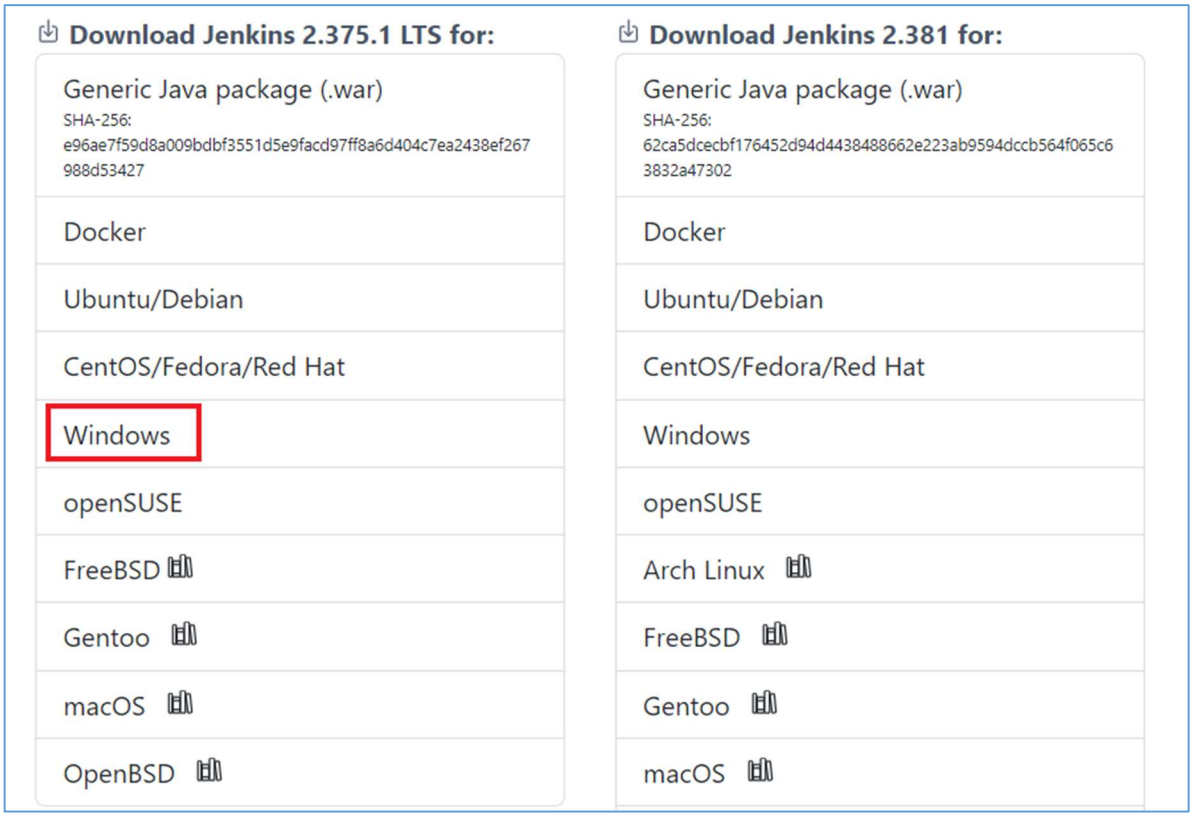
Jenkins

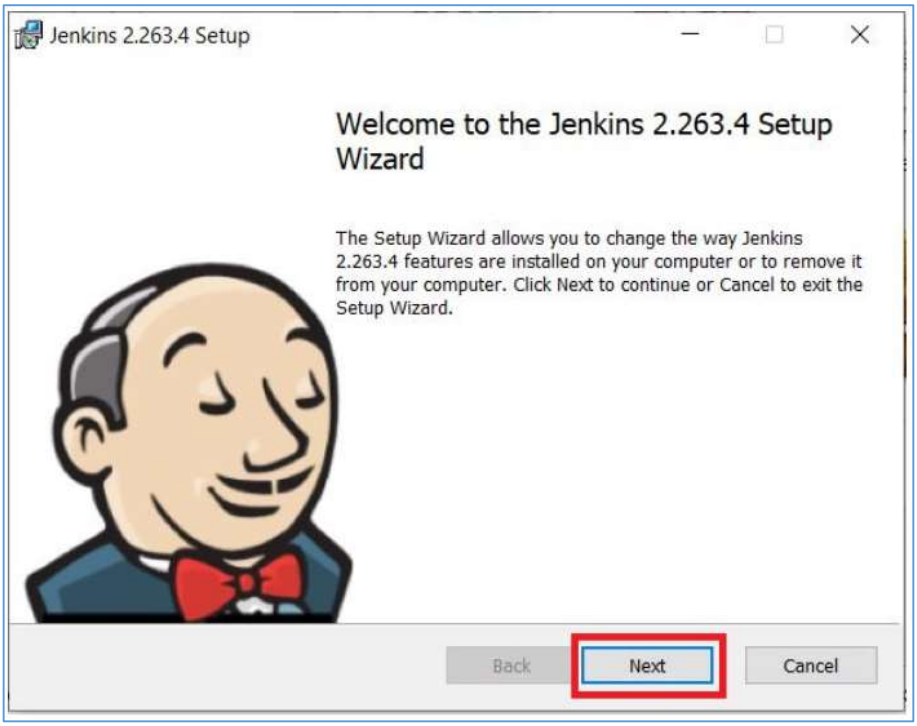
1. Download and install Jenkins on Windows x64, CentOS and Amazon Linux.
2. Managing and installing plugins.
3. First simple job examle in Jenkins.
4. Simple job example in Jenkins with deploy.
5. Simple job example in Jenkins with deploy using plug. “Publish over SSH”.
6. Simple CI/CD pipeline example.
7. Simple CI/CD pipeline example using AWS.
8. Jenkins nodes(agent/~~slave~~).
9. Simple job using Jenkins agent.
10. Simple Jenkis pipeline example.
11. Simple Jenkis pipeline script from SCM (Source Control Managment).
12. Jenkis for Java app using Maven.

1. Download and install Jenkins on Windows x64, CentOS and Amazon Linux.

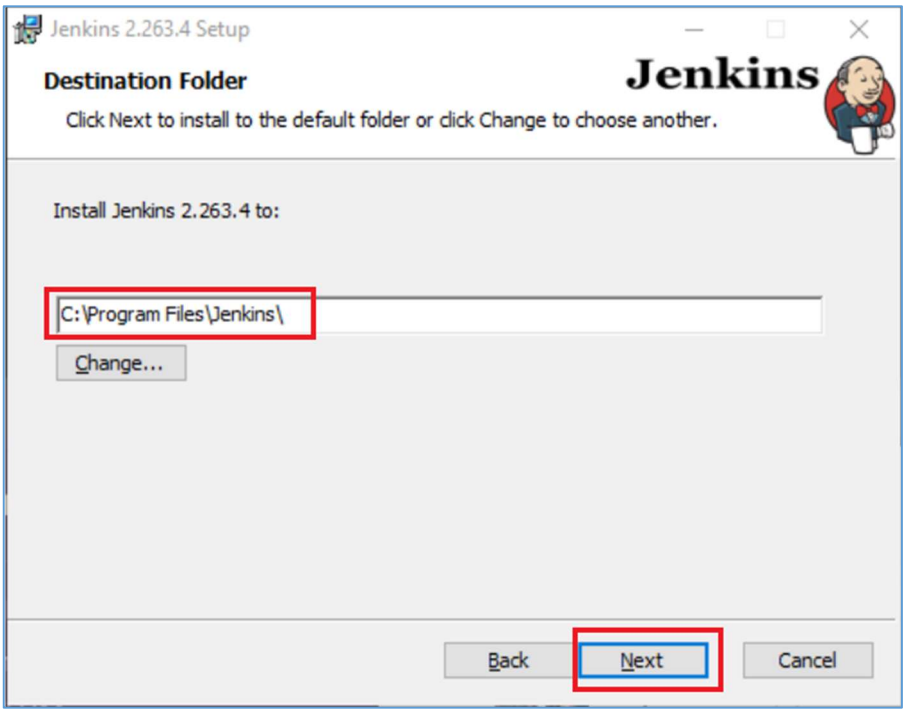
- **Windows**

a. Download newest version Jenkins from website and run installer





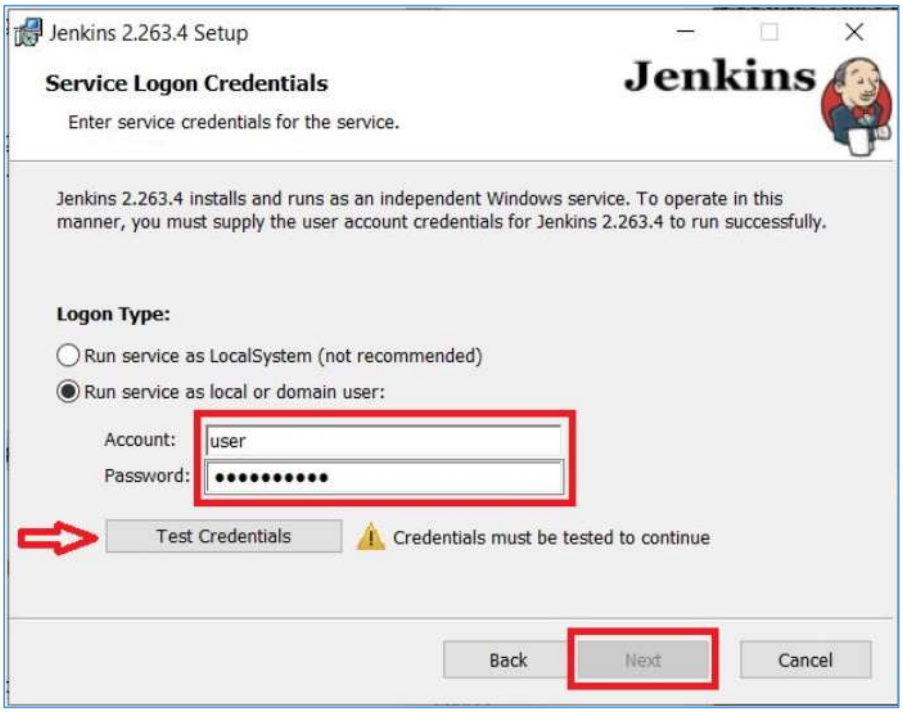
b. Select destination folder



c. Service logon credentials

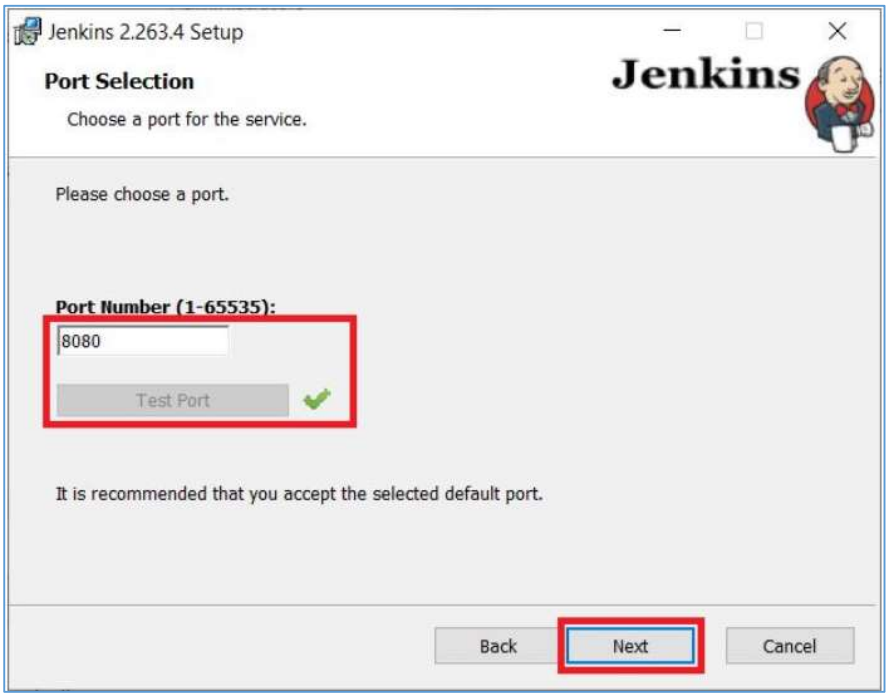
When Installing Jenkins, it is recommended to install and run Jenkins as an independent windows service using a **local** or **domain user** as it is much safer than running Jenkins using **LocalSystem(Windows equivalent of root)** which will grant Jenkins full access to your machine and services.

To run Jenkins service using a local or domain user, specify the domain user name and password with which you want to run Jenkins, click on Test Credentials to test your domain credentials and click on Next.



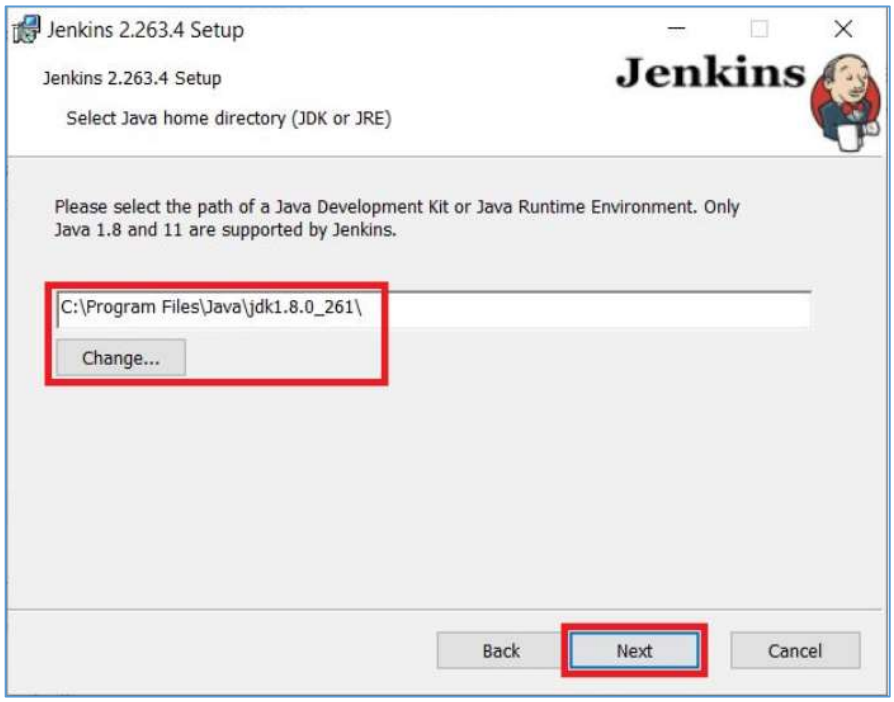
d. Port selection

Specify the port on which Jenkins will be running, Test Port button to validate whether the specified port if free on your machine or not. Consequently, if the port is free, it will show a green tick mark as shown below, then click on Next.



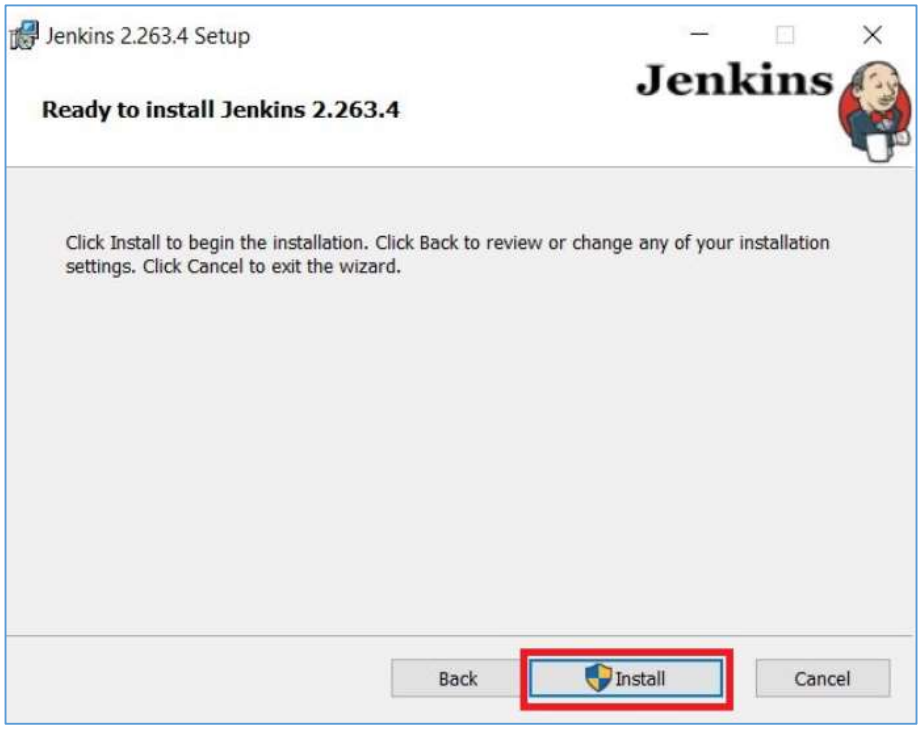
e. Select Java home directory

The installation process checks for Java on your machine and prefills the dialog with the Java home directory. If the needed Java version is not installed on your machine, you will be prompted to install it.



f. Install Jenkins

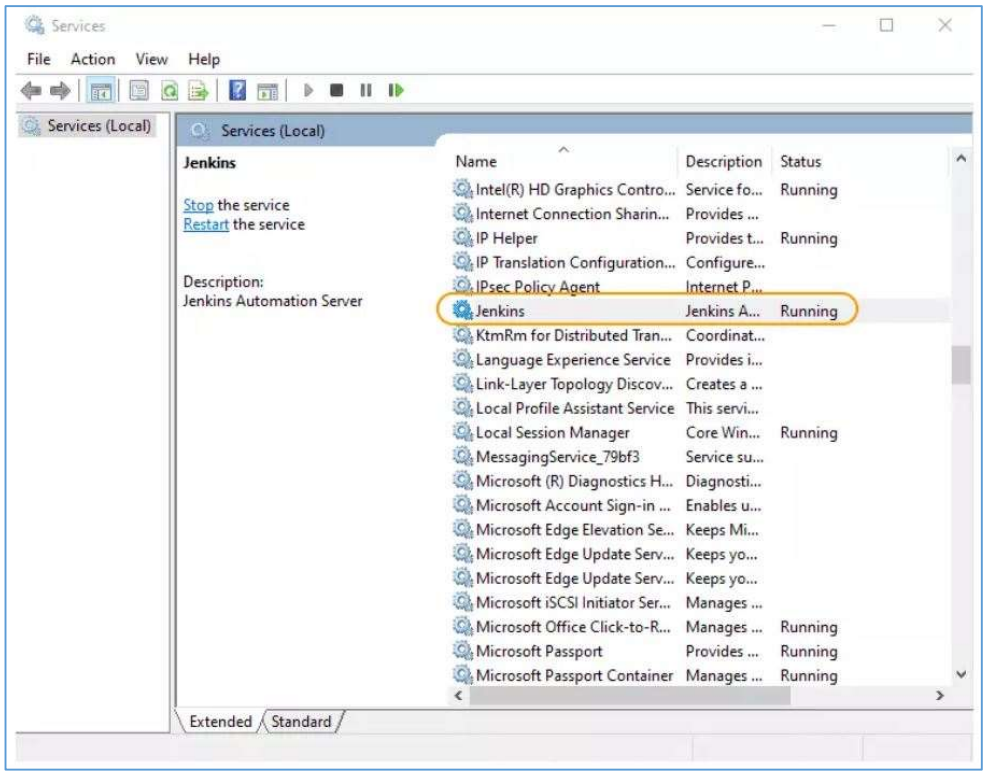
Click on the Install button to start the installation of Jenkins.



g. Finish Jenkins instalation

Once the installation completes, click on Finish to complete the installation.

Jenkins will be installed as a Windows Service. You can validate this by browsing the services section, as shown below:



- **CentOS 7**

a. Download and install (package manager)

You can install Jenkins through yum on Red Hat Enterprise Linux, CentOS, and other Red Hat based distributions. You need to choose either the Jenkins Long Term Support release or the Jenkins weekly release.

When installing the latest Jenkins version on Centos 7 make sure java 11 is the main alternative among other java versions installed:

```
$ alternatives --config java
```

Long Term Support release:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
  https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install java-11-openjdk
sudo yum install jenkins
sudo systemctl daemon-reload
```

Weekly release:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
  https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo yum upgrade
# Add required dependencies for the jenkins package
sudo yum install java-11-openjdk
sudo yum install jenkins
```

b. Start Jenkins

You can enable the Jenkins service to start at boot with the command:

```
sudo systemctl enable jenkins
```

You can start the Jenkins service with the command:

```
sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

```
sudo systemctl status jenkins
```

```
[vagrant@vm2 ~]$
[vagrant@vm2 ~]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-12-11 11:53:36 UTC; 57min ago
 Main PID: 653 (java)
   CGroup: /system.slice/jenkins.service
           └─653 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=%C/jenk...

Dec 11 12:36:18 vm2 jenkins[653]: 2022-12-11 12:36:18.090+0000 [id=339]        INFO        c.n.j.p.d....check
Dec 11 12:36:18 vm2 jenkins[653]: 2022-12-11 12:36:18.090+0000 [id=339]        INFO        c.n.j.p.d....leted
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.091+0000 [id=372]        INFO        c.n.j.p.d....gered
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.091+0000 [id=372]        INFO        c.n.j.p.d.Docke...
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.092+0000 [id=372]        INFO        c.n.j.p.d....check
Dec 11 12:41:18 vm2 jenkins[653]: 2022-12-11 12:41:18.092+0000 [id=372]        INFO        c.n.j.p.d....leted
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.092+0000 [id=394]        INFO        c.n.j.p.d....gered
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]        INFO        c.n.j.p.d.Docke...
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]        INFO        c.n.j.p.d....check
Dec 11 12:46:18 vm2 jenkins[653]: 2022-12-11 12:46:18.093+0000 [id=394]        INFO        c.n.j.p.d....leted
Hint: Some lines were ellipsized, use -l to show in full.
[vagrant@vm2 ~]$
```

* If you have a firewall installed, you must add Jenkins as an exception. You must change YOURPORT in the script below to the port you want to use. Port 8080 is the most common.

```
YOURPORT=8080
PERM="--permanent"
SERV="$PERM --service=jenkins"

firewall-cmd $PERM --new-service=jenkins
firewall-cmd $SERV --set-short="Jenkins ports"
firewall-cmd $SERV --set-description="Jenkins port exceptions"
firewall-cmd $SERV --add-port=$YOURPORT/tcp
firewall-cmd $PERM --add-service=jenkins
firewall-cmd --zone=public --add-service=http --permanent
firewall-cmd --reload
```

- **Amazon Linux**

a. Prerequisites

An AWS account. If you don't have one, you can register here.
An Amazon EC2 key pair. If you don't have one, refer to Creating a key pair.
An AWS IAM User with programmatic key access and permissions to launch EC2 instances

b. Creating a key pair

c. Creating security group

d. Launching an Amazon EC2 instance

e. Connecting to your Linux instance

f. Installing and configuring Jenkins

Completing the previous steps enables you to download and install Jenkins on AWS. To download and install Jenkins:

1. Ensure that your software packages are up to date on your instance by uing the following command to perform a quick software update:

```
[ec2-user ~]$ sudo yum update –y
```

2. Add the Jenkins repo using the following command:

```
[ec2-user ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

3. Import a key file from Jenkins-CI to enable installation from the package:

```
[ec2-user ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
[ec2-user ~]$ sudo yum upgrade
```

4. Install Java:

```
[ec2-user ~]$ sudo amazon-linux-extras install java-openjdk11 -y
```

5. Install Jenkins:

```
[ec2-user ~]$ sudo yum install jenkins -y
```

6. Enable the Jenkins service to start at boot:

```
[ec2-user ~]$ sudo systemctl enable jenkins
```

7. Start Jenkins as a service:

```
[ec2-user ~]$ sudo systemctl start jenkins
```

You can check the status of the Jenkins service using the command:

```
[ec2-user@ip-172-31-14-46 jobs]$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-12-04 12:16:03 UTC; 1 weeks 0 days ago
```

g. [Configuring Jenkins]

Jenkins is now installed and running on your EC2 instance. To configure Jenkins:

1. Connect to http://<your_server_public_DNS>:8080 from your browser. You will be able to access Jenkins through its management interface:

2. As prompted, enter the password found in **/var/lib/jenkins/secrets/initialAdminPassword**.
   Use the following command to display this password:
   [ec2-user ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

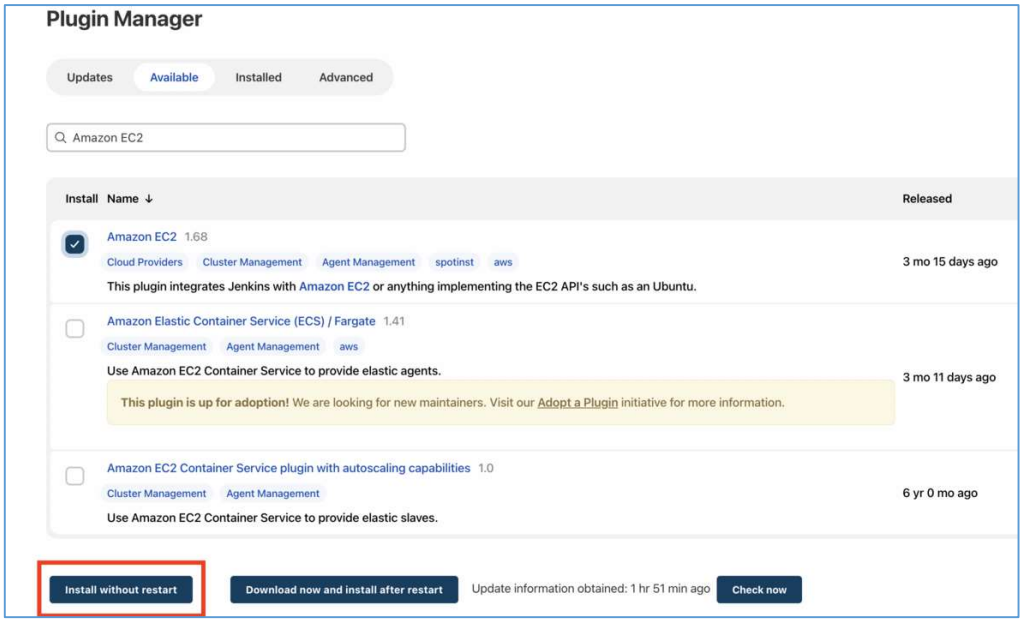
3. The Jenkins installation script directs you to the **Customize Jenkins page**. Click **Install suggested plugins**.

4. Once the installation is complete, the **Create First Admin User** will open. Enter your information, and then select **Save and Continue**.

5. On the left-hand side, select **Manage Jenkins**, and then select **Manage Plugins**.

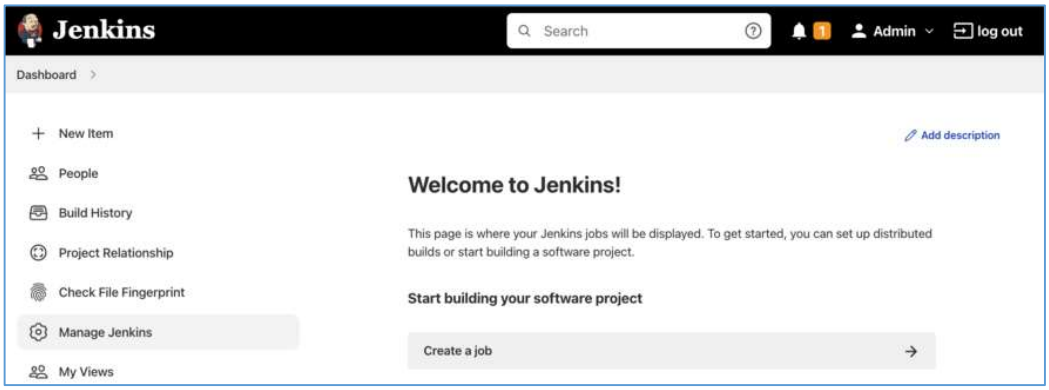
6. Select the **Available** tab, and then enter **Amazon EC2 plugin** at the top right.

7. Select the checkbox next to **Amazon EC2 plugin**, and then select **Install without restart**.

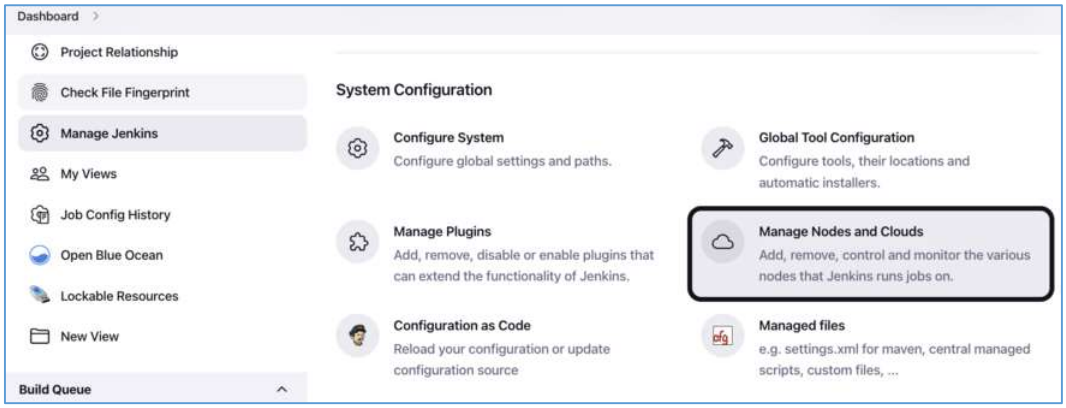


8. Once the installation is done, select **Back to Dashboard**.

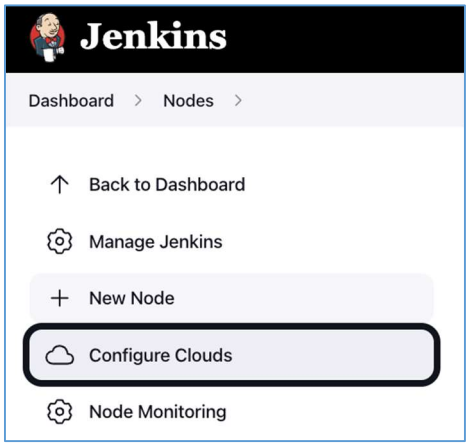
9. Select **Configure a cloud** if there are no existing nodes or clouds. If you already have other nodes or clouds set up, select **Manage Jenkins.**



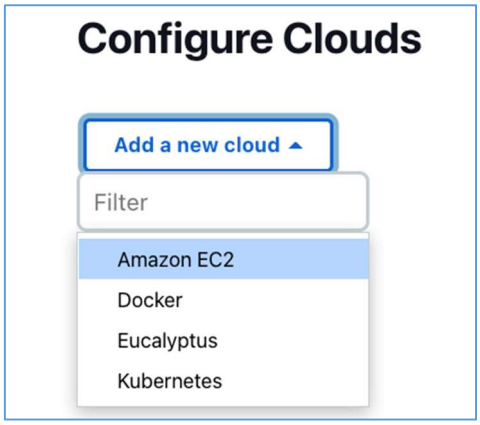
10. After navigating to **Manage Jenkins**, select **Configure Nodes and Clouds** from the left hand side of the page.



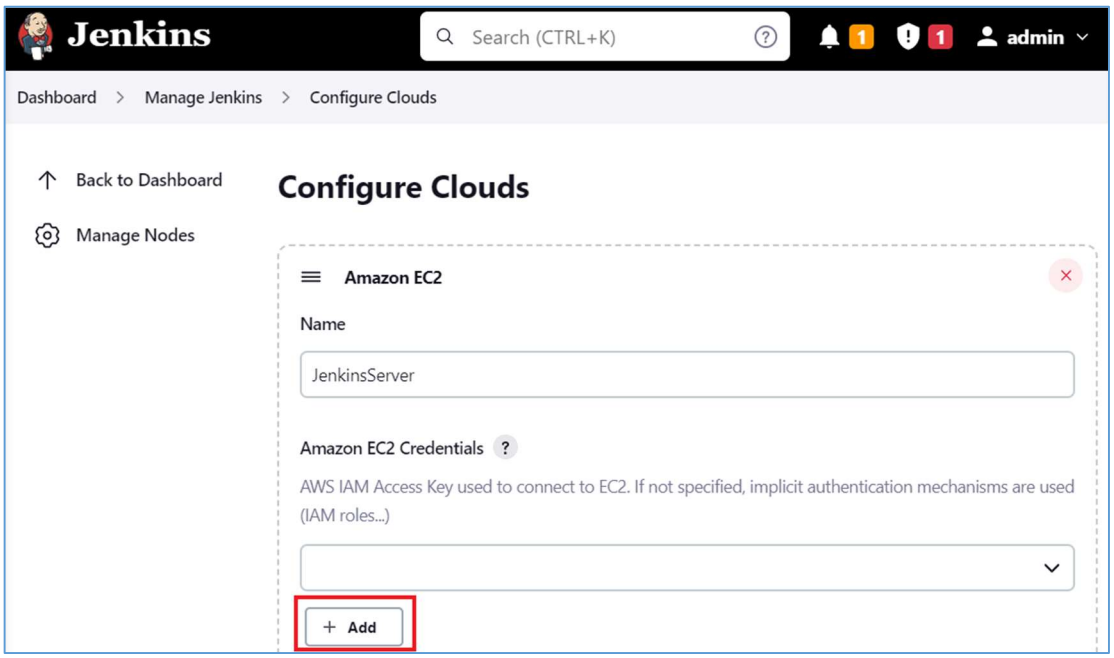
11. From here, select **Configure Clouds**.



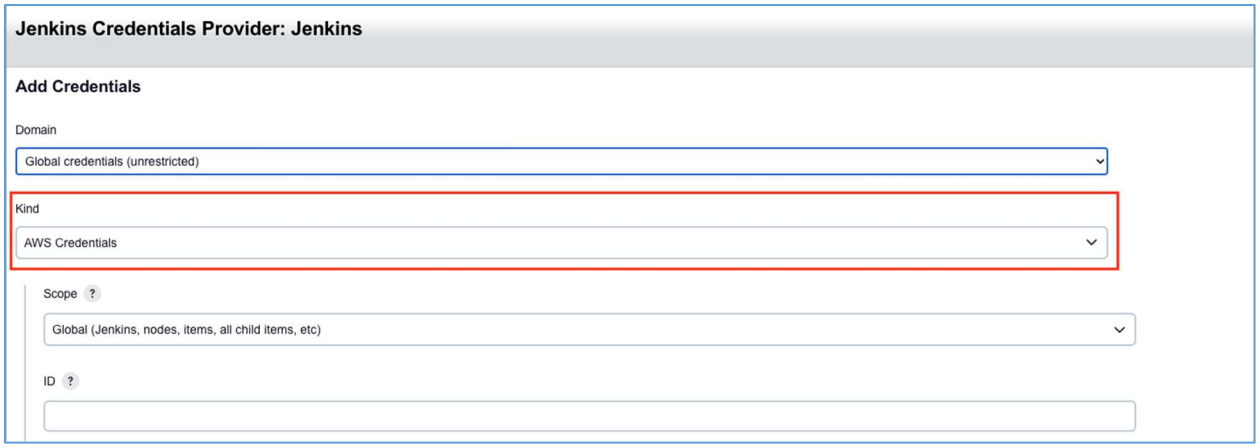
12. Select **Add a new cloud**, and select **Amazon EC2**. A collection of new fields appears.



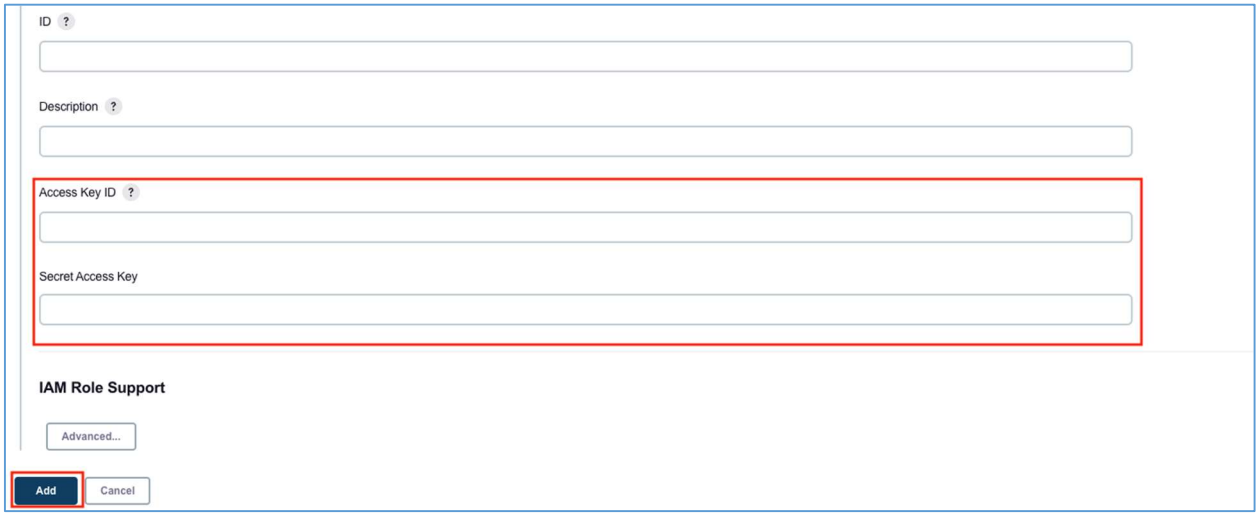
13. Click **Add** under Amazon EC2 Credentials



From **the Jenkins** Credentials Provider, select AWS Credentials as the **Kind**.

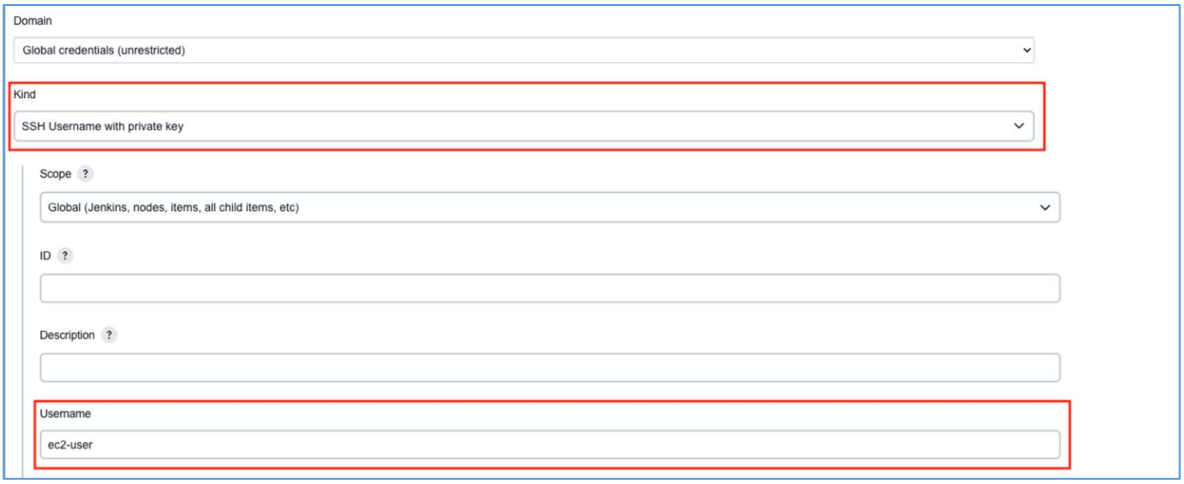


Scroll down and enter in the IAM User programmatic access keys with permissions to launch EC2 instances and select **Add**.

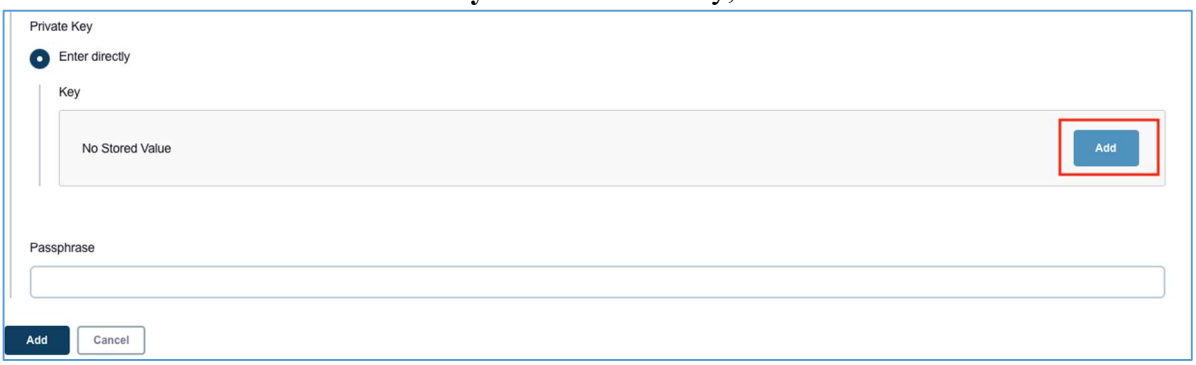


Scroll down to select your region using the drop-down, and select **Add** for the EC2 Key Pair's Private Key.

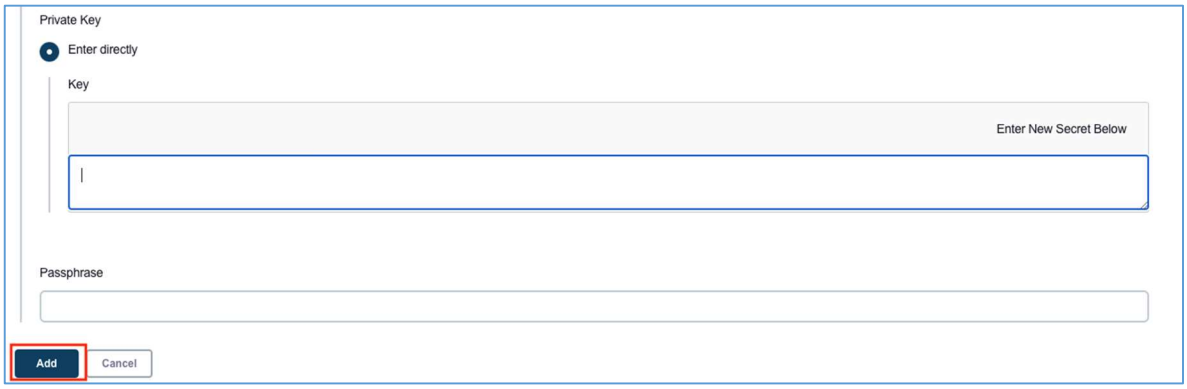
From the Jenkins Credentials Provider, select SSH Username with private key as the Kind and set the Username to ec2-user.



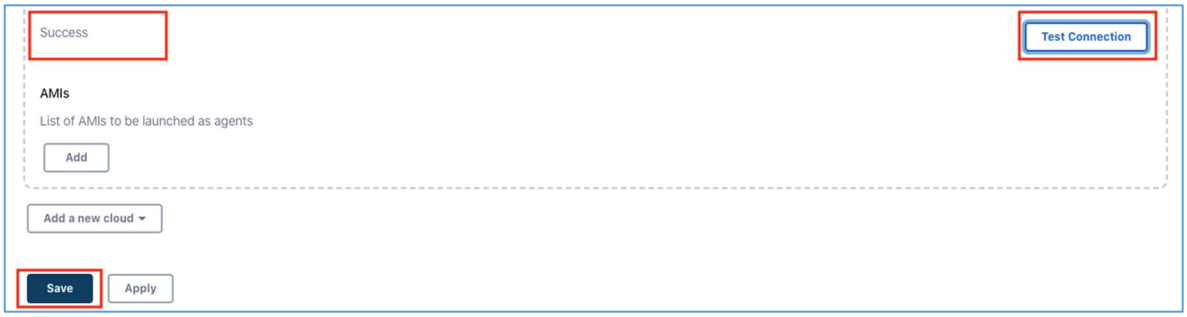
Scroll down and select **Enter Directly** under Private Key, then select **Add**.



Open the private key pair you created in the creating a key pair step and paste in the contents from "-----BEGIN RSA PRIVATE KEY-----" to "-----END RSA PRIVATE KEY-----". Select **Add** when completed.



Scroll down to "Test Connection" and ensure it states "Success". Select **Save** when done

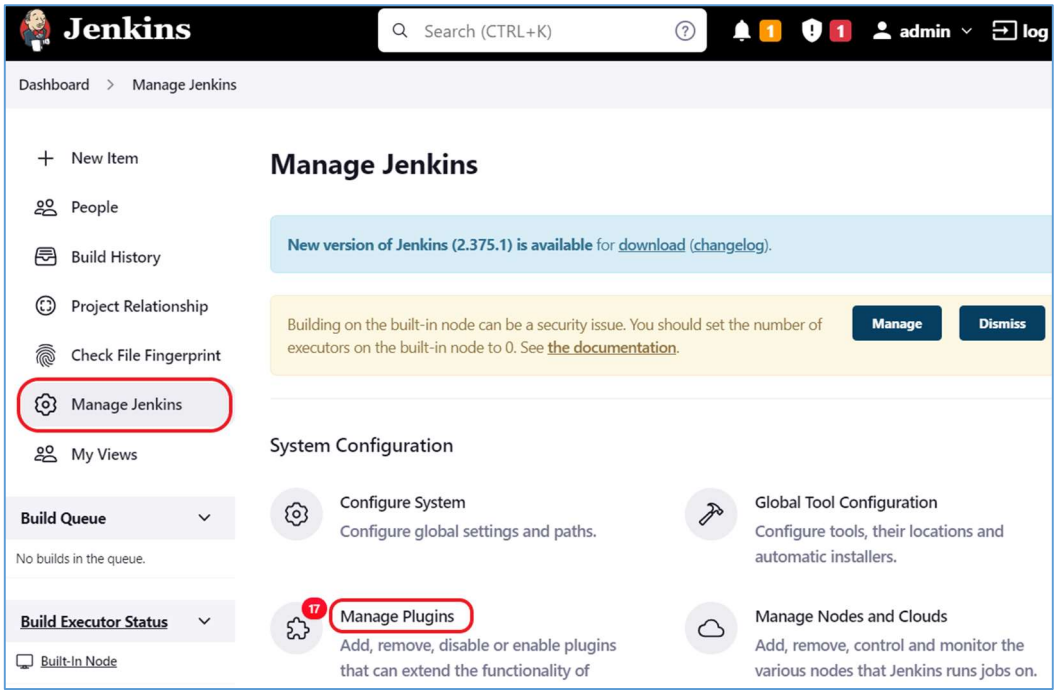


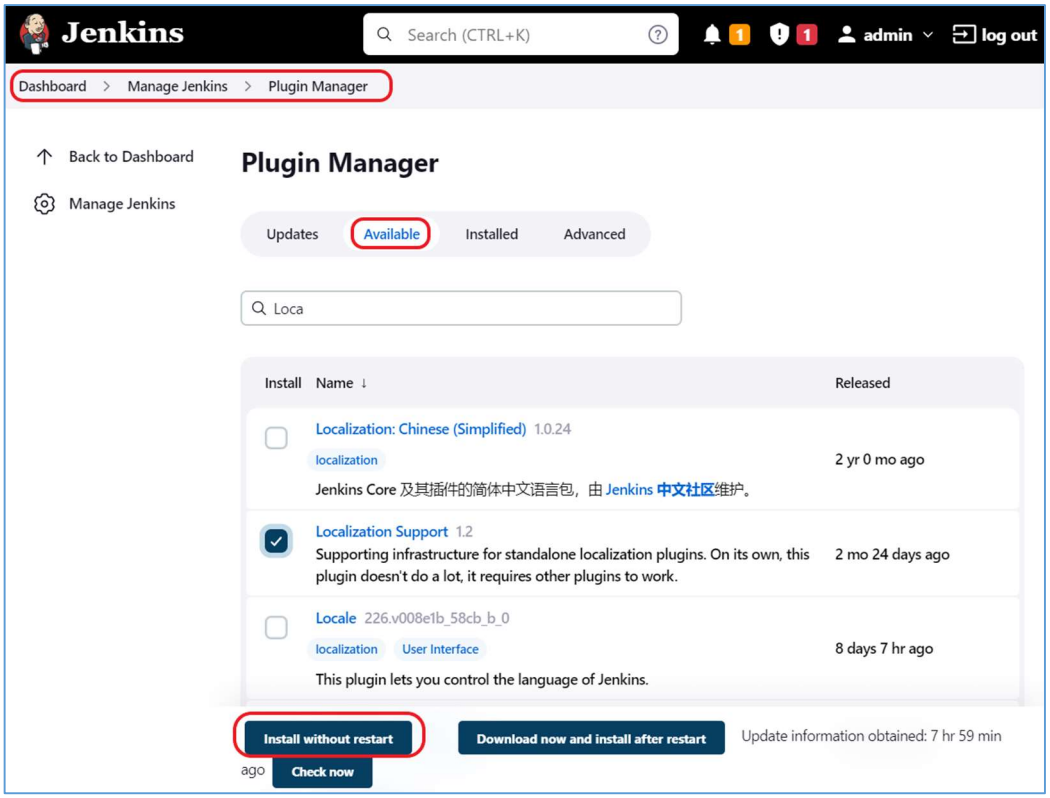
2. Managing and installing plugins

Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization- or user-specific needs. There are over a thousand different plugins which can be installed on a Jenkins controller and to integrate various build tools, cloud providers, analysis tools, and much more.

Jenkins provides two methods for installing plugins on the controller:

a. Using the "**Plugin Manager**" in the web UI.





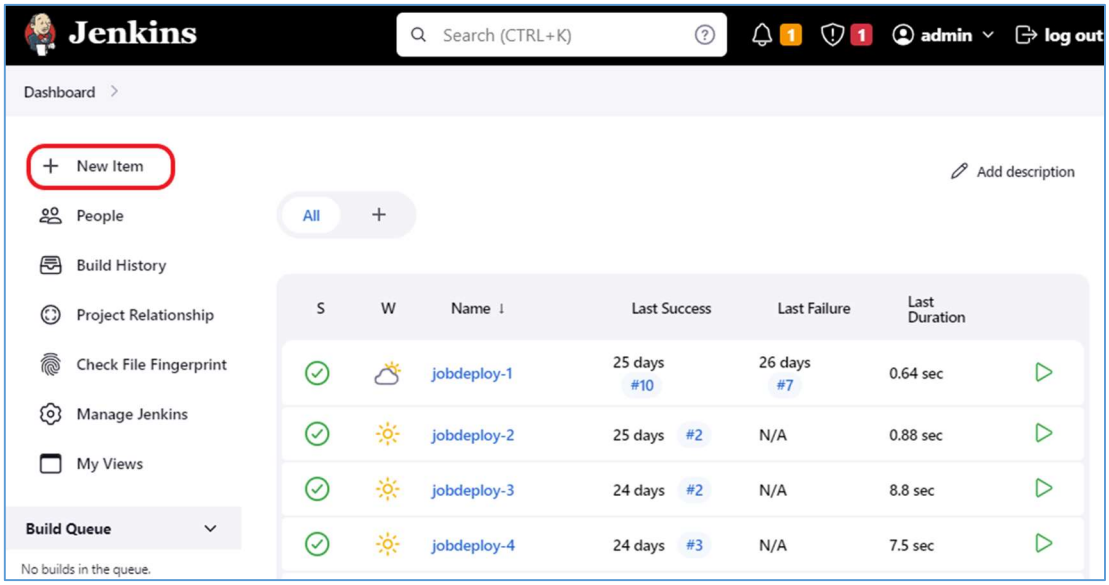
b. Using the Jenkins CLI install-plugin command

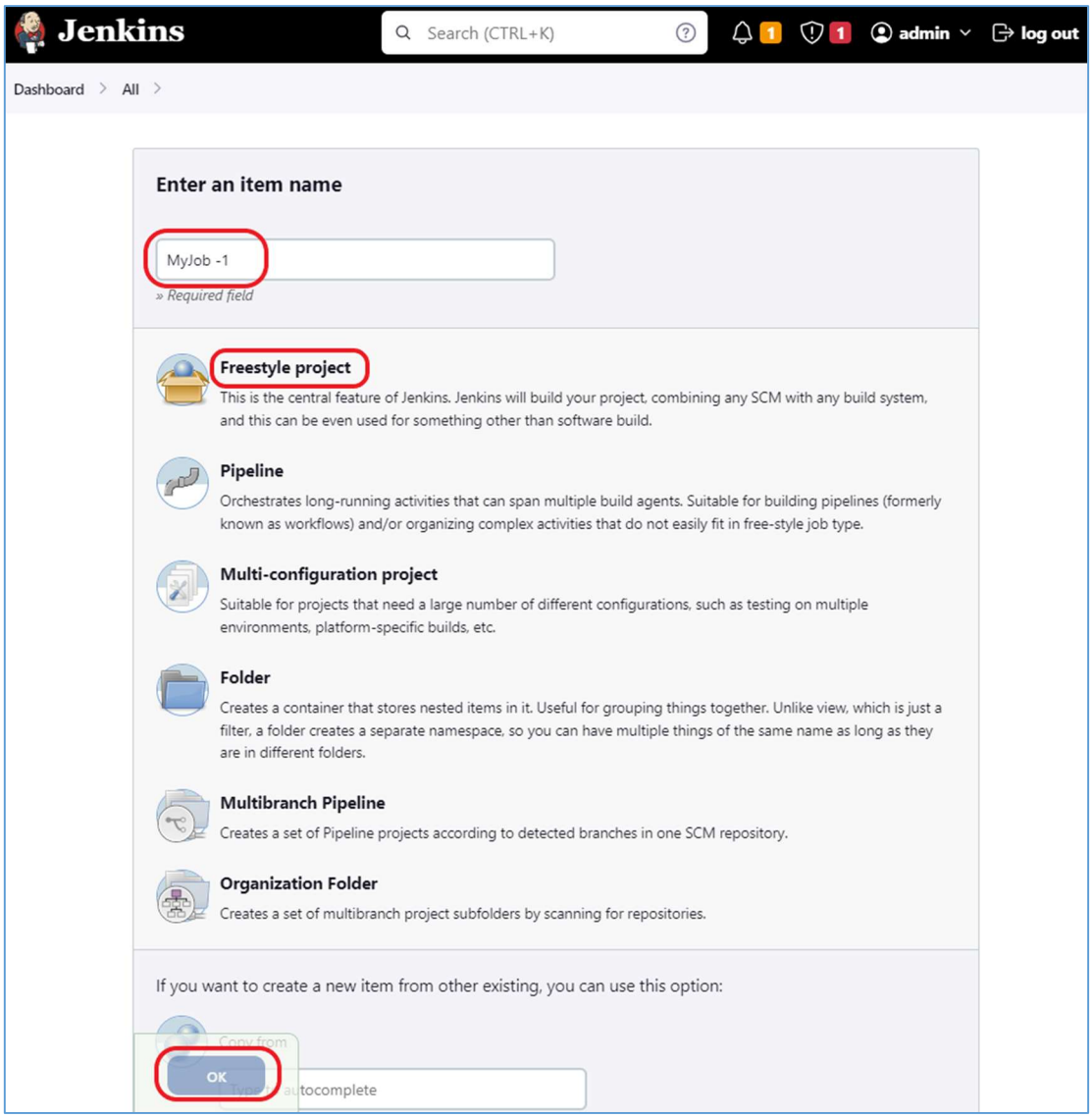
Administrators may also use the Jenkins CLI which provides a command to install plugins. Scripts to manage Jenkins environments, or configuration management code, may need to install plugins without direct user interaction in the web UI. The Jenkins CLI allows a command line user or automation tool to download a plugin and its dependencies.

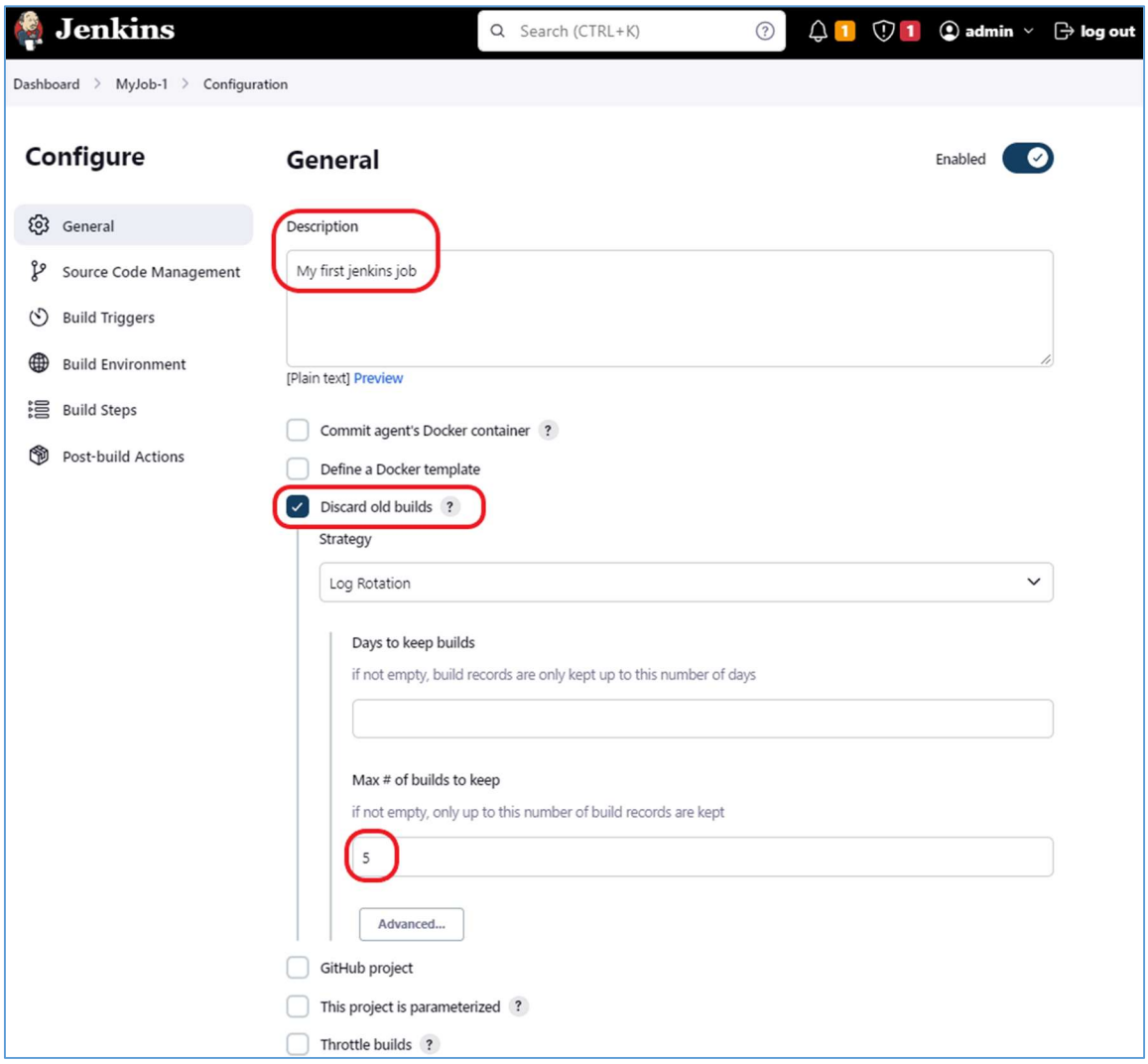
```
java -jar jenkins-cli.jar -s http://localhost:8080/ install-plugin SOURCE ... [-deploy] [-name VAL] [-restart]

Installs a plugin either from a file, an URL, or from update center.

 SOURCE    : If this points to a local file, that file will be installed. If
             this is an URL, Jenkins downloads the URL and installs that as a
             plugin.Otherwise the name is assumed to be the short name of the
             plugin in the existing update center (like "findbugs"),and the
             plugin will be installed from the update center.
-deploy   : Deploy plugins right away without postponing them until the reboot.
-name VAL : If specified, the plugin will be installed as this short name
             (whereas normally the name is inferred from the source name
             automatically).
-restart  : Restart Jenkins upon successful installation.
```

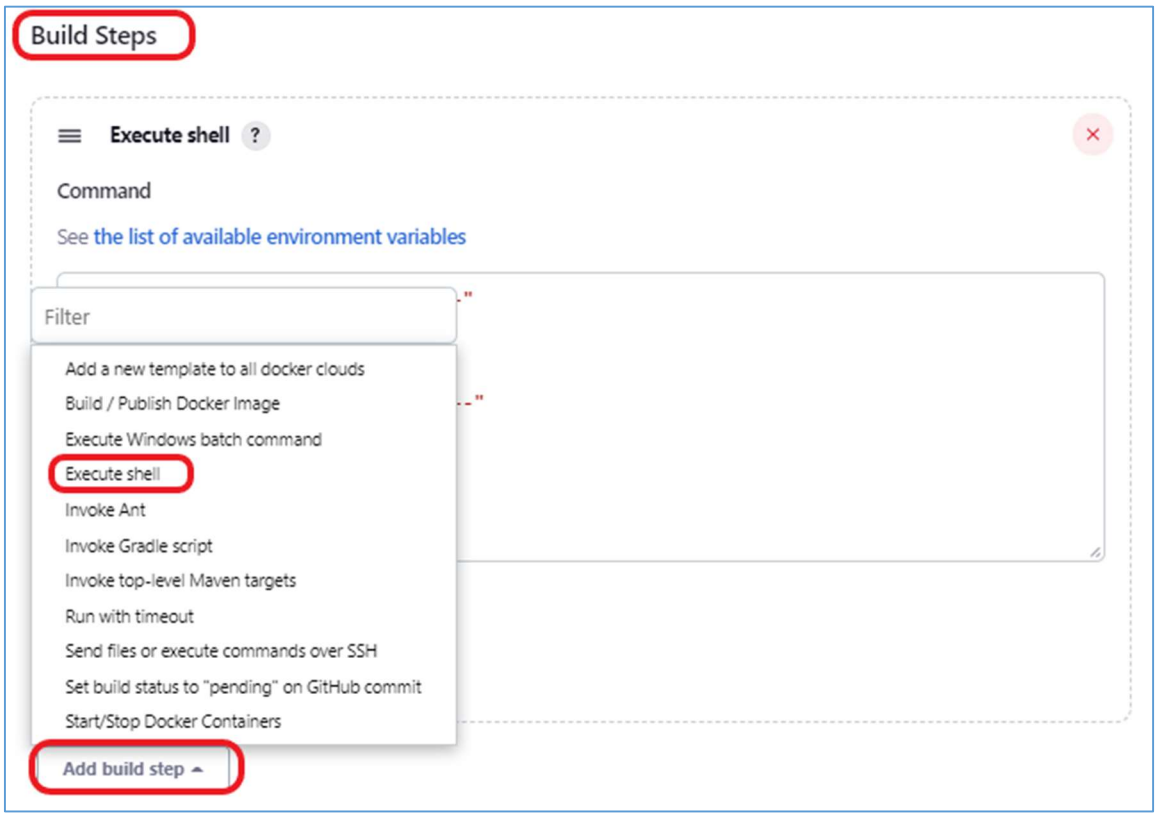
3. First simple job examle in Jenkins.

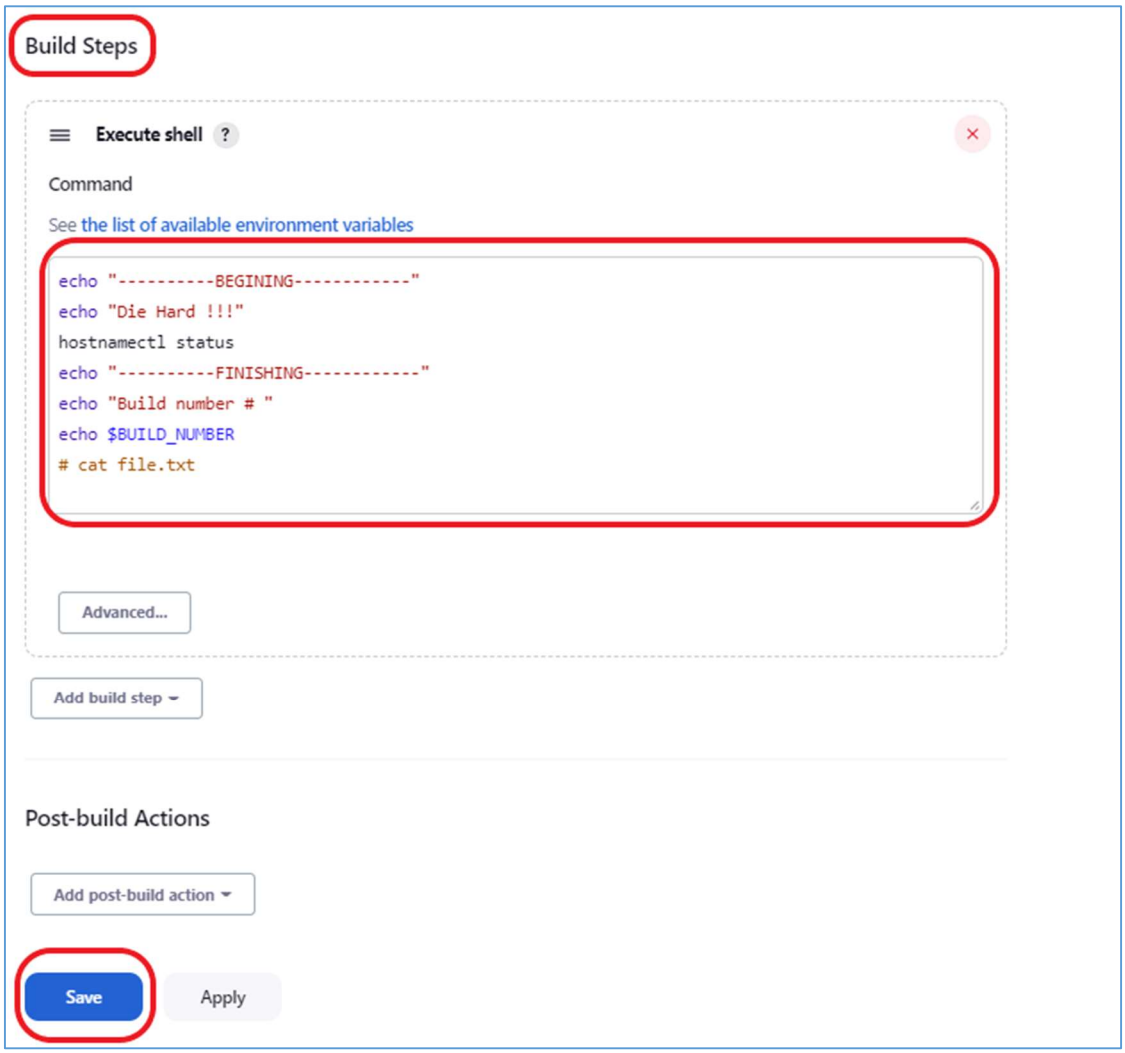
a. Create new job.



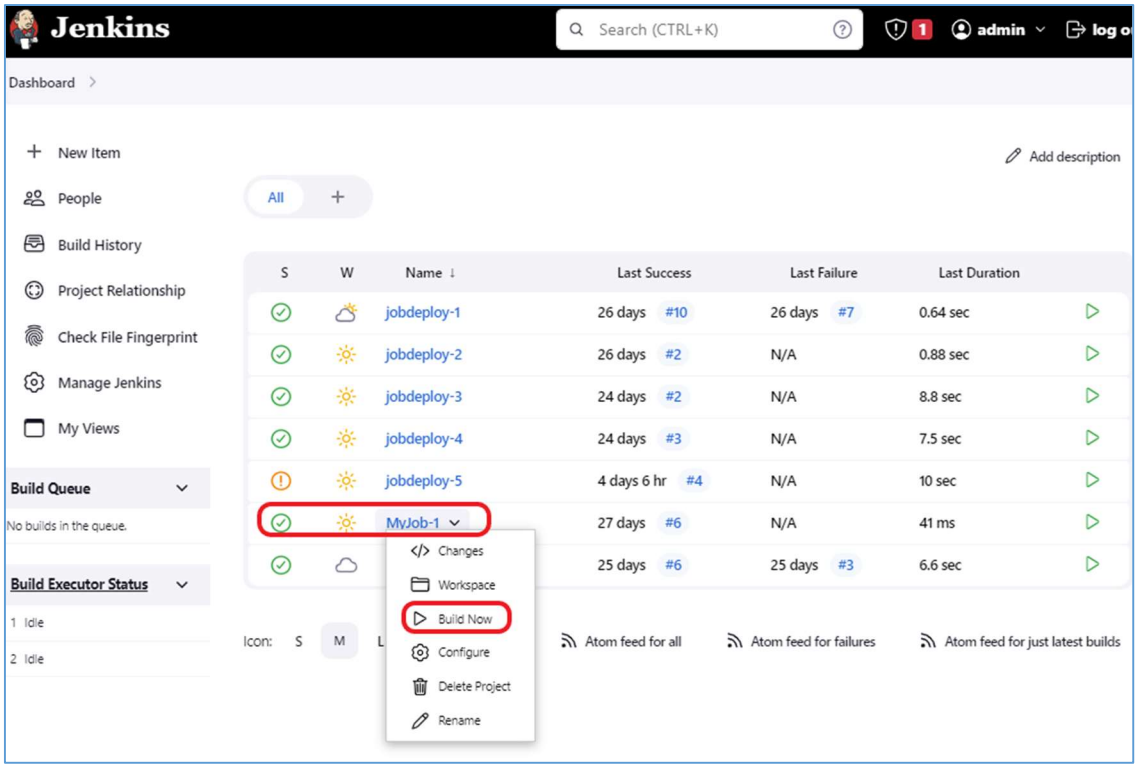


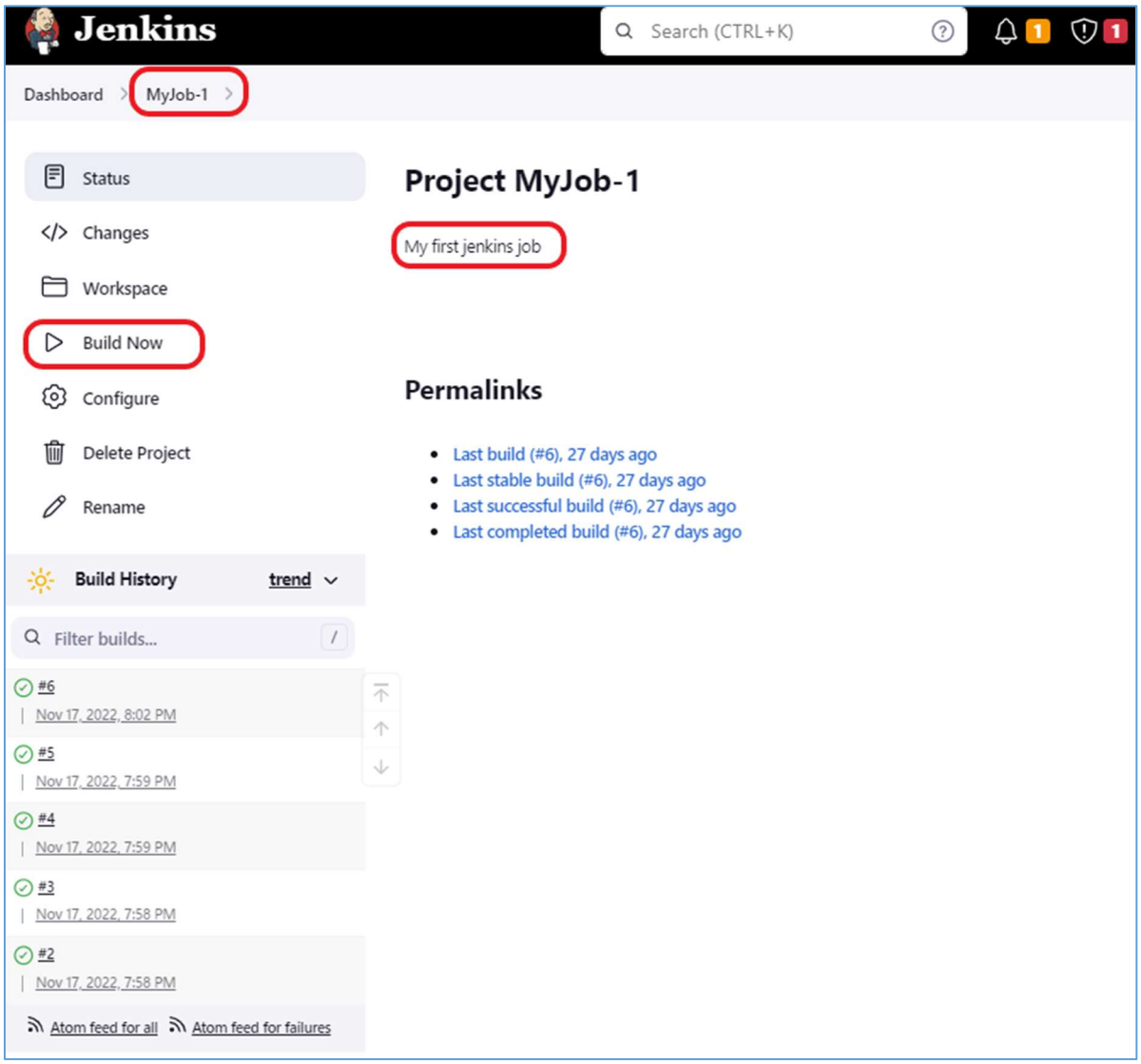

Jenkins
Search (CTRL+K)
admin
log out
Dashboard > MyJob-1 > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
My first jenkins job
[Plain text] Preview
Commit agent's Docker container
Define a Docker template
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
5
Advanced...
GitHub project
This project is parameterized
Throttle builds


Build Steps
Execute shell
Command
See the list of available environment variables
Filter
Add a new template to all docker clouds
Build / Publish Docker Image
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Send files or execute commands over SSH
Set build status to "pending" on GitHub commit
Start/Stop Docker Containers
Add build step

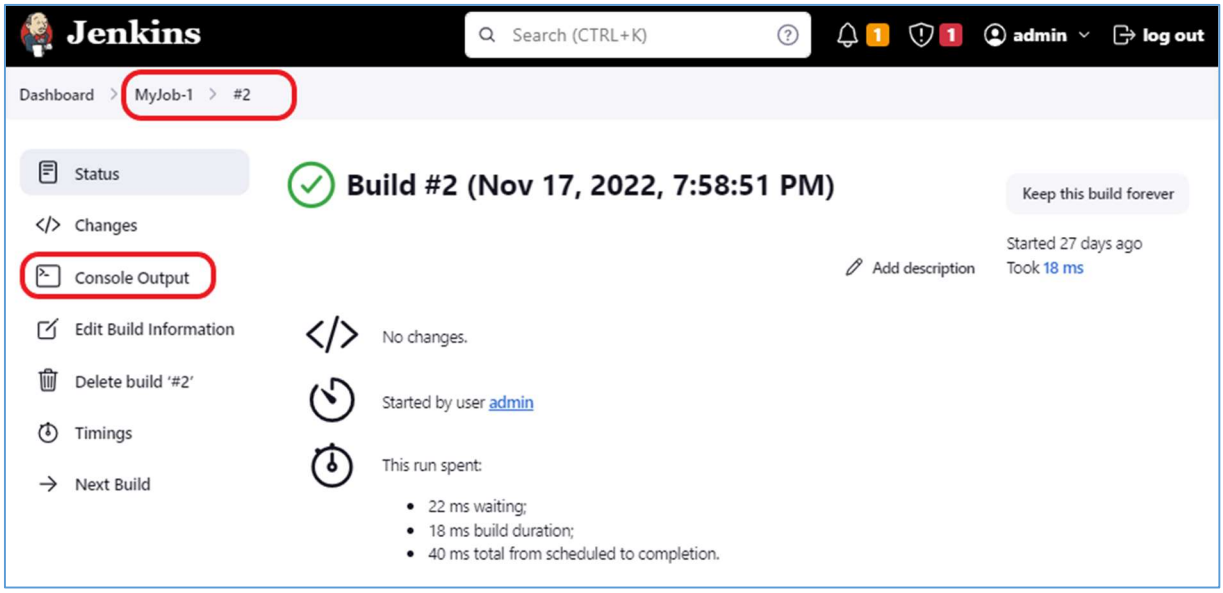

Build Steps
Execute shell
Command
See the list of available environment variables
echo "----------BEGINING----------"
echo "Die Hard !!!"
hostnamectl status
echo "----------FINISHING----------"
echo "Build number # "
echo $BUILD_NUMBER
cat file.txt
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

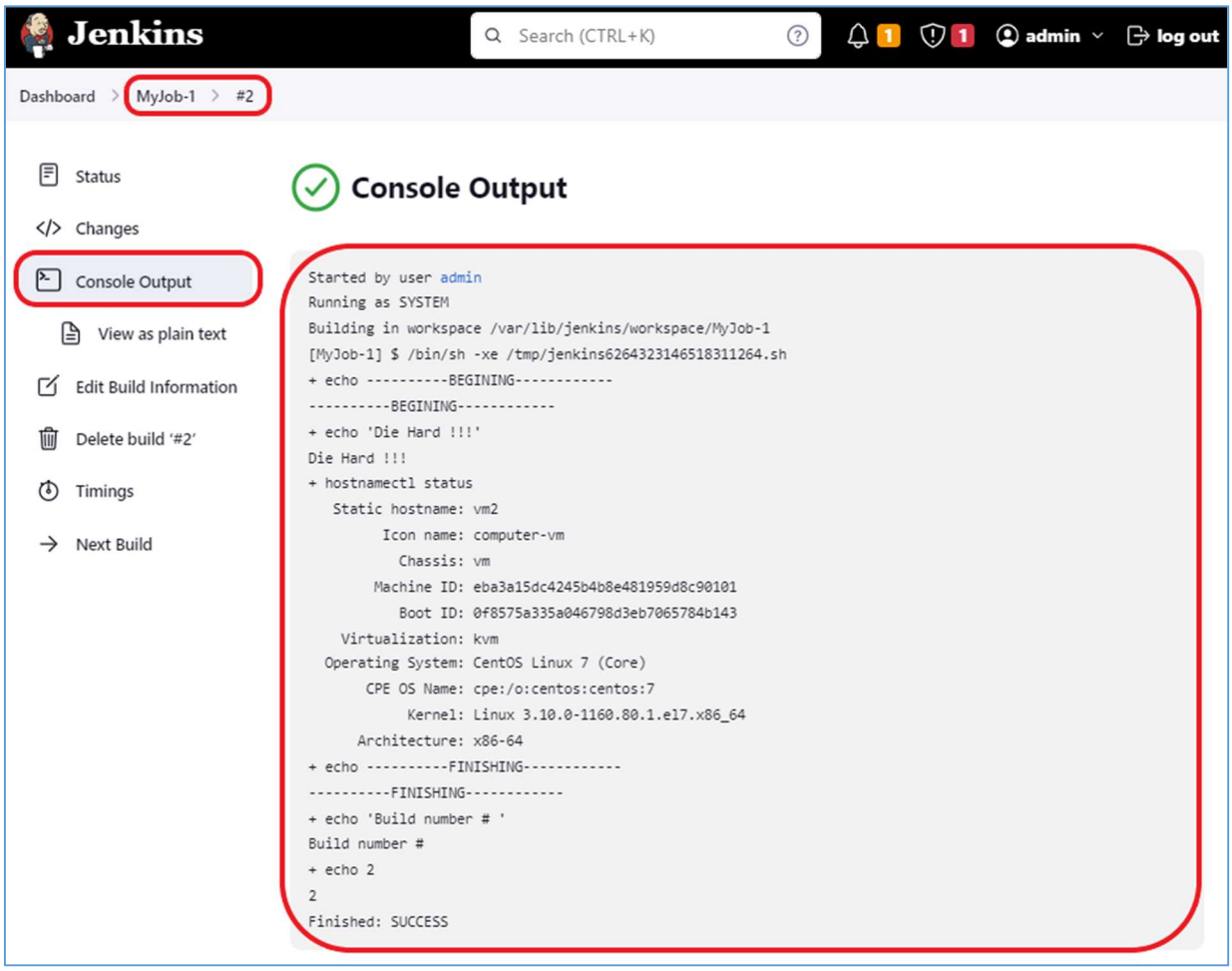
b. Build job.


Jenkins
Search (CTRL+K)
admin
Dashboard
New Item
People
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
jobdeploy-1 26 days #10 26 days #7 0.64 sec
jobdeploy-2 26 days #2 N/A 0.88 sec
jobdeploy-3 24 days #2 N/A 8.8 sec
jobdeploy-4 24 days #3 N/A 7.5 sec
jobdeploy-5 4 days 6 hr #4 N/A 10 sec
MyJob-1 27 days #6 N/A 41 ms
25 days #6 25 days #3 6.6 sec
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Icon: S M
Atom feed for all
Atom feed for failures
Atom feed for just latest builds



c. Console output.





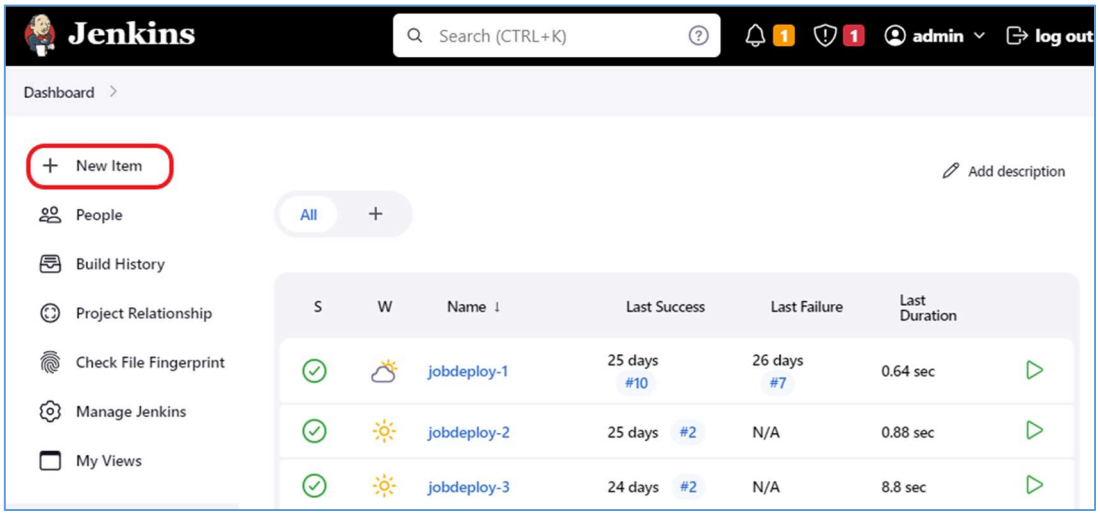

```
Last login: Thu Dec 15 13:32:36 2022 from 10.0.2.2
[vagrant@vm2 ~]$
[vagrant@vm2 ~]$
[vagrant@vm2 ~]$ cd /var/lib/jenkins/workspace/
[vagrant@vm2 workspace]$
[vagrant@vm2 workspace]$
[vagrant@vm2 workspace]$ ll
total 0
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 07:15 jobdeploy-1
drwxr-xr-x. 2 jenkins jenkins 24 Nov 19 14:36 jobdeploy-2
drwxr-xr-x. 3 jenkins jenkins 53 Nov 20 18:04 jobdeploy-3
drwxr-xr-x. 2 jenkins jenkins  6 Nov 21 10:58 jobdeploy-3@tmp
drwxr-xr-x. 3 jenkins jenkins 53 Nov 21 12:20 jobdeploy-4
drwxr-xr-x. 2 jenkins jenkins  6 Nov 21 12:20 jobdeploy-4@tmp
drwxr-xr-x. 3 jenkins jenkins 53 Dec 11 11:58 jobdeploy-5
drwxr-xr-x. 2 jenkins jenkins  6 Dec 11 11:58 jobdeploy-5@tmp
drwxr-xr-x. 2 jenkins jenkins  6 Nov 17 19:58 MyJob-1
drwxr-xr-x. 3 jenkins jenkins 53 Nov 20 17:11 mypipeline_example-1
drwxr-xr-x. 2 jenkins jenkins  6 Nov 20 16:54 mypipeline_example-1@tmp
```

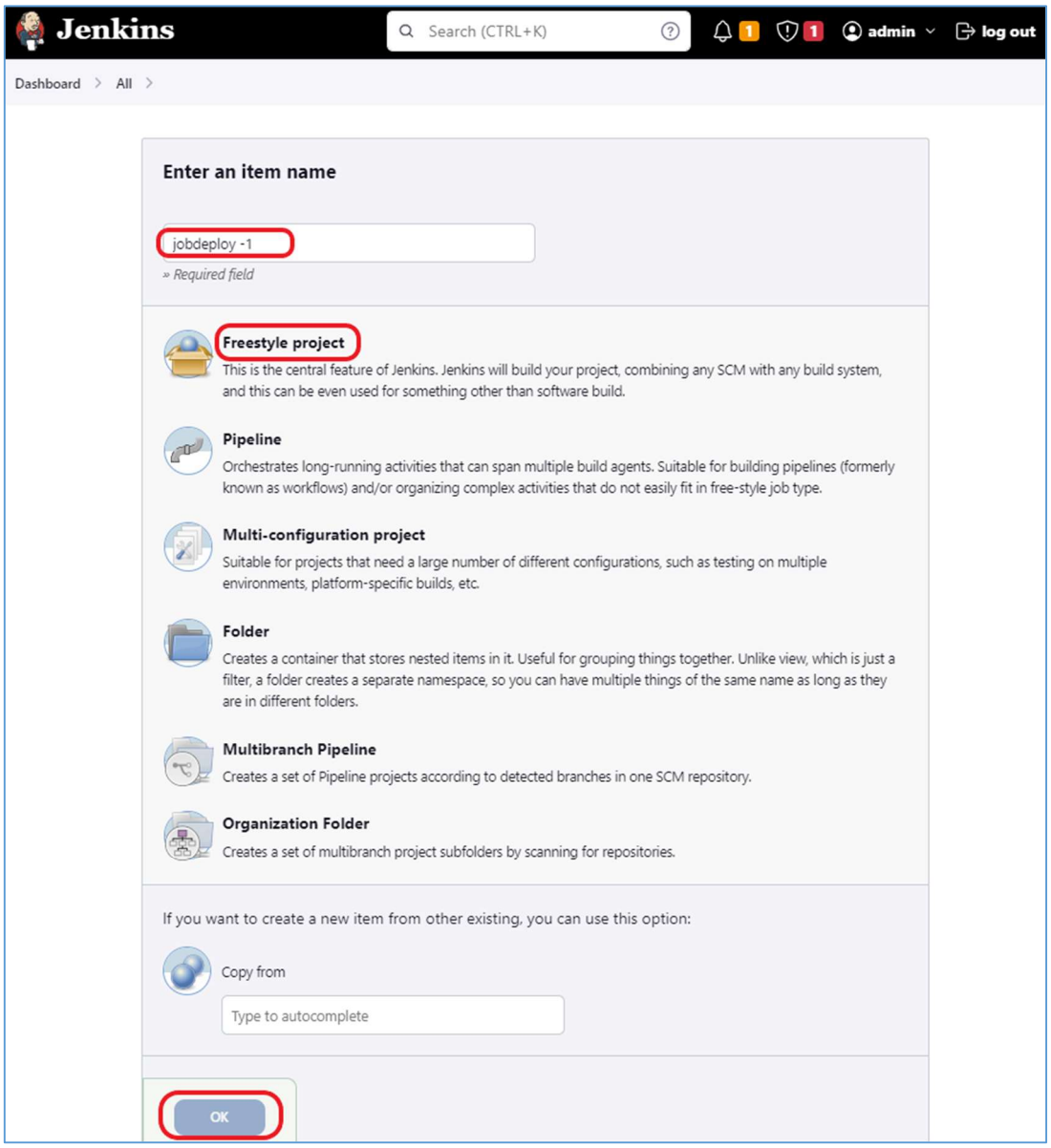
```
[vagrant@vm2 jenkins]$ cd jobs/
[vagrant@vm2 jobs]$
[vagrant@vm2 jobs]$
[vagrant@vm2 jobs]$ ll
total 0
drwxr-xr-x. 3 jenkins jenkins 61 Nov 19 14:03 jobdeploy-1
drwxr-xr-x. 3 jenkins jenkins 61 Nov 19 14:40 jobdeploy-2
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 10:58 jobdeploy-3
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 12:20 jobdeploy-4
drwxr-xr-x. 3 jenkins jenkins 84 Dec 11 11:58 jobdeploy-5
drwxr-xr-x. 3 jenkins jenkins 61 Nov 18 16:39 MyJob-1
drwxr-xr-x. 3 jenkins jenkins 61 Nov 21 10:57 mypipeline_example-1
[vagrant@vm2 jobs]$ cd MyJob-1/
[vagrant@vm2 MyJob-1]$
[vagrant@vm2 MyJob-1]$ ll
total 8
drwxr-xr-x. 7 jenkins jenkins   86 Nov 18 16:56 builds
-rw-r--r--. 1 jenkins jenkins 1202 Nov 18 16:39 config.xml
-rw-r--r--. 1 jenkins jenkins    2 Nov 17 20:02 nextBuildNumber
[vagrant@vm2 MyJob-1]$
[vagrant@vm2 MyJob-1]$ cd builds/
[vagrant@vm2 builds]$
[vagrant@vm2 builds]$ ll
total 4
drwxr-xr-x. 2 jenkins jenkins  55 Nov 17 19:58 2
drwxr-xr-x. 2 jenkins jenkins  55 Nov 17 19:58 3
drwxr-xr-x. 2 jenkins jenkins  55 Nov 17 19:59 4
drwxr-xr-x. 2 jenkins jenkins  55 Nov 17 19:59 5
drwxr-xr-x. 2 jenkins jenkins  55 Nov 17 20:02 6
-rw-r--r--. 1 jenkins jenkins   0 Nov 17 19:54 legacyIds
-rw-r--r--. 1 jenkins jenkins 126 Nov 17 20:02 permalinks
[vagrant@vm2 builds]$
```

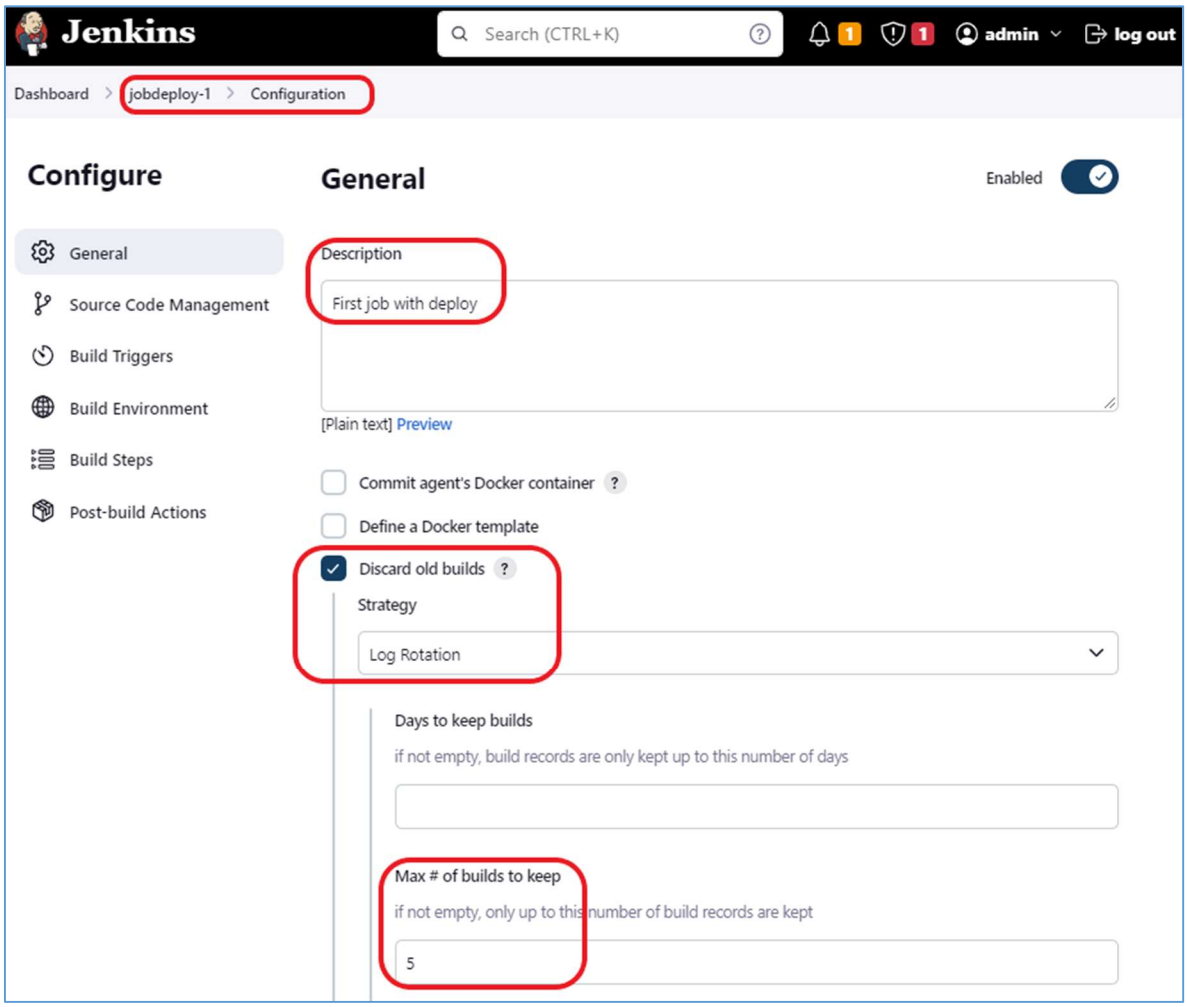
```
[vagrant@vm2 builds]$ cd 2/
[vagrant@vm2 2]$
[vagrant@vm2 2]$ ll
total 12
-rw-r--r--. 1 jenkins jenkins 1220 Nov 17 19:58 build.xml
-rw-r--r--. 1 jenkins jenkins    6 Nov 17 19:58 changelog.xml
-rw-r--r--. 1 jenkins jenkins 1064 Nov 17 19:58 log
[vagrant@vm2 2]$
```

4. Simple job example in Jenkins with deploy.

a. Create new job.







b. Establish **SSH** connection with or without login/password.

- Generate ssh key-pair

```
$ ssh-keygen -t rsa -f "key_name"
```

- Copy public key to the remote host where you will deploy app

```
$ ssh-copy-id "user_name@host_name"
```

- We have to use **`-o StrictHostKeyCheking=no`** to deploy index.html to the remote host without login/password

```
$ scp -v -o StrictHostKeyChecking=no index.html "user_name@host_name":/var/www/html
```

instead of preconfigured fingerprint:

```
$ scp -v index.html "user_name@host_name":/var/www/html
```

- We have to copy file **"id_rsa"**(chmod 400) to **/var/lib/jenkins/.ssh**

- The **/var/lib/jenkins/.ssh** directory and files inside of it should be owned by **jenkins**

c. Install **Apache server** and establish privileges for **/var/www**.

- Install **Apache** on remote host where you will deploy app **(**example Ubuntu 20.04**)**

```
$ sudo apt update
$ sudo apt install apache2
$ sudo ufw allow 'Apache'
$ sudo ufw status
$ sudo sysstemctl status apache2
```

- Establish privileges for **/var/www**

```
vagrant@vm1:/var/www/html$ sudo su
root@vm1:/var/www/html#
root@vm1:/var/www/html# cd /var/www/
root@vm1:/var/www# ls -lh
total 4.0K
drwxrwxr-x 2 root vagrant 4.0K Nov 19 10:25 html
root@vm1:/var/www# sudo chmod -R g+rw /var/www/
root@vm1:/var/www#
root@vm1:/var/www# ls -la
total 12
drwxrwxr-x  3 root vagrant 4096 Nov 11 16:19 .
drwxr-xr-x 14 root root    4096 Nov 11 16:19 ..
drwxrwxr-x  2 root vagrant 4096 Nov 19 10:25 html
root@vm1:/var/www# cd html/
root@vm1:/var/www/html#
root@vm1:/var/www/html# ls -la
total 24
drwxrwxr-x 2 root    vagrant  4096 Nov 19 10:25 .
drwxrwxr-x 3 root    vagrant  4096 Nov 11 16:19 ..
-rw-rw-r-- 1 vagrant vagrant  2918 Nov 21 12:20 index.html
-rwxrwxr-x 1 root    vagrant 10918 Nov 19 07:32 index_old.html
root@vm1:/var/www/html# sudo chgrp -R vagrant /var/www/
root@vm1:/var/www/html# sudo chmod -R g+rw /var/www/
root@vm1:/var/www/html#
root@vm1:/var/www/html# ls -la
total 24
drwxrwxr-x 2 root    vagrant  4096 Nov 19 10:25 .
drwxrwxr-x 3 root    vagrant  4096 Nov 11 16:19 ..
-rw-rw-r-- 1 vagrant vagrant  2918 Nov 21 12:20 index.html
-rwxrwxr-x 1 root    vagrant 10918 Nov 19 07:32 index_old.html
root@vm1:/var/www/html#
```

d. Build Steps in Jenkins.

- Find out IP and hostname of the remote host(webserver Ubuntu 20.04)

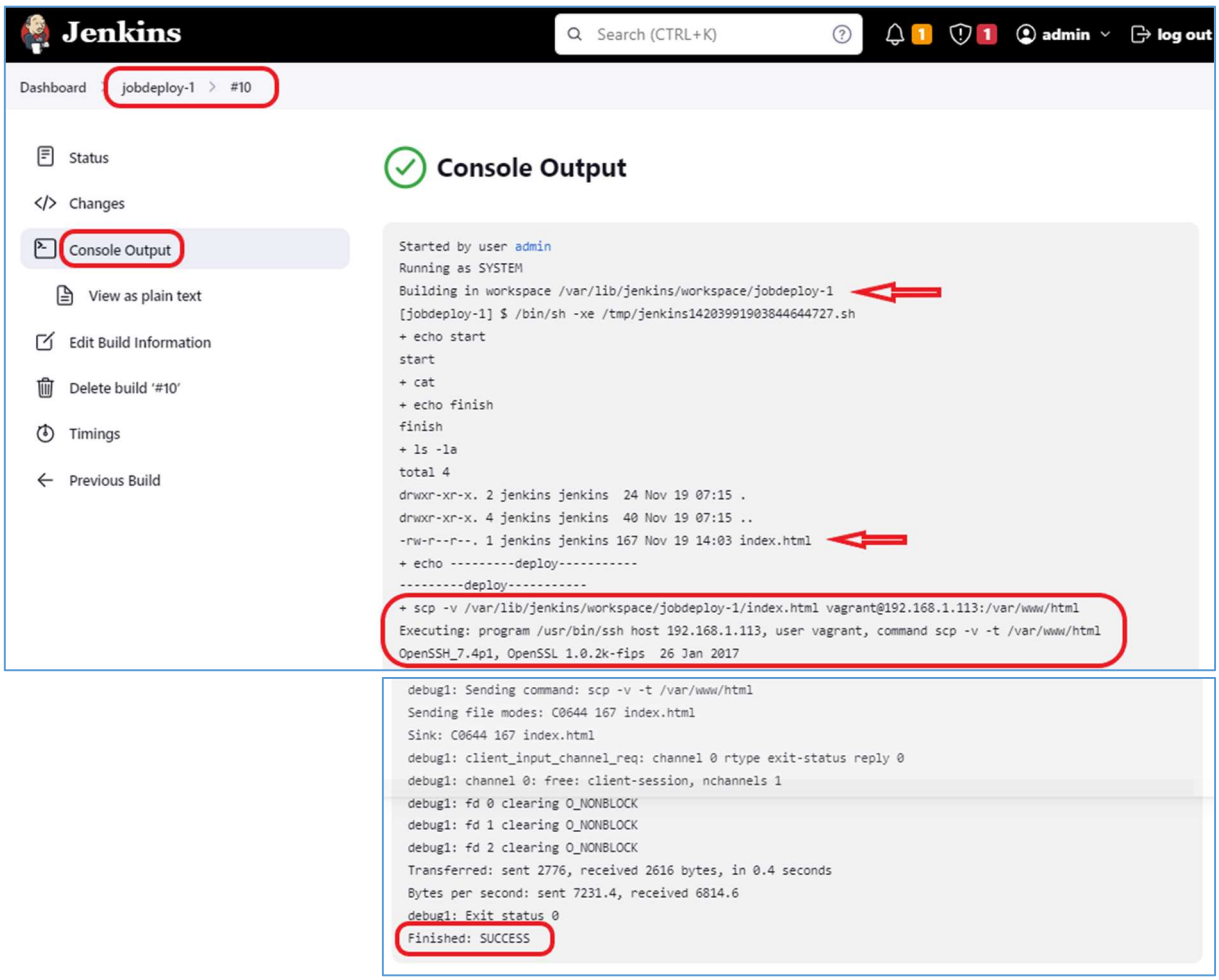
```
vagrant@vm1:/var/www/html$ hostname -I
10.0.2.15 192.168.1.113
vagrant@vm1:/var/www/html$ whoami
vagrant
vagrant@vm1:/var/www/html$
```

- Copy the file **"index.html"** to the remote host via ssh

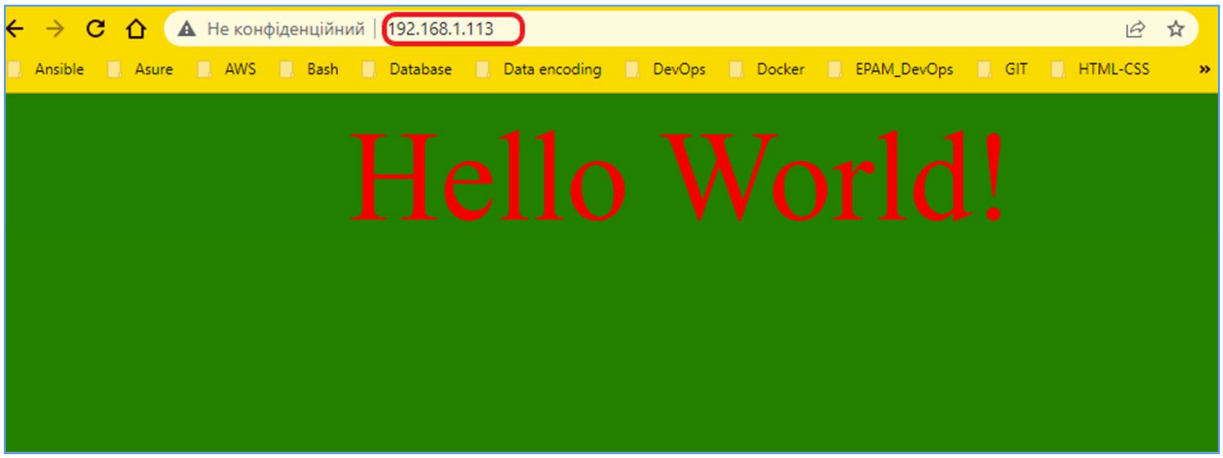

```
echo "start"
cat <<EOF > index.html
<html>
<head>
    <title>TEST</title>
</head>
<body bgcolor=green>
    <p style = "color: red; text-align: center; font-size: 100px;">Hello World!</p>
</body>
</html>
EOF
echo "finish"
ls -la
echo "---------deploy----------"
# copy file "index.html" to remote host over ssh.
scp -v /var/lib/jenkins/workspace/jobdeploy-1/index.html vagrant@192.168.1.113:/var/www/html

# copy file "index.html" to remote host over ssh and without checking key.
# scp -v -o StrictHostKeyChecking=no index.html vagrant@192.168.1.113:/var/www/html
```

e. Build job.

- Console output

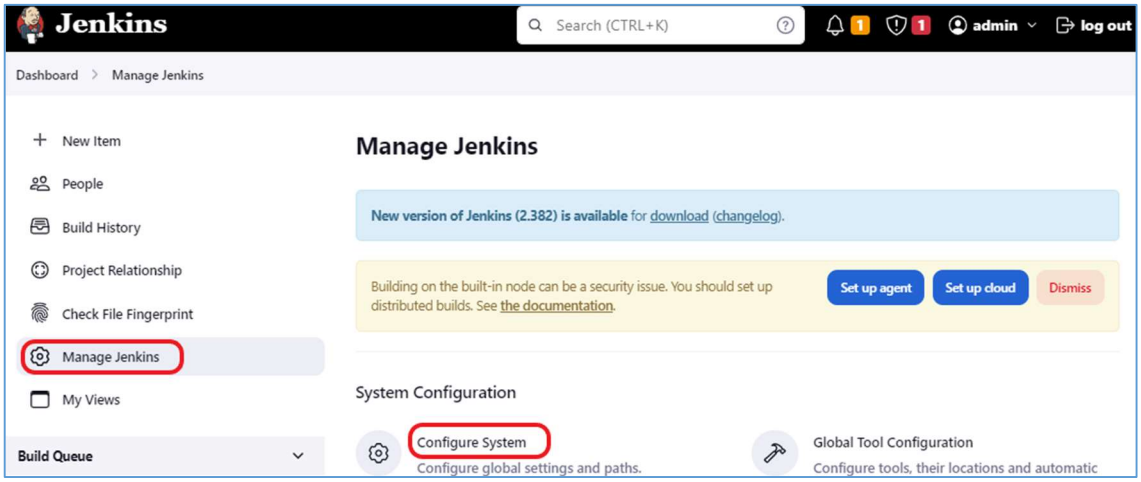


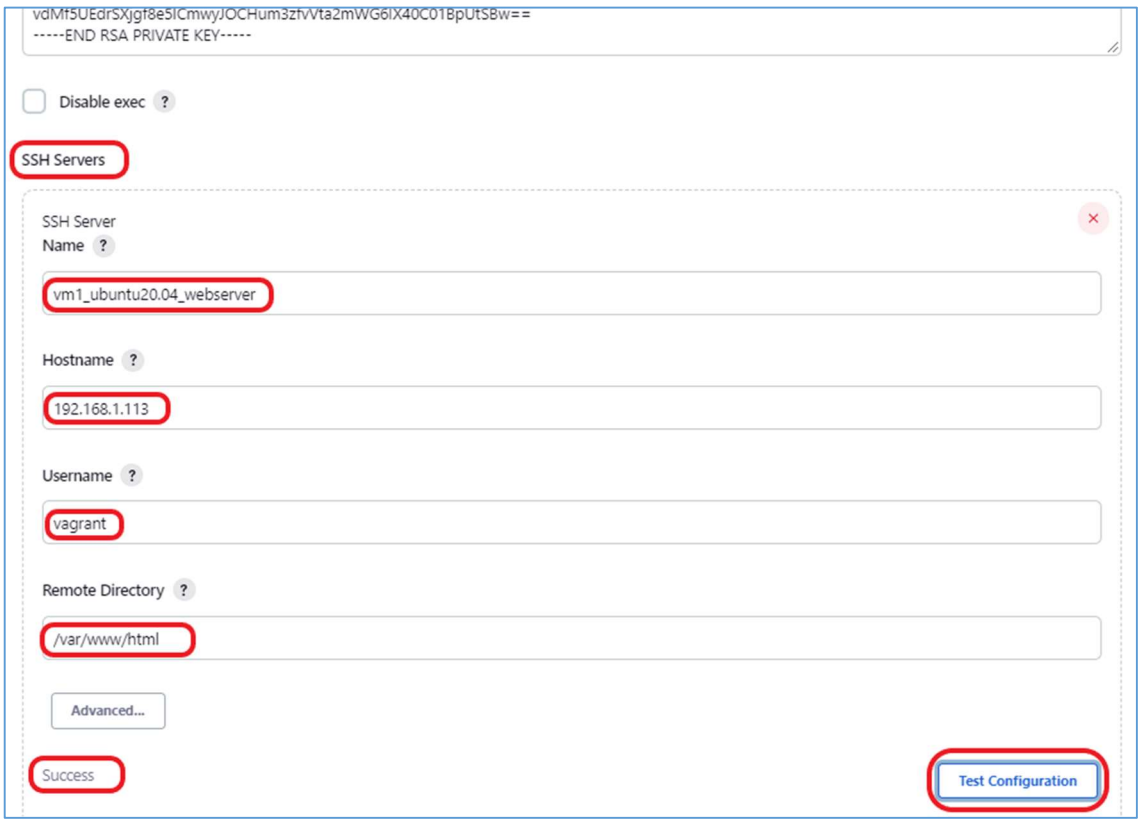
- Checking our website



5. Simple job example in Jenkins with deploy using plugin "Publish over SSH".

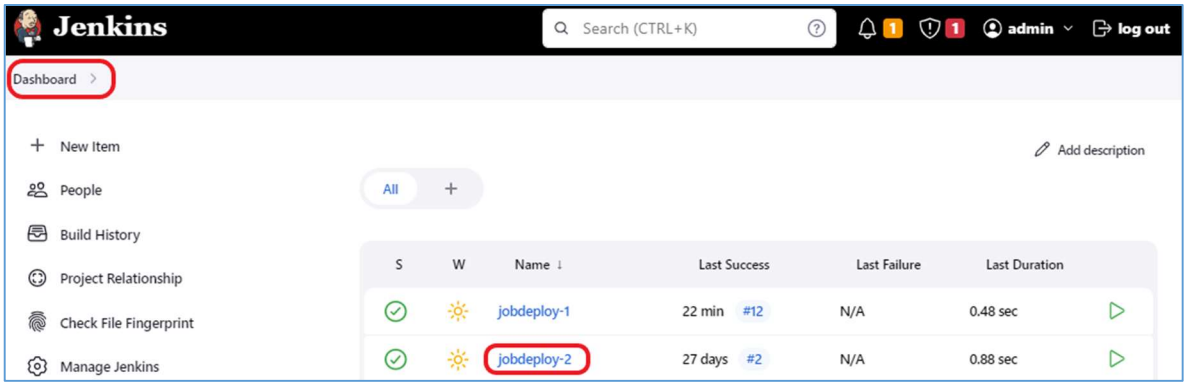
a. Configuration and settings of the plugin "Publish over SSH".

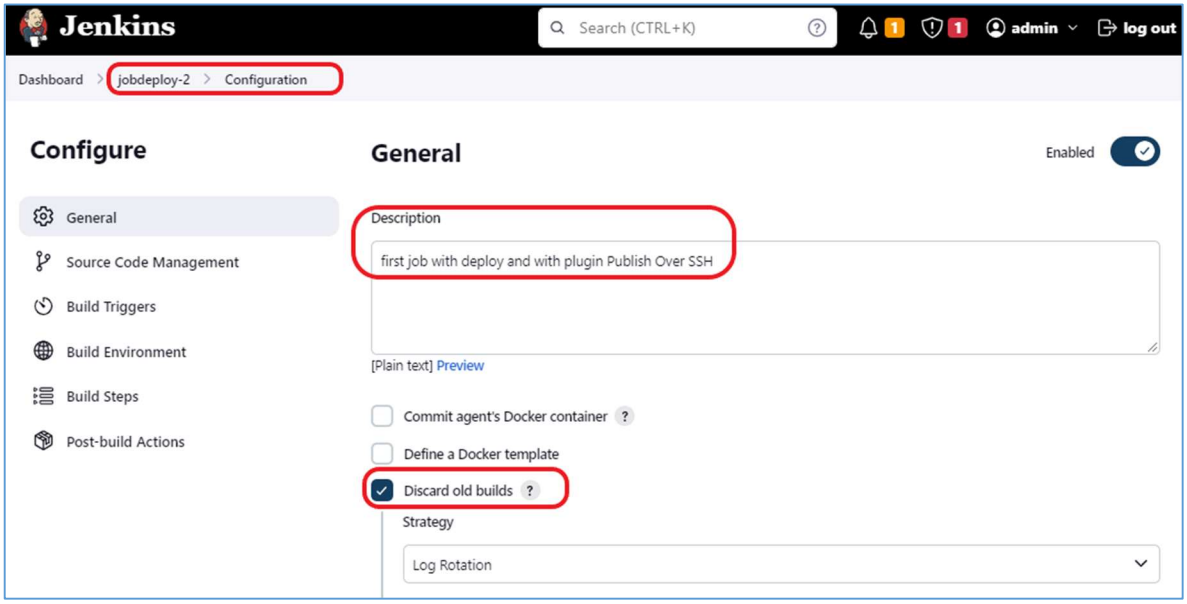


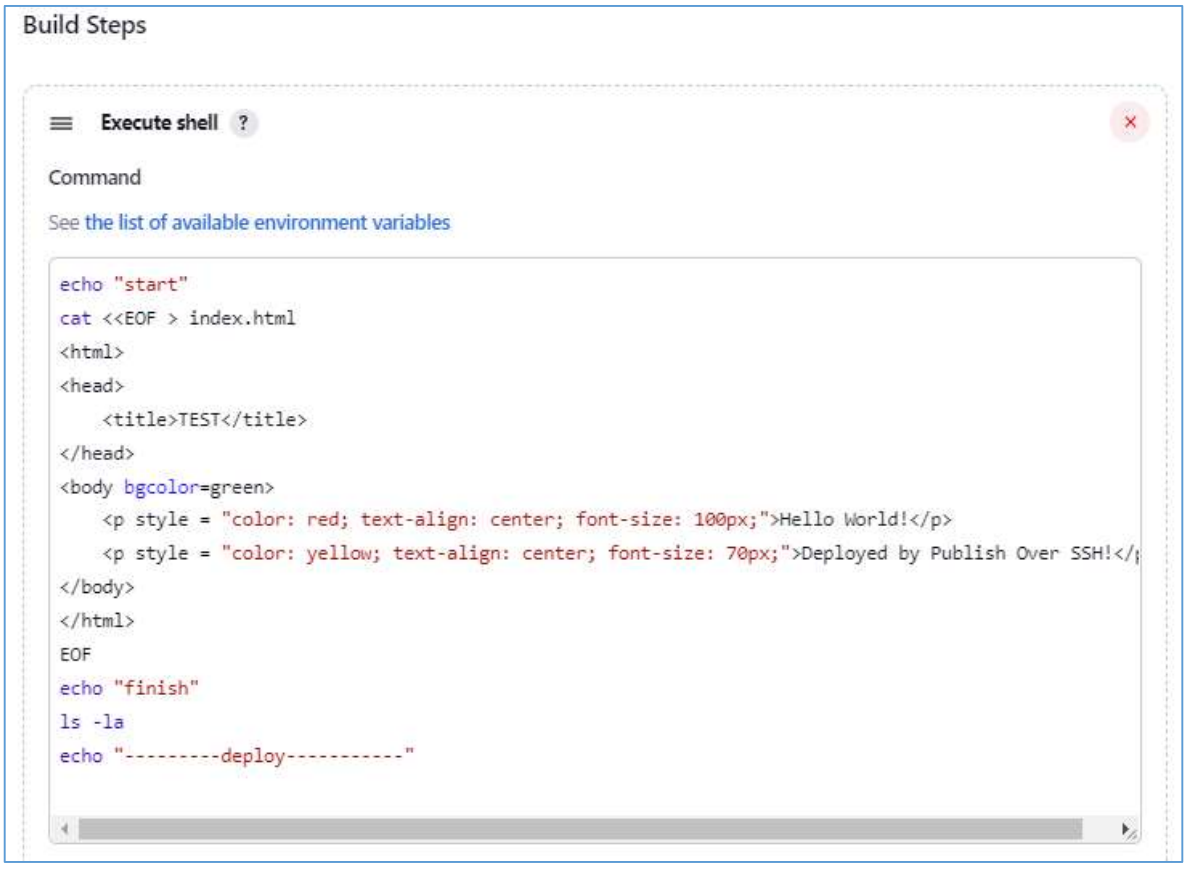


*** User (vagrant) must have access to the directory (/var/www/html)**

b. Create new job.

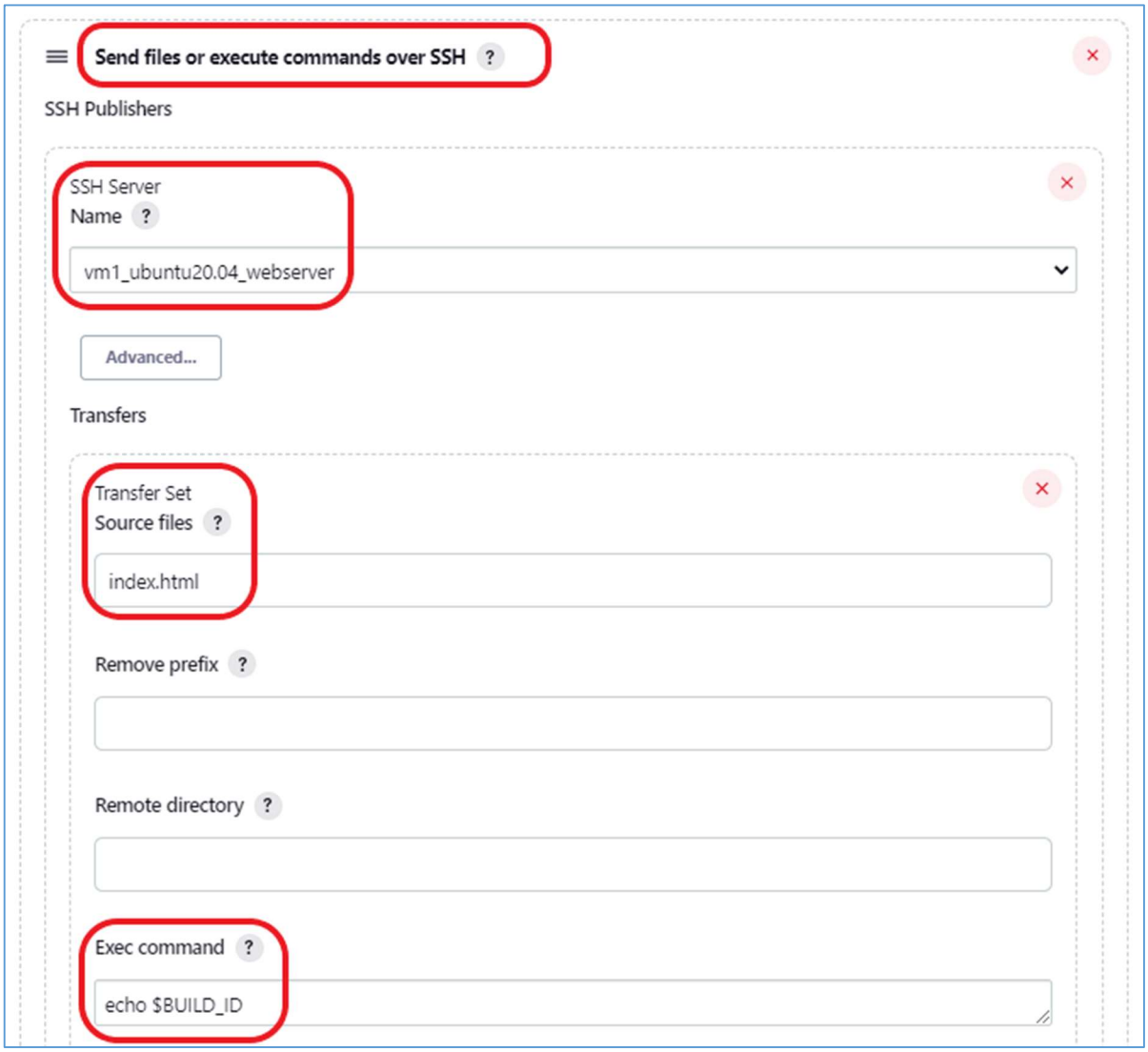






c. Establish **SSH** connection using plugin "**Publish over SSH**".

In the SSH Server Name section use the ssh server name which you configured earlier.



d. Install **Apache server** and establish privileges for **/var/www**.

- Install **Apache** on remote host where you will deploy app **(**example Ubuntu 20.04**)**

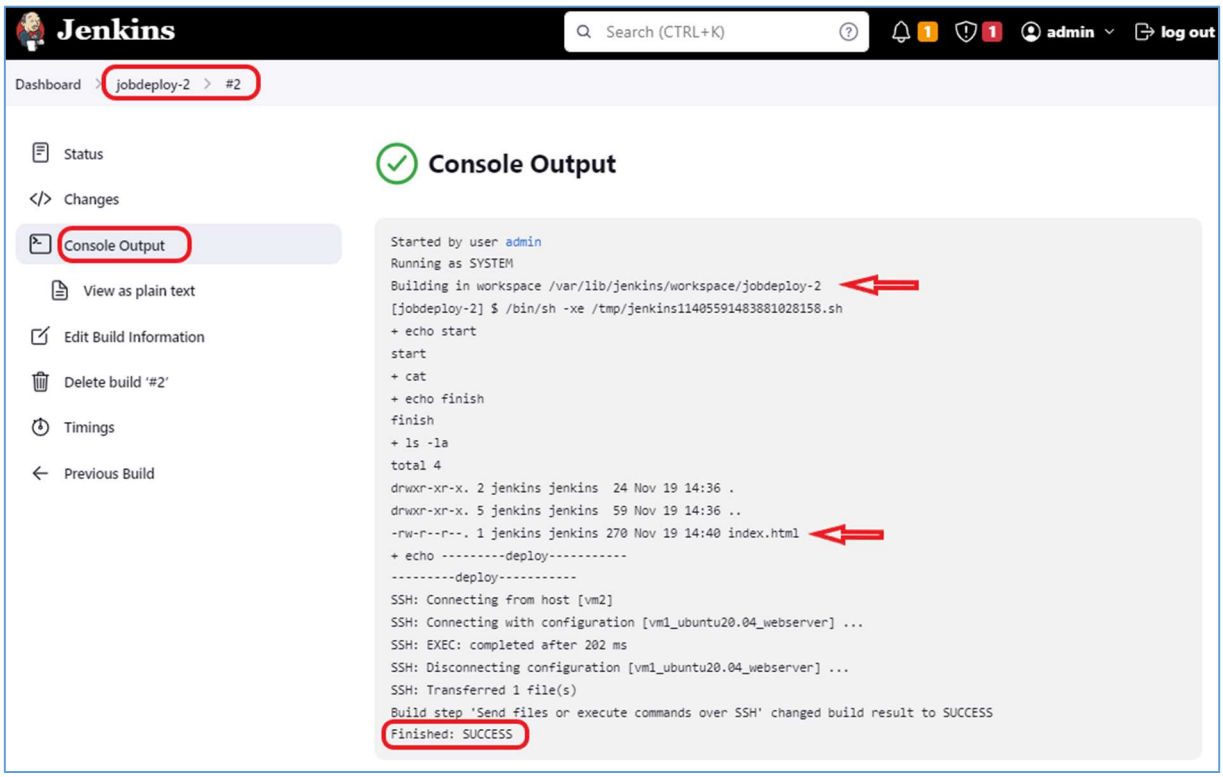
```
$ sudo apt update
$ sudo apt install apache2
$ sudo ufw allow 'Apache'
$ sudo ufw status
$ sudo sysstemctl status apache2
```

- Establish privileges for **/var/www**

```
vagrant@vm1:/var/www/html$ sudo su
root@vm1:/var/www/html#
root@vm1:/var/www/html# cd /var/www/
root@vm1:/var/www# ls -lh
total 4.0K
drwxrwxr-x 2 root vagrant 4.0K Nov 19 10:25 html
root@vm1:/var/www# sudo chmod -R g+rw /var/www/
root@vm1:/var/www#
root@vm1:/var/www# ls -la
total 12
drwxrwxr-x  3 root vagrant 4096 Nov 11 16:19 .
drwxr-xr-x 14 root root    4096 Nov 11 16:19 ..
drwxrwxr-x  2 root vagrant 4096 Nov 19 10:25 html
root@vm1:/var/www# cd html/
root@vm1:/var/www/html#
root@vm1:/var/www/html# ls -la
total 24
drwxrwxr-x 2 root    vagrant  4096 Nov 19 10:25 .
drwxrwxr-x 3 root    vagrant  4096 Nov 11 16:19 ..
-rw-rw-r-- 1 vagrant vagrant  2918 Nov 21 12:20 index.html
-rwxrwxr-x 1 root    vagrant 10918 Nov 19 07:32 index_old.html
root@vm1:/var/www/html# sudo chgrp -R vagrant /var/www/
root@vm1:/var/www/html# sudo chmod -R g+rw /var/www/
root@vm1:/var/www/html#
root@vm1:/var/www/html# ls -la
total 24
drwxrwxr-x 2 root    vagrant  4096 Nov 19 10:25 .
drwxrwxr-x 3 root    vagrant  4096 Nov 11 16:19 ..
-rw-rw-r-- 1 vagrant vagrant  2918 Nov 21 12:20 index.html
-rwxrwxr-x 1 root    vagrant 10918 Nov 19 07:32 index_old.html
root@vm1:/var/www/html#
```

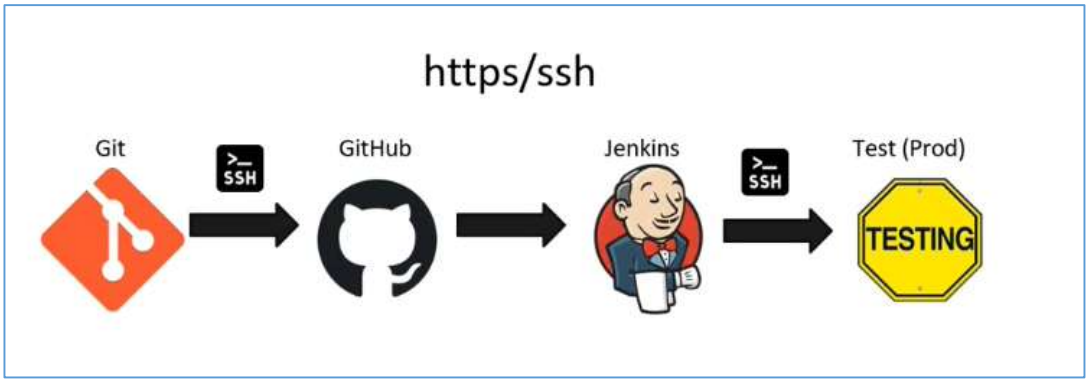
e. Build job.

- Console output



- Checking our website

6. Simple CI/CD pipeline example.



a. Configuration and settings credentials GitHub.

- Generate ssh key-pair on your Linux host

```
$ ssh-keygen -t rsa -C "your_GitHub_email@example.com"
```

- Adding your SSH key to the ssh-agent

```
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/"key_name"
```

- Adding a new SSH public key to your GitHub account

```
$ clip < ~/.ssh/"public_key_name.pub"
```

* If clip isn't working, you can locate the hidden .ssh folder, open the file in your favorite text editor, and copy it to your clipboard.

Then go to the your GitHub account in section **Settings - SSH and GPG keys** and add new ssh public key. *(see next page)*

- Adding github.com to the **/var/lib/jenkins/.ssh/known_hosts**

```
$ sudo ssh-keyscan github.com >>
/var/lib/jenkins/.ssh/known_hosts"
```

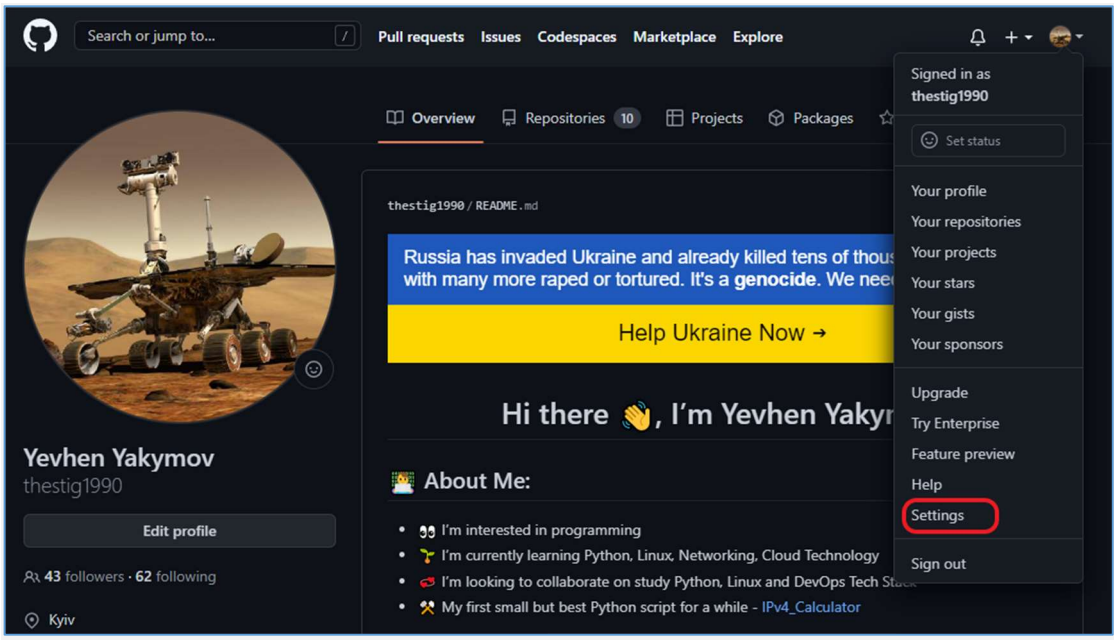
- Dashboard > Manage Jenkins > Configure Global Security > Git Host Key Verification Configuration > Host Key Verification Strategy > Known hosts file

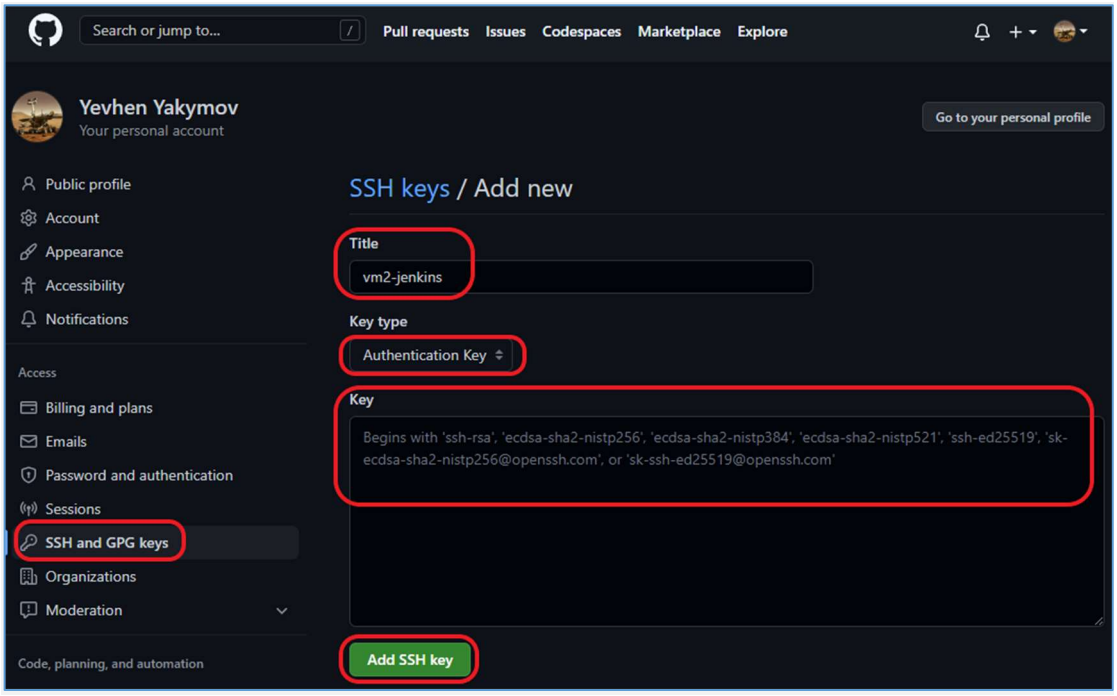
- We have to copy file "**key_name**"(chmod 400) to **/var/lib/jenkins/.ssh**

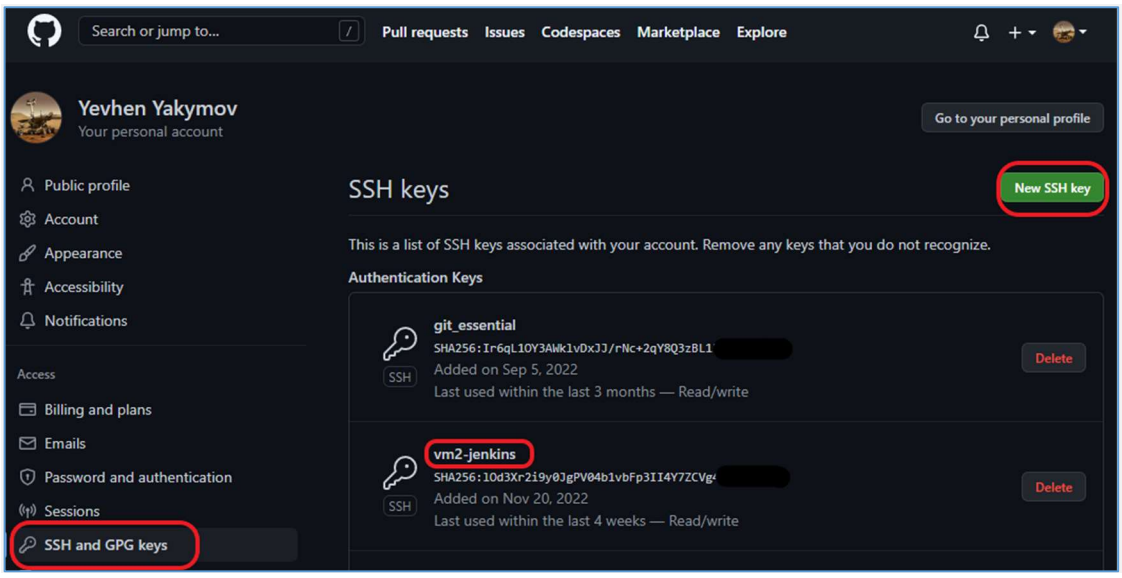
- Testing your connection to the GitHub from **jenkins** user

```
$ ssh -T git@github.com
```

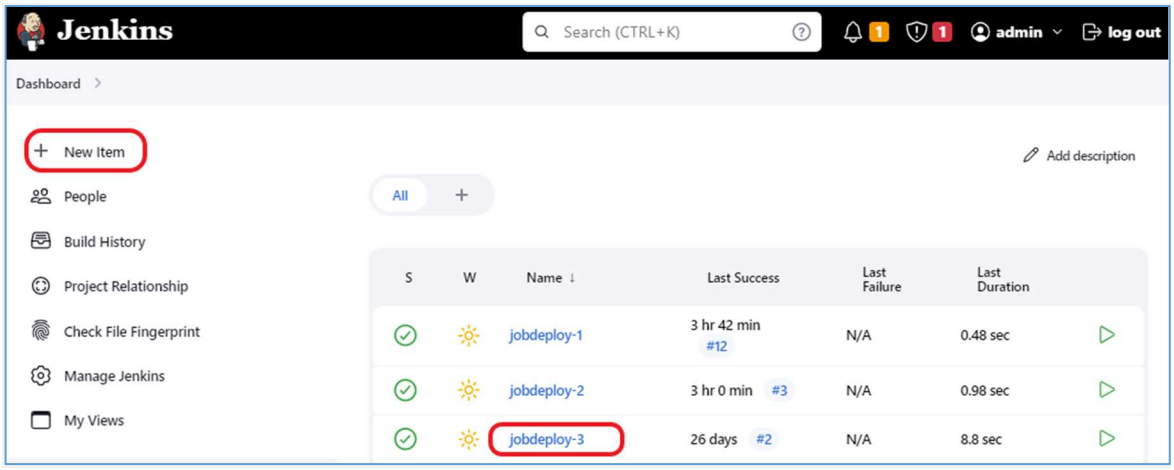
```
-bash-4.2$ ssh -T git@github.com
Hi thestig1990! You've successfully authenticated, but GitHub does not provide shell access.
-bash-4.2$
```

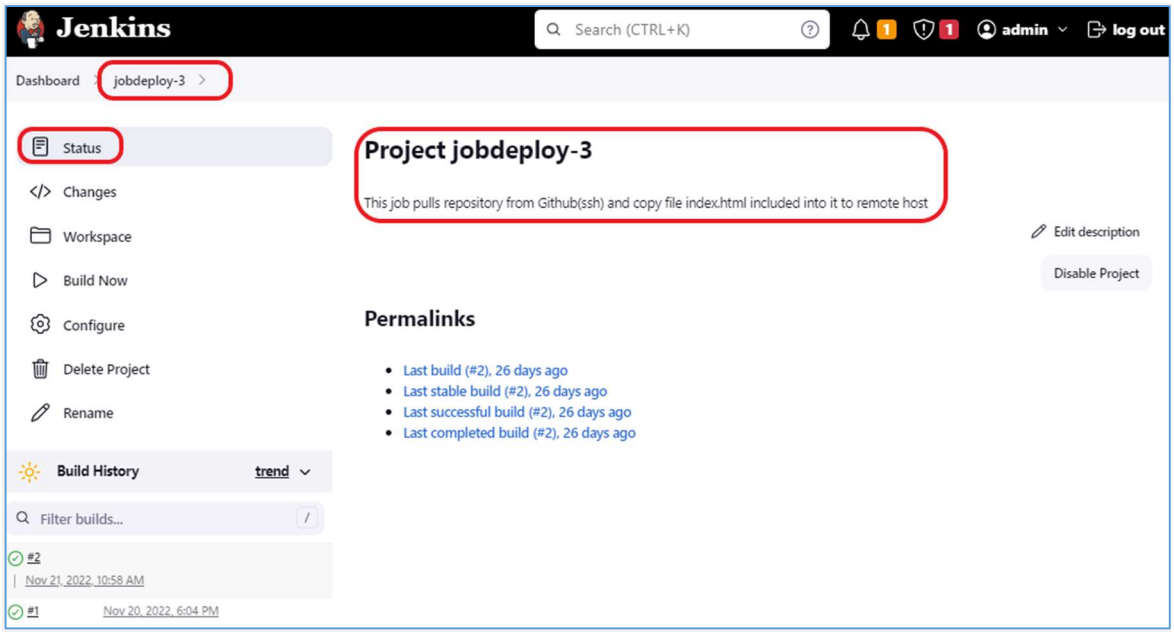

Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Signed in as thestig1990
Overview
Repositories 10
Projects
Packages
Set status
Your profile
Your repositories
Your projects
Your stars
Your gists
Your sponsors
Upgrade
Try Enterprise
Feature preview
Help
Settings
Sign out
thestig1990 / README.md
Russia has invaded Ukraine and already killed tens of thou
with many more raped or tortured. It's a genocide. We nee
Help Ukraine Now →
Hi there 👋, I'm Yevhen Yakyr
About Me:
I'm interested in programming
I'm currently learning Python, Linux, Networking, Cloud Technology
I'm looking to collaborate on study Python, Linux and DevOps Tech Stack
My first small but best Python script for a while - IPv4_Calculator
Yevhen Yakymov
thestig1990
Edit profile
43 followers · 62 following
Kyiv


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Yevhen Yakymov
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Moderation
Code, planning, and automation
SSH keys / Add new
Title
vm2-jenkins
Key type
Authentication Key
Key
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'
Add SSH key

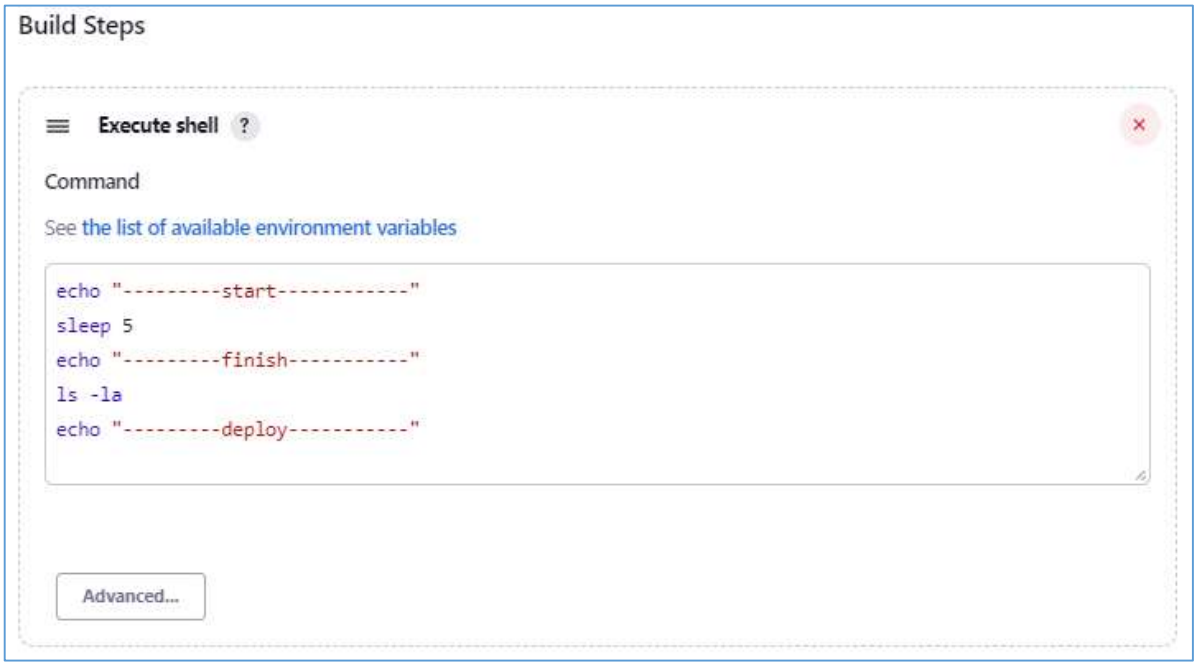

Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
Yevhen Yakymov
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
SSH keys
New SSH key
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.
Authentication Keys
git_essential
SHA256:Ir6qL1OY3AWklvDxJJ/rNc+2qY8Q3zBL1
SSH
Added on Sep 5, 2022
Last used within the last 3 months — Read/write
Delete
vm2-jenkins
SHA256:lOd3Xr2i9y0JgPV04b1vbFp3II4Y7ZCVg
SSH
Added on Nov 20, 2022
Last used within the last 4 weeks — Read/write
Delete

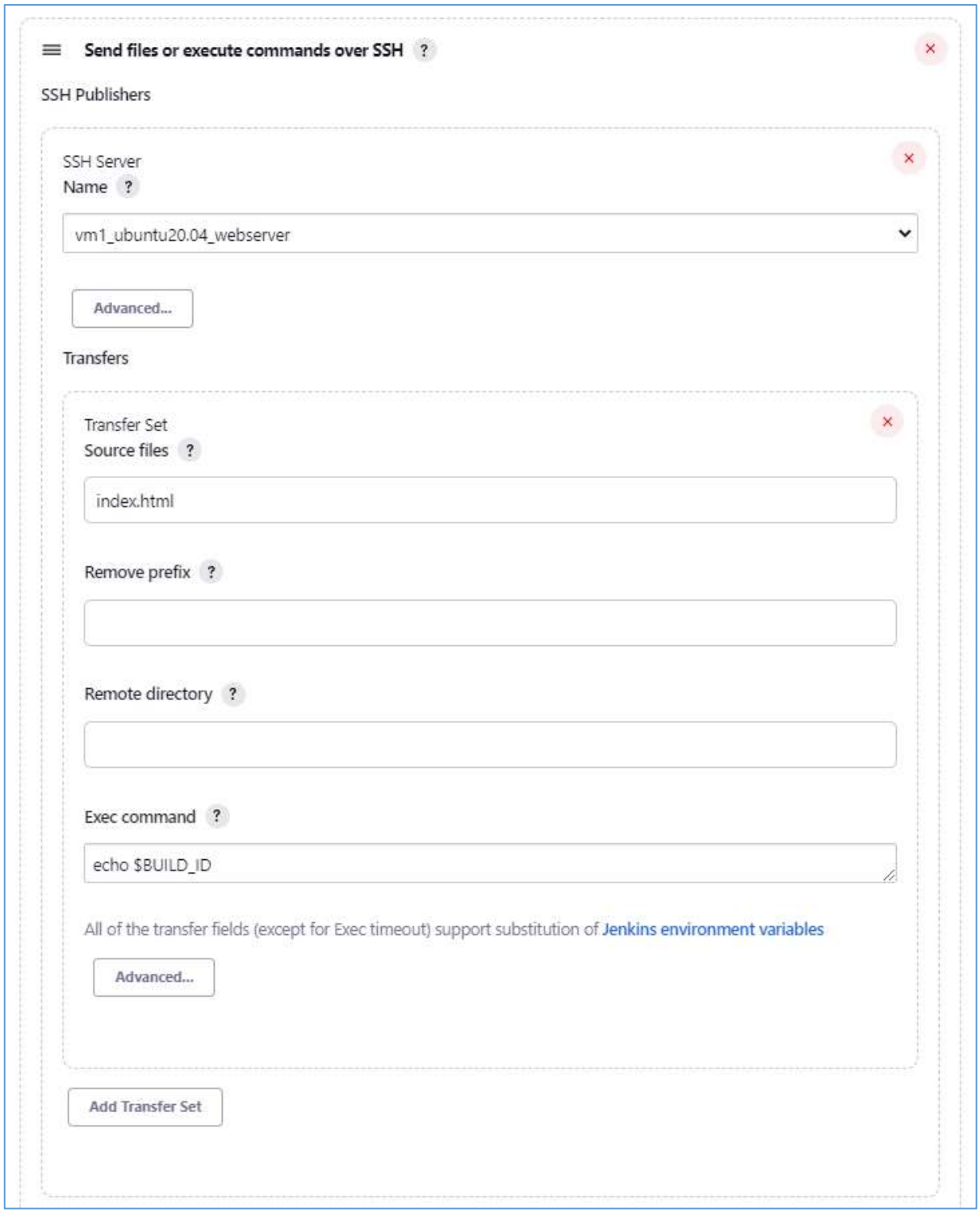
b. Create new job.



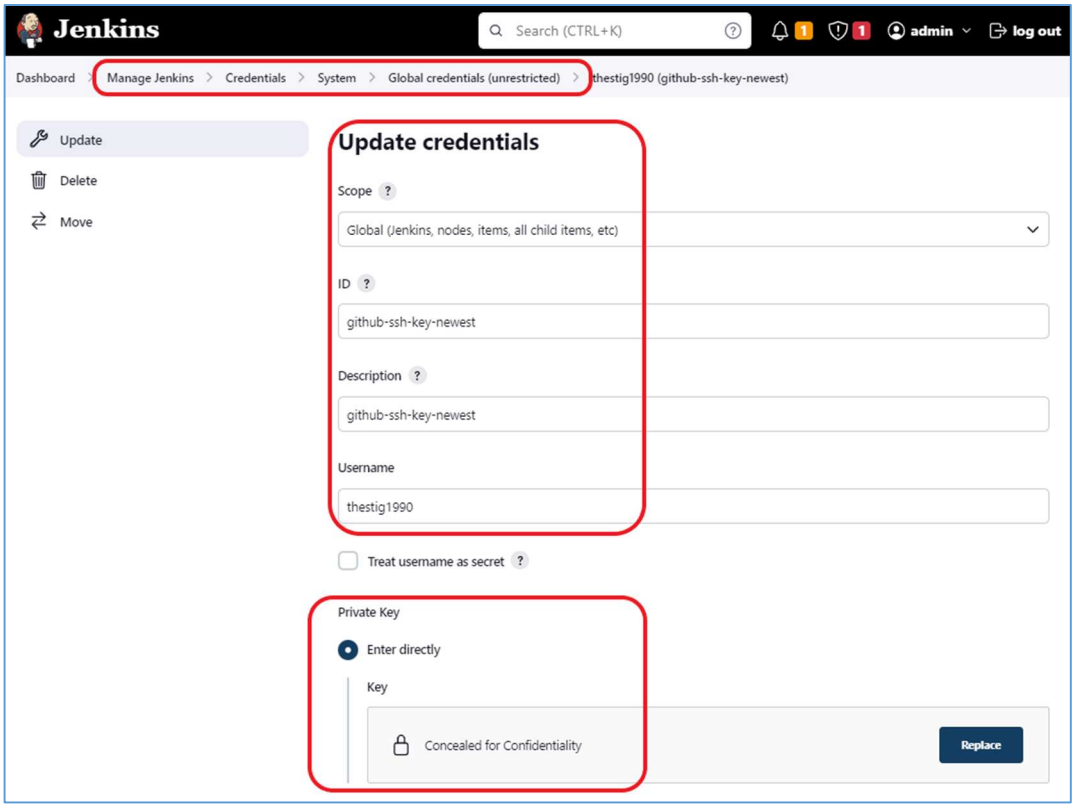


c. Build steps.

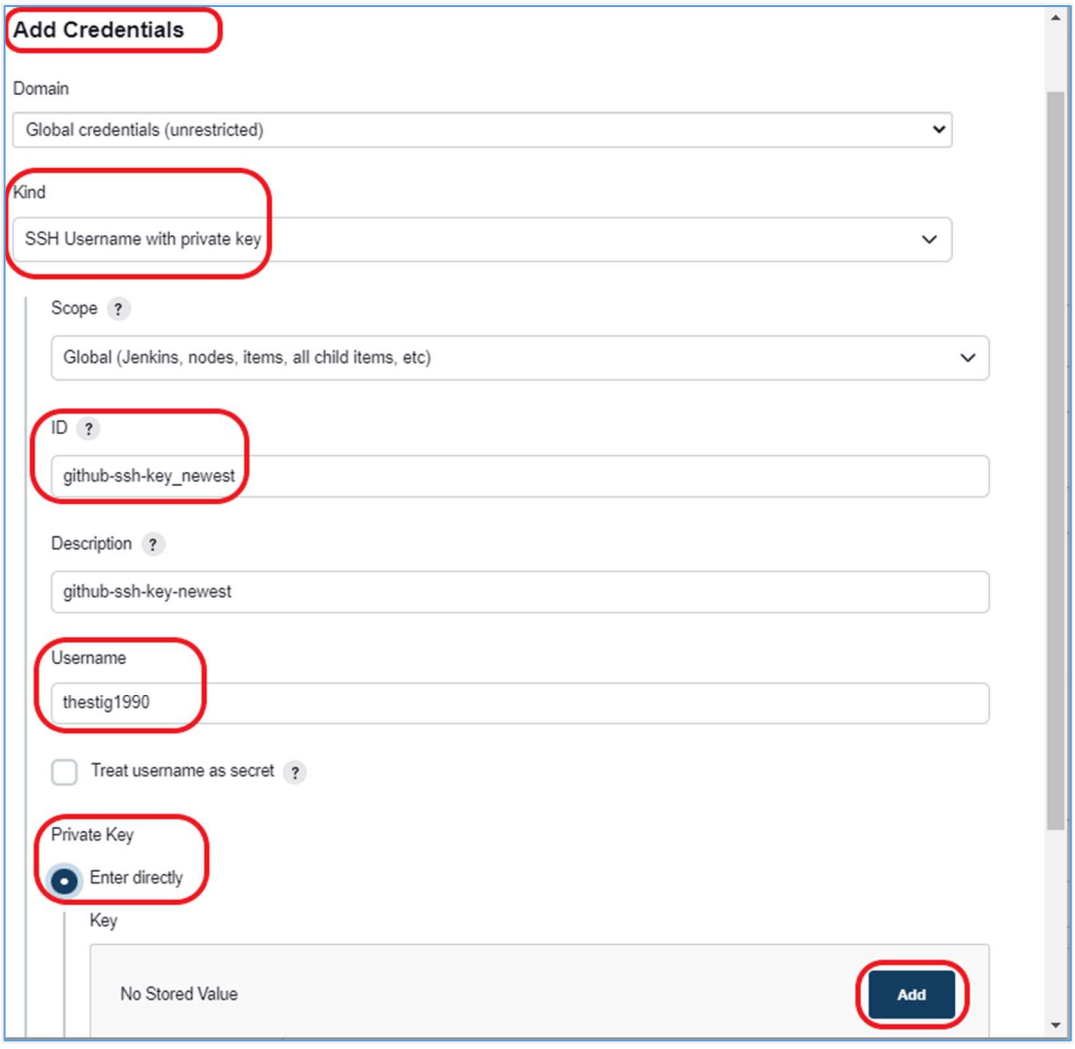


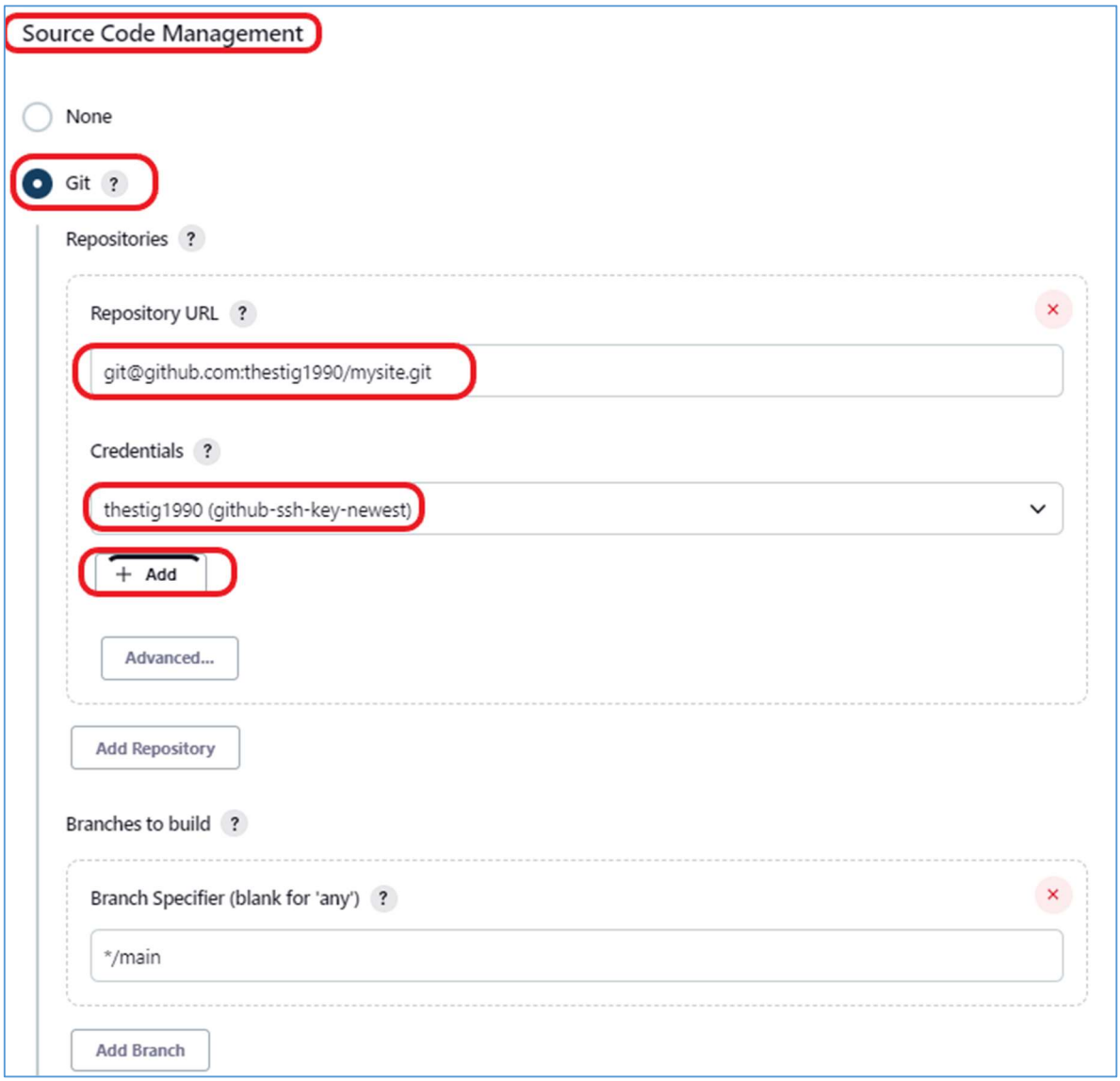

Send files or execute commands over SSH
SSH Publishers
SSH Server
Name
vm1_ubuntu20.04_webserver
Advanced...
Transfers
Transfer Set
Source files
index.html
Remove prefix
Remote directory
Exec command
echo $BUILD_ID
All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables
Advanced...
Add Transfer Set

d. Adding Jenkins Credentials Provider.

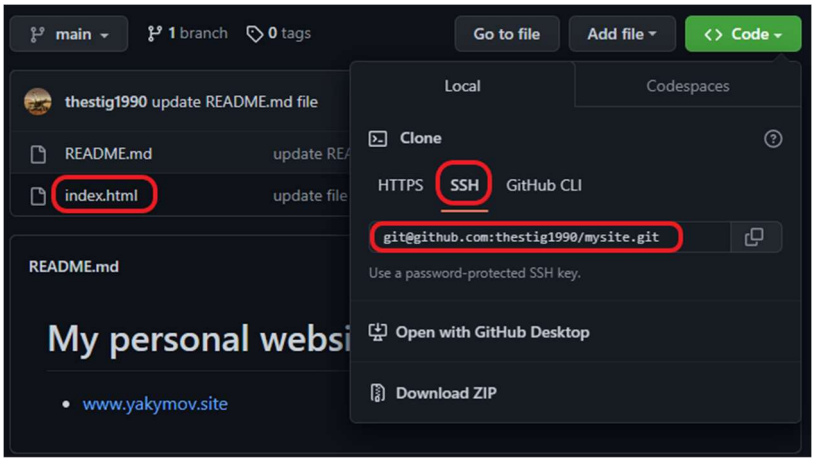


or





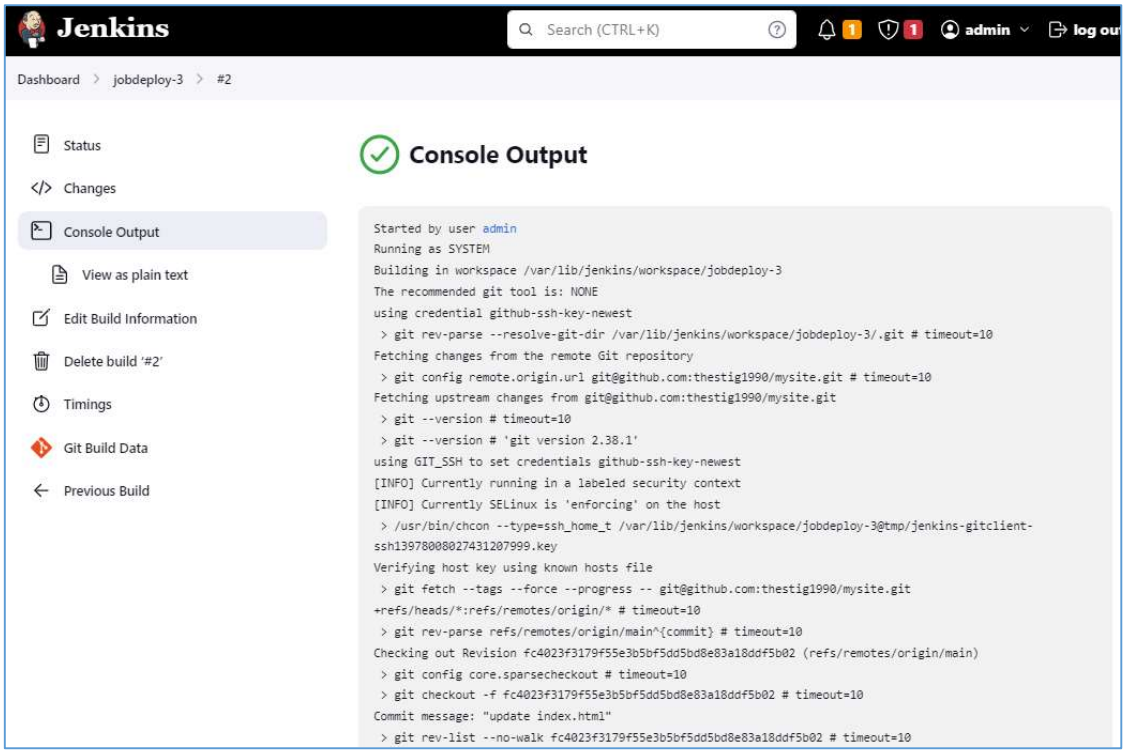
- **ID** - An internal unique ID by which these credentials are identified from jobs and other configuration
- **Username** - your username in GitHub
- **Private key** – private key that your create earlier.
- **Repository URL** - specify the URL or path of the git repository.



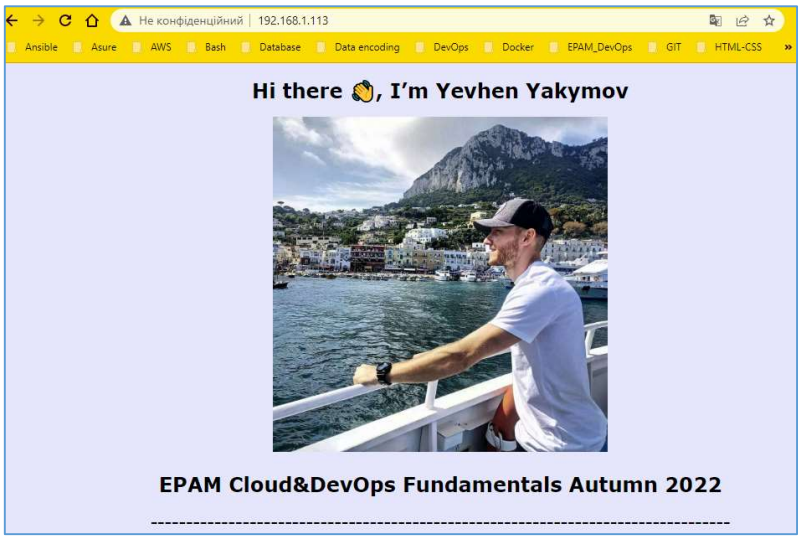
- **Credentials** - Credential used to check out sources

e. Build job.

- Console output



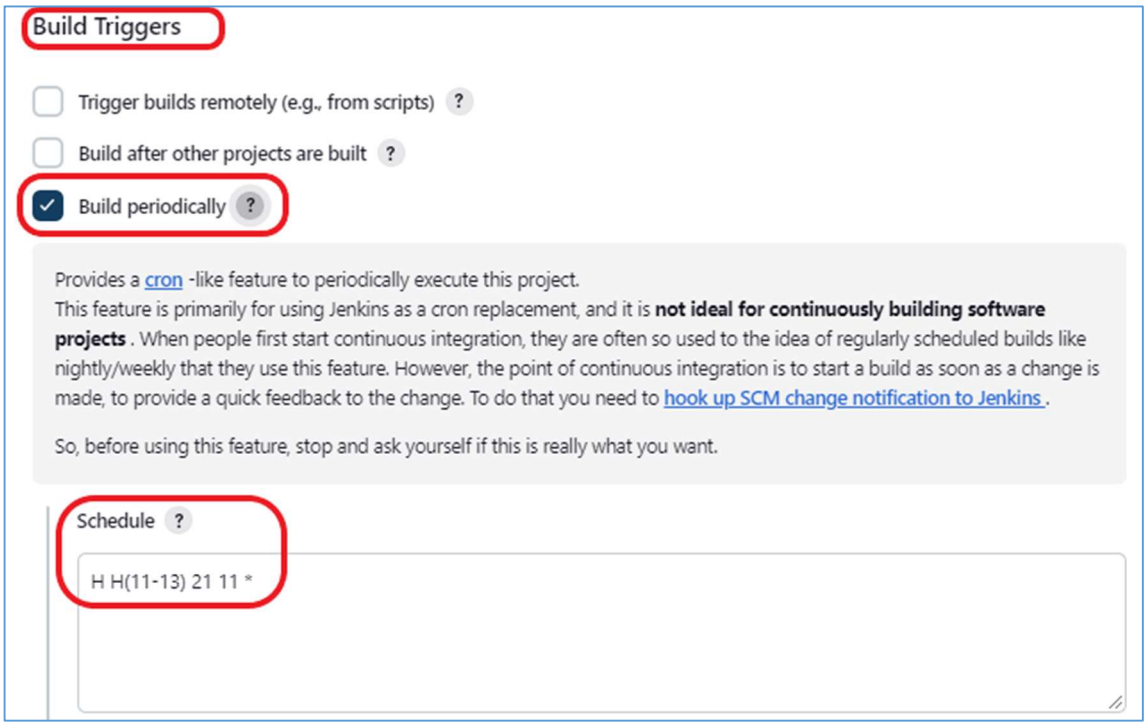
- Checking the availability of the website



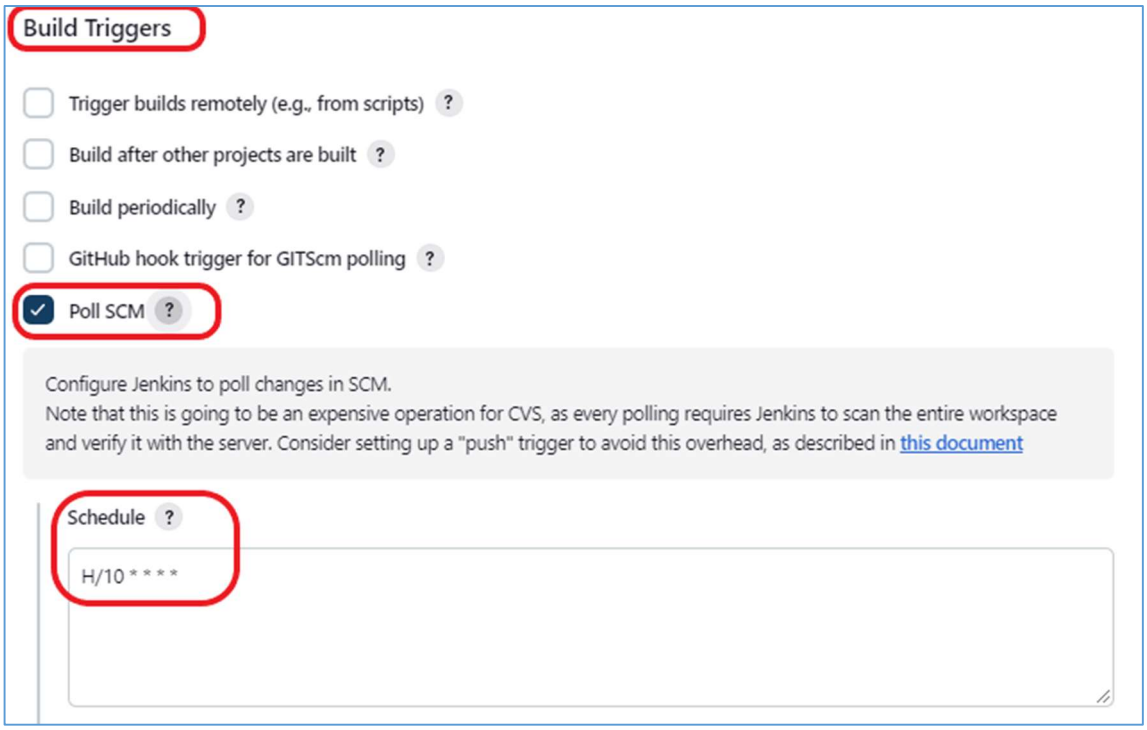
f. More about Build Triggers.

If you wanna that the process of pulling file from GitHub repo and copying this file to the web server to be automatic you need to use Build Triggers.

- Build periodically (use **Schedule** like crontab by Linux)

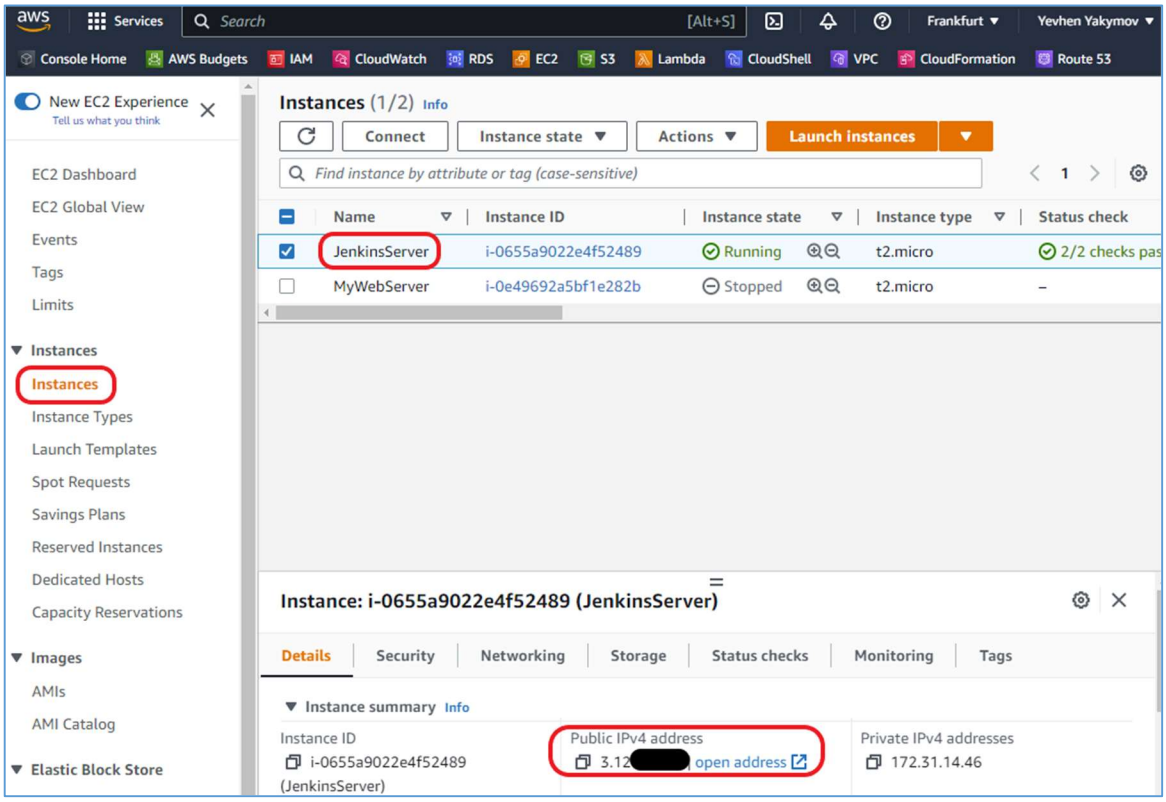


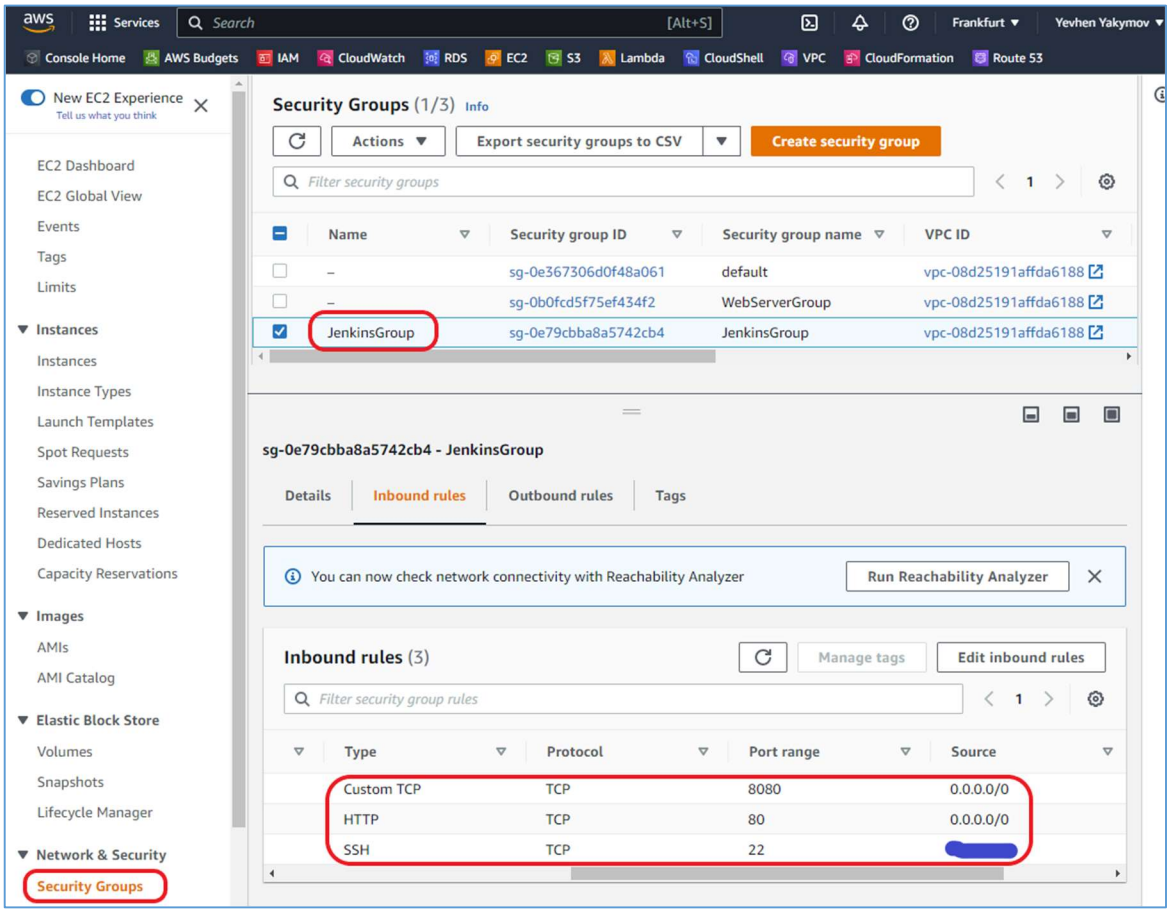
- Poll SCM (in this case Jenkins cheking github repo for changes every 10 min and if there are the build is launched automatically)

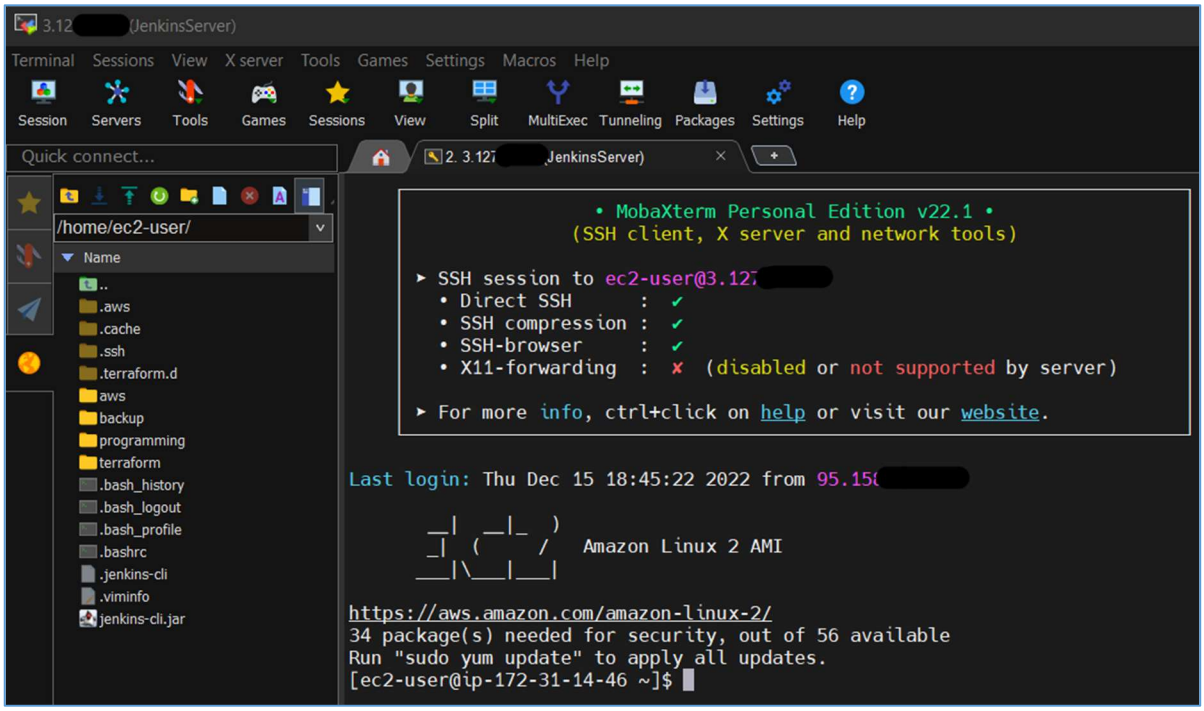


7. Simple CI/CD pipeline example using AWS.

- Launch the AWS EC2 Instance in accordance with the steps in AWS hometask.



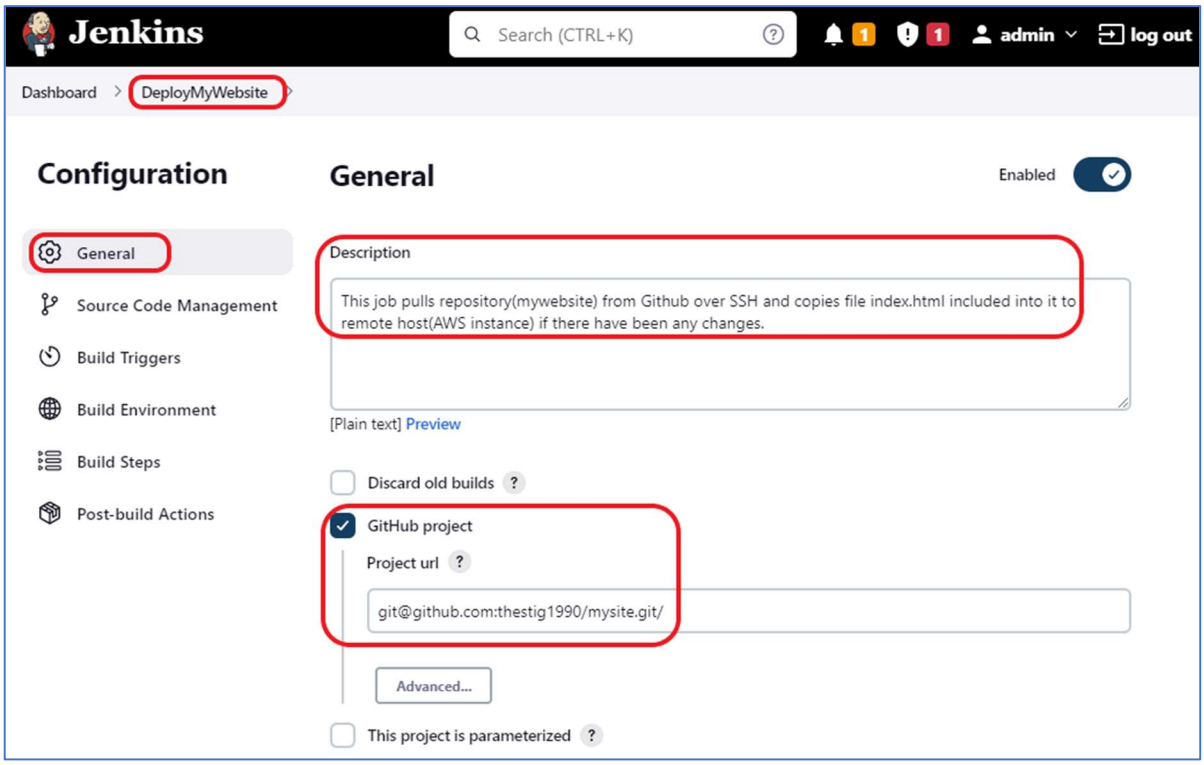




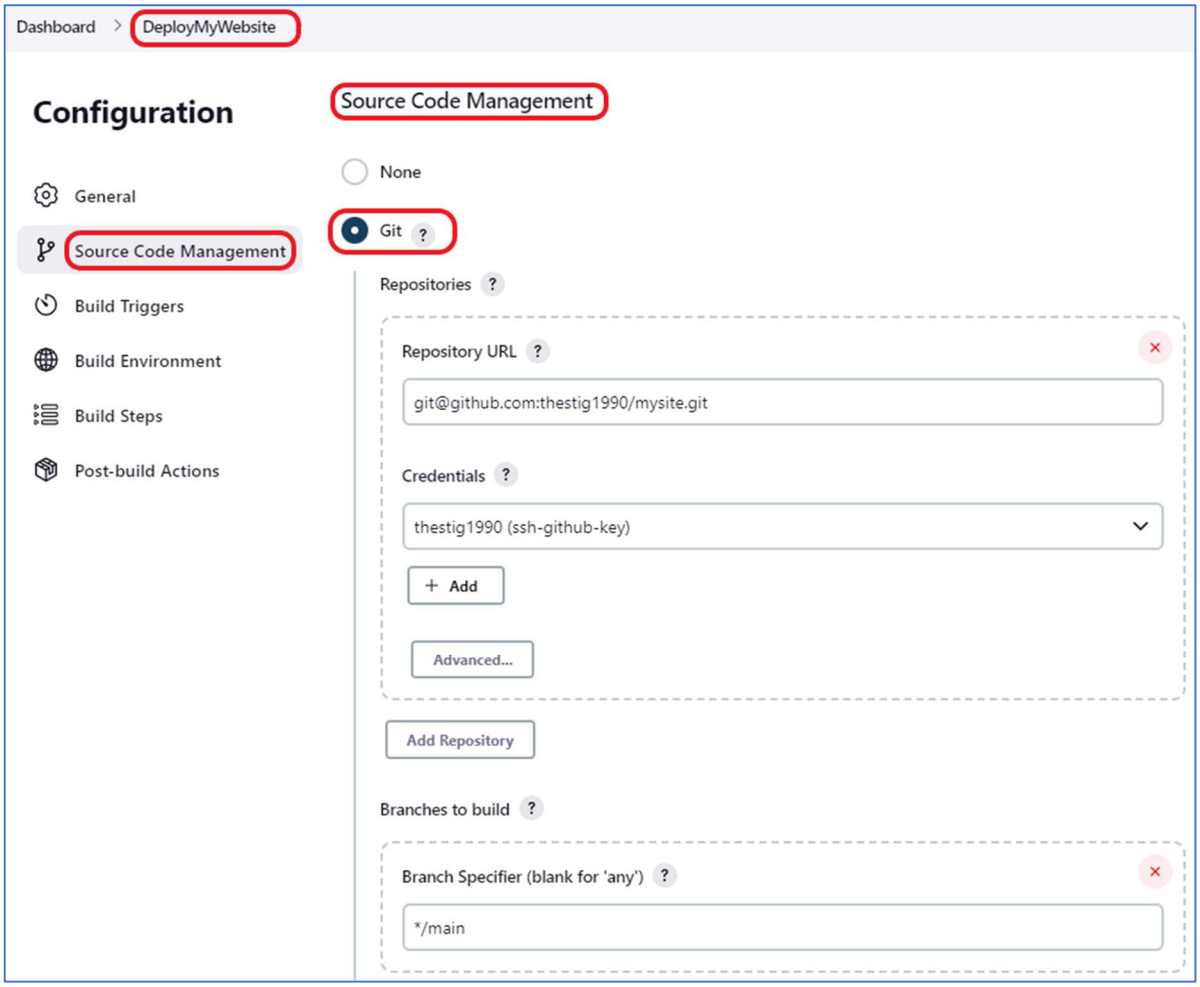
- Install and integrate Jenkins with AWS EC2 Instance (*see instruction*).

- Create job.

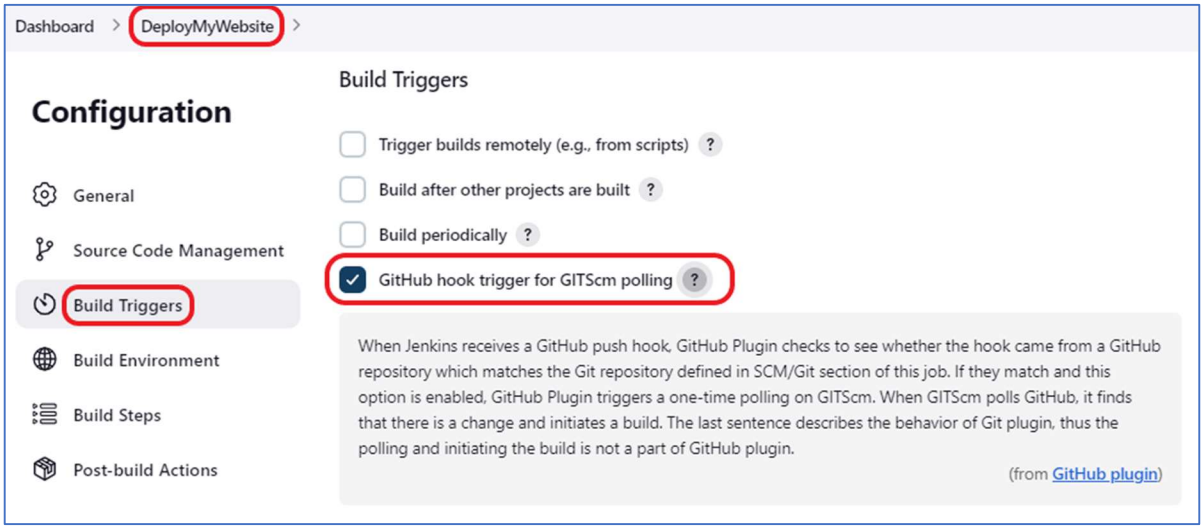
- **Desription** of the job and turn on **GitHub project** with github repo url.



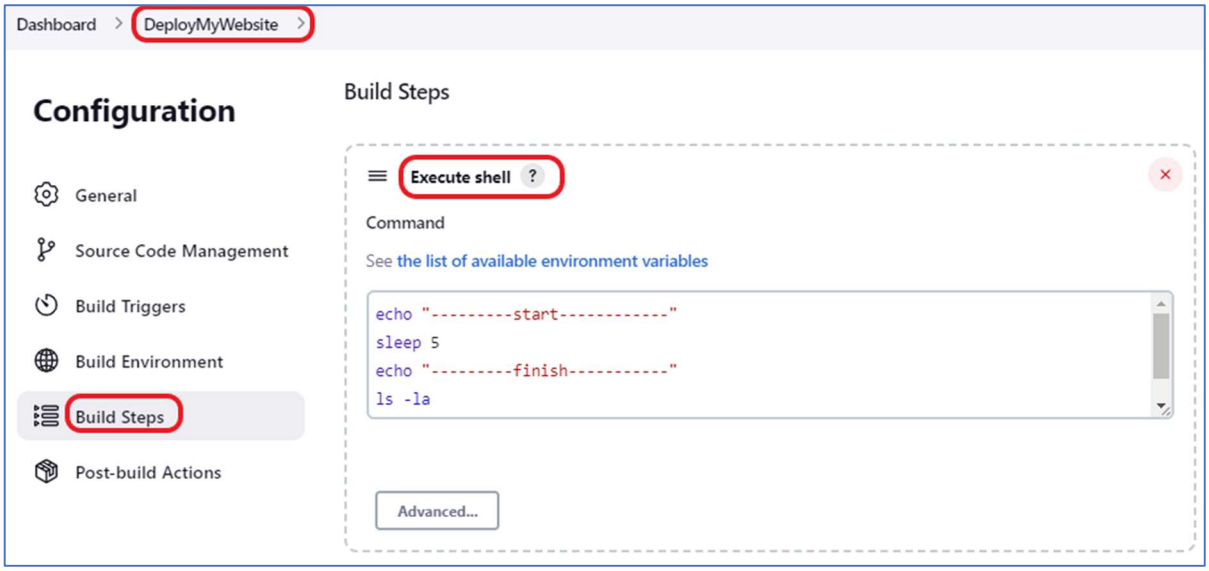
- Description of the **Sourse Code Management** *(see instraction)*



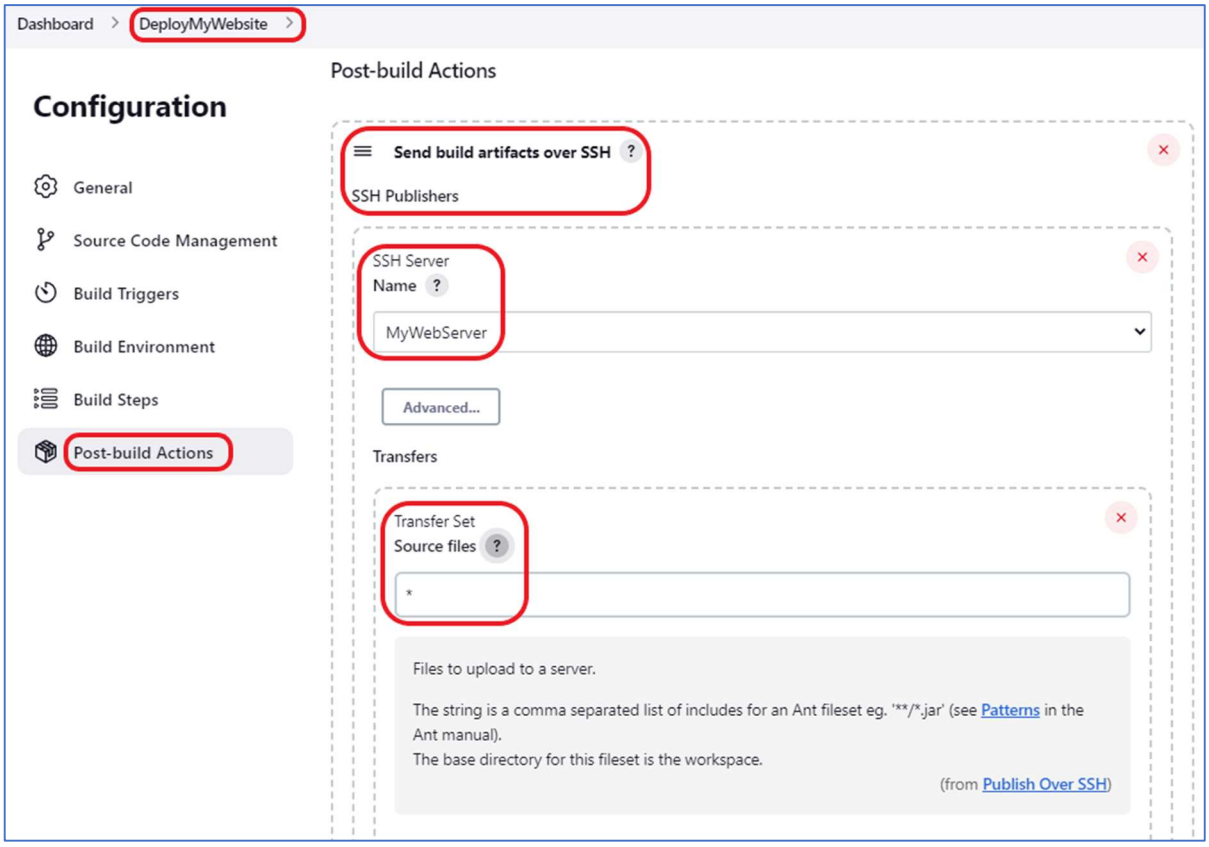
- Build Triggers using **GitHub trigger for GITScm polling**



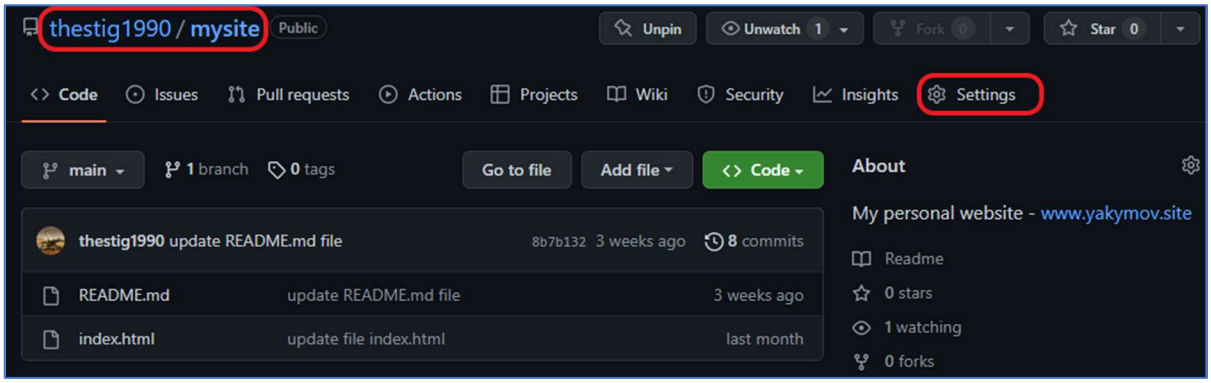
- Build Steps

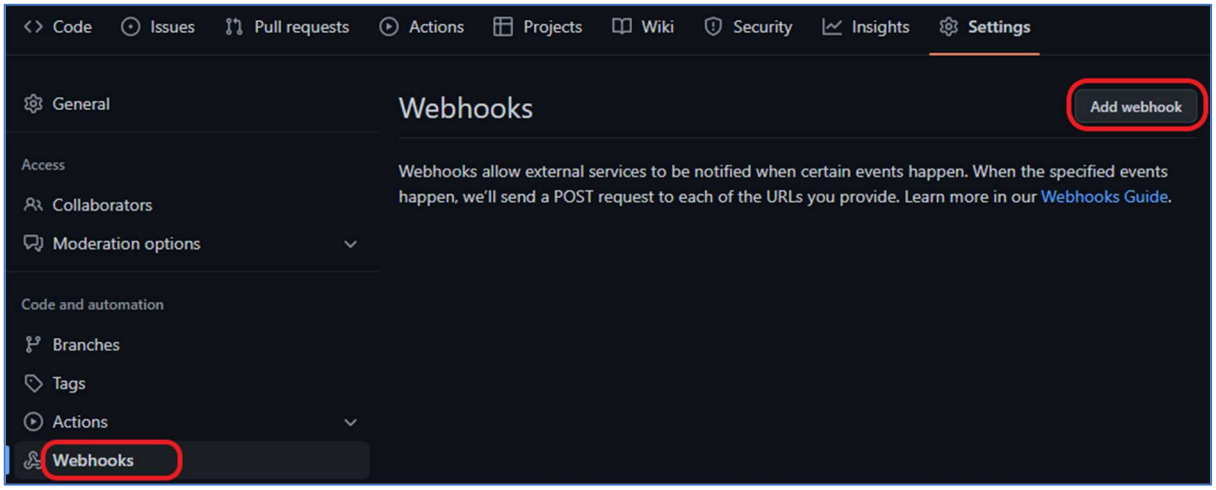


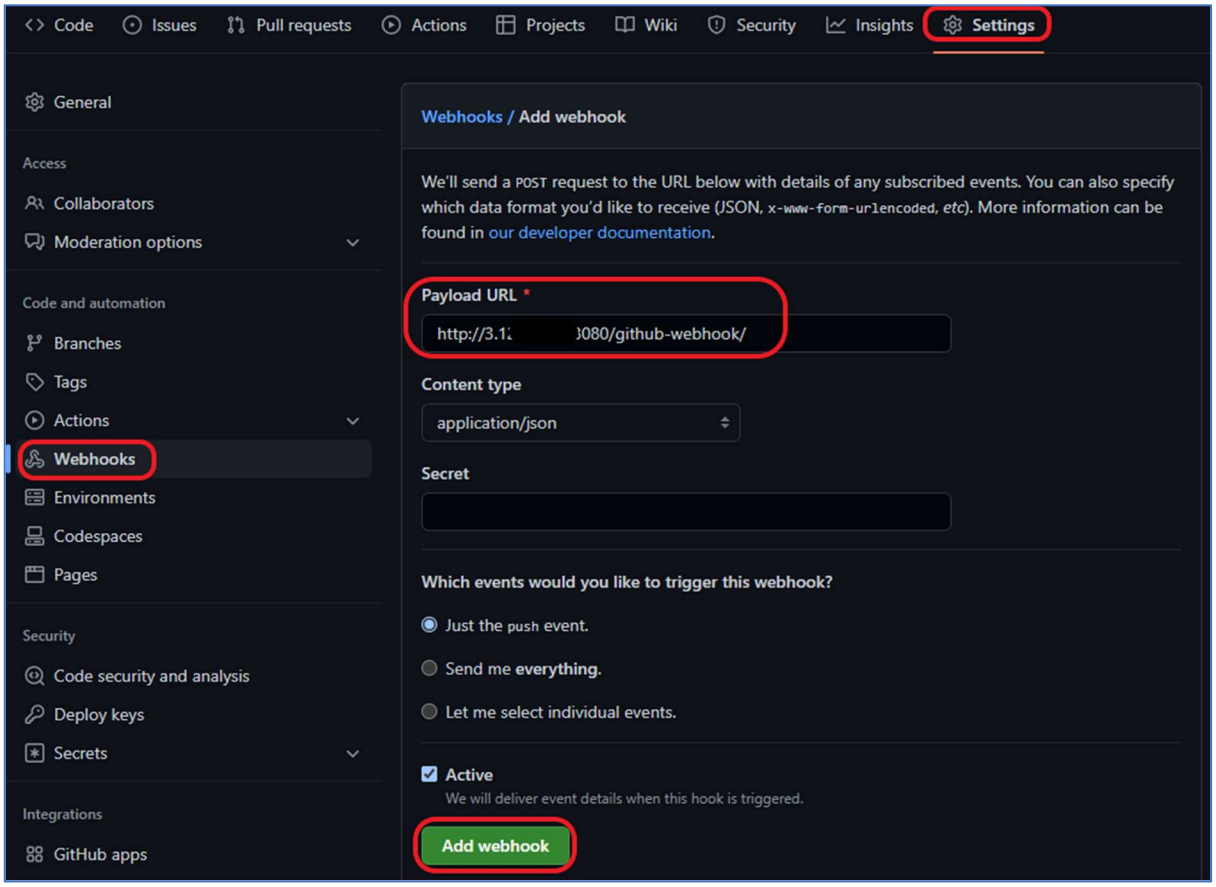
- Post-build Actions (*Publish Over SSH*)



- Record the settings on GitHub in the **Webhooks** section.





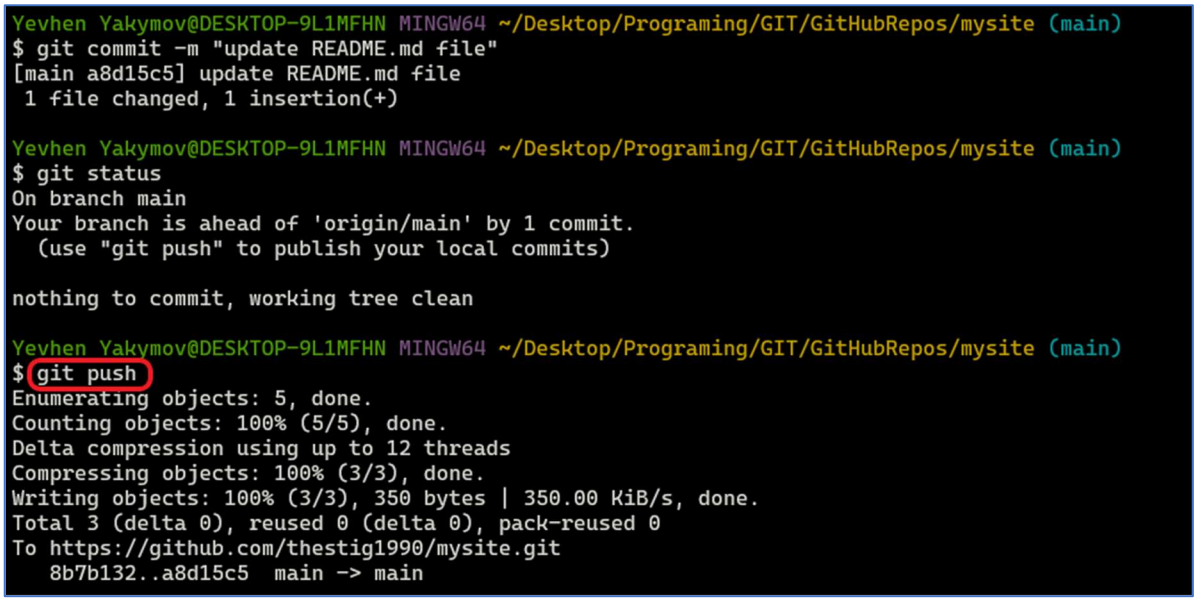


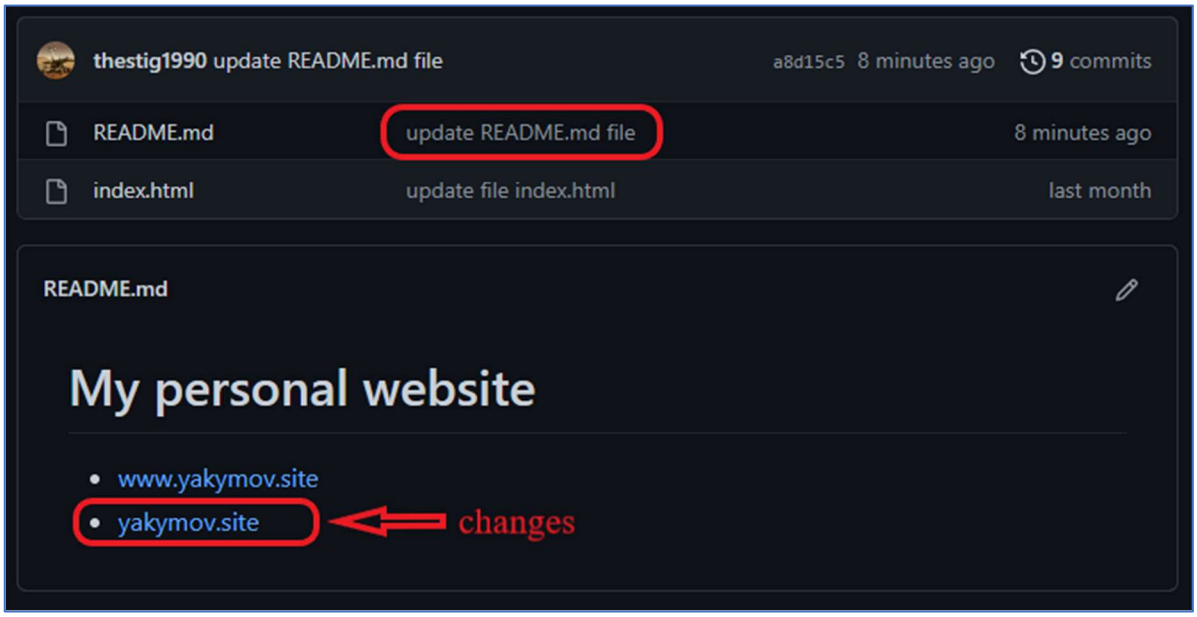
* **Payload URL** - http://jenkins_ip:port/github-webhook/
http://192.168.1.145:8080/github-webhook/ - example.

- Build job.

In our case jobs Build will start when we push some changes to the github repo

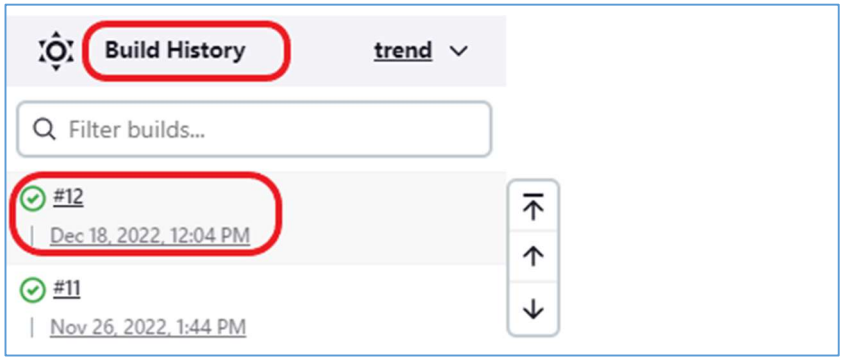
- Push some changes to the github repo



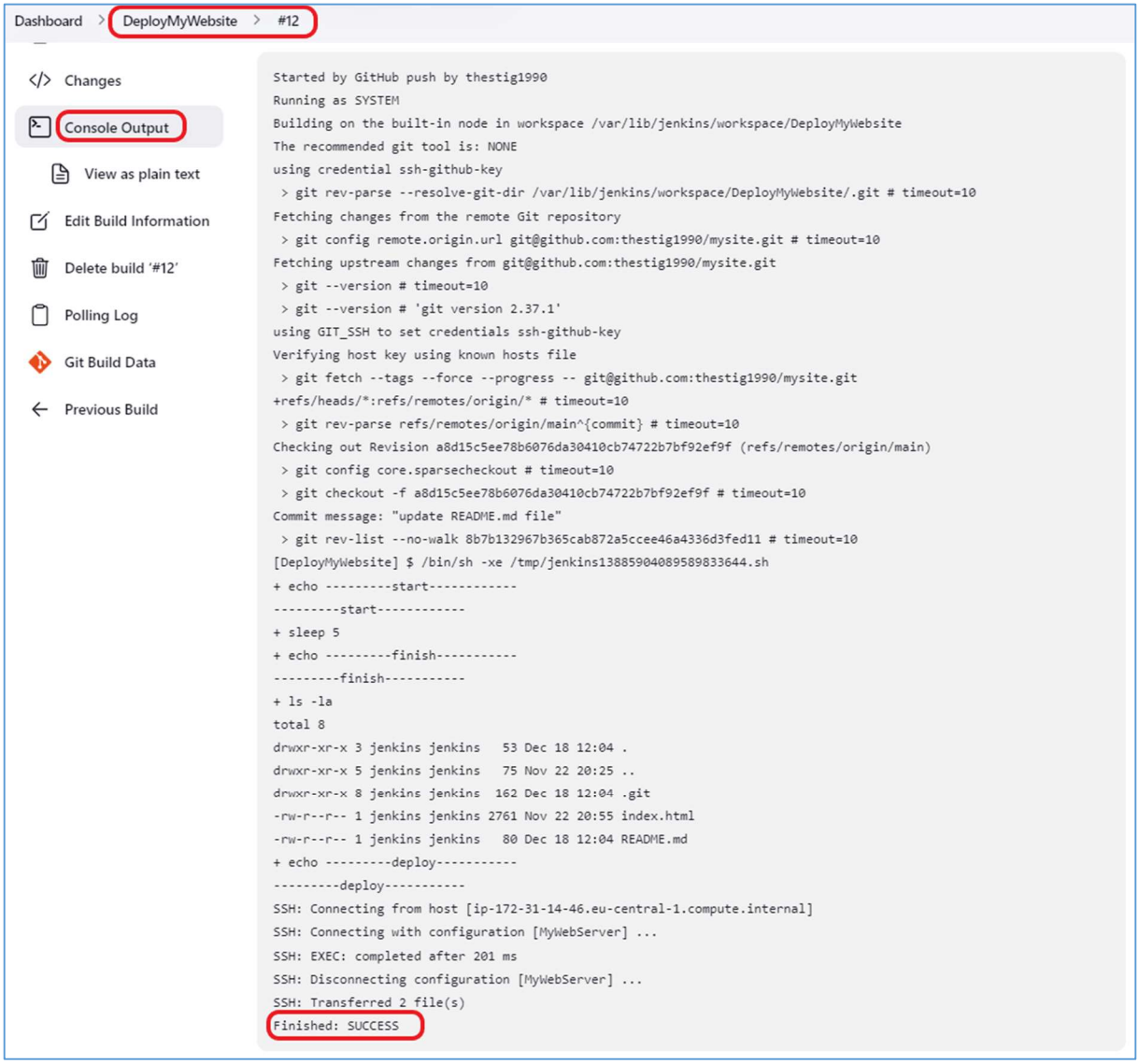


- Job description

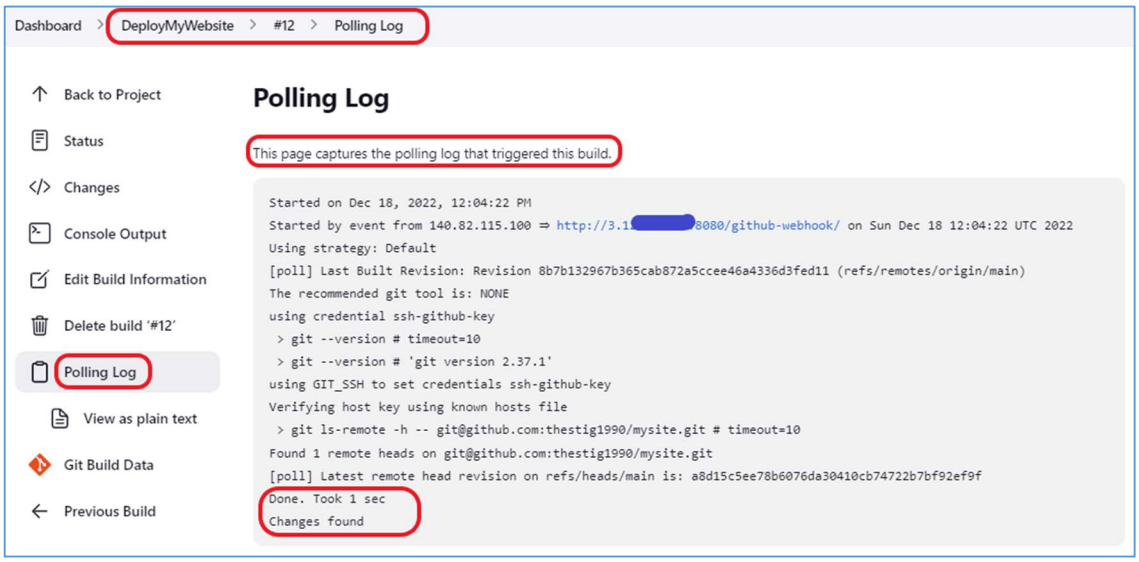
- Build History



- Console output



- Polling Log



- Checking the availability of the website

8. Jenkins nodes(agent/~~slave~~).

Jenkins nodes(agent):

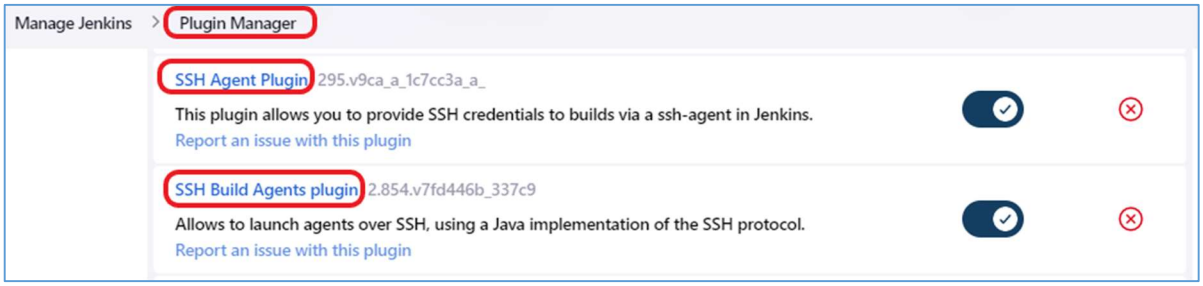
- Run more builds on Jenkins nodes(agent)
- Decrease load to Jenkins controller
- Differrent OS'es can be used on Jenkins nodes(agent) for different application

Jenkins controller require:

- Java instalation on Jenkins agent

```
$ sudo apt-get  install openjdk-11-jdk  (for Ubuntu)
```

- Network connection beetwen Jenkins agent and Jenkins controller
- Jenkins controller needs two plugins to communicate with Jenkins agent:



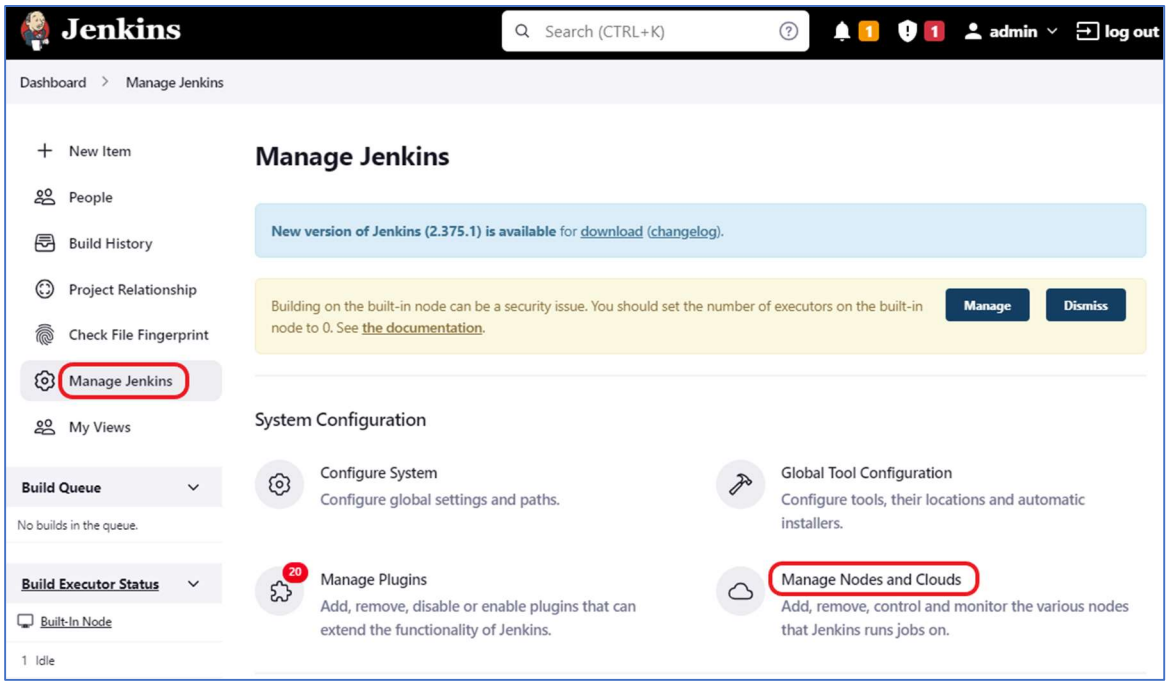
- You must generate **ssh key** on Jenkins controller and copy **public key** to the Jenkins agent before their usage will be available:

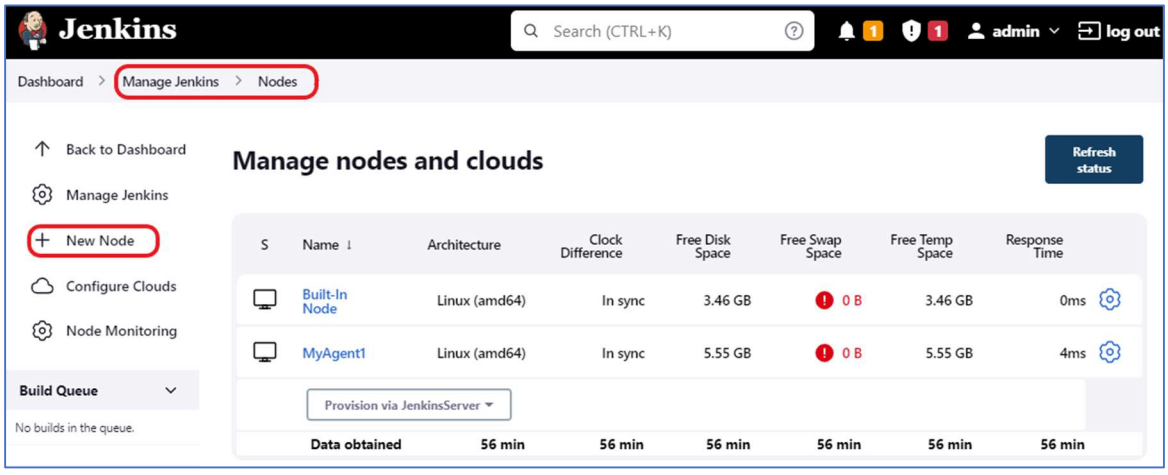
```
$ ssh-keygen -t rsa -f "key_name"
$ ssh-copy-id "user_name@host_name"
```

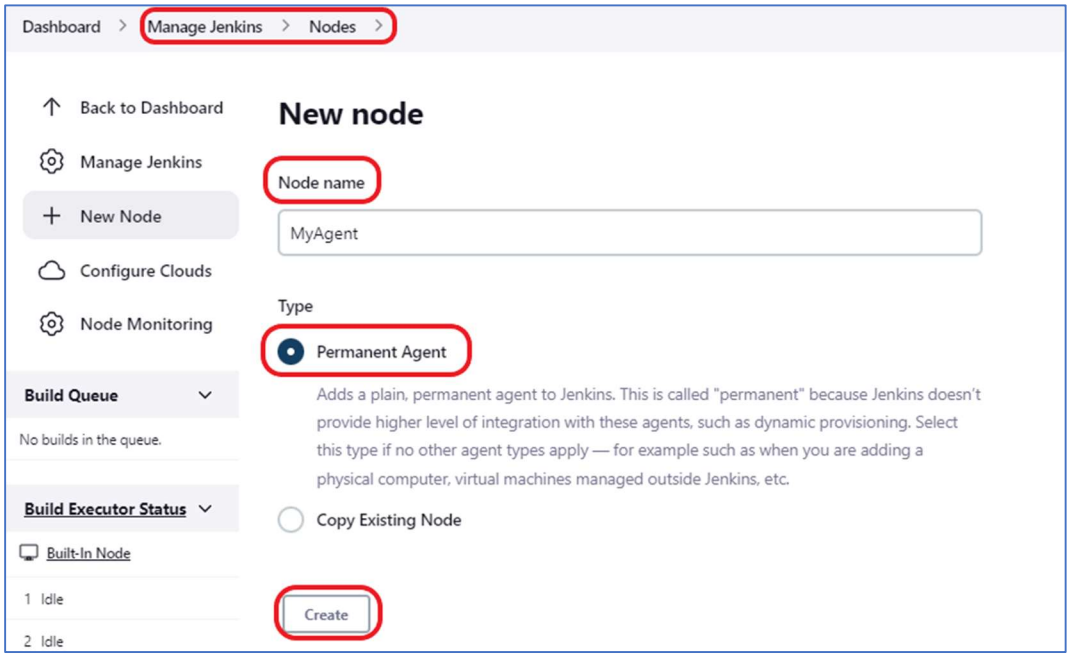
- On Jenkins agent (Ubuntu) should be done:

```
$ sudo apt apdate                          #update repo list
 *$ ip a                                   #check IP for SSH
 *$ pwd                                    #check where are you now
 *$ mkdir -p /home/jenkins                 #home directory for Jenkins
$ sudo apt-get install openjdk-11-jdk      #needed for Jenkins
```

Creating Jenkins nodes(agent):

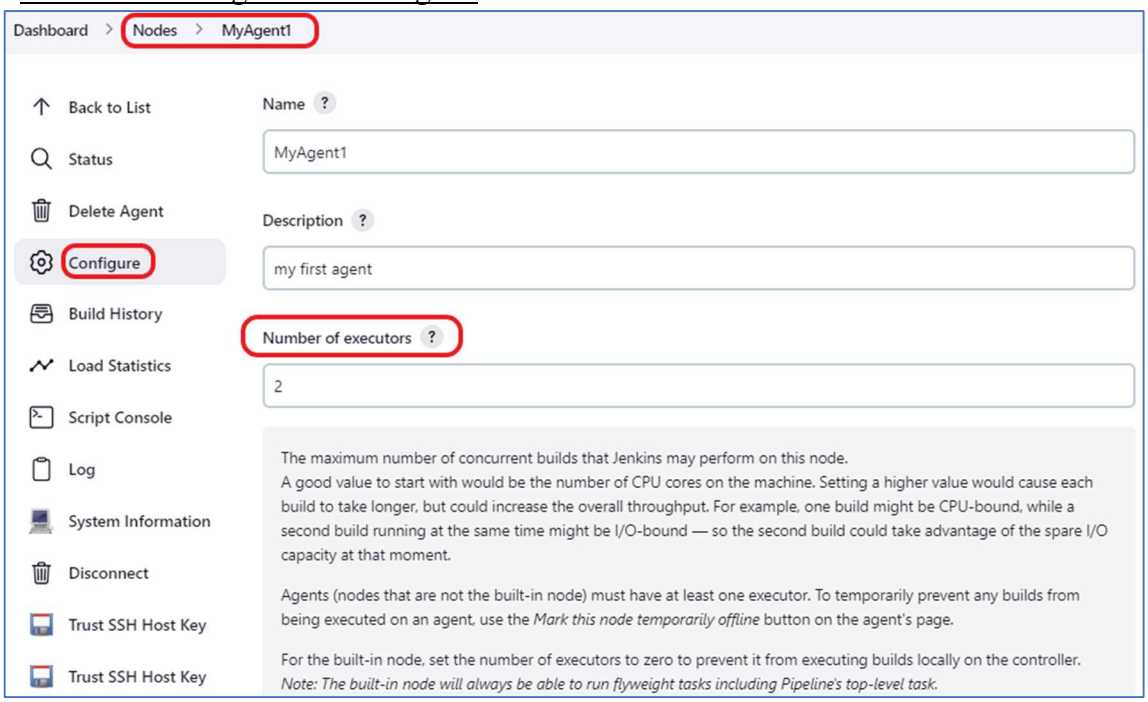


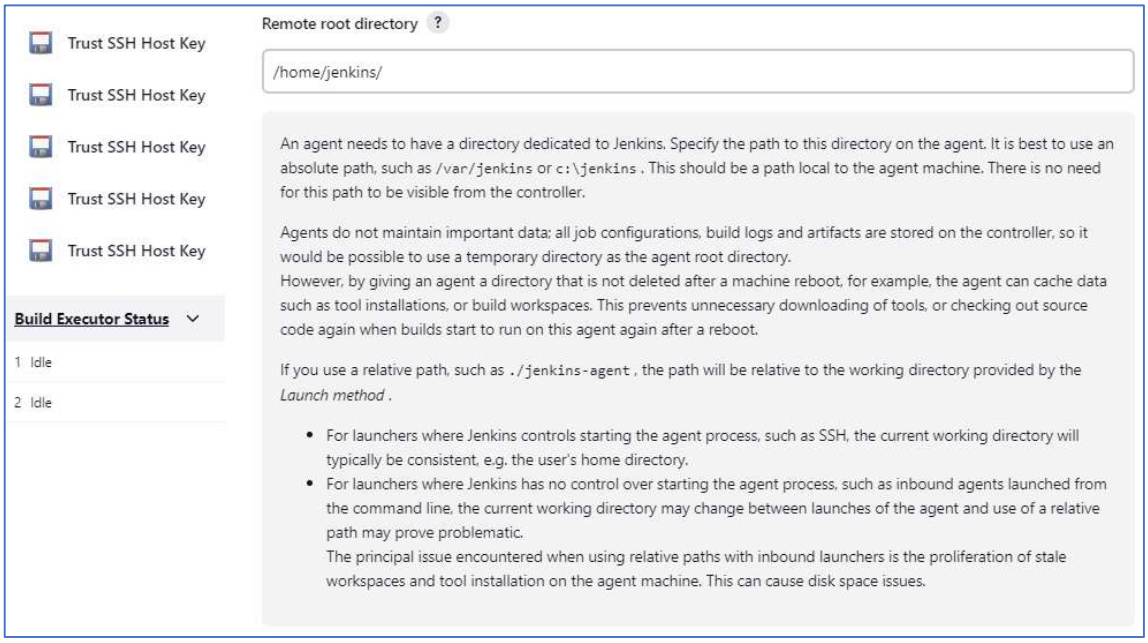


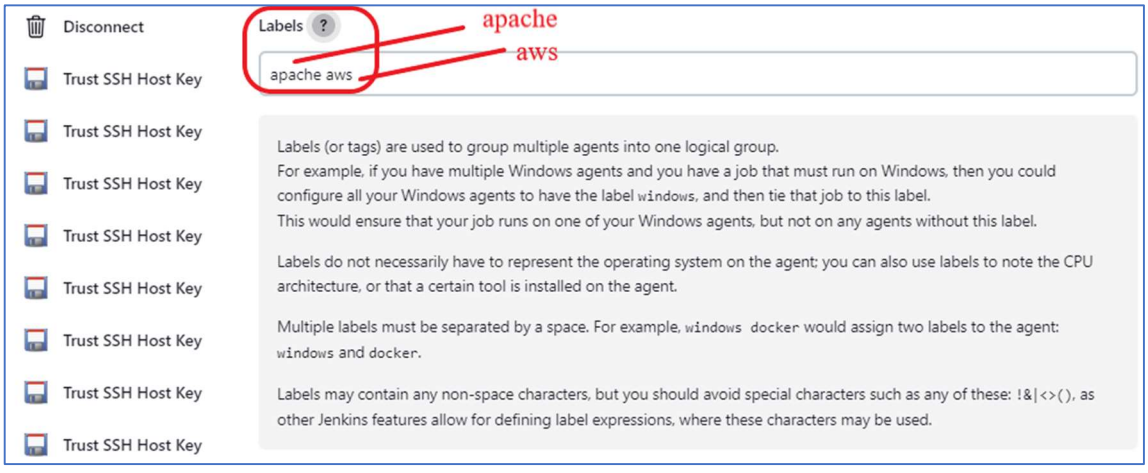


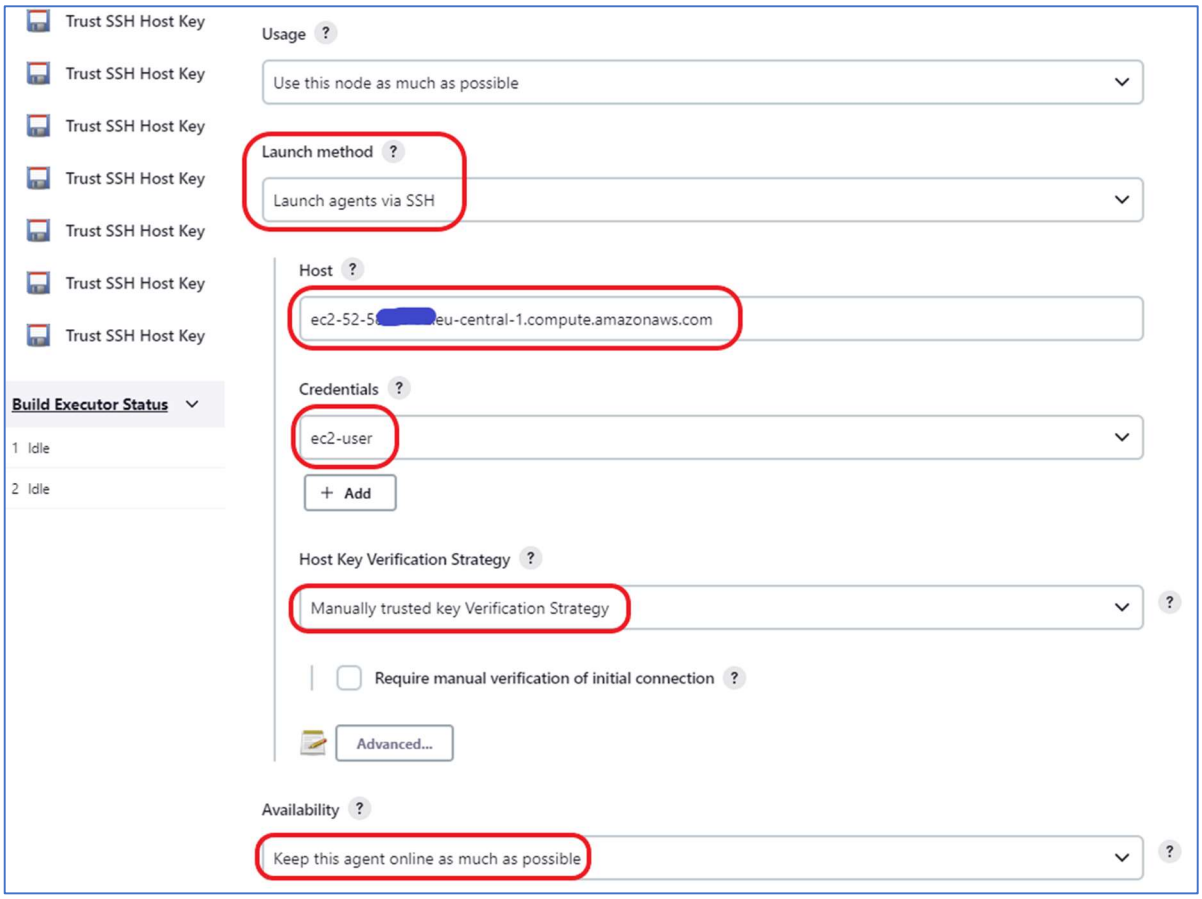
9. Simple job using Jenkins agent.

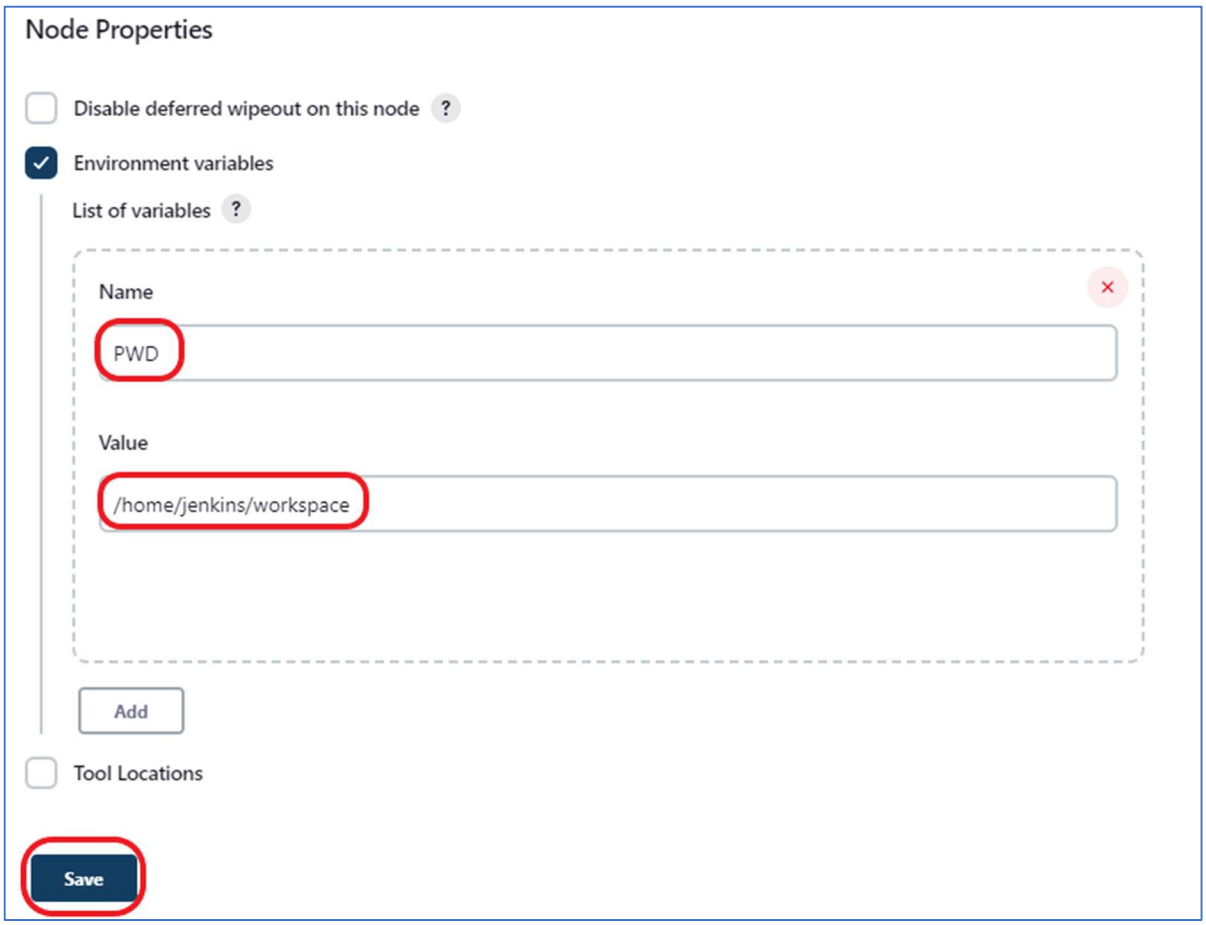
- Create and configure Jenkins agent.



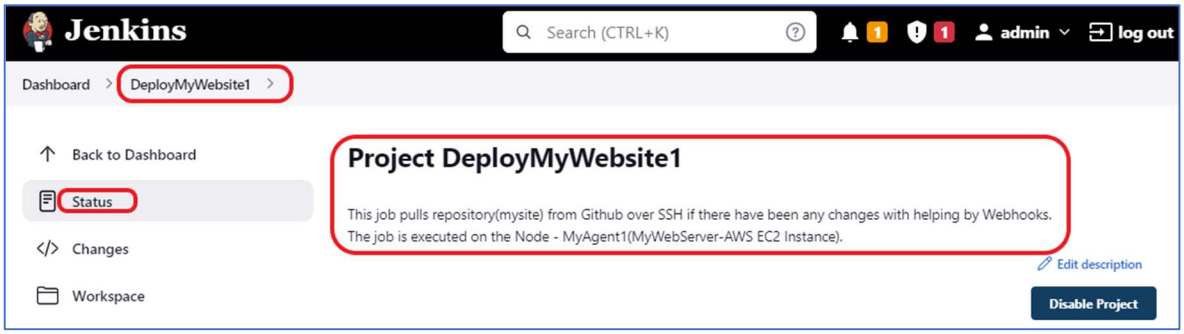


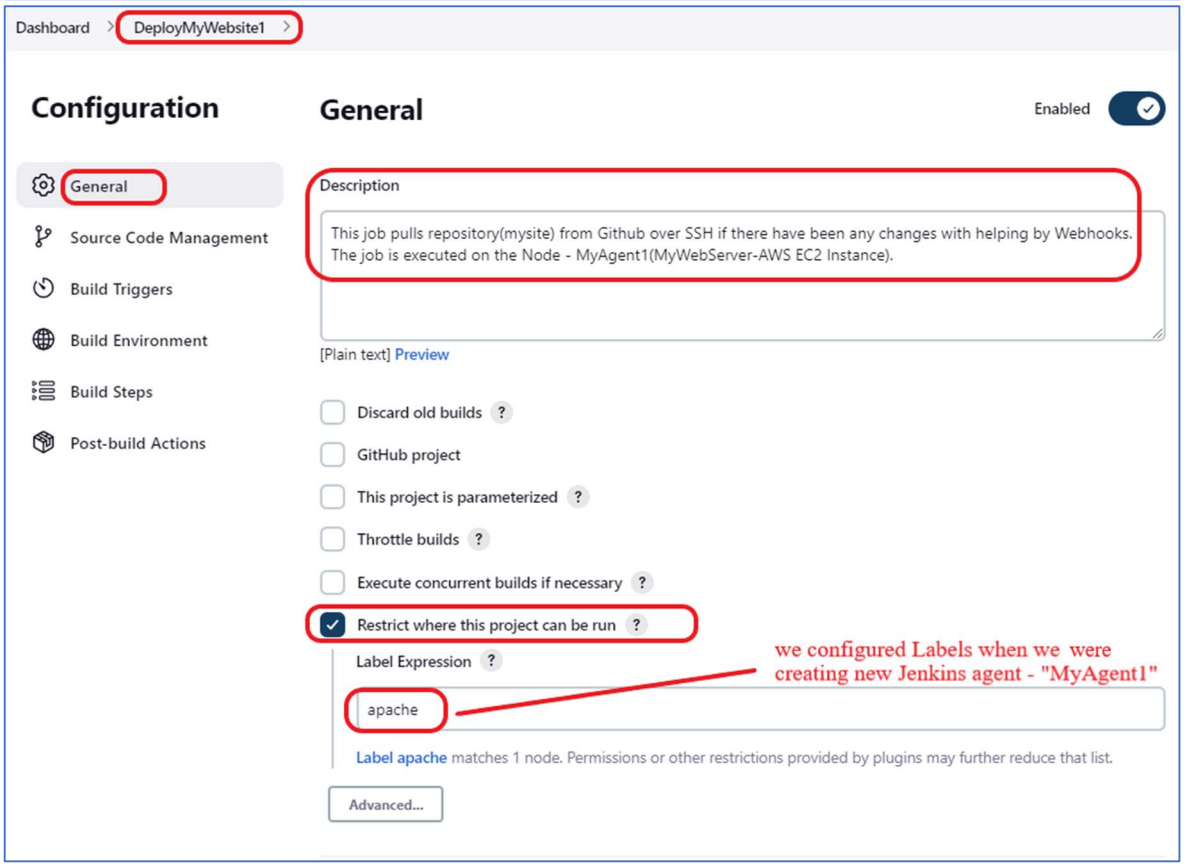


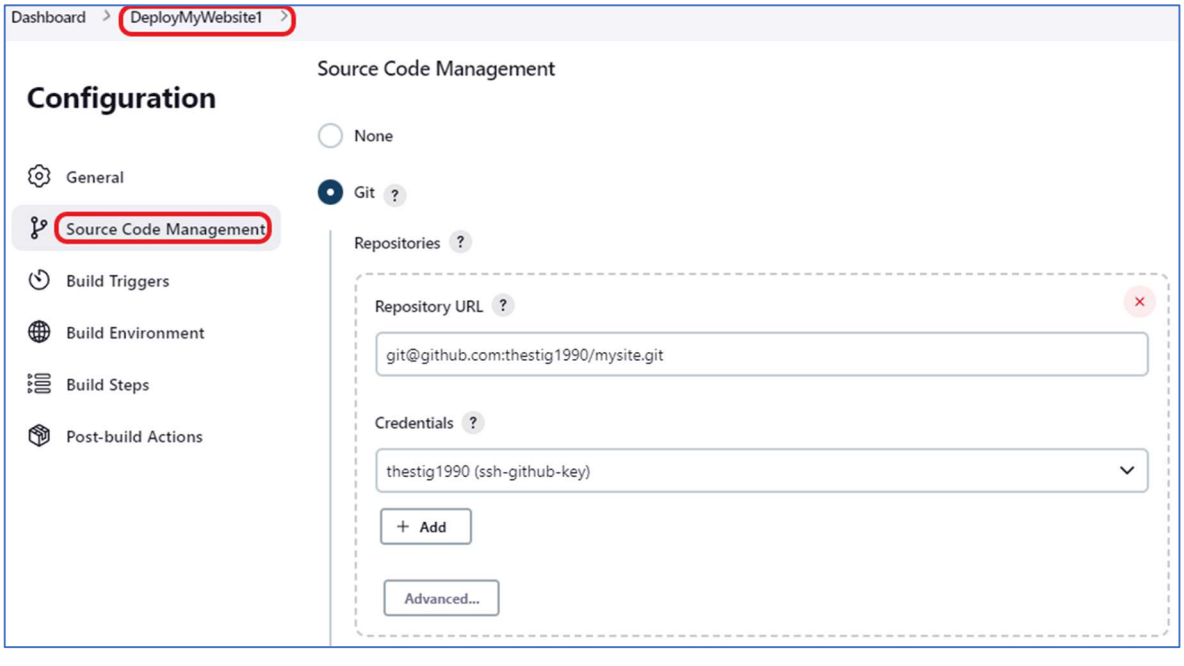

Trust SSH Host Key
Trust SSH Host Key
Trust SSH Host Key
Trust SSH Host Key
Trust SSH Host Key
Trust SSH Host Key
Trust SSH Host Key
Build Executor Status
1 Idle
2 Idle
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
ec2-52-5 eu-central-1.compute.amazonaws.com
Credentials
ec2-user
Add
Host Key Verification Strategy
Manually trusted key Verification Strategy
Require manual verification of initial connection
Advanced...
Availability
Keep this agent online as much as possible

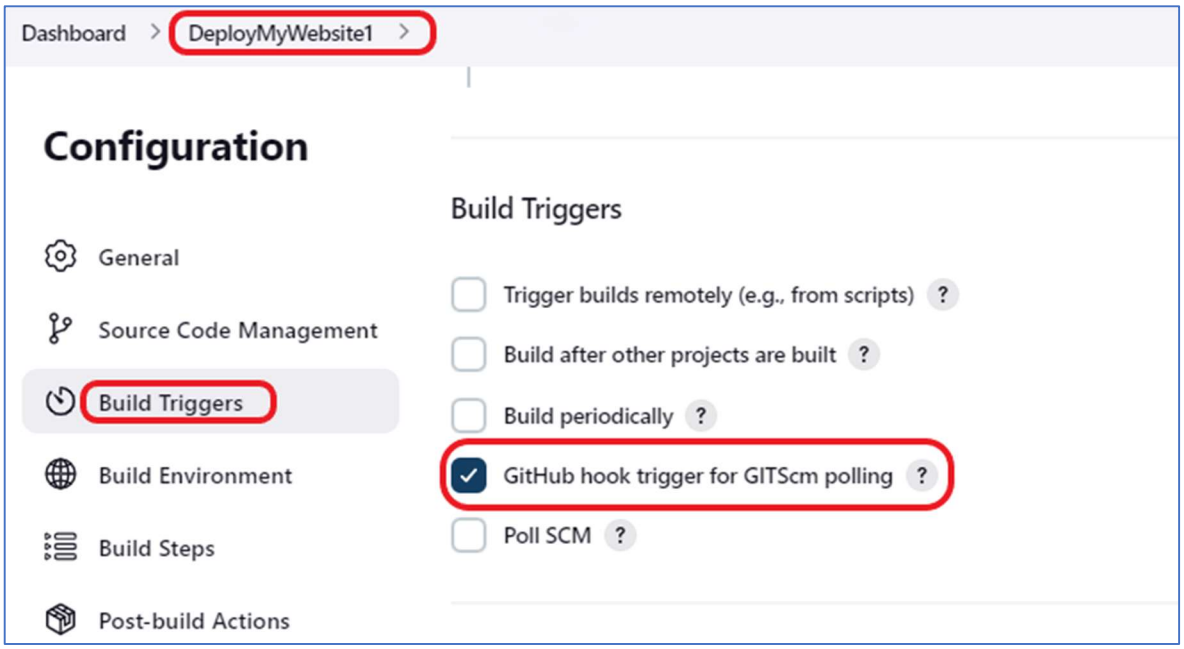

Node Properties
Disable deferred wipeout on this node
Environment variables
List of variables
Name
PWD
Value
/home/jenkins/workspace
Add
Tool Locations
Save

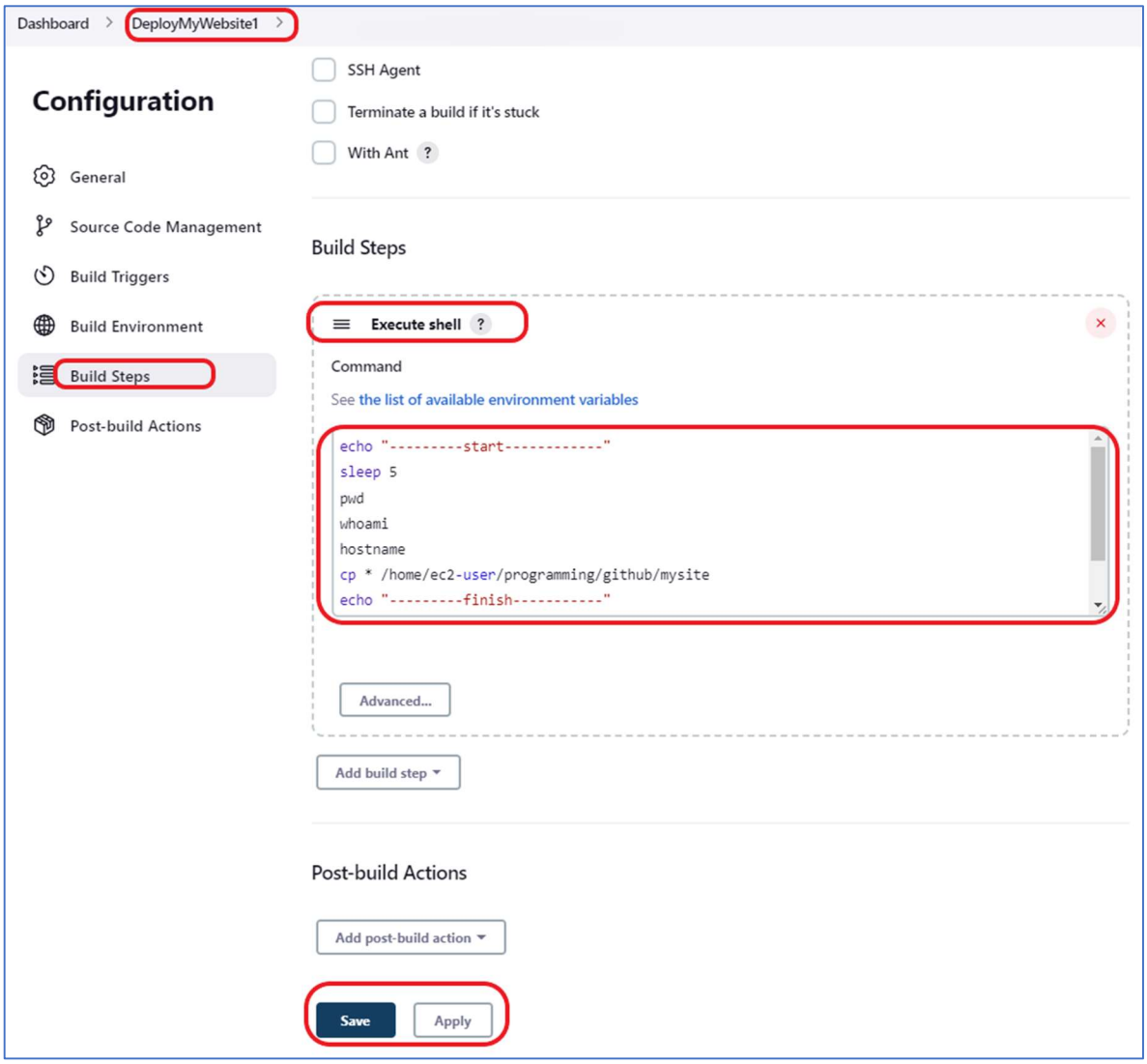
- Create job


Jenkins
Search (CTRL+K)
admin
log out
Dashboard
DeployMyWebsite1
Back to Dashboard
Status
Changes
Workspace
Project DeployMyWebsite1
This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks.
The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).
Edit description
Disable Project

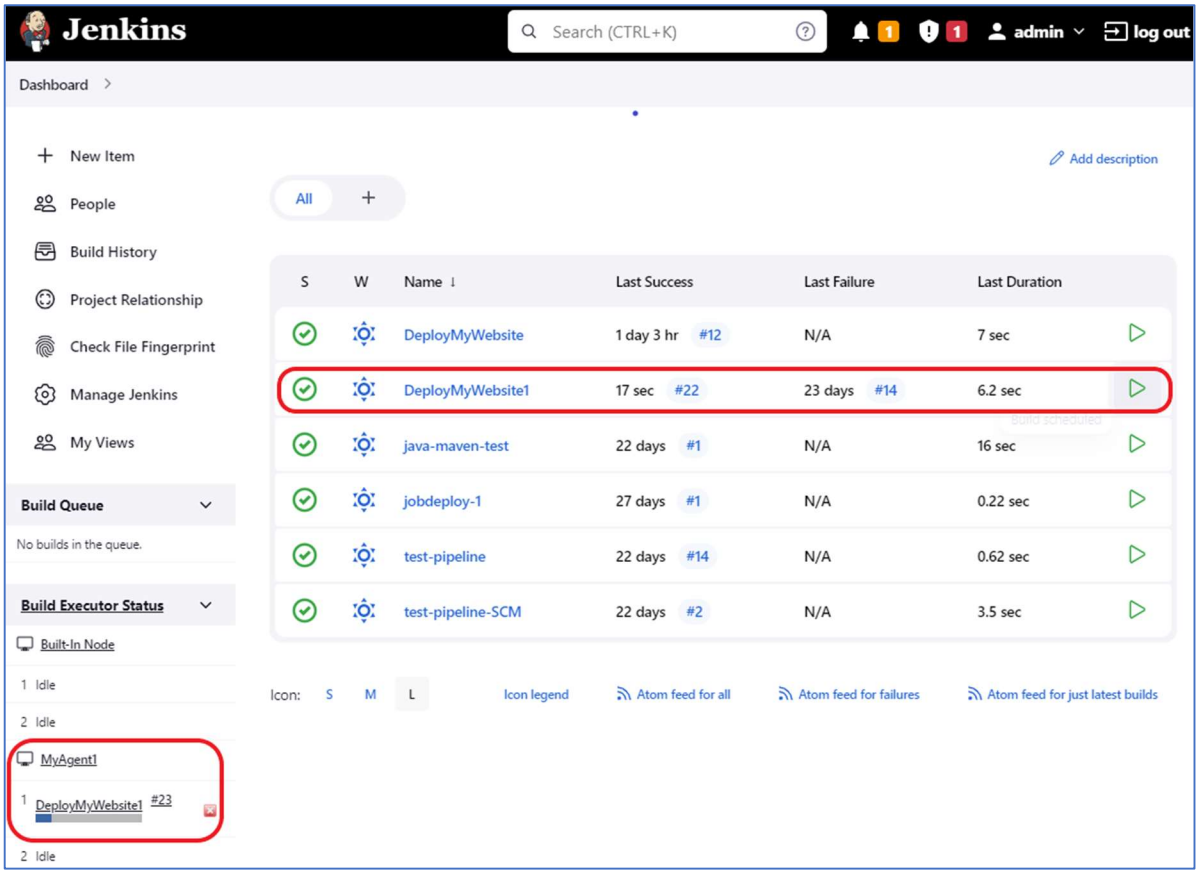

Dashboard
DeployMyWebsite1
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks.
The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).
[Plain text] Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
we configured Labels when we were creating new Jenkins agent - "MyAgent1"
apache
Label apache matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.
Advanced...

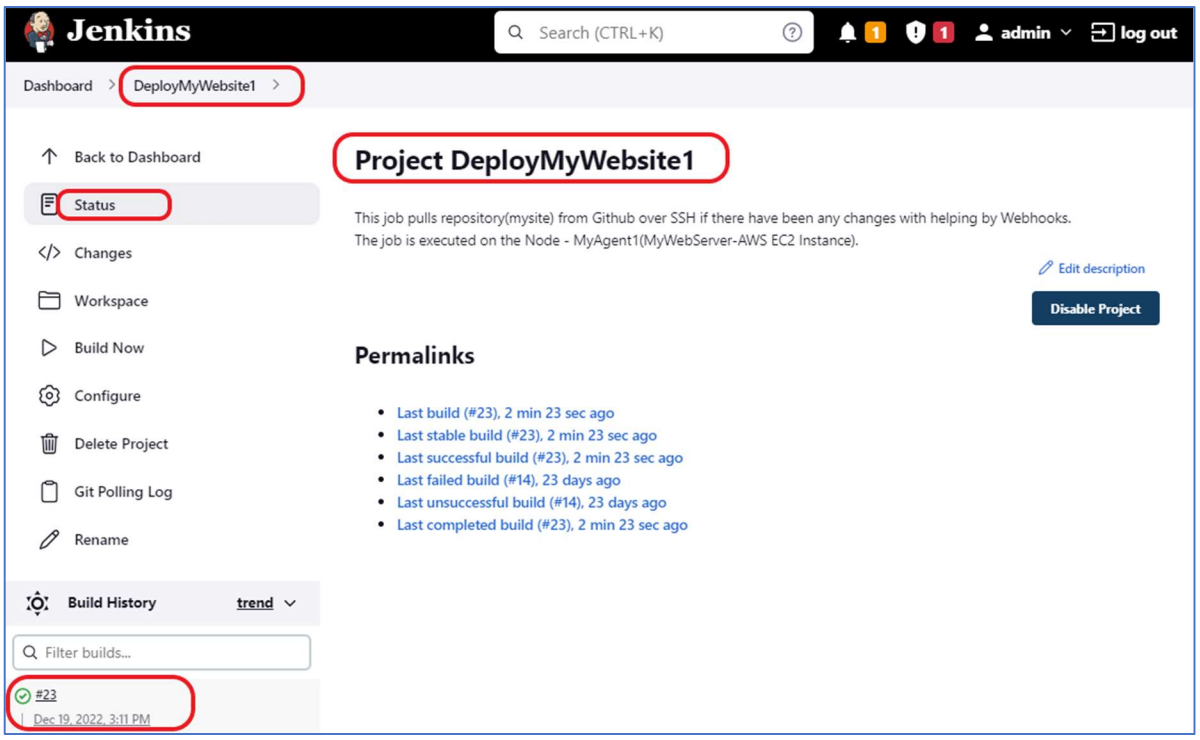

Dashboard
DeployMyWebsite1
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
git@github.com:thestig1990/mysite.git
Credentials
thestig1990 (ssh-github-key)
Add
Advanced...


Dashboard
DeployMyWebsite1
Configuration
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

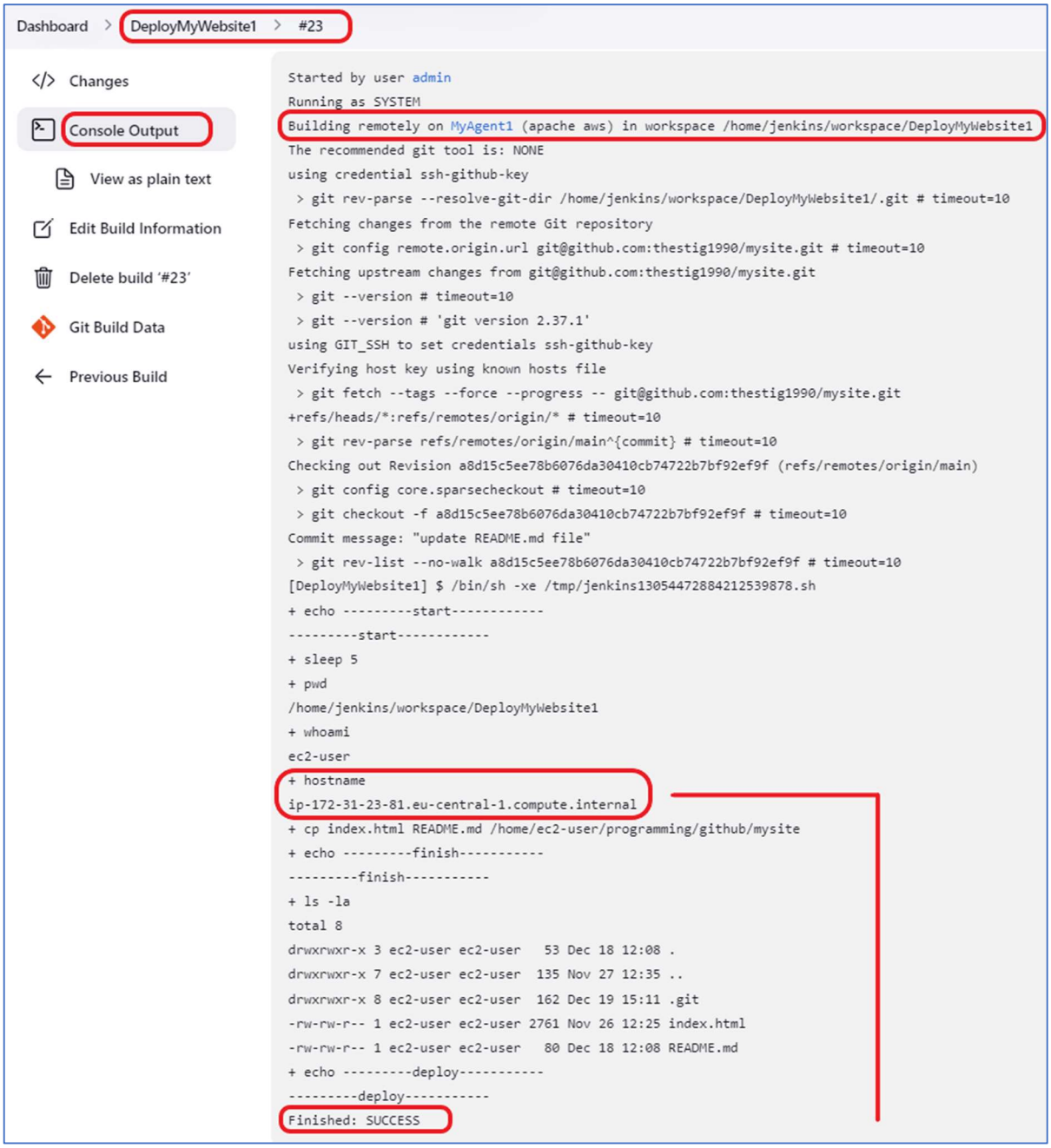

Dashboard
DeployMyWebsite1
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
SSH Agent
Terminate a build if it's stuck
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
echo "---------start-----------"
sleep 5
pwd
whoami
hostname
cp * /home/ec2-user/programming/github/mysite
echo "---------finish----------"
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply

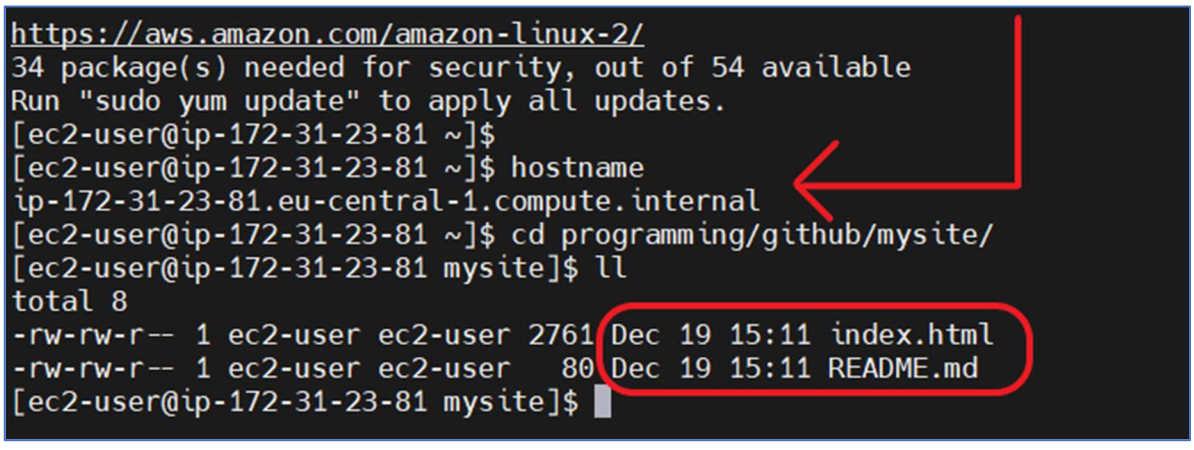
- Build job


Jenkins
Search (CTRL+K)
admin
log out
Dashboard
New Item
People
Build History
Project Relationship
Check File Fingerprint
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node
1 Idle
2 Idle
MyAgent1
1 DeployMyWebsite1 #23
2 Idle
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
DeployMyWebsite
1 day 3 hr #12
N/A
7 sec
DeployMyWebsite1
17 sec #22
23 days #14
6.2 sec
java-maven-test
22 days #1
N/A
16 sec
jobdeploy-1
27 days #1
N/A
0.22 sec
test-pipeline
22 days #14
N/A
0.62 sec
test-pipeline-SCM
22 days #2
N/A
3.5 sec
Icon: S M L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds


Jenkins
Search (CTRL+K)
admin
log out
Dashboard
DeployMyWebsite1
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Git Polling Log
Rename
Build History
trend
Filter builds...
#23
Dec 19, 2022, 3:11 PM
Project DeployMyWebsite1
This job pulls repository(mysite) from Github over SSH if there have been any changes with helping by Webhooks.
The job is executed on the Node - MyAgent1(MyWebServer-AWS EC2 Instance).
Edit description
Disable Project
Permalinks
Last build (#23), 2 min 23 sec ago
Last stable build (#23), 2 min 23 sec ago
Last successful build (#23), 2 min 23 sec ago
Last failed build (#14), 23 days ago
Last unsuccessful build (#14), 23 days ago
Last completed build (#23), 2 min 23 sec ago

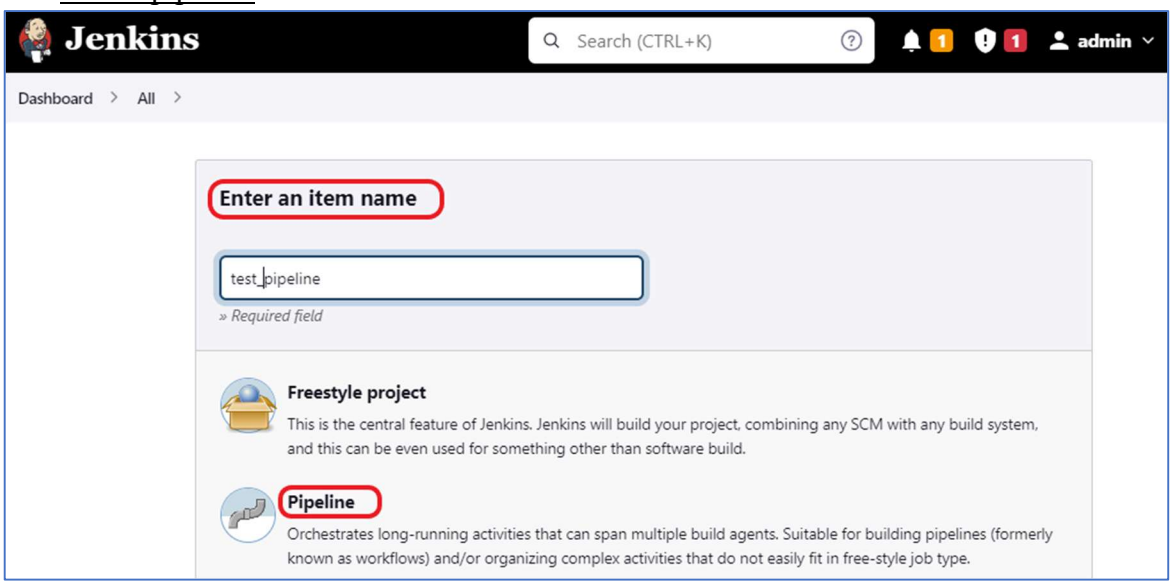
- Console output

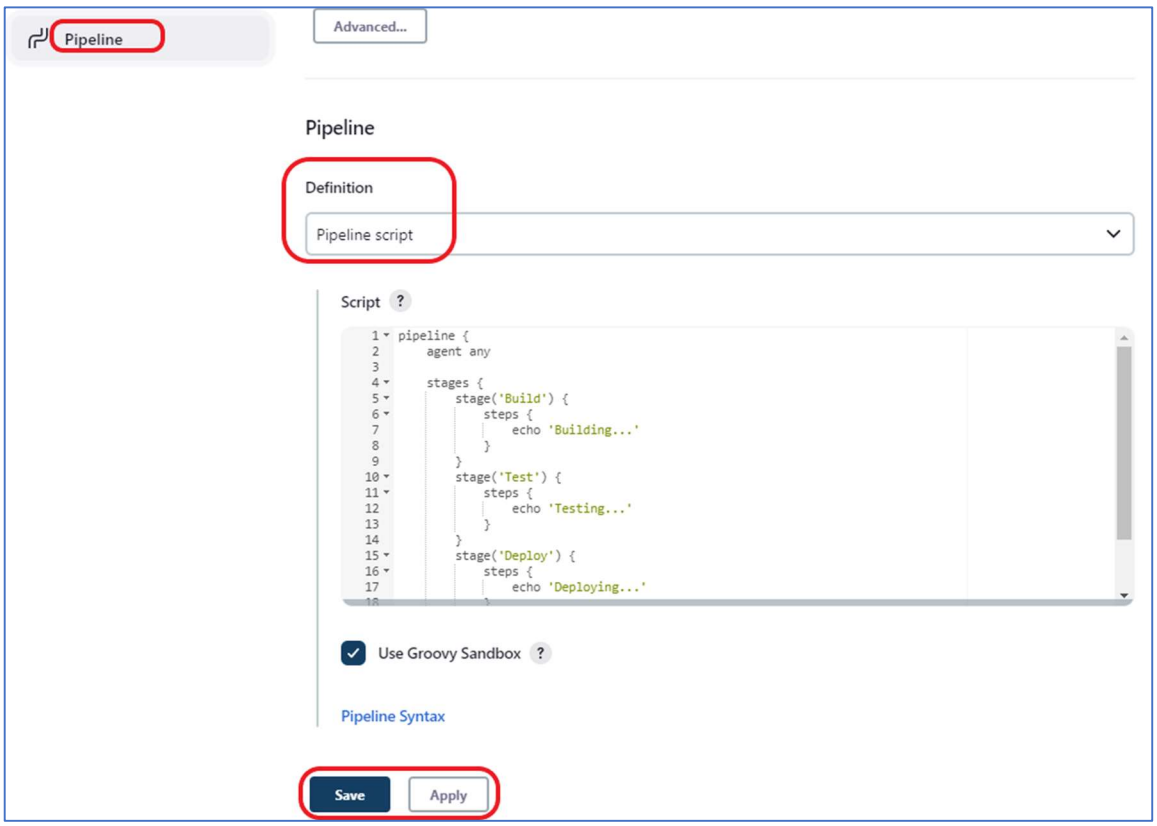




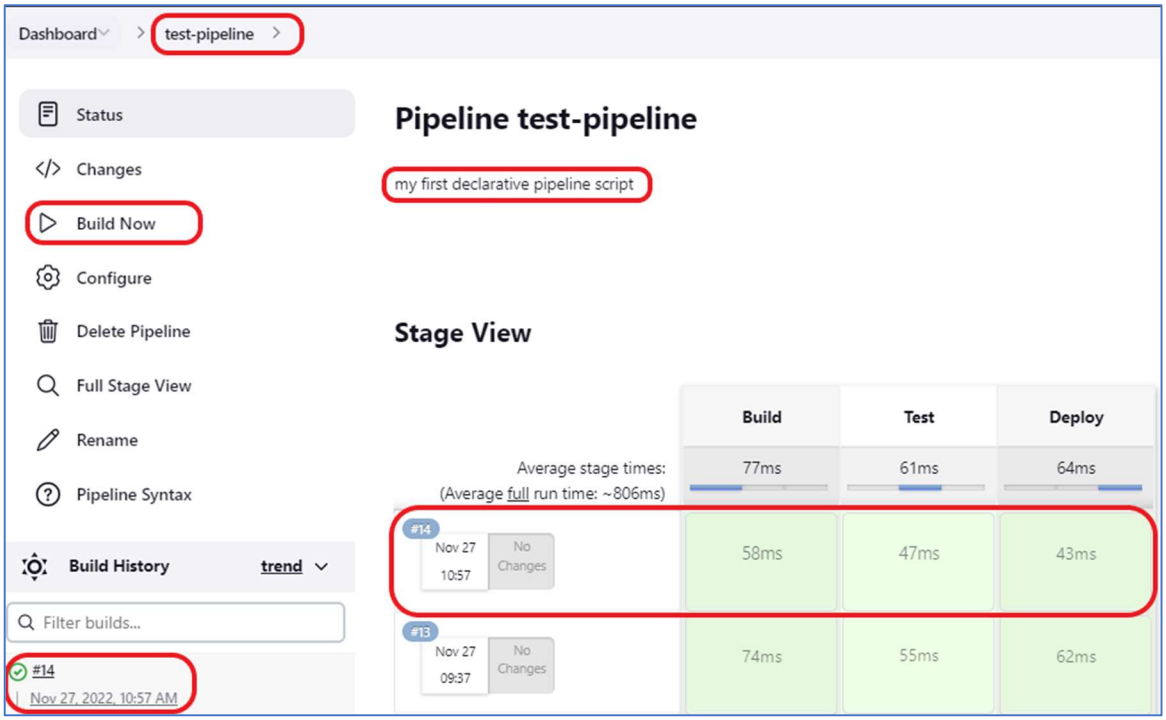
10. Simple Jenkis pipeline example.

- Create pipeline

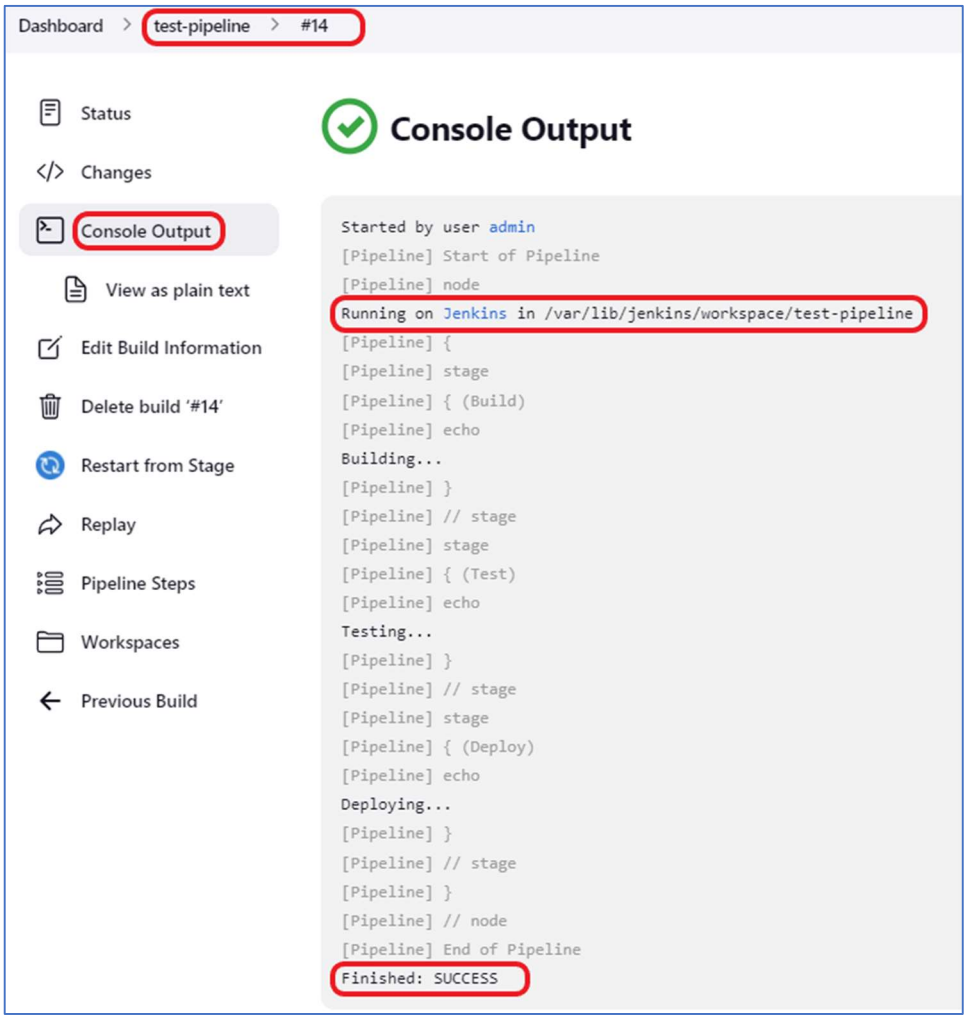




- Build pipeline

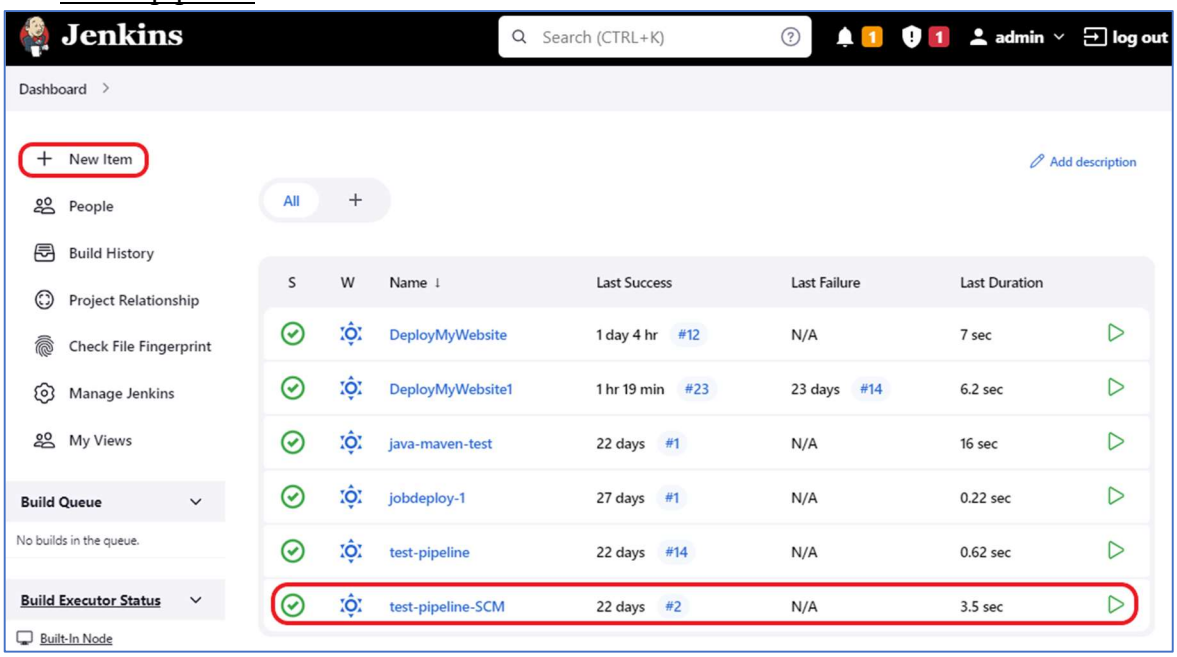


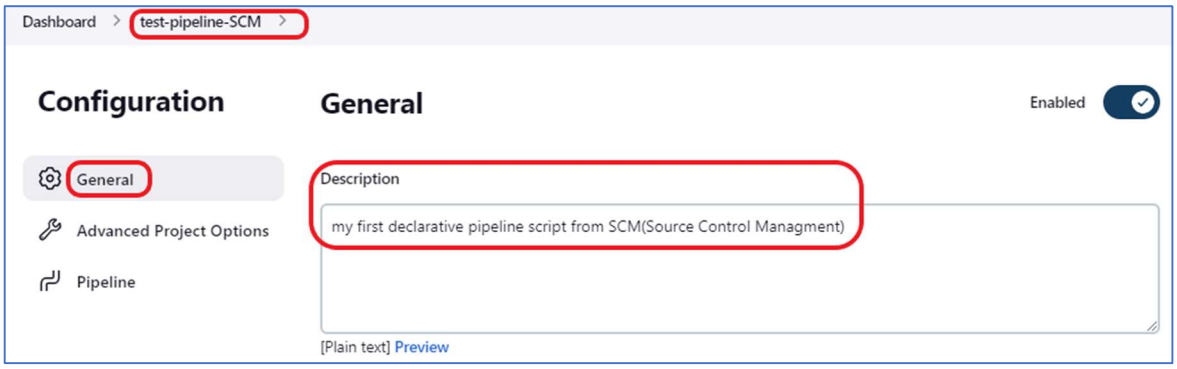
- Console output

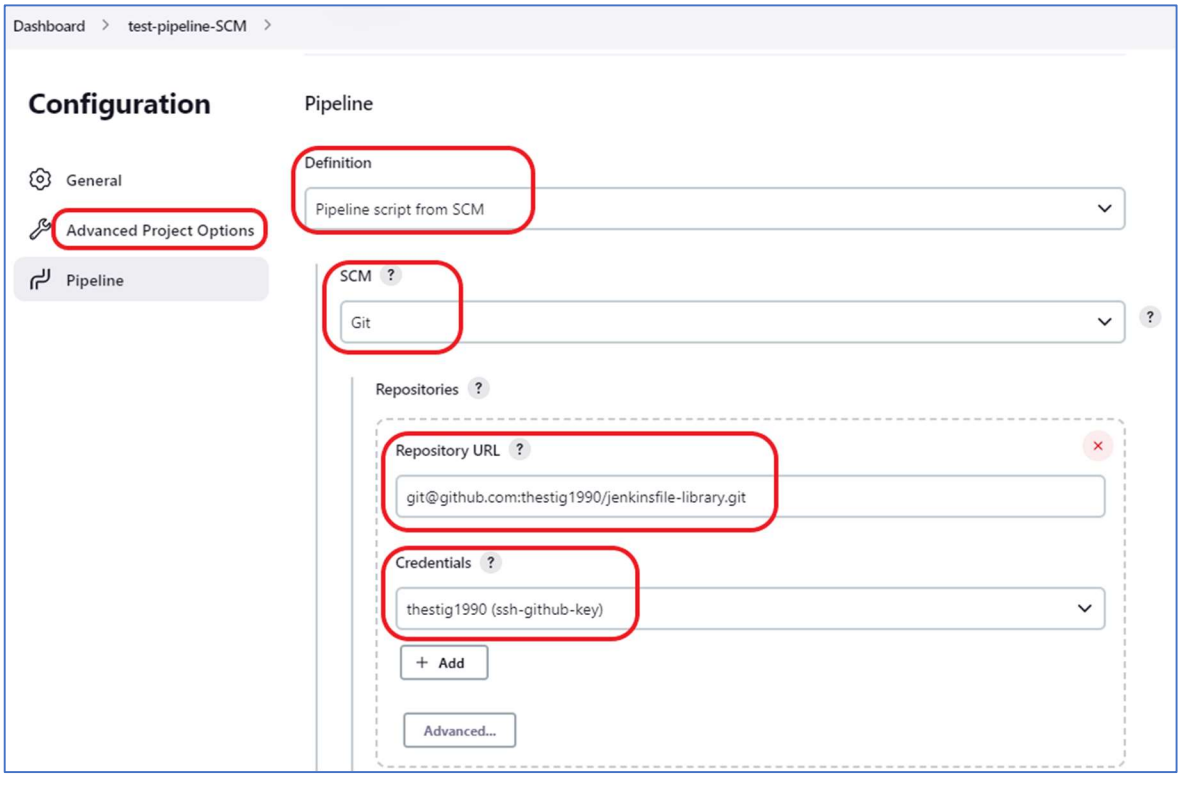


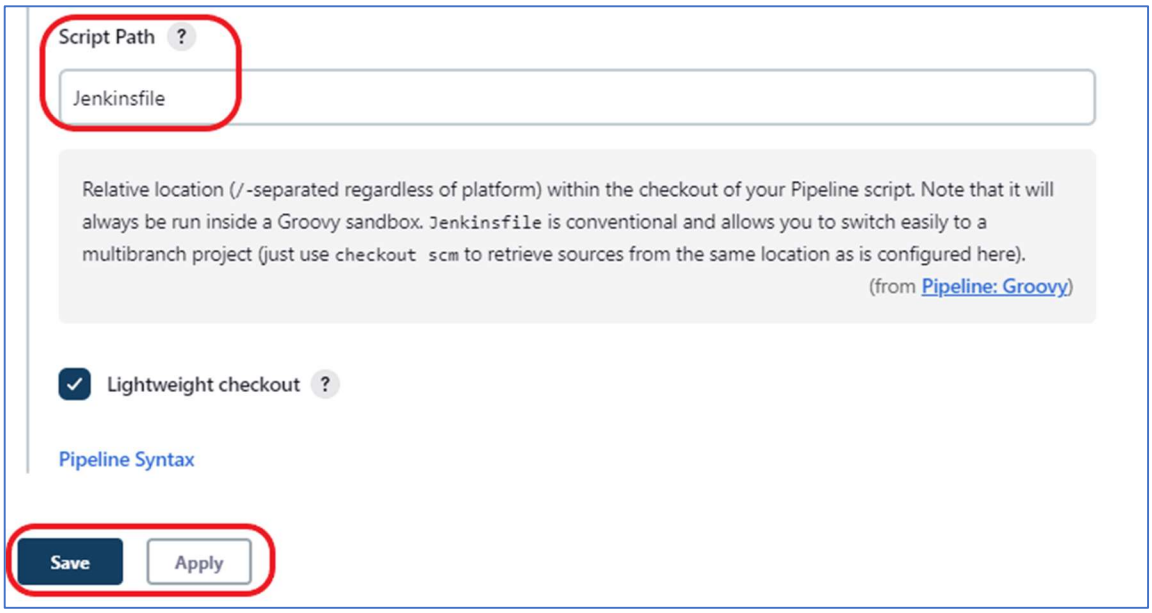
11. Simple Jenkis pipeline script from SCM (Source Control Managment).

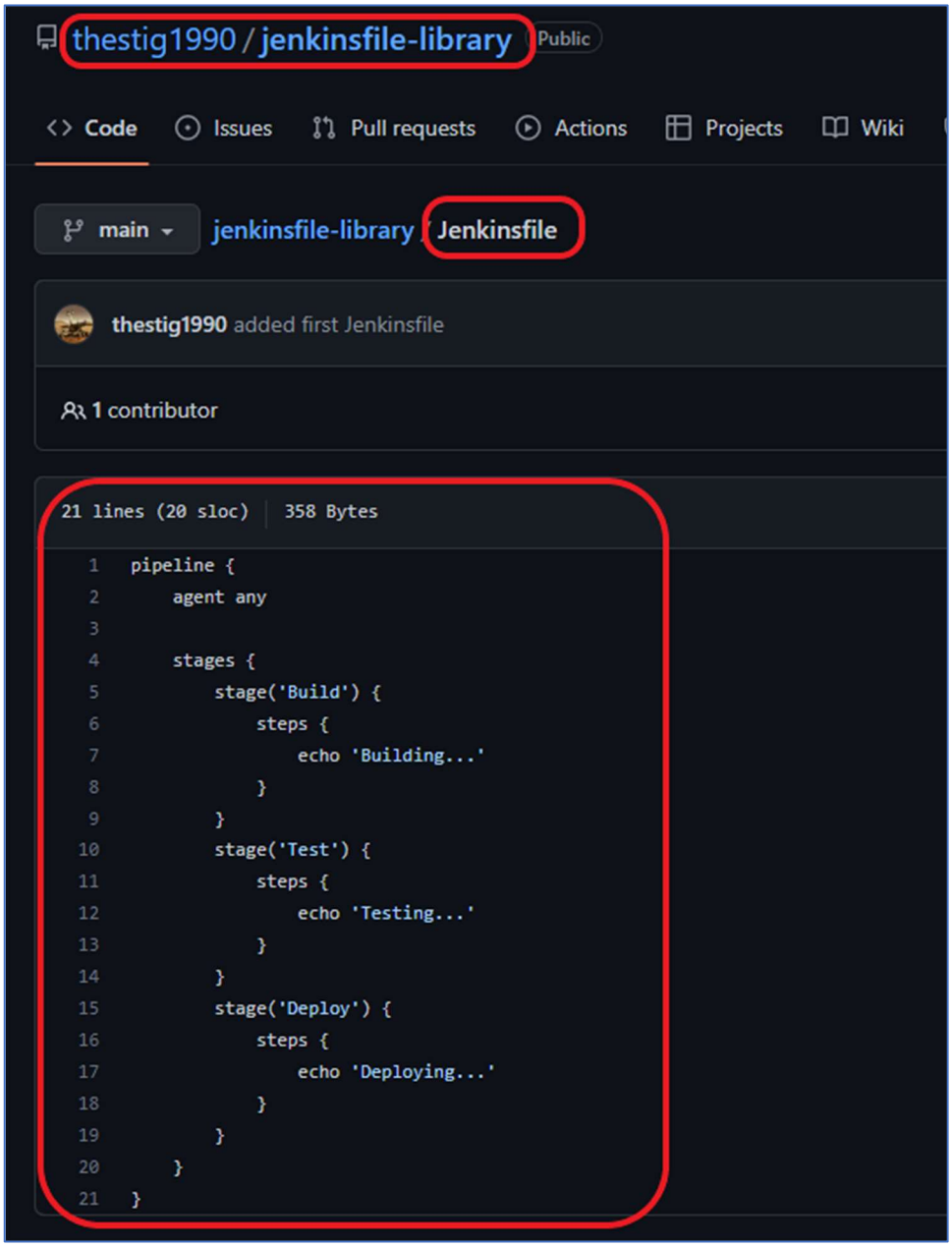
- Create pipeline



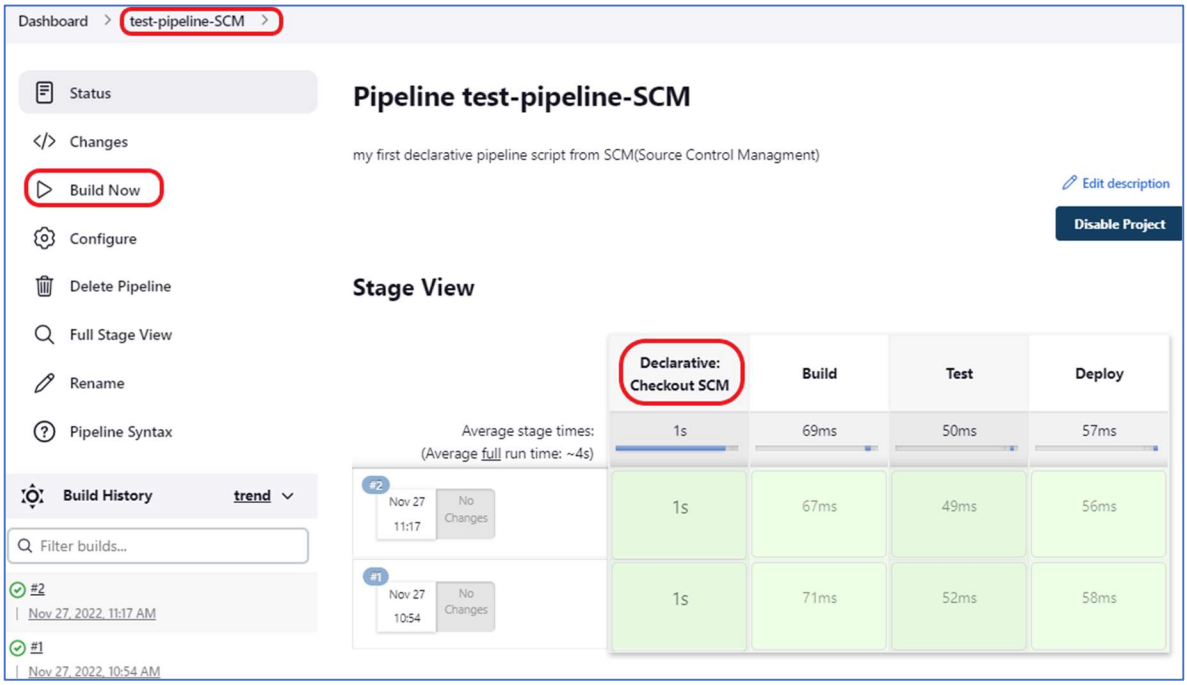




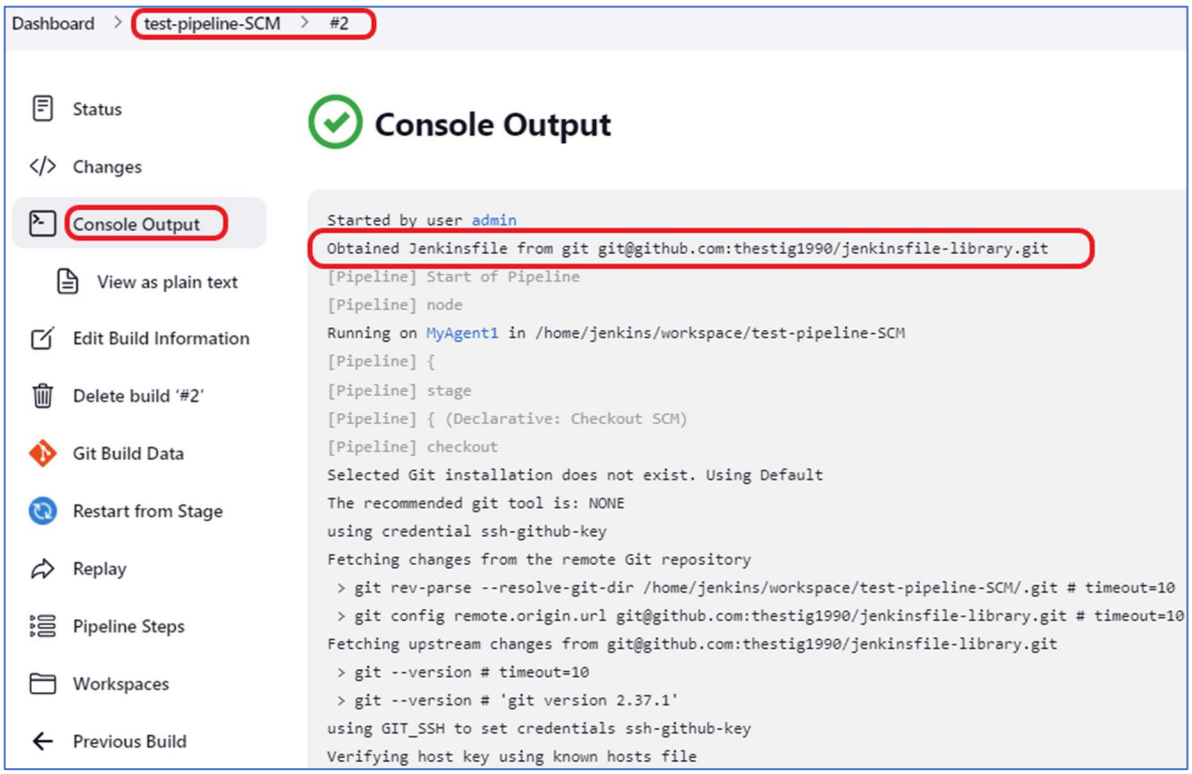

Script Path ?
Jenkinsfile
Relative location (/-separated regardless of platform) within the checkout of your Pipeline script. Note that it will always be run inside a Groovy sandbox. Jenkinsfile is conventional and allows you to switch easily to a multibranch project (just use checkout scm to retrieve sources from the same location as is configured here).
(from Pipeline: Groovy)
Lightweight checkout ?
Pipeline Syntax
Save
Apply

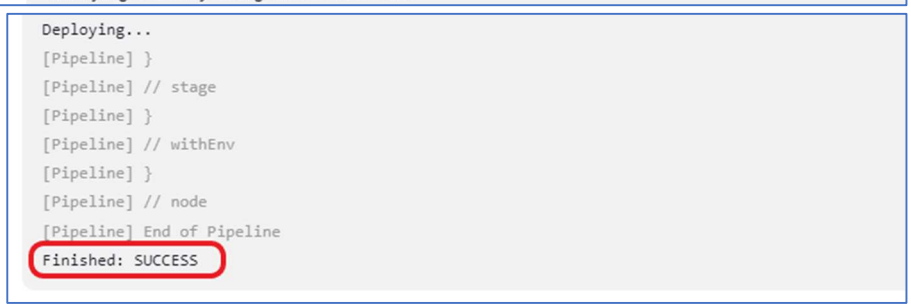

thestig1990 / jenkinsfile-library
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
main
jenkinsfile-library / Jenkinsfile
thestig1990 added first Jenkinsfile
1 contributor
21 lines (20 sloc) | 358 Bytes
1 pipeline {
2 agent any
3
4 stages {
5 stage('Build') {
6 steps {
7 echo 'Building...'
8 }
9 }
10 stage('Test') {
11 steps {
12 echo 'Testing...'
13 }
14 }
15 stage('Deploy') {
16 steps {
17 echo 'Deploying...'
18 }
19 }
20 }
21 }

- Build pipeline



- Console output

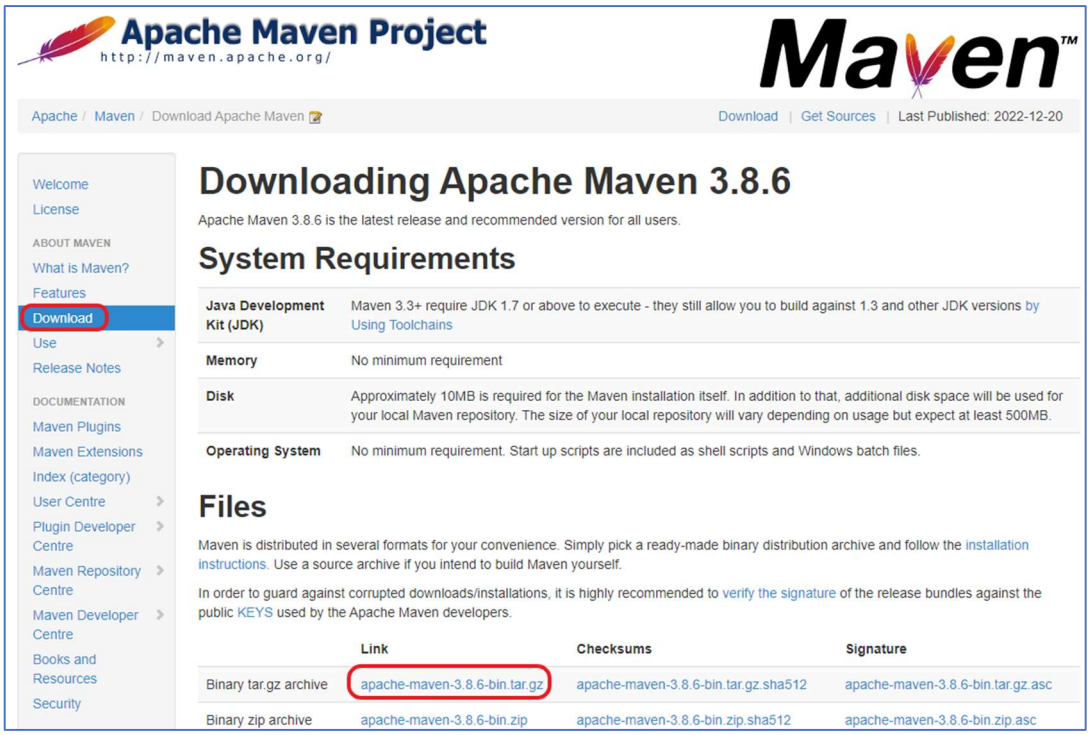


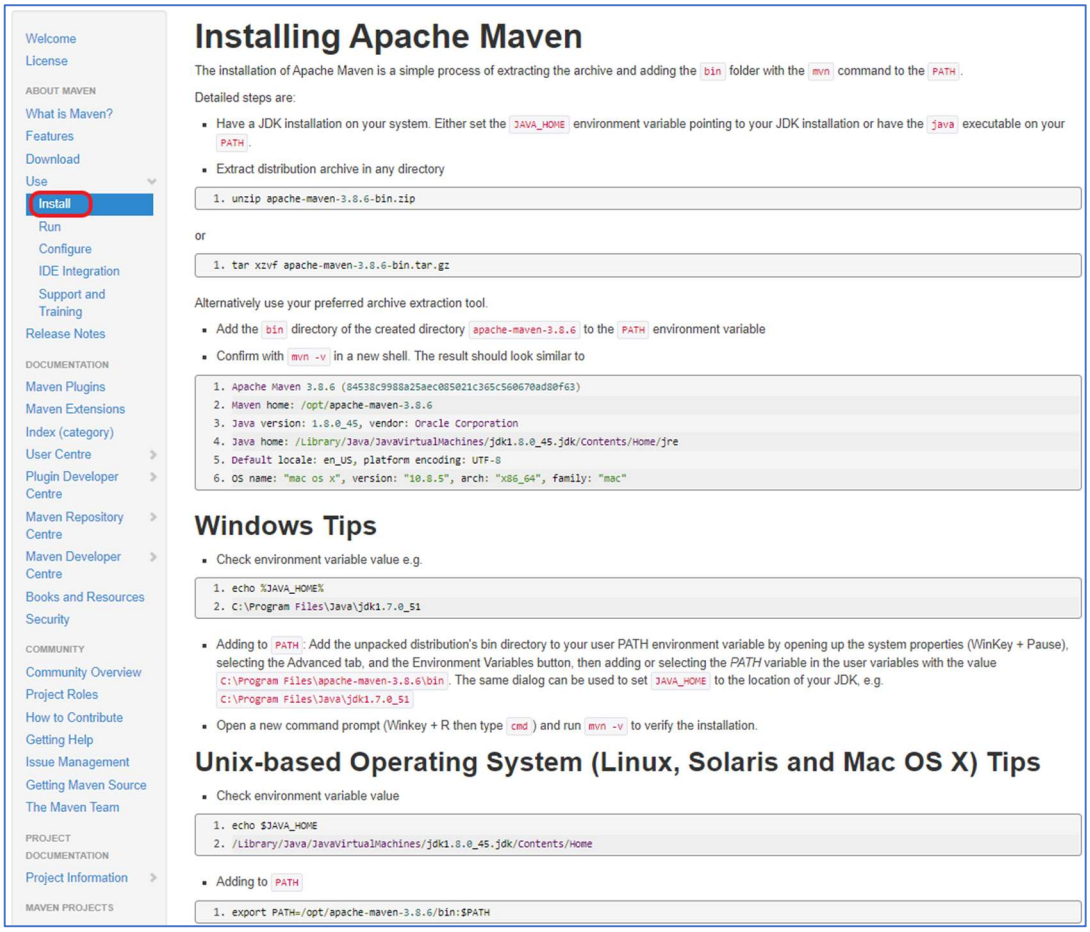


12. Jenkis for Java app using Maven.

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), **Maven** can manage a project's build, reporting and documentation from a central piece of information.

- Download and install **Maven**





- Java project structure

```
[vagrant@vm2 java-test]$ tree
.
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── mycompany
    │               └── app
    │                   └── App.java
    └── test
        └── java
            └── com
                └── mycompany
                    └── app
                        └── AppTest.java

11 directories, 3 files
[vagrant@vm2 java-test]$
```

```
[vagrant@vm2 java-test]$ cat src/main/java/com/mycompany/app/App.java
package com.mycompany.app;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
[vagrant@vm2 java-test]$ cat src/test/java/com/mycompany/app/AppTest.java
package com.mycompany.app;

import static org.junit.Assert.assertTrue;

import org.junit.Test;

/**
 * Unit test for simple App.
 */
public class AppTest
{
    /**
     * Rigorous Test :-)
     */
    @Test
    public void shouldAnswerWithTrue()
    {
        assertTrue( true );
    }
}
```

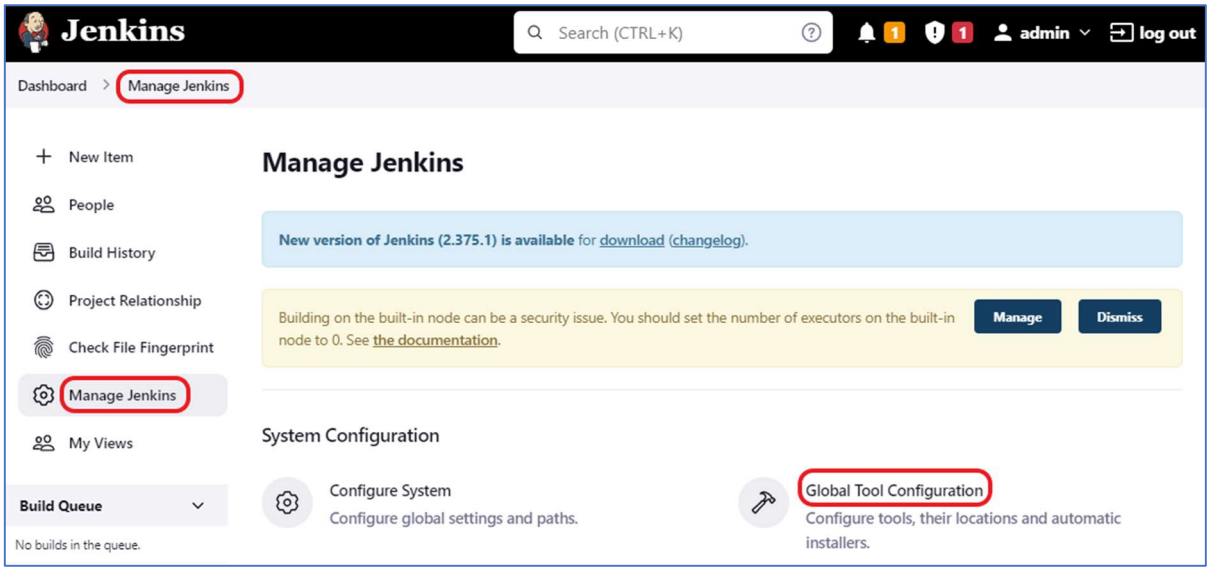
- Add Maven bin to the PATH and checking installation

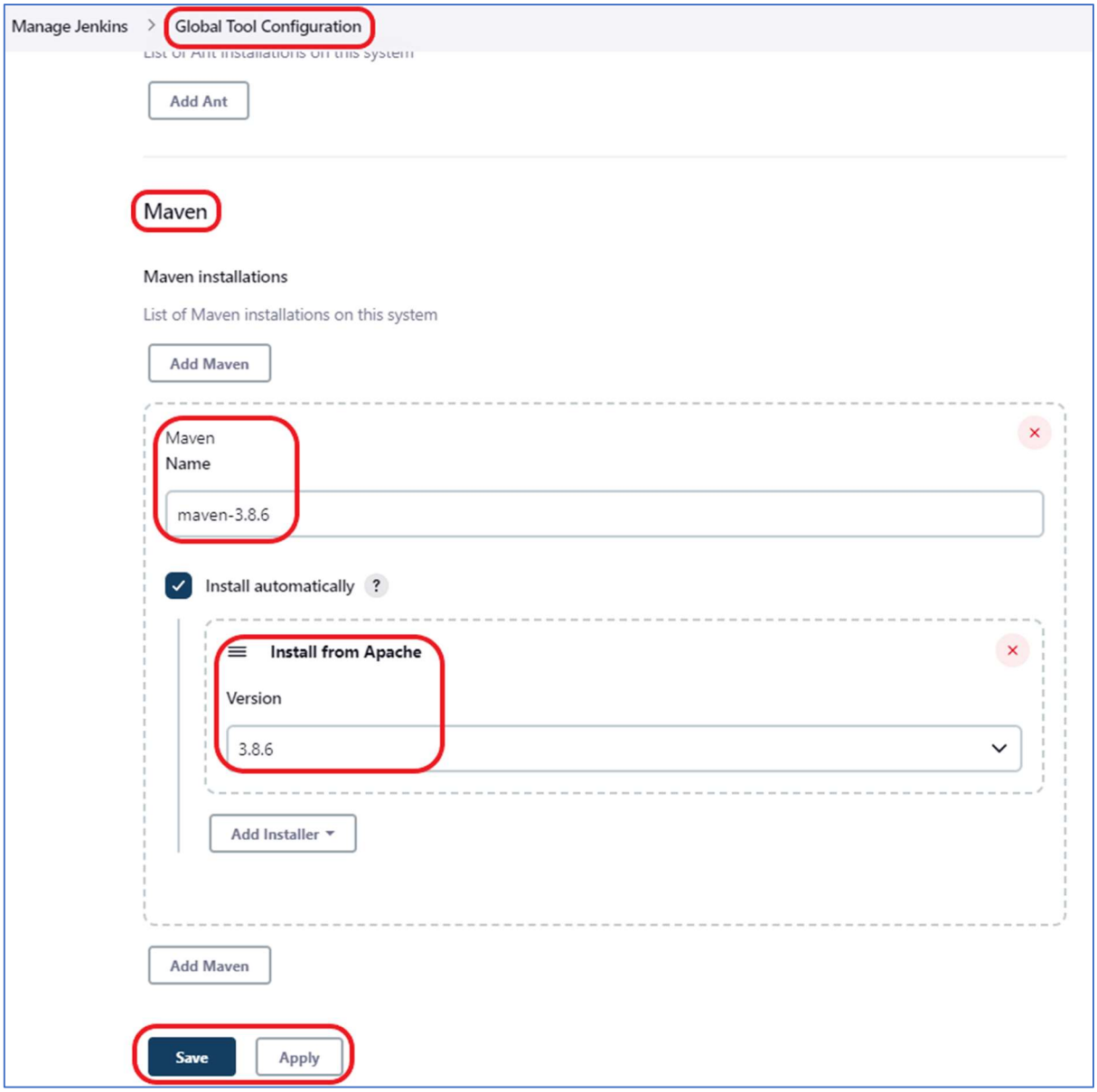
```
[vagrant@vm2 apache-maven-3.8.6]$ export PATH=$PATH:/home/vagrant/install/apache-maven-3.8.6/bin/
[vagrant@vm2 apache-maven-3.8.6]$ mvn -v
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: /home/vagrant/install/apache-maven-3.8.6
Java version: 11.0.17, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-11-openjdk-11.0.17.0.8-2
.el7_9.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1160.80.1.el7.x86_64", arch: "amd64", family: "unix"
[vagrant@vm2 apache-maven-3.8.6]$
```

- Run Maven test for java app

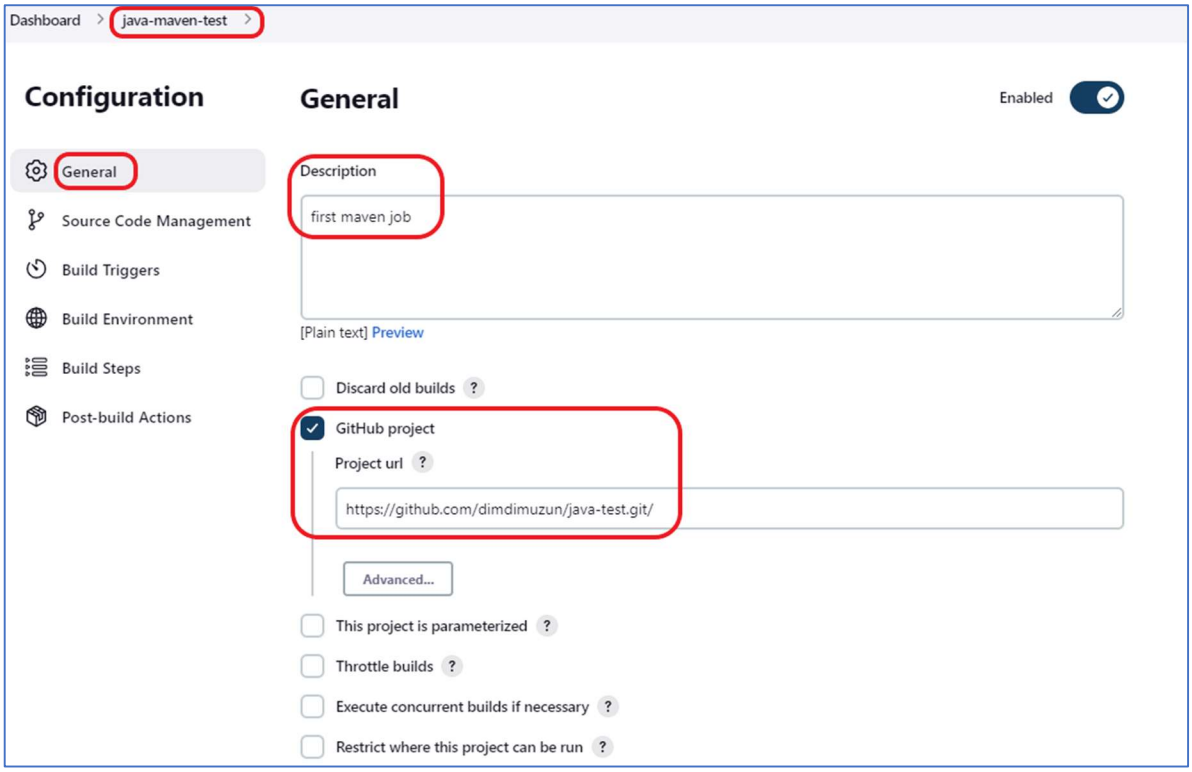
```
[vagrant@vm2 apache-maven-3.8.6]$ cd /home/vagrant/git/java-test/
[vagrant@vm2 java-test]$
[vagrant@vm2 java-test]$
[vagrant@vm2 java-test]$ ll
total 4
-rw-rw-r--. 1 vagrant vagrant 2653 Dec 20 14:58 pom.xml
drwxrwxr-x. 4 vagrant vagrant   30 Dec 20 14:58 src
[vagrant@vm2 java-test]$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mycompany.app:my-app >----------------------
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-res
ources-plugin/3.0.2/maven-resources-plugin-3.0.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-reso
urces-plugin/3.0.2/maven-resources-plugin-3.0.2.pom (7.1 kB at 8.5 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
providers/2.22.1/surefire-providers-2.22.1.pom (2.5 kB at 34 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire
-junit4/2.22.1/surefire-junit4-2.22.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
junit4/2.22.1/surefire-junit4-2.22.1.jar (85 kB at 813 kB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.085 s - in com.mycompany
.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20.422 s
[INFO] Finished at: 2022-12-20T15:20:41Z
[INFO] ------------------------------------------------------------------------
[vagrant@vm2 java-test]$
```

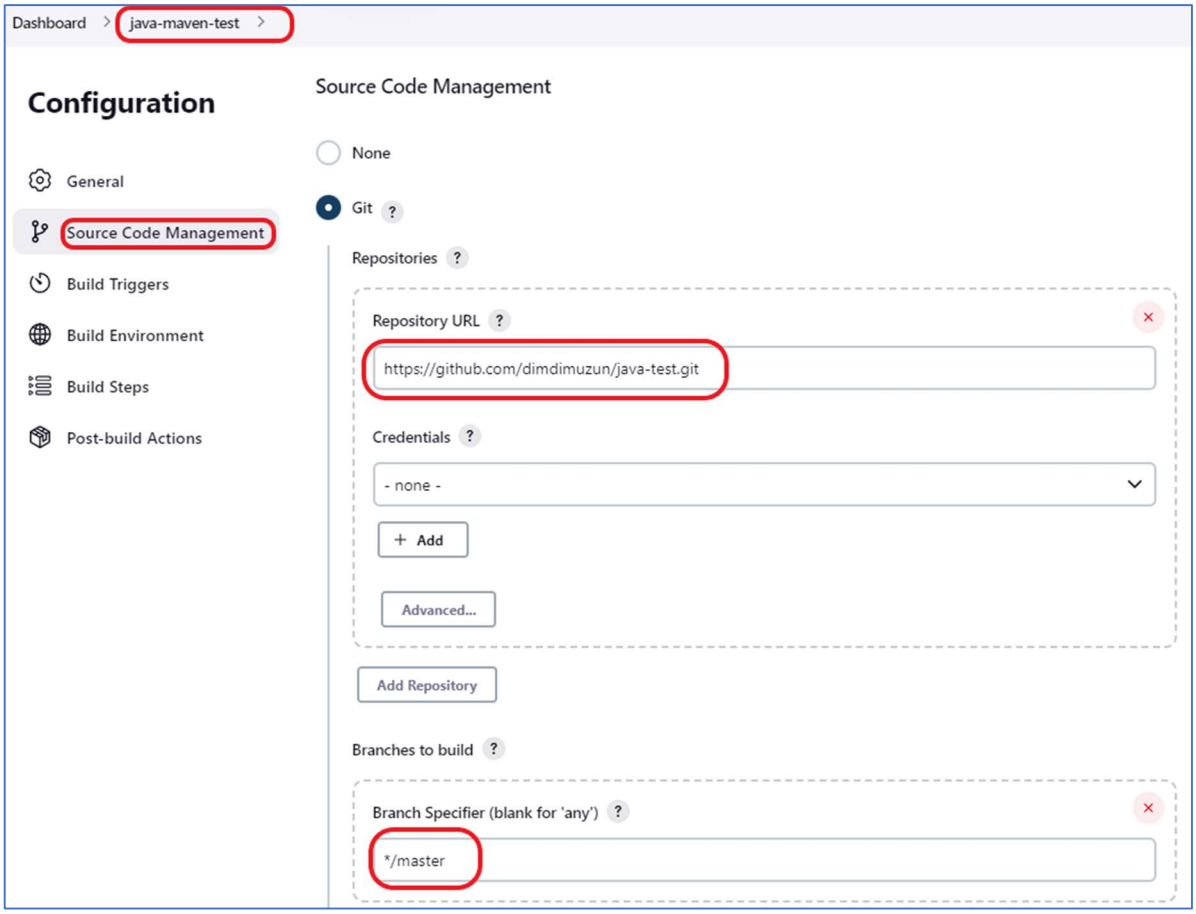
- Clobal Tool Configuration. Adding Maven to the Jenkins.

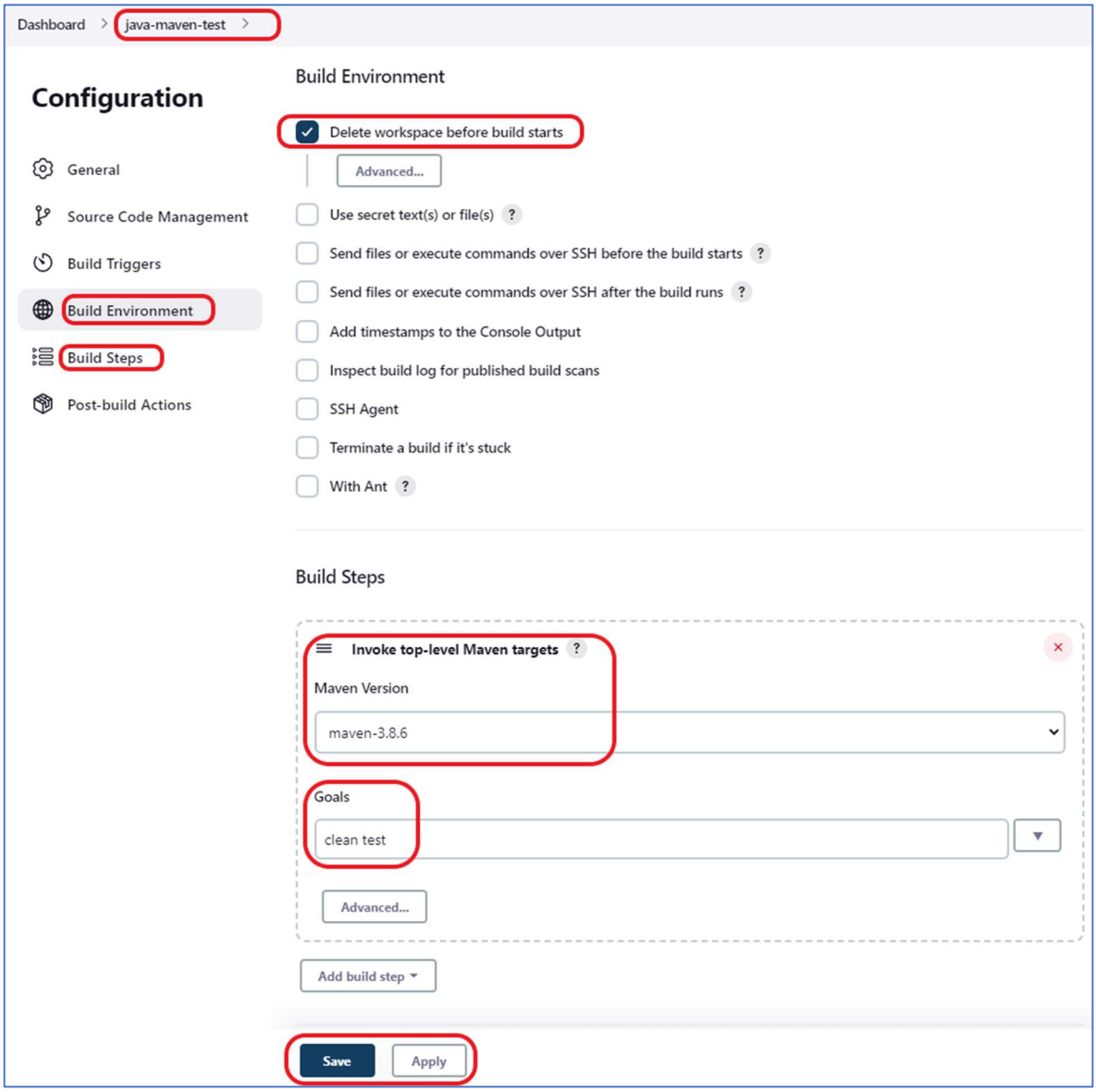




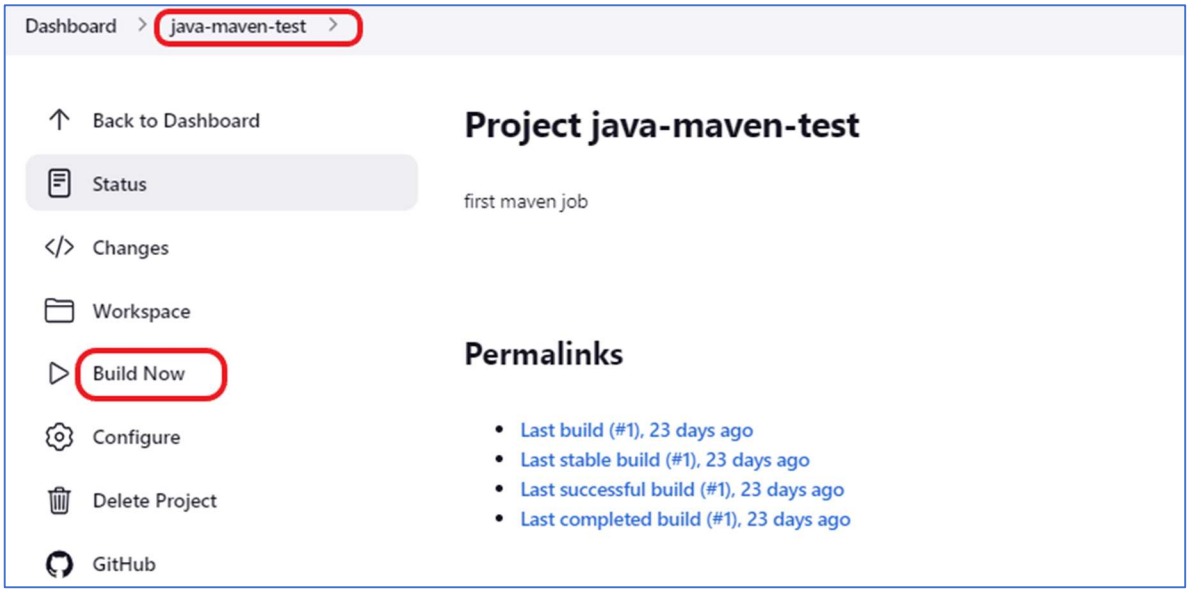
- Create job


Dashboard
java-maven-test
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
first maven job
[Plain text] Preview
Discard old builds
GitHub project
Project url
https://github.com/dimdimuzun/java-test.git/
Advanced...
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run


Dashboard
java-maven-test
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/dimdimuzun/java-test.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master


Dashboard
java-maven-test
Configuration
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Build Environment
Delete workspace before build starts
Advanced...
Use secret text(s) or file(s)
Send files or execute commands over SSH before the build starts
Send files or execute commands over SSH after the build runs
Add timestamps to the Console Output
Inspect build log for published build scans
SSH Agent
Terminate a build if it's stuck
With Ant
Build Steps
Invoke top-level Maven targets
Maven Version
maven-3.8.6
Goals
clean test
Advanced...
Add build step
Save
Apply

- Build job


Dashboard
java-maven-test
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
GitHub
Project java-maven-test
first maven job
Permalinks
Last build (#1), 23 days ago
Last stable build (#1), 23 days ago
Last successful build (#1), 23 days ago
Last completed build (#1), 23 days ago

- Console output