

## ANSIBLE REFACTORING AND STATIC ASSIGNMENTS (IMPORTS AND ROLES)

In this project we will continue working with 'ansible-config-mgt' repository and make some improvements of our code. Now you need to refactor our Ansible code, create assignments, and learn how to use the imports functionality. Imports allows to effectively re-use previously created playbooks in a new playbook - it helps to organize tasks and reuse them when needed.

Firstly, let's explain code refactoring.

**Refactoring** is a general term in computer programming. It means making changes to the source code without changing expected behaviour of the software. The main idea of refactoring is to enhance code readability, increase maintainability and extensibility, reduce complexity, add proper comments without affecting the logic. It involves making changes to the code's internal structure, organization, and implementation while preserving its functionality.

### STEP 1- JENKINS JOB ENHANCEMENT

Before we begin, let us make some changes to our Jenkins job - now every new change in the codes creates a separate directory which is not very convenient when we want to run some commands from one place. Besides, it consumes space on Jenkins server with each subsequent change. Let us enhance it by introducing a new Jenkins project/job - we will require **Copy Artifact** plugin.

1. Go to your Jenkins-Ansible server and create a new directory called 'ansible-config-artifact' - which will be utilized to store all artifacts after each build.

```
$ sudo mkdir /home/ubuntu/ansible-config-artifact
```

2. Change permissions to this directory, so Jenkins could save files there -

```
$ chmod -R 0777 /home/ubuntu/ansible-config-artifact.
```

```

ubuntu@ip-172-31-37-181:~$ sudo mkdir ansible-config-artifact
ubuntu@ip-172-31-37-181:~$ ls
ansible-config-artifact  ansible-config-mgt  inventory  test-folder
ubuntu@ip-172-31-37-181:~$ cd ansible-config-mgt/
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ ls
Ansible.md  inventory  playbooks
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$
Session was closed
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1039-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Jul  6 12:15:41 UTC 2023

System load:  0.0               Processes:            99
Usage of /:   53.3% of 7.57GB    Users logged in:     0
Memory usage: 61%              IPv4 address for eth0: 172.31.37.181
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

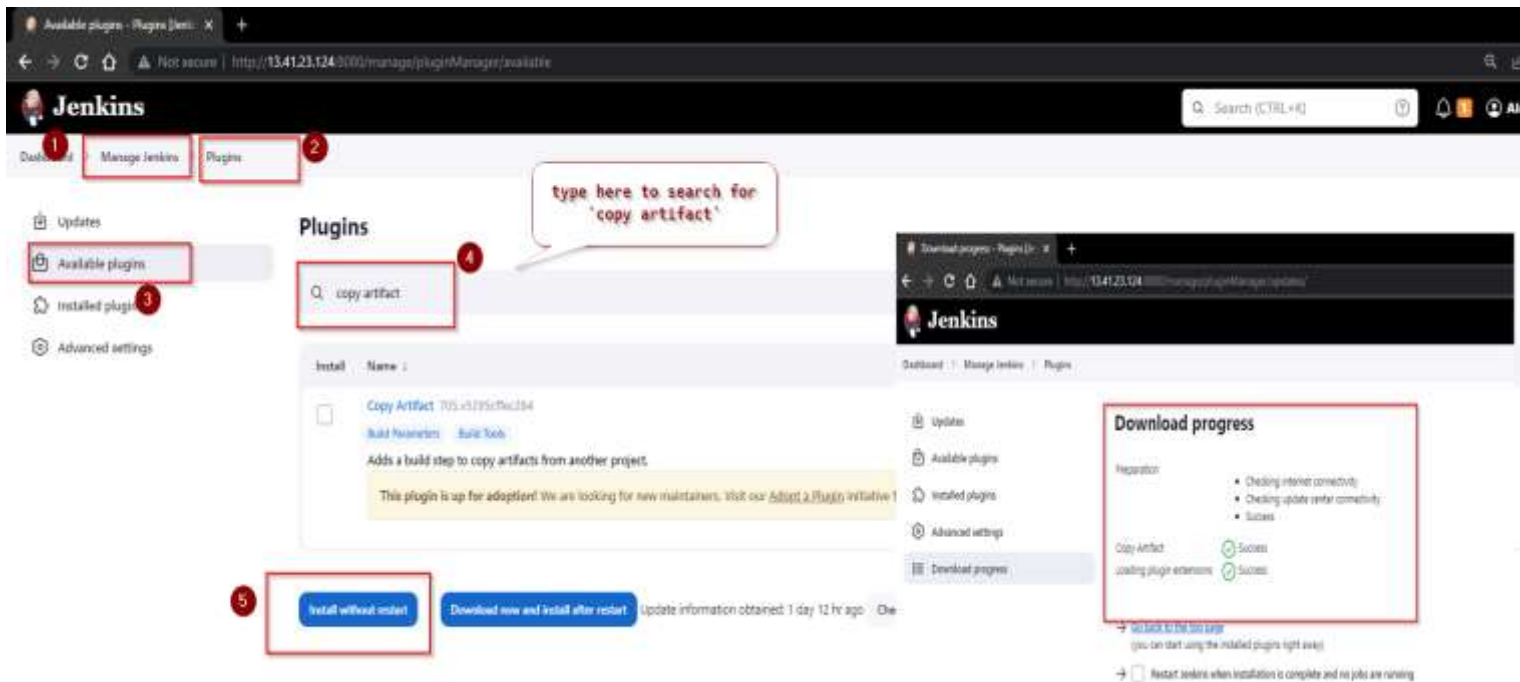
38 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Jul  5 17:44:15 2023 from 109.158.217.167
ubuntu@ip-172-31-37-181:~$ ls
ansible-config-artifact  ansible-config-mgt  inventory  test-folder
ubuntu@ip-172-31-37-181:~$ sudo chmod -R 777 /home/ubuntu/ansible-config-artifact/
ubuntu@ip-172-31-37-181:~$

```

3. Go to Jenkins web console -> Manage Jenkins -> Manage Plugins -> on Available plugins tab search for Copy Artifact and install this plugin without restarting Jenkins.



4. We will create a new Freestyle project and name it 'save\_artifacts'.

5. This project will be triggered by completion of your existing ansible project. Hence, we will configure it accordingly:

### Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

### General

Enabled

Description

[Plain text](#) [Preview](#)

☒ Discard old builds

Strategy

Log Rotation

Days to keep builds

If not empty, build records are only kept up to this number of days

Max # of builds to keep

If not empty, only up to this number of build records are kept

2

Advanced

- ☐ GitHub project
- ☐ Permission to Copy Artifact
- ☐ This project is parameterised
- ☐ Throttle builds

save\_artifacts Config [Jenkins] x Dashboard [Jenkins] x +

Not secure | http://13.41.23.124:8080/job/save\_artifacts/configure

Dashboard > save\_artifacts > Configuration

Advanced v

## Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

### Source Code Management

☒ None

☐ Git ?

### Build Triggers

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

☒ Build after other projects are built ?

Projects to watch

ansible,

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Always trigger, even if the build is aborted

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

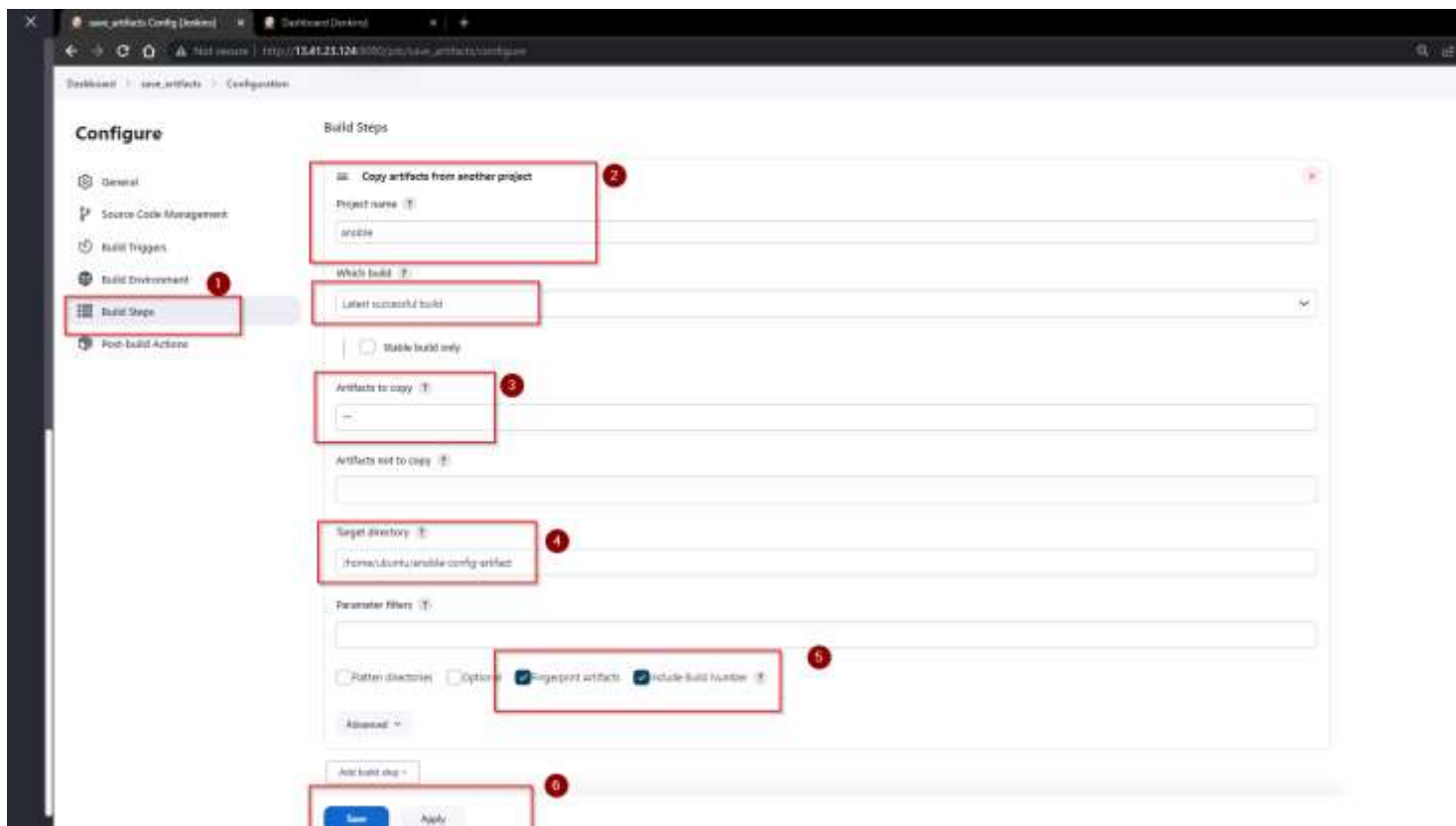
### Build Environment

☐ Delete workspace before build starts

Save Apply

**Please note:** Depending on the number of builds you might want to keep, say only last 2 or 5 build results, you can configure number of builds to keep in order to save space on the server.

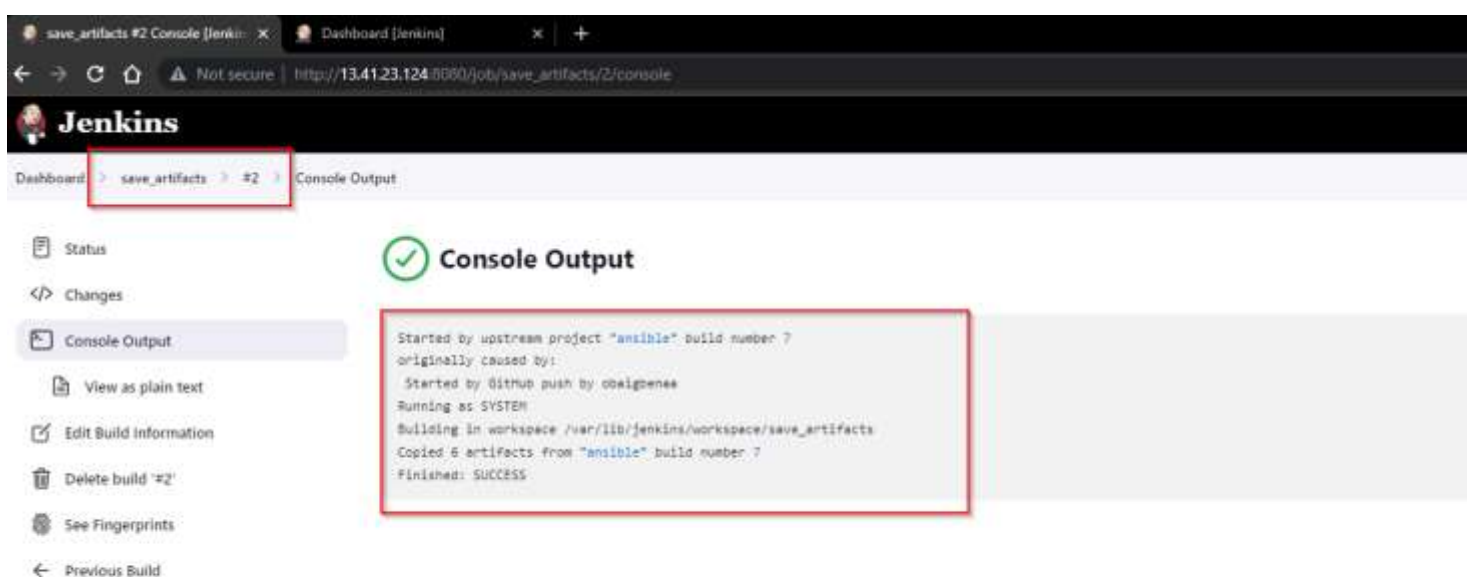
- The main idea of 'save\_artifacts project' is to save artifacts into **/home/ubuntu/ansible-config-artifact** directory. To achieve this, create a Build step and choose **Copy artifacts from other project**, specify **ansible** as a source project and **/home/ubuntu/ansible-config-artifact** as a target directory.



7. Test your set up by making some changes in README.MD file inside your **ansible-config-mgt repository** (right inside main/master branch).

If both Jenkins jobs have completed one after another – you shall see your files inside **/home/ubuntu/ansible-config-artifact directory** and it will be updated with every commit to your master branch.

Now your Jenkins pipeline is neater and cleaner!.



*Image above shows build triggered in 'save-artifacts' project.*

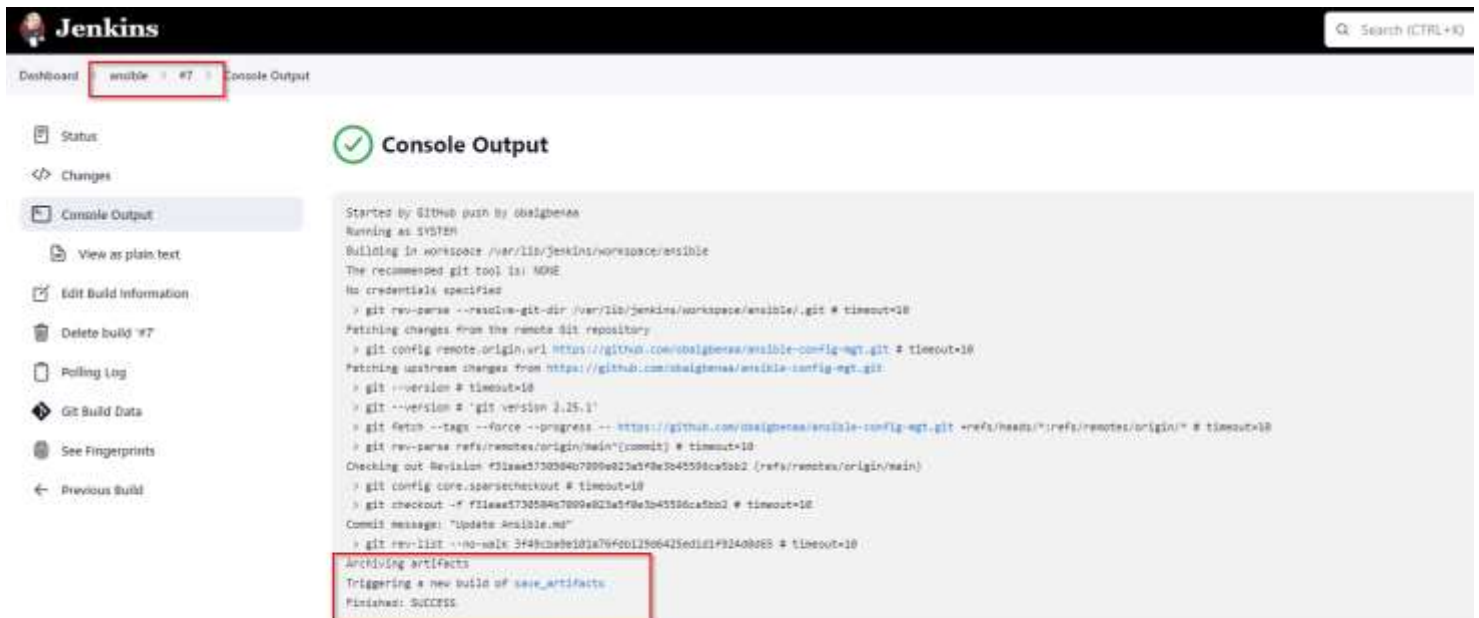


Image above also shows build triggered simultaneously in 'ansible' project on Jenkins.

```
ubuntu@ip-172-31-37-181:~$ cd ansible-config-artifact/  
ubuntu@ip-172-31-37-181:~/ansible-config-artifact$ ls  
7  
ubuntu@ip-172-31-37-181:~/ansible-config-artifact$ cd 7/  
ubuntu@ip-172-31-37-181:~/ansible-config-artifact/7$ ls  
Ansible.md  inventory  playbooks  
ubuntu@ip-172-31-37-181:~/ansible-config-artifact/7$
```

## STEP 2. REFACTOR ANSIBLE CODE BY IMPORTING OTHER PLAYBOOKS INTO site.yml

Before starting to refactor the codes, ensure that you have pulled down the latest code from master (main) branch, and created a new branch, name it *refactor*.

```
$ git pull
```

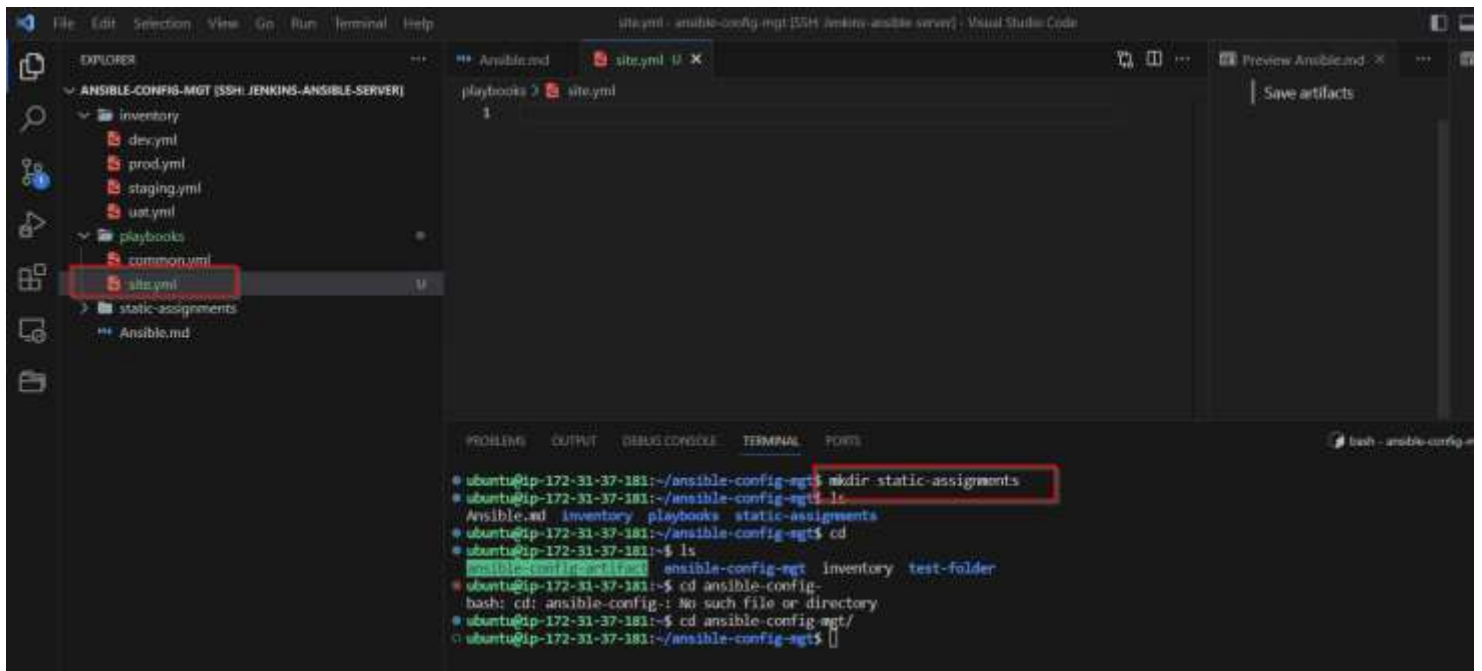
```
$ git branch refactor
```



```
● ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git pull
Already up to date.
● ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git branch refactor
● ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git branch
  feature/proj-45
* main
  refactor
○ ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ █
```

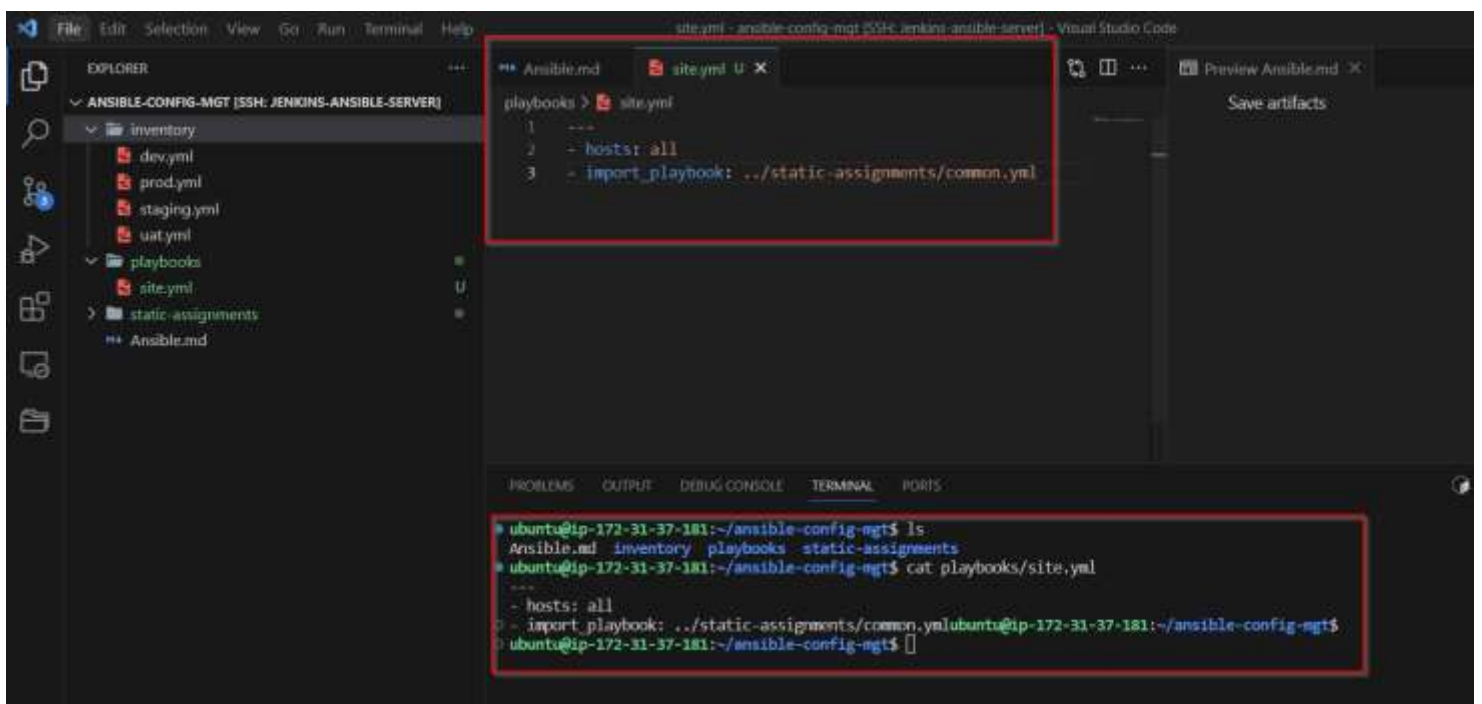
1. Within **playbooks** folder, create a new file and name it **site.yml** – This file will now be considered as an entry point into the entire infrastructure configuration. Other playbooks will be included here as a reference. In other words, **site.yml** will become a parent to all other playbooks that will be developed. Including **common.yml** that you created previously. Don't worry, you will understand more what this means shortly.??
2. Create a new folder in root of the repository and name it **static-assignments**. The static-assignments folder is where all other children playbooks will be stored. This is merely for easy organization of your work. It is not an Ansible specific concept, therefore you can choose how you want to organize your work. You will see why the folder name has a prefix of static very soon. For now, just follow along.??



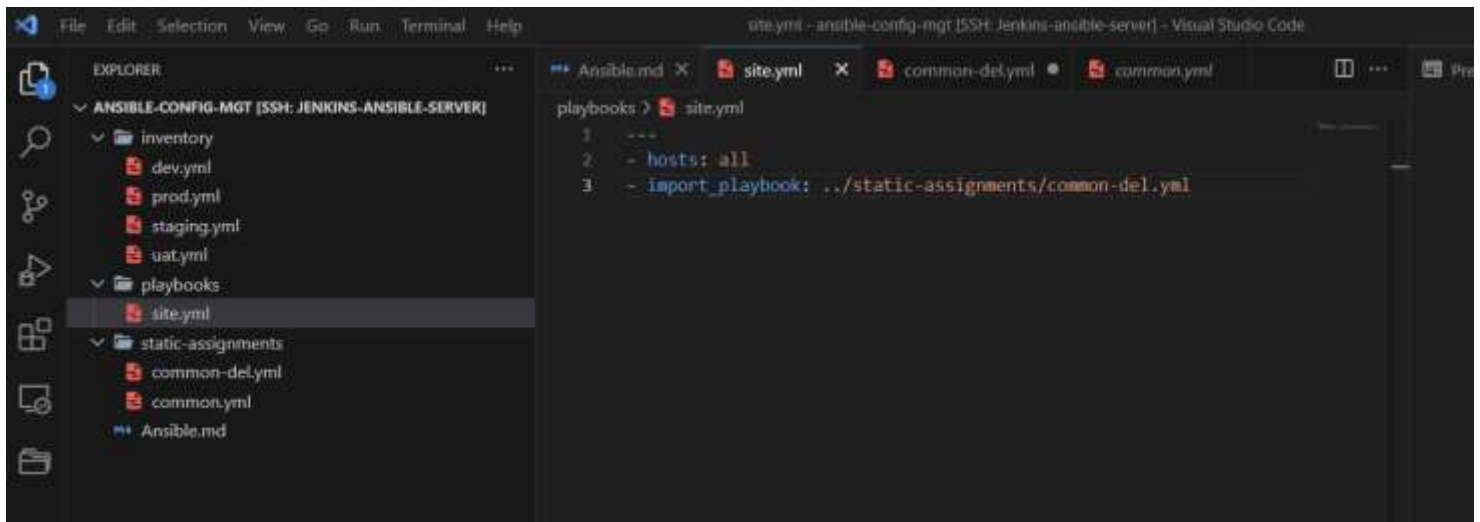


3. Move **common.yml** file into the newly created **static-assignments** folder.

4. Inside **site.yml** file, import **common.yml** playbook.



update **site.yml** with **- import\_playbook: ../static-assignments/common-del.yml** instead of **common.yml** and run it against **dev** servers:



```
$ cd /home/ubuntu/ansible-config-mgt/
```

```
$ ansible-playbook -i inventory/dev.yml playbooks/site.yml
```

**Wait!!!** Before running the `ansible-playbook` command we need to push our code changes unto github.

### Commit your code into GitHub:

1. Use git commands to **add**, **commit** and **push** your branch to GitHub.

```
$ git status -this shows us what branch we are currently working on.
```

In this case, we are currently on our `feature/project-45` branch.

```
$ git add <selected files>
```

```
$ git add . (to add every change in the ansible-config-mgt repository)
```

```
$ git push
```

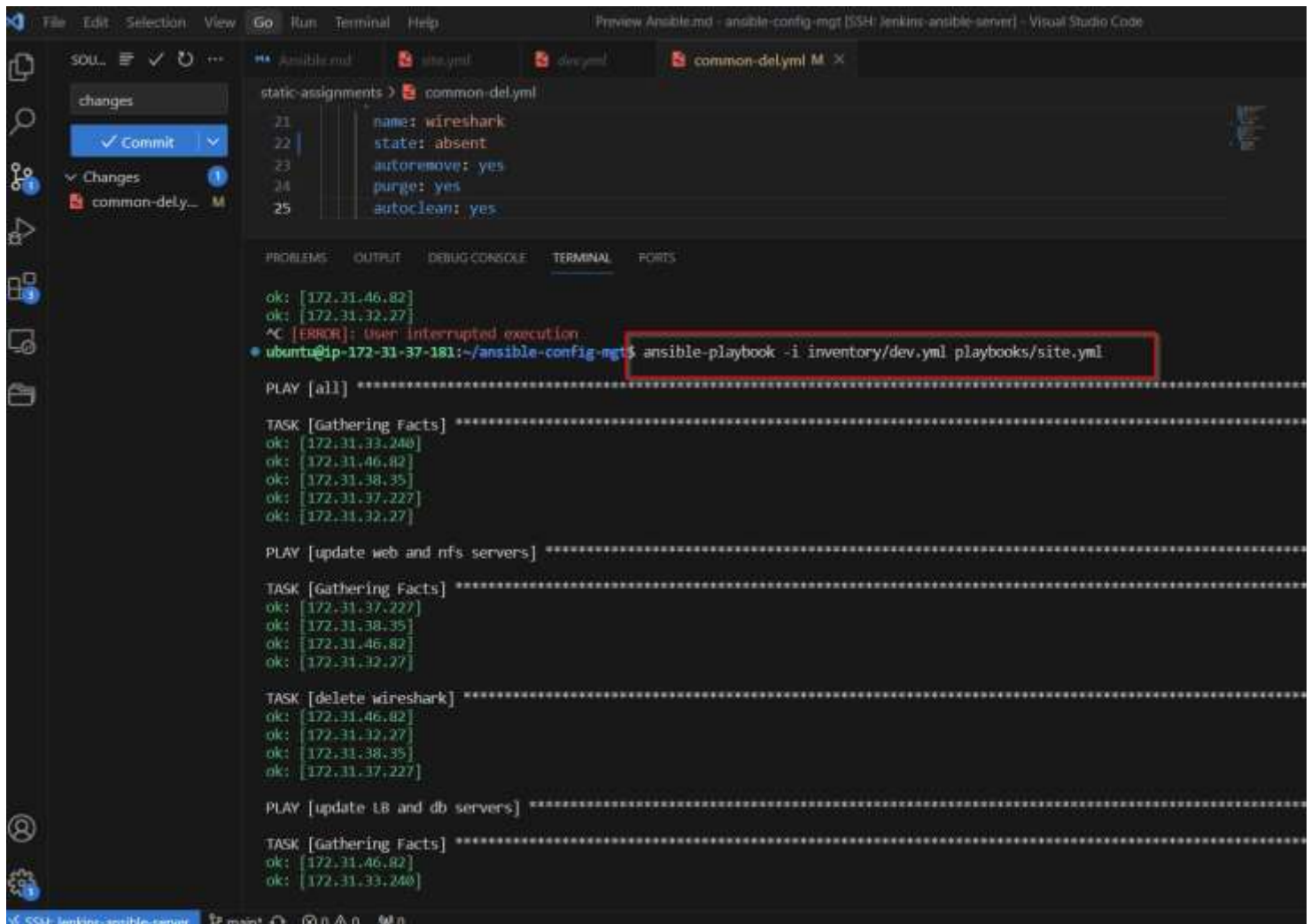
```
site.yml - ansible-config-mgt [SSH: Jenkins-ansible-server] - Visual Studio Code
SOURCE CONTROL
Message (Ctrl+Enter to commit on 'main')
[✓] Commit
playbooks > site.yml
1 ---
2 - hosts: all
3 - import_playbook: ../static-assignments/common-del.yml
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ERROR! the playbook: playbooks/site.yml could not be found
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ ansible-playbook -i inventory/dev.yml playbooks/site.yml
ERROR! the playbook: playbooks/site.yml could not be found
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    playbooks/common.yml

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    playbooks/site.yml
    static-assignments/

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git add
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git add .
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git commit -m "added site and common-del"
[main b3dc3d] added site and common-del
3 files changed, 3 insertions(+)
create mode 100644 playbooks/site.yml
create mode 100644 static-assignments/common-del.yml
rename {playbooks -> static-assignments}/common.yml (100%)
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 559 bytes | 279.00 Kib/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/obaigbenaa/ansible-config-mgt.git
7672bb2..b3dc3d main -> main
ubuntu@ip-172-31-37-181:~/ansible-config-mgt$
```

\$ ansible-playbook -i inventory/dev.yml playbooks/site.yml



Make sure that wireshark is deleted on all the servers by running  
`$ wireshark --version`

```
[ec2-user@ip-172-31-32-27 ~]$ wireshark --version
-bash: wireshark: command not found
[ec2-user@ip-172-31-32-27 ~]$ wireshark --version
-bash: wireshark: command not found
[ec2-user@ip-172-31-32-27 ~]$
```

Now you have learned how to use `import_playbooks` module and you have a ready solution to install/delete packages on multiple servers with just one command.

### STEP 3. CONFIGURE UAT WEBSERVERS WITH A ROLE 'WEBSERVER'

We have our nice and clean *dev* environment, so let us put it aside and configure 2 new Web Servers as *uat*. We could write tasks to configure Web Servers in the same playbook, but it would be too messy, instead, we will use a dedicated *role* to make our configuration reusable.

1. Launch 2 fresh EC2 instances using RHEL 8 image, we will use them as our *uat* servers, so give them names accordingly - **Web1-UAT** and **Web2-UAT**.
2. To create a role, you must create a directory called **roles/**, relative to the playbook file or in **/etc/ansible/** directory.
3. Use an Ansible utility called **ansible-galaxy** inside **ansible-config-mgt/roles** directory (you need to create roles directory upfront)

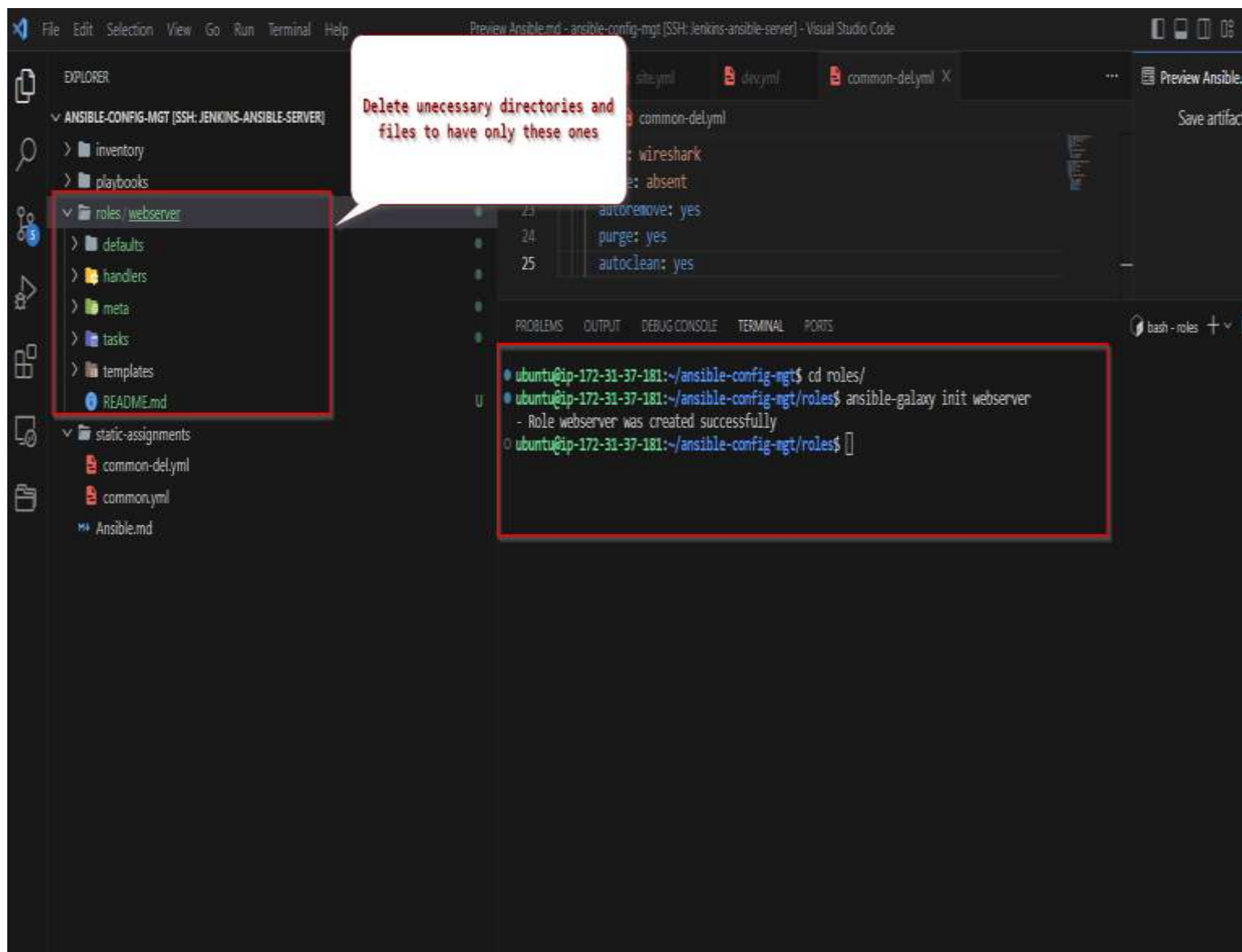
```
$ mkdir roles
```

```
$ cd roles
```

```
$ ansible-galaxy init webserver
```

You can also do this directly on VSCode editor or via the terminal.

Under the directory **roles/webserver**, remove unnecessary directories and files so that the **roles** structure looks exactly like this below.



```
└─ webservice
   ├── README.md
   ├── defaults
   │   └─ main.yml
   ├── handlers
   │   └─ main.yml
   ├── meta
   │   └─ main.yml
   ├── tasks
   │   └─ main.yml
   └─ templates
```