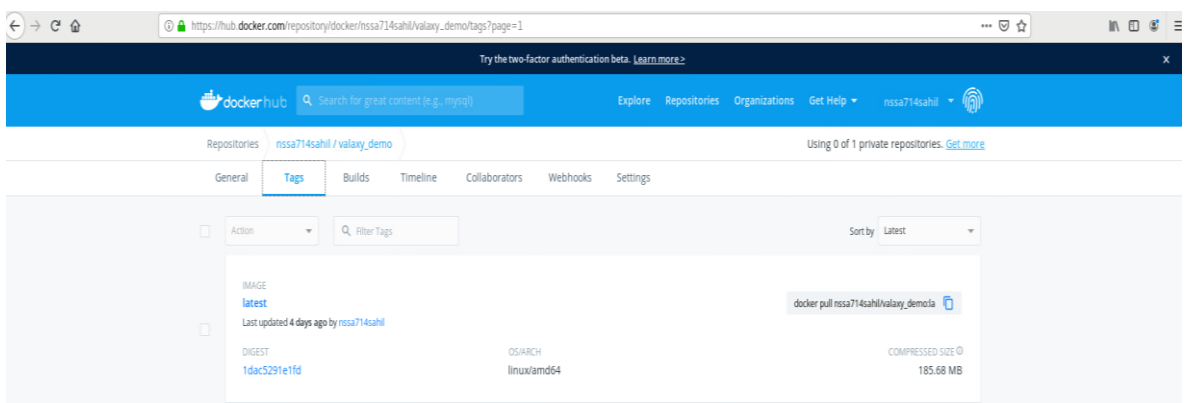
PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.10.2]

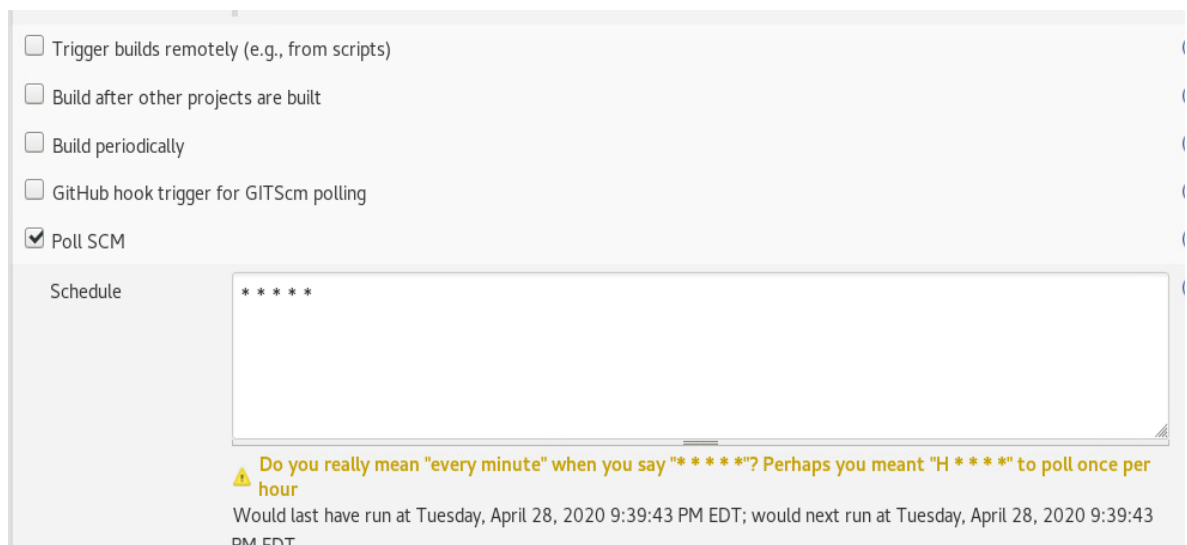
TASK [create docker image] *****
changed: [192.168.10.2]

PLAY RECAP *****
192.168.10.2 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

SSH: EXEC: completed after 3,882 ms
SSH: Disconnecting configuration [ansible_server] ...
SSH: Transferred 0 file(s)
Finished: SUCCESS
```



13. Make Jenkins job to poll SCM periodically. So whenever there's a change in source GitHub repository our job will be automatically built.



Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Build periodically

GitHub hook trigger for GITScm polling

☒ Poll SCM

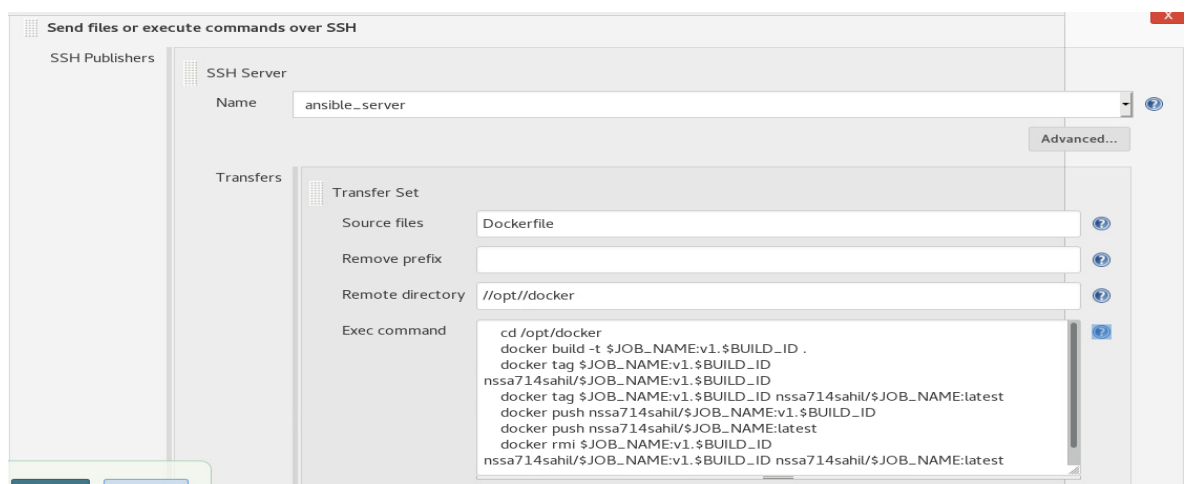
Schedule

* * * * *

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Tuesday, April 28, 2020 9:39:43 PM EDT; would next run at Tuesday, April 28, 2020 9:39:43 PM EDT

14. We modified the existing Dockerfile to include the version of images



Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: ansible_server

Advanced...

Transfers

Transfer Set

Source files: Dockerfile

Remove prefix:

Remote directory: //opt//docker

Exec command:

```
cd /opt/docker
docker build -t $JOB_NAME:v1.$BUILD_ID .
docker tag $JOB_NAME:v1.$BUILD_ID nssa714sahil/$JOB_NAME:v1.$BUILD_ID
docker tag $JOB_NAME:v1.$BUILD_ID nssa714sahil/$JOB_NAME:latest
docker push nssa714sahil/$JOB_NAME:v1.$BUILD_ID
docker push nssa714sahil/$JOB_NAME:latest
docker rmi $JOB_NAME:v1.$BUILD_ID
nssa714sahil/$JOB_NAME:v1.$BUILD_ID nssa714sahil/$JOB_NAME:latest
```

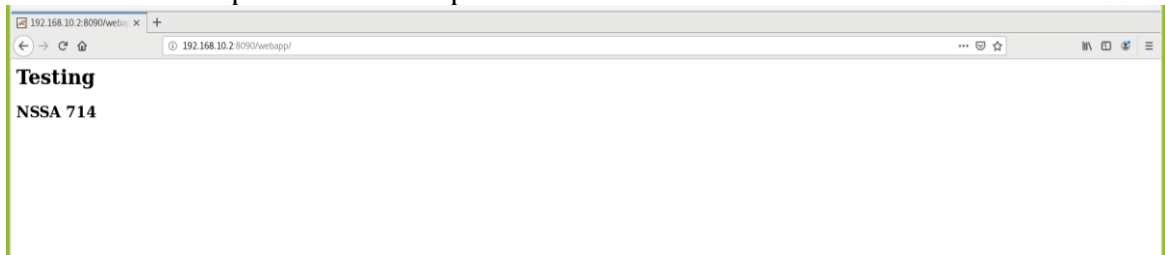
BUILD_ID - The current build id.

JOB_NAME - Name of the project of this build. This is the name you gave your job when you first set it up.

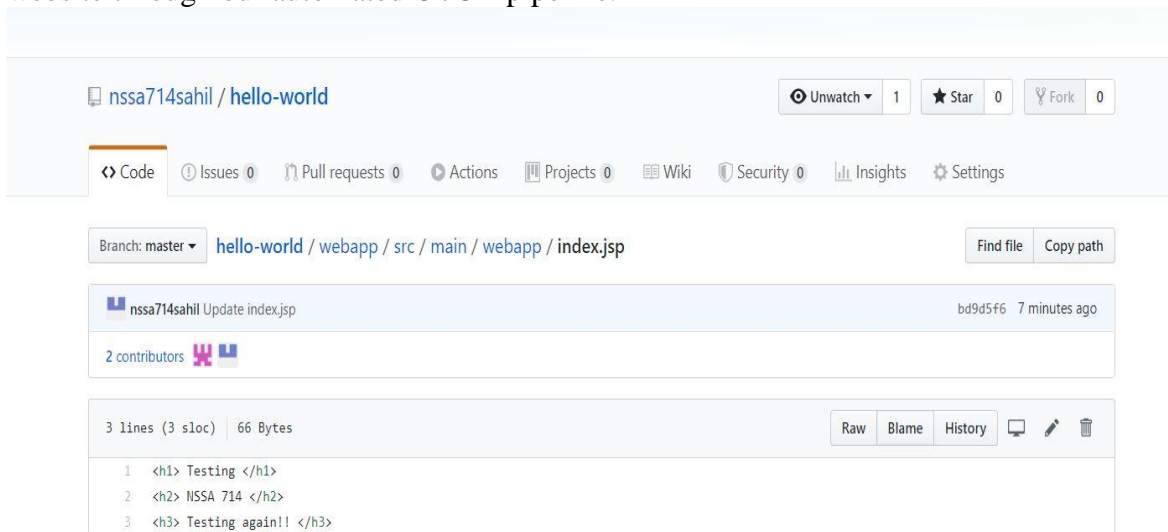
15. Verified that the docker hub now contains multiple images stored along with their version information

4. Result

1. With our first commit the tomcat server was used to display the content of .war file which was a simple .html file on port 8090 on ansible client node.



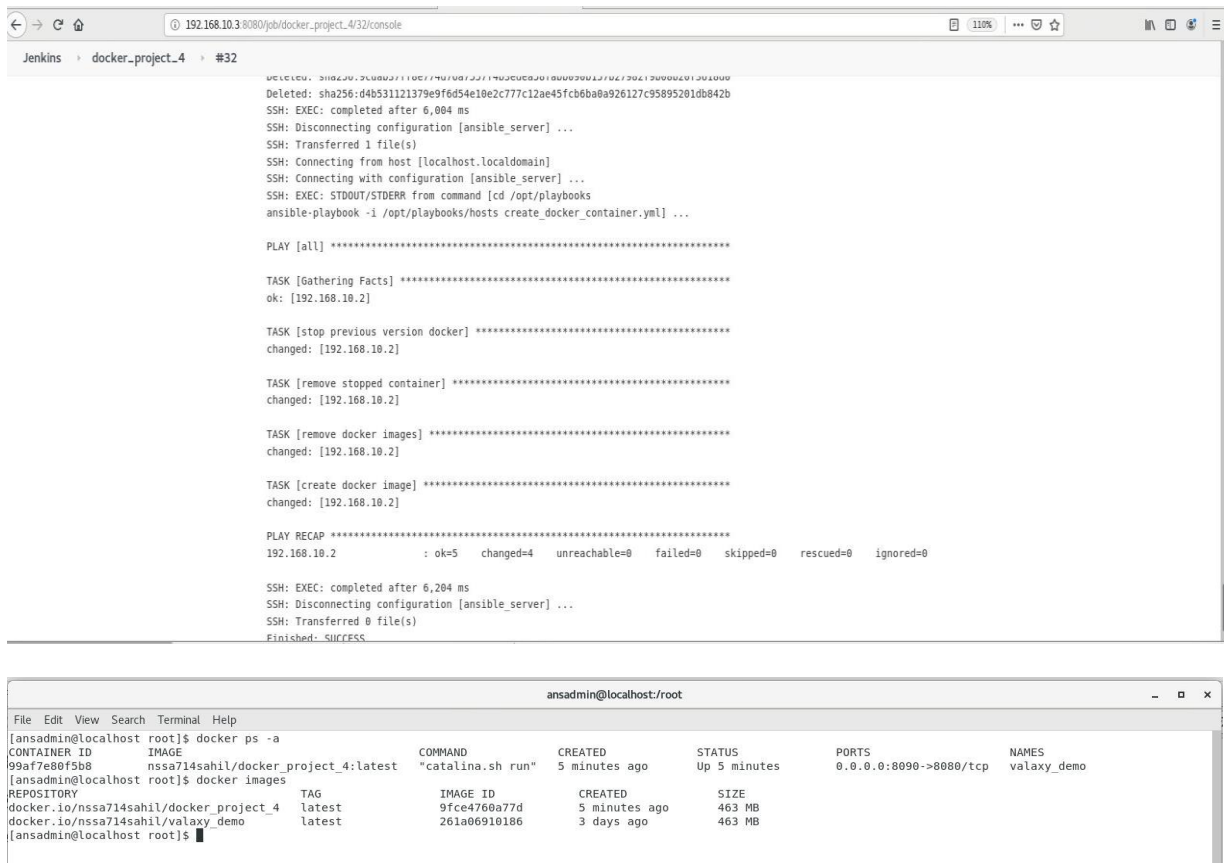
2. We made some changes to our website's index file on GitHub which got reflected on our website through our automated CI/CD pipeline.



3. When we commit the code, Jenkins starts building the job automatically due to SCM polling.



4. Jenkins has successfully built our job and the container and docker image has been deployed in our Ansible client node.



The top screenshot shows the Jenkins console output for a job named 'docker_project_4'. The output displays the execution of an Ansible playbook, including tasks for gathering facts, stopping the previous version of Docker, removing the stopped container, removing Docker images, and creating a new Docker image. The playbook completed successfully.

```
Deleted: sha256:d4b531121379e9f6d54e10e2c777c12ae45fcb6ba0a926127c55895201db842b
SSH: EXEC: completed after 6,004 ms
SSH: Disconnecting configuration [ansible_server] ...
SSH: Transferred 1 file(s)
SSH: Connecting from host [localhost.localdomain]
SSH: Connecting with configuration [ansible_server] ...
SSH: EXEC: STDOUT/STDERR from command [cd /opt/playbooks
ansible-playbook -i /opt/playbooks/hosts create_docker_container.yml] ...

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [192.168.10.2]

TASK [stop previous version docker] *****
changed: [192.168.10.2]

TASK [remove stopped container] *****
changed: [192.168.10.2]

TASK [remove docker images] *****
changed: [192.168.10.2]

TASK [create docker image] *****
changed: [192.168.10.2]

PLAY RECAP *****
192.168.10.2      : ok=5   changed=4   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

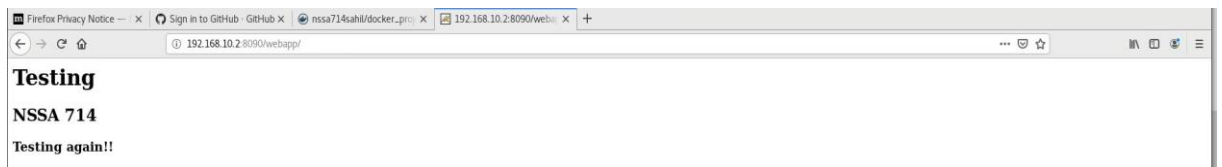
SSH: EXEC: completed after 6,204 ms
SSH: Disconnecting configuration [ansible_server] ...
SSH: Transferred 0 file(s)
Finished: SUCCESS
```

The bottom screenshot shows a terminal window on the Ansible client node (localhost). The user runs the command 'docker ps -a', which displays a table of Docker containers. The table shows a container named 'valaxy_demo' with ID '99af7e80f5b8', created 5 minutes ago, and running the 'catalina.sh' command. The user also runs 'docker images', which displays a table of Docker images. The table shows two images: 'docker.io/nssa714sahil/docker_project_4:latest' (ID '9fce4760a77d', created 5 minutes ago, size 463 MB) and 'docker.io/nssa714sahil/valaxy_demo:latest' (ID '261a06910186', created 3 days ago, size 463 MB).

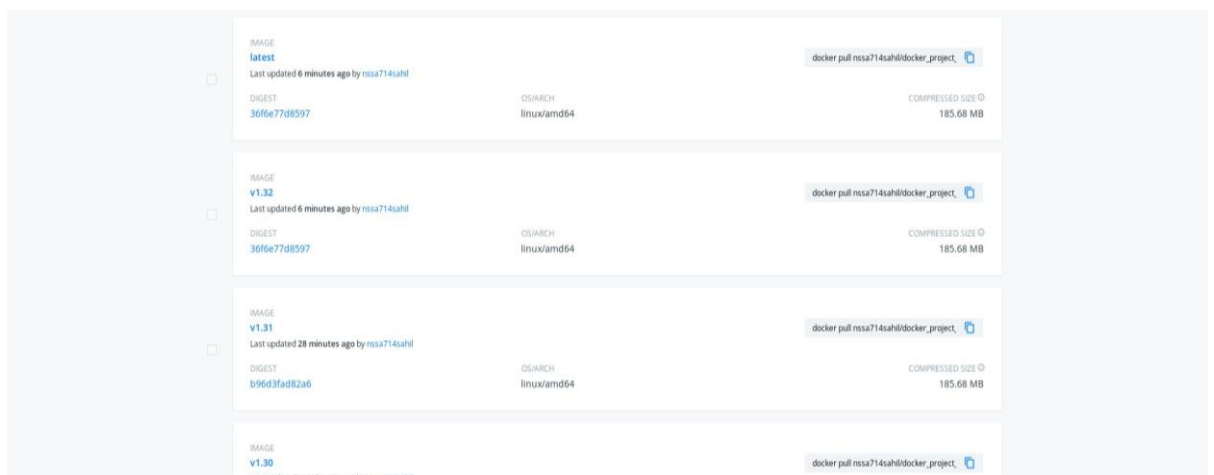
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
99af7e80f5b8	nssa714sahil/docker_project_4:latest	"catalina.sh run"	5 minutes ago	Up 5 minutes	0.0.0.0:8090->8080/tcp	valaxy_demo

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker.io/nssa714sahil/docker_project_4	latest	9fce4760a77d	5 minutes ago	463 MB
docker.io/nssa714sahil/valaxy_demo	latest	261a06910186	3 days ago	463 MB

5. We can see our website has been updated.



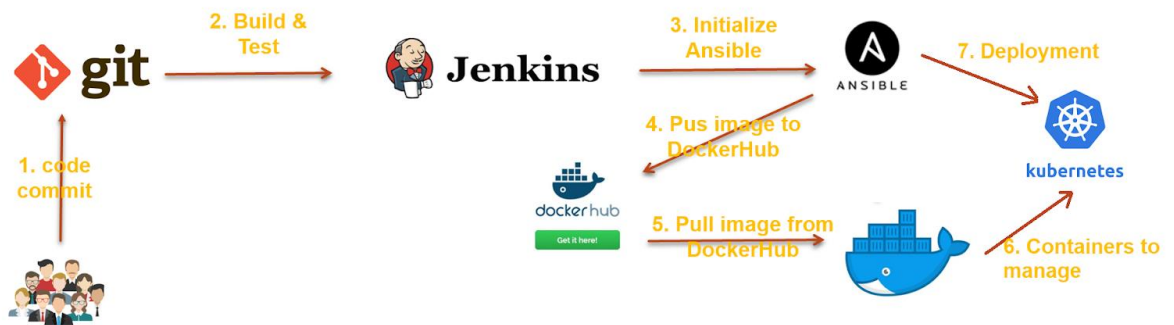
6. At the same time the docker image has been uploaded to docker hub with the latest version and build number. We can verify that version control is working.



5. Future Work

There are numerous tools available in the market for automation of network and DevOps. Though this project is a standalone model, replacing individual tools for better efficiency and easier implementation is always the goal. We could use Vagrant, a tool to manage multiple virtual machines, to work as an infrastructure base; SonarSource to check the quality of our Jenkins processes; Nexus to handle the binaries of the program files we will get from GitHub. Going cloud will save a lot of hassle in managing all these resources, so using AWS or Azure will be efficient. We could add features like load-balancing, Kubernetes, complex ansible-playbook cater to different needs of clients. Kubernetes will help to scale and manage containers and load balancing will ensure the load is distributed properly over all the nodes.

Potential network infrastructure is as shown below:



6. References

- <https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch15s03.html>
- <https://docs.docker.com/engine/reference/commandline/image/>
- <https://docs.ansible.com/ansible/latest/plugins/inventory/yaml.html>
- https://www.tutorialspoint.com/ansible/ansible_yaml_basics.htm
- https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html
- <https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch15s03.html>
- <https://thenewstack.io/docker-basics-how-to-use-dockerfiles/>
- <http://java.boot.by/wcd-guide/ch02s04.html>
- https://docs.ansible.com/ansible/latest/user_guide/become.html
- github.com/ValaxyTech

Video Link: <https://tinyurl.com/ybj3m24x>

Skip from 0:55 to 1:30