TOOLING WEBSITE DEPLOYMENT AUTOMATION WITH CONTINUOUS INTEGRATION. INTRODUCTION TO JENKINS

In this project we are going to start automating part of our routine tasks with a free and open source automation server – Jenkins. It is one of the most popular CI/CD tools, it was created by a former Sun Microsystems developer Kohsuke Kawaguchi and the project originally had a named "Hudson".

Acording to Circle CI, **Continuous integration (CI)** is a software development strategy that increases the speed of development while ensuring the quality of the code that teams deploy. Developers continually commit code in small increments (at least daily, or even several times a day), which is then automatically built and tested before it is merged with the shared repository.

In our project we are going to utilize Jenkins CI capabilities to make sure that every change made to the source code in GitHub `https://github.com/<yourname>/tooling` will automatically be updated to the Tooling Website.

Task

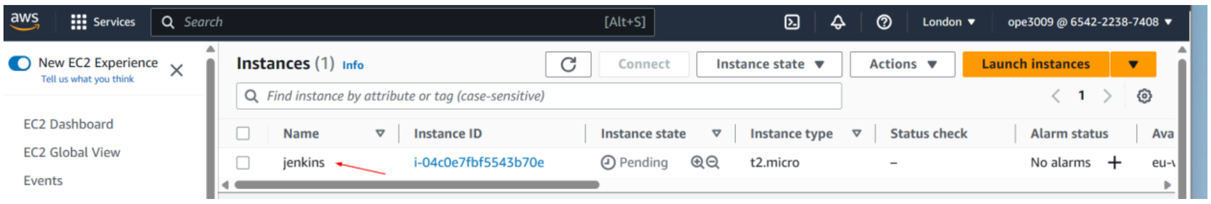
Enhance the architecture prepared in Project 8 by adding a Jenkins server, configure a job to automatically deploy source codes changes from Git to NFS server.

Here is how your updated architecture will look like upon competion of this project:

INSTALL AND CONFIGURE JENKINS SERVER

Step 1 – Install Jenkins server.

1. Create an AWS EC2 server based on Ubuntu Server 20.04 LTS and name it "Jenkins"



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Ava
jenkins	i-04c0e7fbf5543b70e	Pending	t2.micro	–	No alarms	eu-\

2. Install JDK (since Jenkins is a Java-based application)

```
sudo apt update
sudo apt install default-jdk-headless
```

3. Install Jenkins

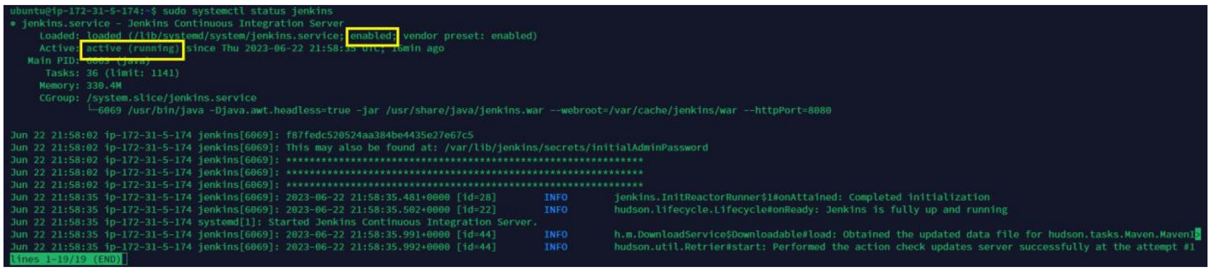
```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key |
sudo tee \
    /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
    https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update
sudo apt-get install fontconfig openjdk-11-jre
sudo apt-get install jenkins
```

Make sure Jenkins is up and running

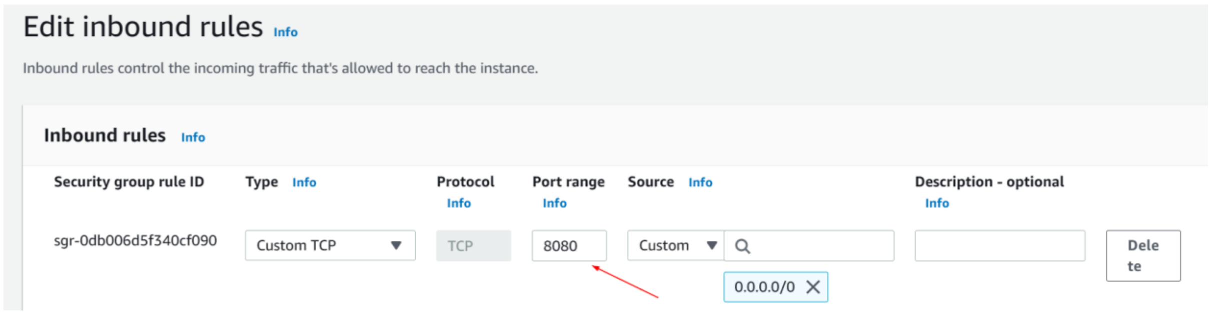
```
sudo systemctl status jenkins
```



```
ubuntu@ip-172-31-5-174:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-22 21:58:35 UTC; 16min ago
   Main PID: 6069 (java)
      Tasks: 36 (limit: 1141)
     Memory: 330.4M
     CGroup: /system.slice/jenkins.service
             └─6069 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: f87fedc520524aa384be4435e27e67c5
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************************
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************************
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************************
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.481+0000 [id=28]       INFO       jenkins.InitReactorRunner$1#onAttained: Completed initialization
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.502+0000 [id=22]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Jun 22 21:58:35 ip-172-31-5-174 systemd[1]: Started Jenkins Continuous Integration Server.
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.991+0000 [id=44]       INFO       h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenI>
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.992+0000 [id=44]       INFO       hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
lines 1-19/19 (END)
```

4. By default Jenkins server uses TCP port 8080 – open it by creating a new Inbound Rule in your EC2 Security Group



Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-0db006d5f340cf090	Custom TCP	TCP	8080	Custom 0.0.0.0/0		Delete

5. Perform initial Jenkins setup.

From your browser access `http://<Jenkins-Server-Public-IP-Address-or-Public-DNS-Name>:8080`

You will be prompted to provide a default admin password

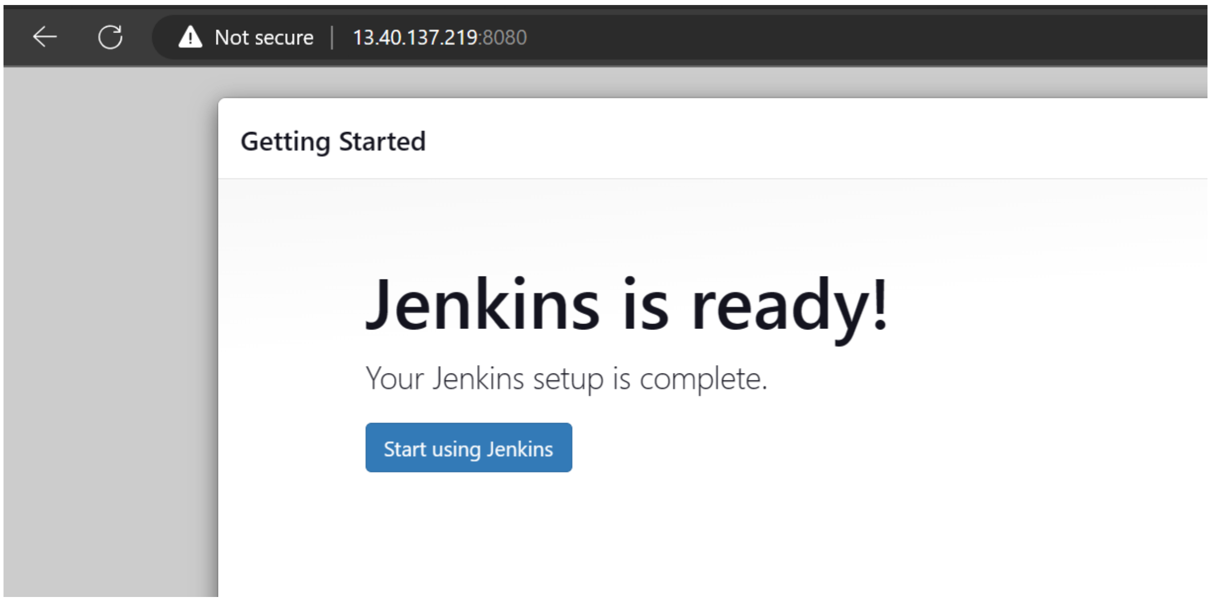
Retrieve it from your server:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Then you will be asked which plugings to install – choose suggested plugins.

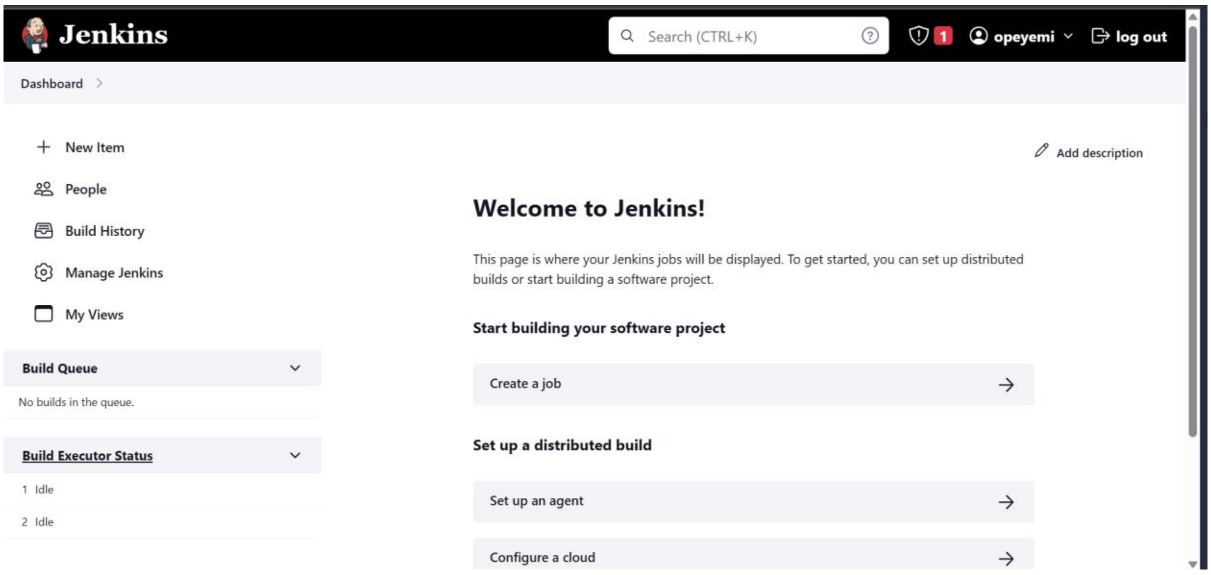
Once plugins installation is done – create an admin user and you will get your Jenkins server address.

The installation is completed!



Jenkins is ready!

Your Jenkins setup is complete.



No builds in the queue.

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

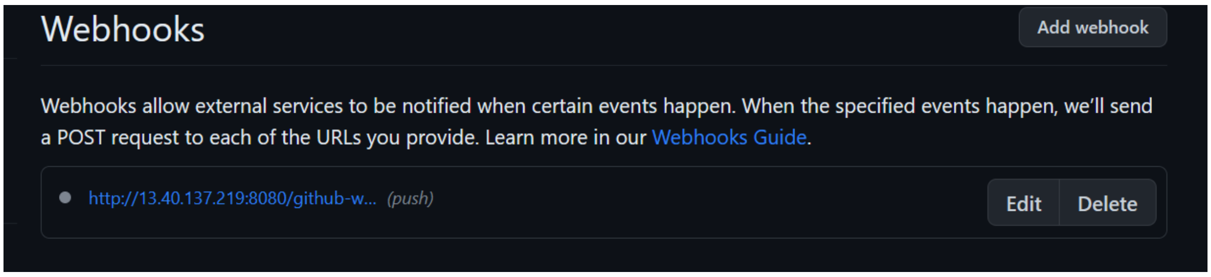
Start building your software project

Set up a distributed build

Step 2 – Configure Jenkins to retrieve source codes from GitHub using Webhooks

In this part, you will learn how to configure a simple Jenkins job/project (these two terms can be used interchangeably). This job will will be triggered by GitHub webhooks and will execute a ‘build’ task to retrieve codes from GitHub and store it locally on Jenkins server.

1. Enable webhooks in your GitHub repository settings



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we’ll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

● http://13.40.137.219:8080/github-w... *(push)*

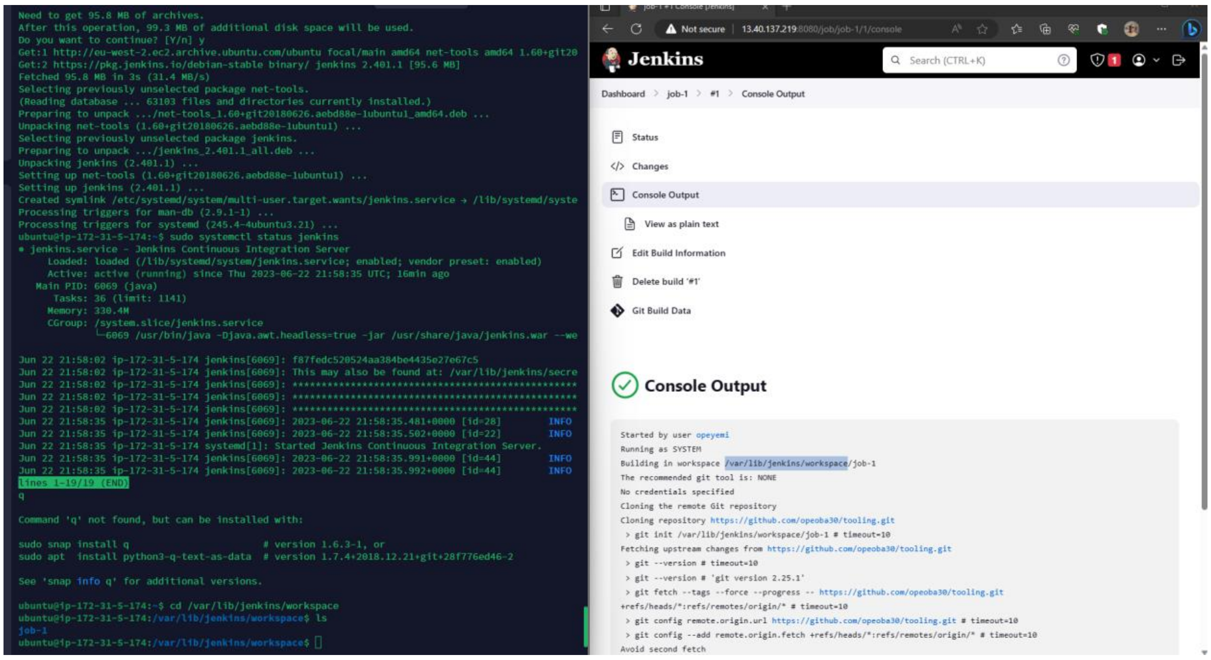
2. Go to Jenkins web console, click "New Item" and create a "Freestyle project"

To connect your GitHub repository, you will need to provide its URL, you can copy from the repository itself

In configuration of your Jenkins freestyle project choose Git repository, provide there the link to your Tooling GitHub repository and credentials (user/password) so Jenkins could access files in the repository.

Save the configuration and let us try to run the build. For now we can only do it manually.
Click "Build Now" button, if you have configured everything correctly, the build will be successfull and you will see it under `#1`

You can open the build and check in "Console Output" if it has run successfully.



```
Need to get 95.8 MB of archives.
After this operation, 99.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu focal/main amd64 net-tools amd64 1.60+git20
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.401.1 [95.6 MB]
Fetched 95.8 MB in 3s (31.4 MB/s)
Selecting previously unselected package net-tools.
(Reading database ... 63103 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20180626.aebd88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.401.1_all.deb ...
Unpacking jenkins (2.401.1) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up jenkins (2.401.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /lib/systemd/syste
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.21) ...
ubuntu@ip-172-31-5-174:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2023-06-22 21:58:35 UTC; 16min ago
   Main PID: 6069 (java)
      Tasks: 36 (limit: 1141)
     Memory: 330.4M
     CGroup: /system.slice/jenkins.service
             └─6069 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --we

Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: f87fedc520524aa384be4435e27e67c5
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: This may also be found at: /var/lib/jenkins/secre
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************
Jun 22 21:58:02 ip-172-31-5-174 jenkins[6069]: *************************************************
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.481+0000 [id=28]        INFO
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.502+0000 [id=22]        INFO
Jun 22 21:58:35 ip-172-31-5-174 systemd[1]: Started Jenkins Continuous Integration Server.
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.991+0000 [id=44]        INFO
Jun 22 21:58:35 ip-172-31-5-174 jenkins[6069]: 2023-06-22 21:58:35.992+0000 [id=44]        INFO
lines 1-19/19 (END)
q

Command 'q' not found, but can be installed with:

sudo snap install q                     # version 1.6.3-1, or
sudo apt  install python3-q-text-as-data  # version 1.7.4+2018.12.21+git+28f776ed46-2

See 'snap info q' for additional versions.

ubuntu@ip-172-31-5-174:~$ cd /var/lib/jenkins/workspace
ubuntu@ip-172-31-5-174:/var/lib/jenkins/workspace$ ls
job-1
ubuntu@ip-172-31-5-174:/var/lib/jenkins/workspace$
```

Console Output

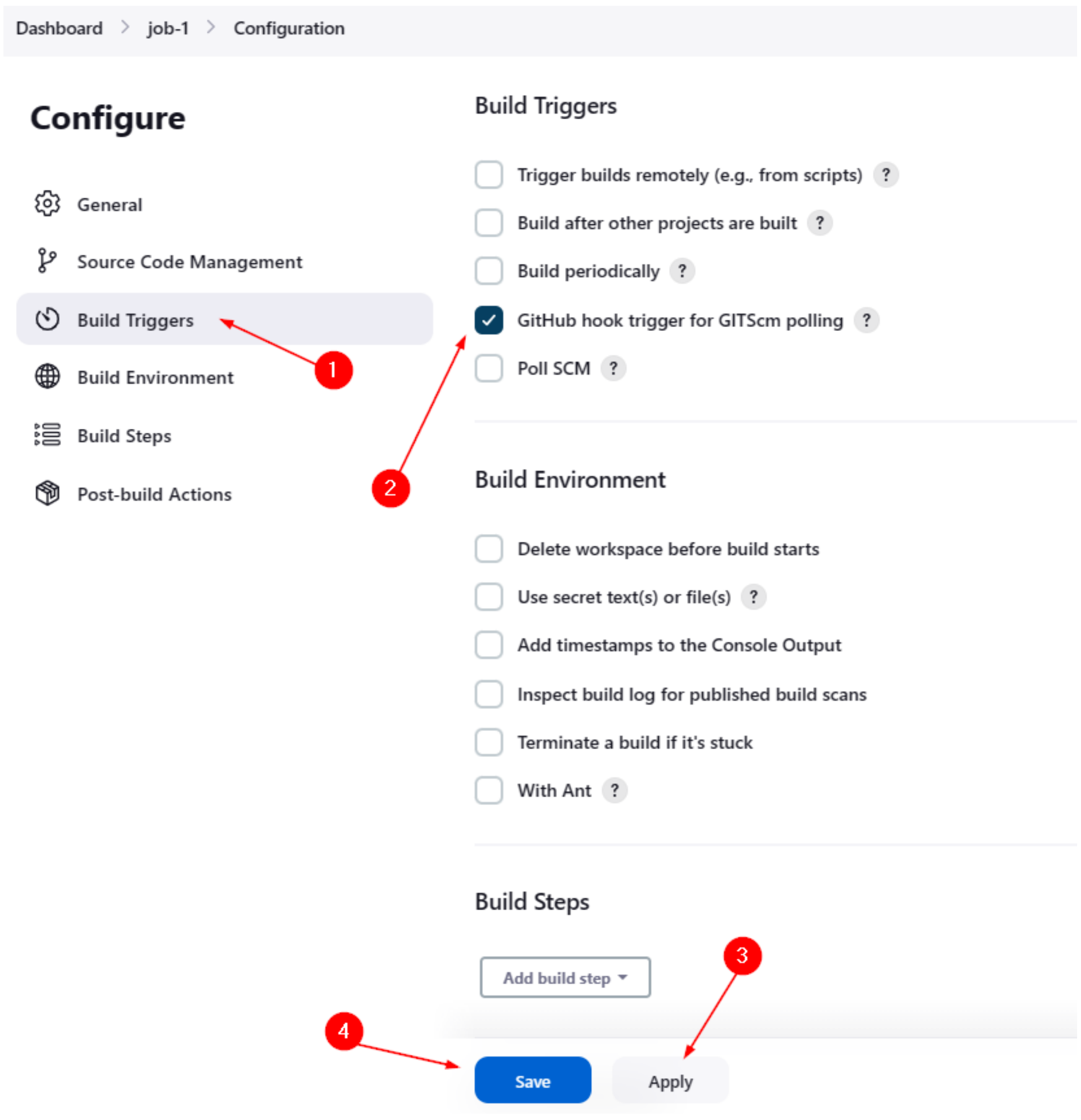
```
Started by user opeyemi
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/job-1
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/opeoba30/tooling.git
 > git init /var/lib/jenkins/workspace/job-1 # timeout=10
Fetching upstream changes from https://github.com/opeoba30/tooling.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/opeoba30/tooling.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/opeoba30/tooling.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
```

If so – congratulations! You have just made your very first Jenkins build!

But this build does not produce anything and it runs only when we trigger it manually. Let us fix it.

3. Click "Configure" your job/project and add these two configurations

Configure triggering the job from GitHub webhook:


Dashboard > job-1 > Configuration

Configure

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

Build Triggers

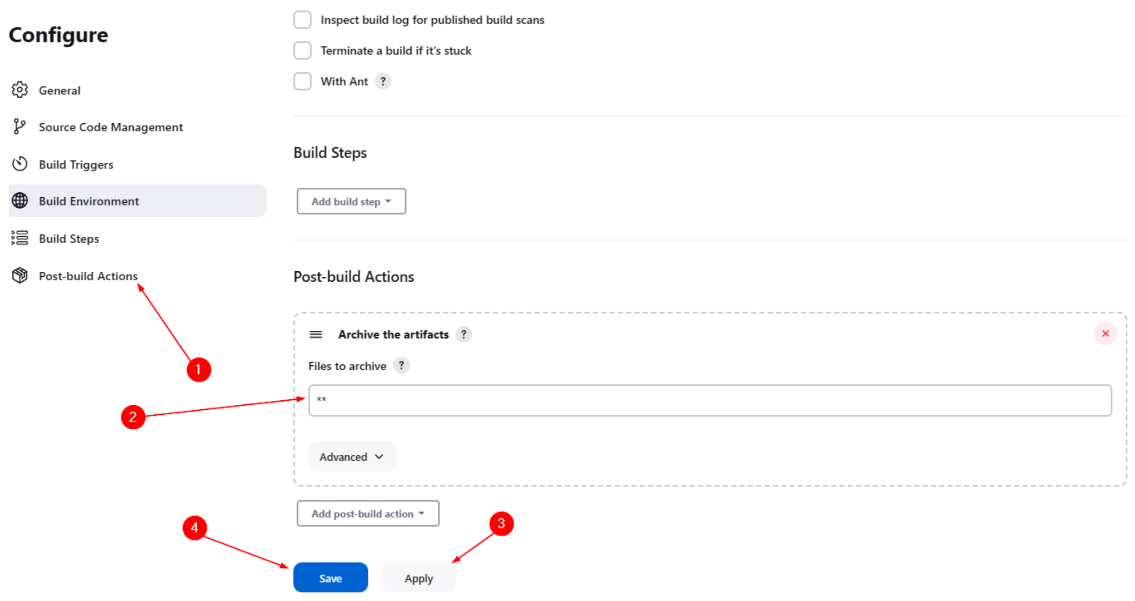
- [ ] Trigger builds remotely (e.g., from scripts) ?
- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [✓] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s) ?
- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published build scans
- [ ] Terminate a build if it's stuck
- [ ] With Ant ?

Build Steps

Configure "Post-build Actions" to archive all the files – files resulted from a build are called "artifacts".



Now, go ahead and make some change in any file in your GitHub repository (e.g. `README.MD` file) and push the changes to the master branch.

You will see that a new build has been launched automatically (by webhook) and you can see its results – artifacts, saved on Jenkins server.

You have now configured an automated Jenkins job that receives files from GitHub by webhook trigger (this method is considered as ‘push’ because the changes are being ‘pushed’ and files transfer is initiated by GitHub). There are also other methods: trigger one job (downstreadm) from another (upstream), poll GitHub periodically and others.

By default, the artifacts are stored on Jenkins server locally

```
ls /var/lib/jenkins/jobs/job-1/builds/<build_number>/archive
```

CONFIGURE JENKINS TO COPY FILES TO NFS SERVER VIA SSH

Step 3 – Configure Jenkins to copy files to NFS server via SSH

Now we have our artifacts saved locally on Jenkins server, the next step is to copy them to our NFS server to `/mnt/apps` directory.

Jenkins is a highly extendable application and there are 1400+ plugins available. We will need a plugin that is called "Publish Over SSH".

1. Install "Publish Over SSH" plugin.

On main dashboard select "Manage Jenkins" and choose "Manage Plugins" menu item.

On "Available" tab search for "Publish Over SSH" plugin and install it

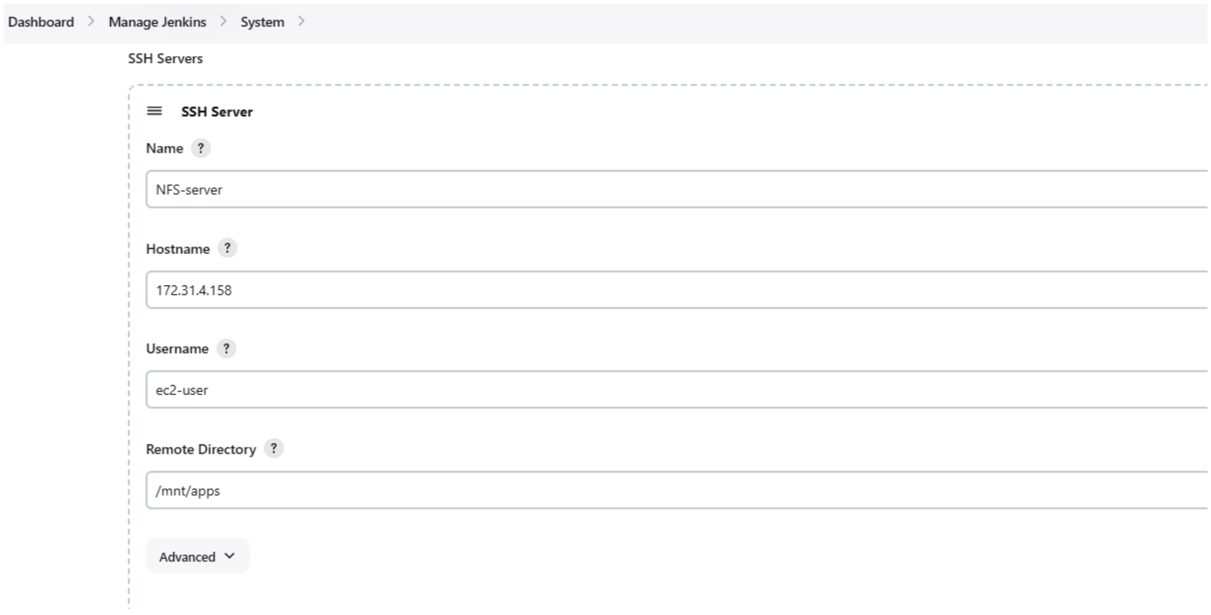
2. Configure the job/project to copy artifacts over to NFS server.

On main dashboard select "Manage Jenkins" and choose "Configure System" menu item.

Scroll down to Publish over SSH plugin configuration section and configure it to be able to connect to your NFS server:

1. Provide a private key (content of .pem file that you use to connect to NFS server via SSH/Putty)
2. Arbitrary name
3. Hostname – can be `private IP address` of your NFS server
4. Username – `ec2-user` (since NFS server is based on EC2 with RHEL 8)
5. Remote directory – `/mnt/apps` since our Web Servers use it as a mointing point to retrieve files from the NFS server

Test the configuration and make sure the connection returns `Success`. Remember, that TCP port 22 on NFS server must be open to receive SSH connections.



Name: NFS-server

Hostname: 172.31.4.158

Username: ec2-user

Remote Directory: /mnt/apps

Save the configuration, open your Jenkins job/project configuration page and add another one "Post-build Action"

Configure it to send all files probuced by the build into our previouslys define remote directory. In our case we want to copy all files and directories – so we use `**`.
If you want to apply some particular pattern to define which files to send – use this syntax.

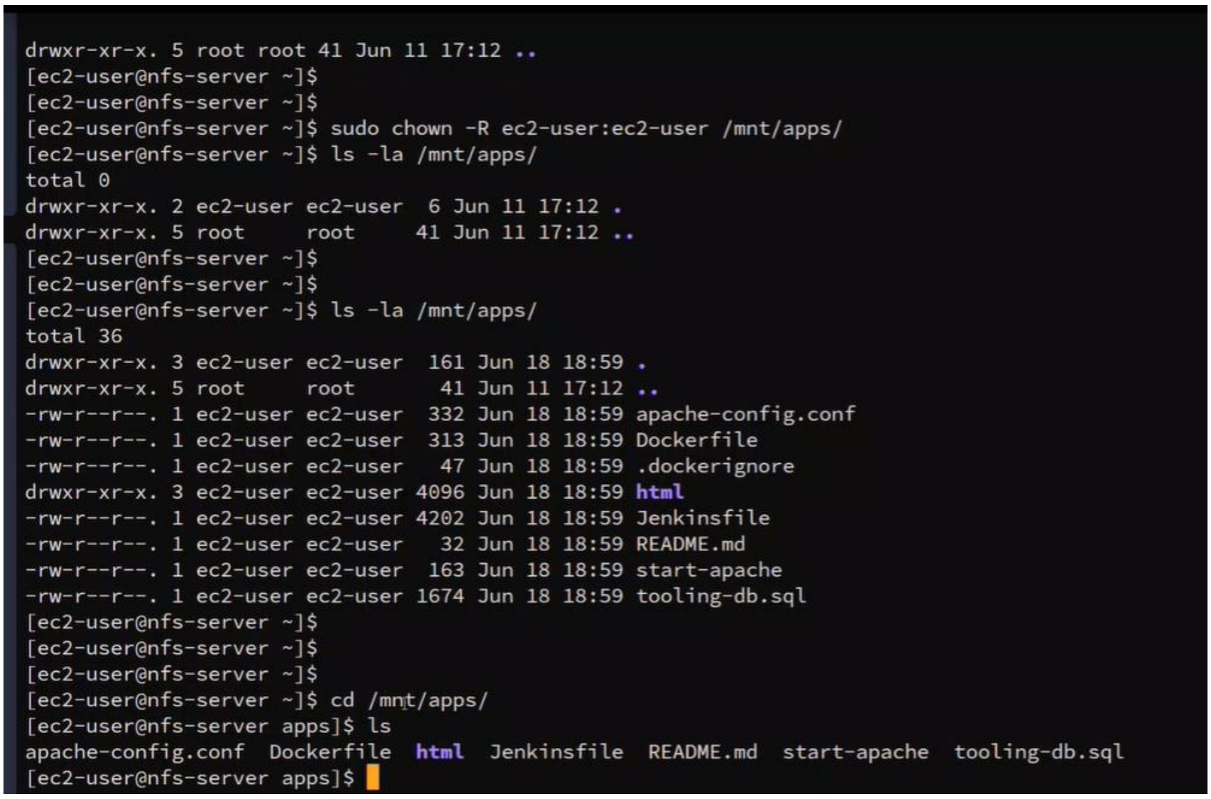
Save this configuration and go ahead, change something in `README.MD` file in your GitHub Tooling repository.

Webhook will trigger a new job and in the "Console Output" of the job you will find something like this:

```
SSH: Transferred 25 file(s)
Finished: SUCCESS
```

To make sure that the files in `/mnt/apps` have been udated – connect via SSH/Putty to your NFS server and check README.MD file

```
cat /mnt/apps/README.md
```



```
drwxr-xr-x. 5 root root 41 Jun 11 17:12 ..
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$ sudo chown -R ec2-user:ec2-user /mnt/apps/
[ec2-user@nfs-server ~]$ ls -la /mnt/apps/
total 0
drwxr-xr-x. 2 ec2-user ec2-user  6 Jun 11 17:12 .
drwxr-xr-x. 5 root     root     41 Jun 11 17:12 ..
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$ ls -la /mnt/apps/
total 36
drwxr-xr-x. 3 ec2-user ec2-user  161 Jun 18 18:59 .
drwxr-xr-x. 5 root     root       41 Jun 11 17:12 ..
-rw-r--r--. 1 ec2-user ec2-user  332 Jun 18 18:59 apache-config.conf
-rw-r--r--. 1 ec2-user ec2-user  313 Jun 18 18:59 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user   47 Jun 18 18:59 .dockerignore
drwxr-xr-x. 3 ec2-user ec2-user 4096 Jun 18 18:59 html
-rw-r--r--. 1 ec2-user ec2-user 4202 Jun 18 18:59 Jenkinsfile
-rw-r--r--. 1 ec2-user ec2-user   32 Jun 18 18:59 README.md
-rw-r--r--. 1 ec2-user ec2-user  163 Jun 18 18:59 start-apache
-rw-r--r--. 1 ec2-user ec2-user 1674 Jun 18 18:59 tooling-db.sql
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$
[ec2-user@nfs-server ~]$ cd /mnt/apps/
[ec2-user@nfs-server apps]$ ls
apache-config.conf  Dockerfile  html  Jenkinsfile  README.md  start-apache  tooling-db.sql
[ec2-user@nfs-server apps]$
```

If you see the changes, you had previously made in your GitHub – the job works as expected.

Congratulations!

You have just implemented your first Continuous Integration solution using Jenkins CI. Watch out for advanced CI configurations in upcoming projects.