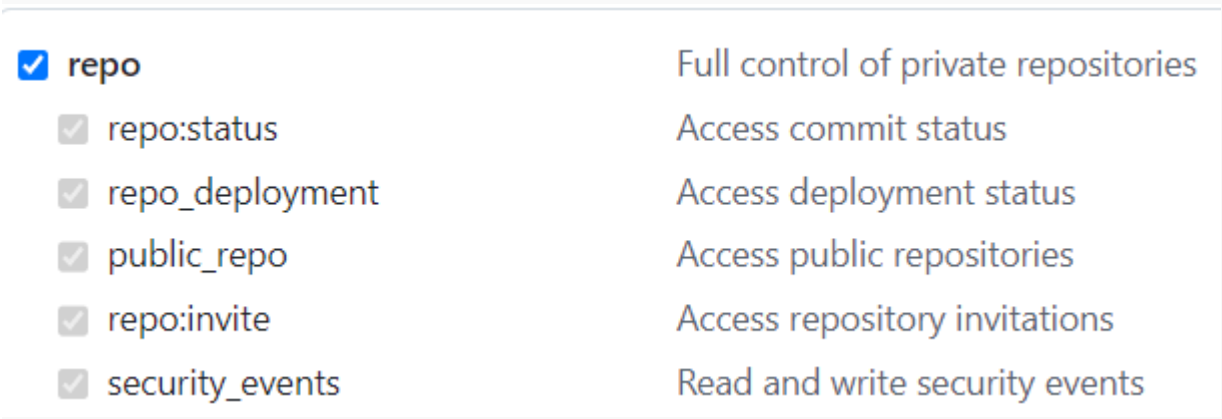
Private: CI/CD: Jenkins Setup Guide

Author: Chrisio (Weixi Guan) Gwaan
Date: 2024-09-12
Note: The following installation guide is based on the Ubuntu system.

Step 1: GitHub Personal Access Token (PAT)

- Go to `GitHub profile` → `Settings` → `Developer settings` → `Personal Access Token`.
- Generate a token with checking the following permission.



- Copy the token and keep it in the note. (Token will only be shown at the first time. If token is lost, you can generate a new one)

Step 2: Install Docker

1. Set up Docker's `apt` repository.

```
# Add Docker's official GPG key:
sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

2. Install the Docker packages.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

3. Check docker container list for verifying if docker is installed successfully.

```
sudo docker ps
```

4. Enable non-root user to use docker command without `sudo`.

```
# Create the docker group if it does not exist
sudo groupadd docker

# Add your user to the docker group
sudo usermod -aG docker $USER

# Log in to the new docker group (to avoid having to log out / log in again; but if not enough, try to reboot)
newgrp docker

# Check if docker can be run without root
docker ps

# (Optional!) Reboot if still got error
reboot
```

Step 3: Install Jenkins and Setup

1. Download and install a Long-Term Support (LTS) release Jenkins. (A LTS is chosen every 12 weeks from the stream of regular releases as the stable release for that time period)

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

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 jenkins
```

2. Installation of Java. (Jenkins requires Java to run)

```
sudo apt update

sudo apt install openjdk-21-jdk

# Check if Java is installed successfully
java -version
```

3. Start Jenkins.

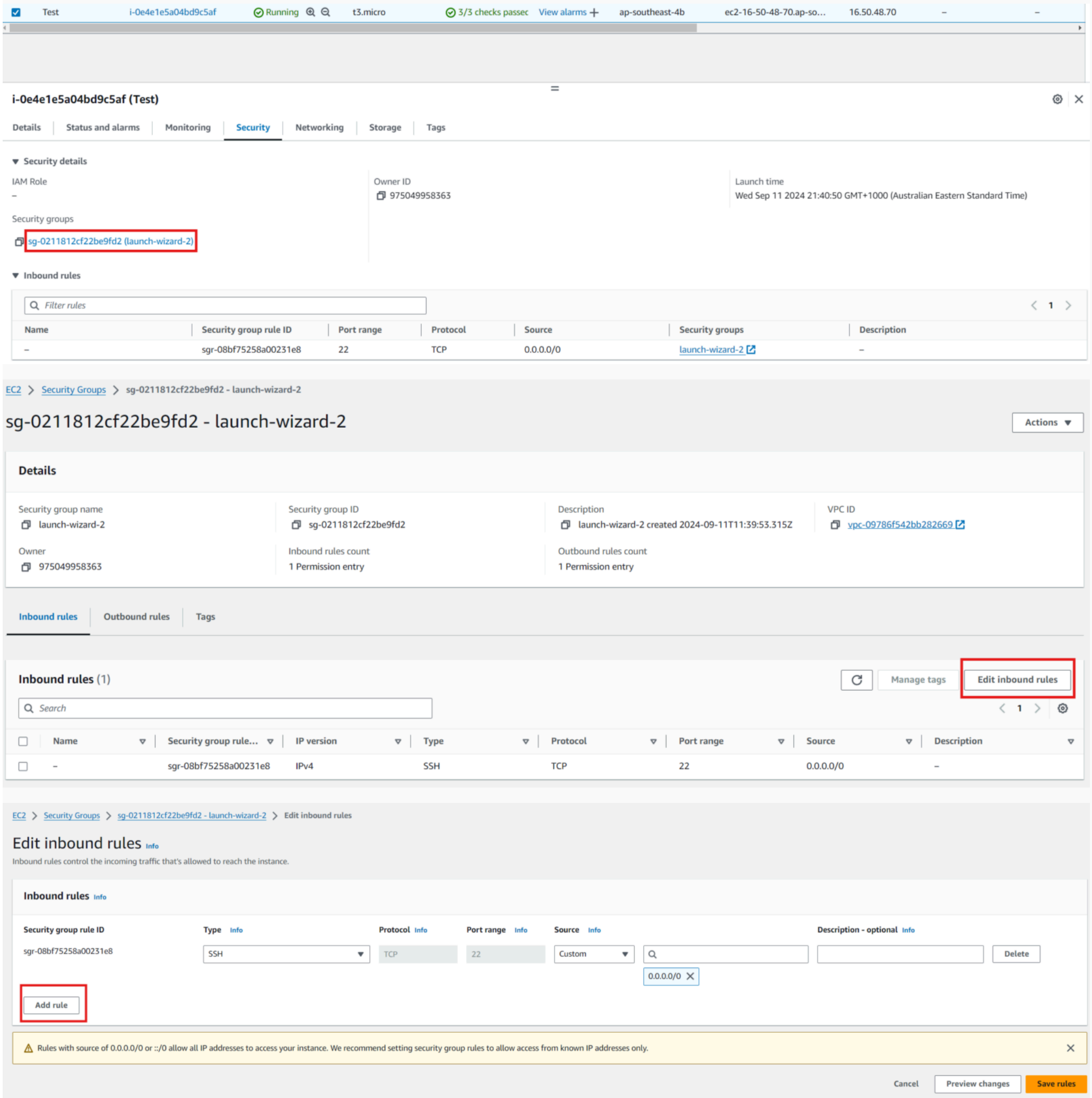
```
# Enable the Jenkins service to start at boot
sudo systemctl enable jenkins

# Start the Jenkins service
sudo systemctl start jenkins

   #if getting error: "System has not been booted with systemd as init system (PID 1). Can't operate.
   #Failed to connect to bus: Host is down"
   #try: sudo service jenkins start  expected output:  "* Starting Jenkins Automation Server jenkins  ..." -Dylan

# Check the status of the Jenkins service
# You should see a generated password for admin, similar example as shown below
sudo systemctl status jenkins

   #if getting error: "System has not been booted with systemd as init system (PID 1). Can't operate.
   #Failed to connect to bus: Host is down"
   #try: sudo service jenkins status expected output: " * jenkins is running" -Dylan

   #to get password (what is provided in the red box in the image below)
   #try: sudo tail -f /var/log/jenkins/jenkins.log expected output: "{numbers & letters AKA PASSWORD} This may also be ..."

   #to open it you can do http://localhost:8080
```

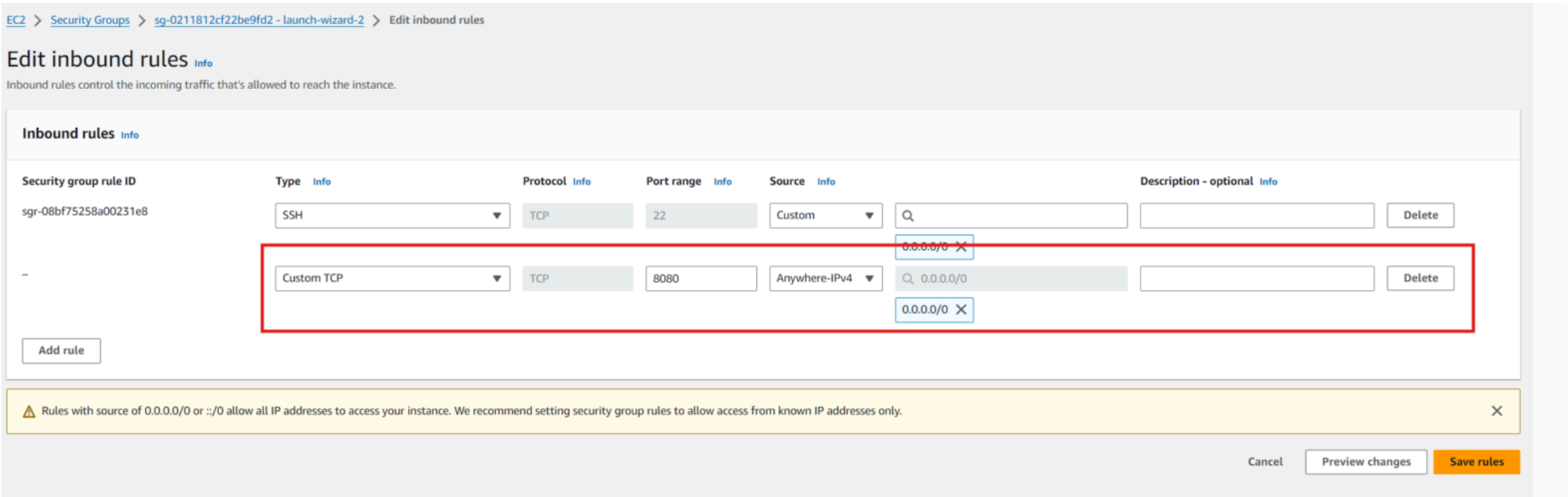
```
ubuntu@ip-172-31-8-50:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-09-11 11:49:42 UTC; 19s ago
   Main PID: 17043 (java)
      Tasks: 42 (limit: 1078)
     Memory: 275.6M (peak: 312.3M)
        CPU: 23.551s
     CGroup: /system.slice/jenkins.service
             └─17043 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 11 11:49:34 ip-172-31-8-50 jenkins[17043]: b2d7e059724e4b238f8bfd4f754d5da9
Sep 11 11:49:34 ip-172-31-8-50 jenkins[17043]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 11 11:49:34 ip-172-31-8-50 jenkins[17043]: ************************************************************
Sep 11 11:49:34 ip-172-31-8-50 jenkins[17043]: ************************************************************
Sep 11 11:49:34 ip-172-31-8-50 jenkins[17043]: ************************************************************
Sep 11 11:49:42 ip-172-31-8-50 jenkins[17043]: 2024-09-11 11:49:42.447+0000 [id=33]        INFO        jenkins.InitReactorRunner$1#onAttained: Completed initialization
Sep 11 11:49:42 ip-172-31-8-50 jenkins[17043]: 2024-09-11 11:49:42.466+0000 [id=24]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Sep 11 11:49:42 ip-172-31-8-50 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 11 11:49:43 ip-172-31-8-50 jenkins[17043]: 2024-09-11 11:49:43.422+0000 [id=49]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Sep 11 11:49:43 ip-172-31-8-50 jenkins[17043]: 2024-09-11 11:49:43.423+0000 [id=49]        INFO        hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
```

As default, Jenkins will create an admin account that has all the permissions for the platform. The login username would be `admin`, and password would be the one with red colour highlight from the terminal. Remember to save this password in your note.

4. Add port 8080 to inbound rule of the EC2 instance in AWS

Click the following highlights:





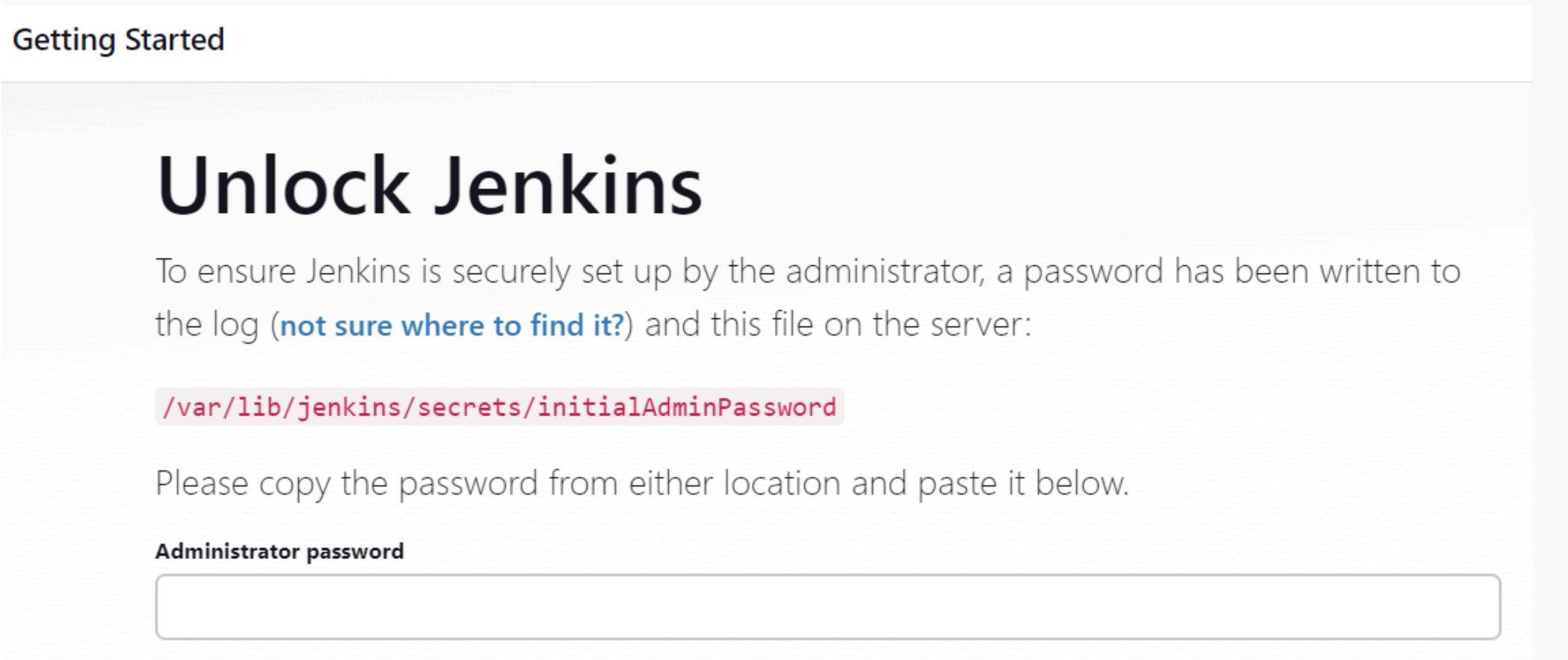
And click `Save rules`.
After the TCP port 8080 is enabled at inbound rule, Jenkins can be accessed publicly from now on.

5. Access Jenkins on browser.

Jenkins uses port 8080 as default, you may access it with the following URL:

```
http://<replace_with_your_domain_here>:8080
```

6. Unlock Jenkins with initial admin password.



7. Install suggested plugins.

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

Getting Started

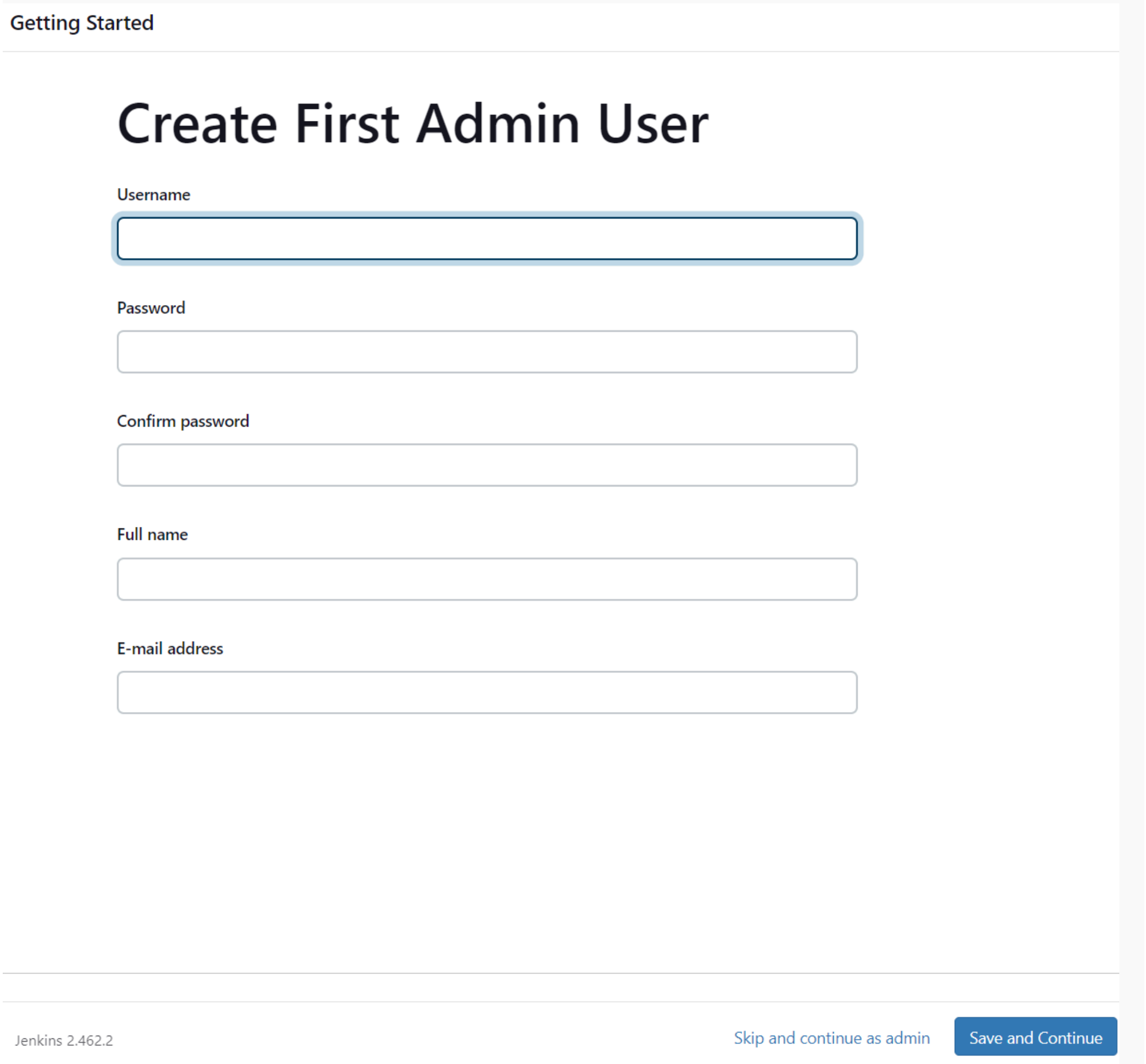
Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline Graph View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	Dark Theme

** - required dependency

Jenkins 2.462.2

8. (Optional) Create first admin user

You can skip and create later.



9. Instance Configuration

You can just leave it as default and click to the next.

Getting Started

Instance Configuration

Jenkins URL: http://ec2-16-50-48-70.ap-southeast-4.compute.amazonaws.com:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.462.2 Not now Save and Finish

10. Awesome! Jenkins is ready to use.

Getting Started

Jenkins is ready!

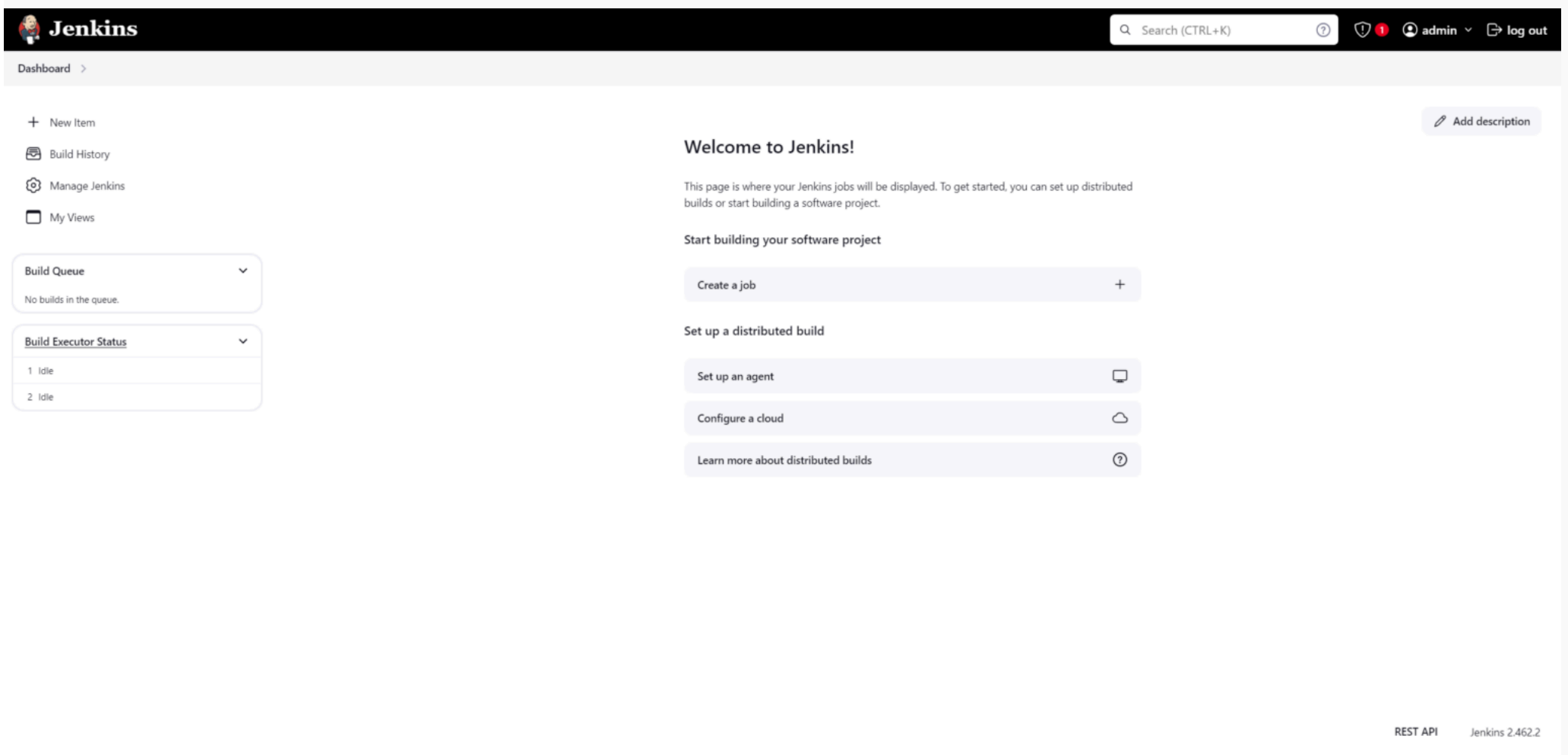
You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

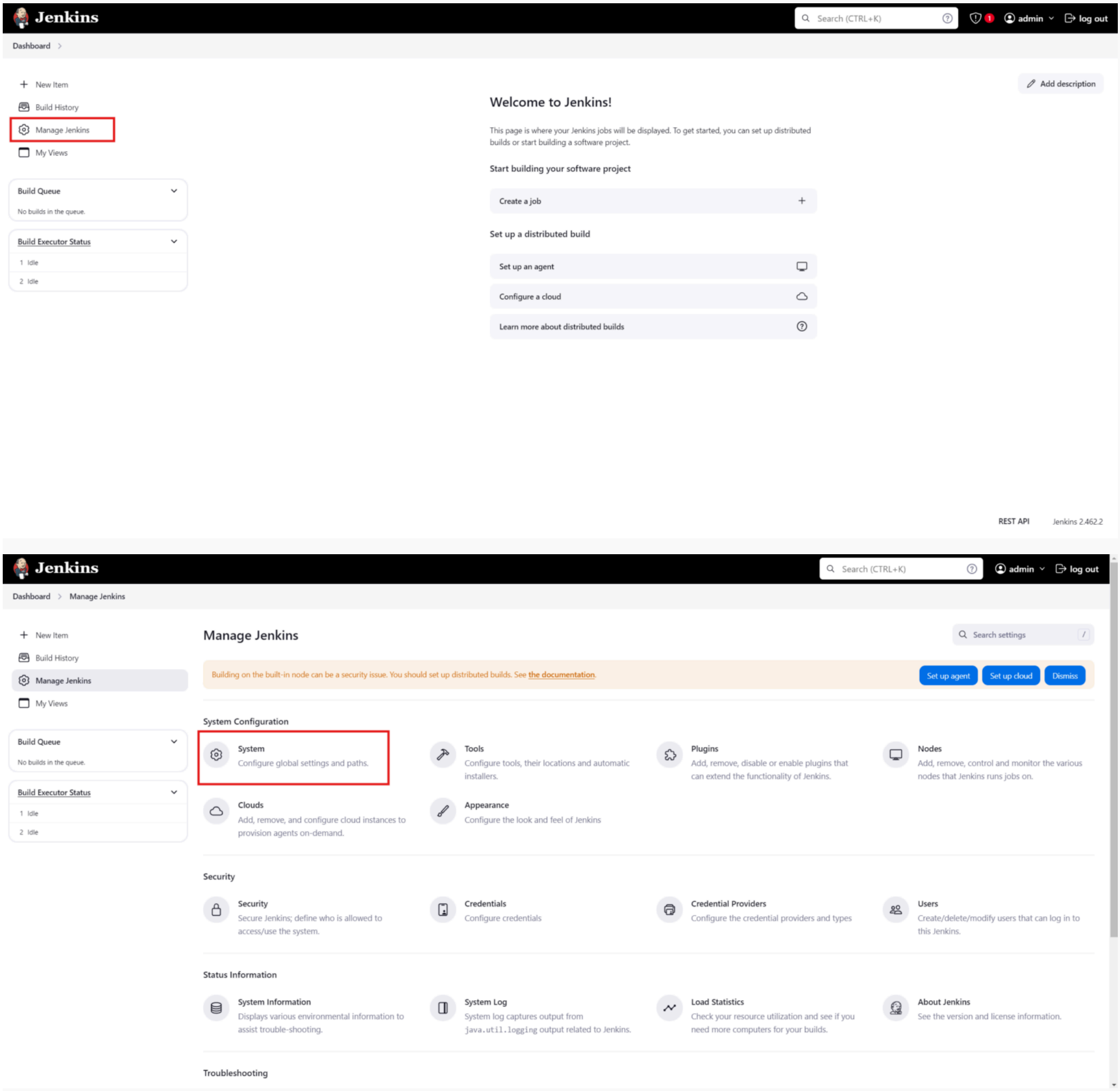
Your Jenkins setup is complete.

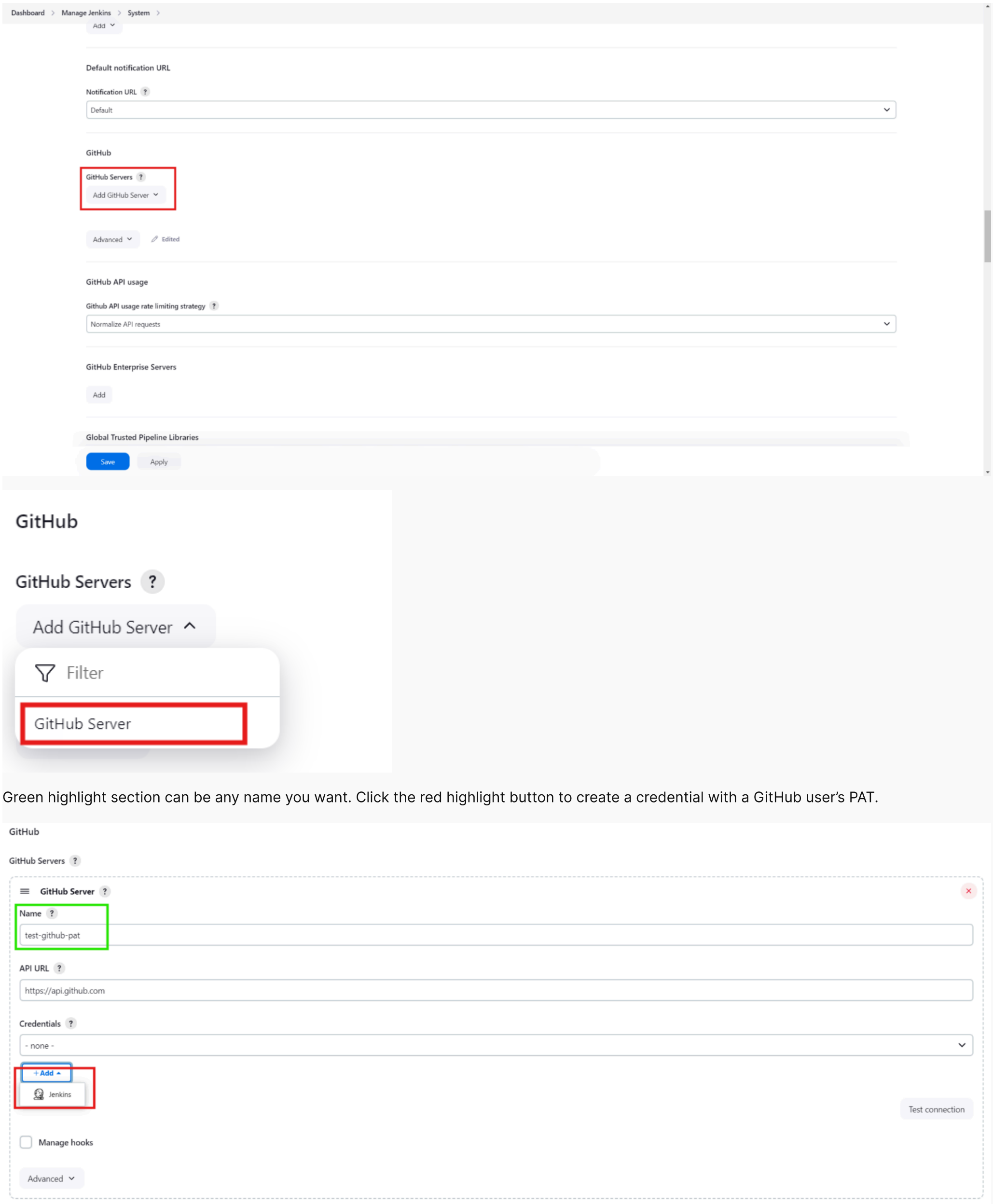
Start using Jenkins

Jenkins 2.462.2

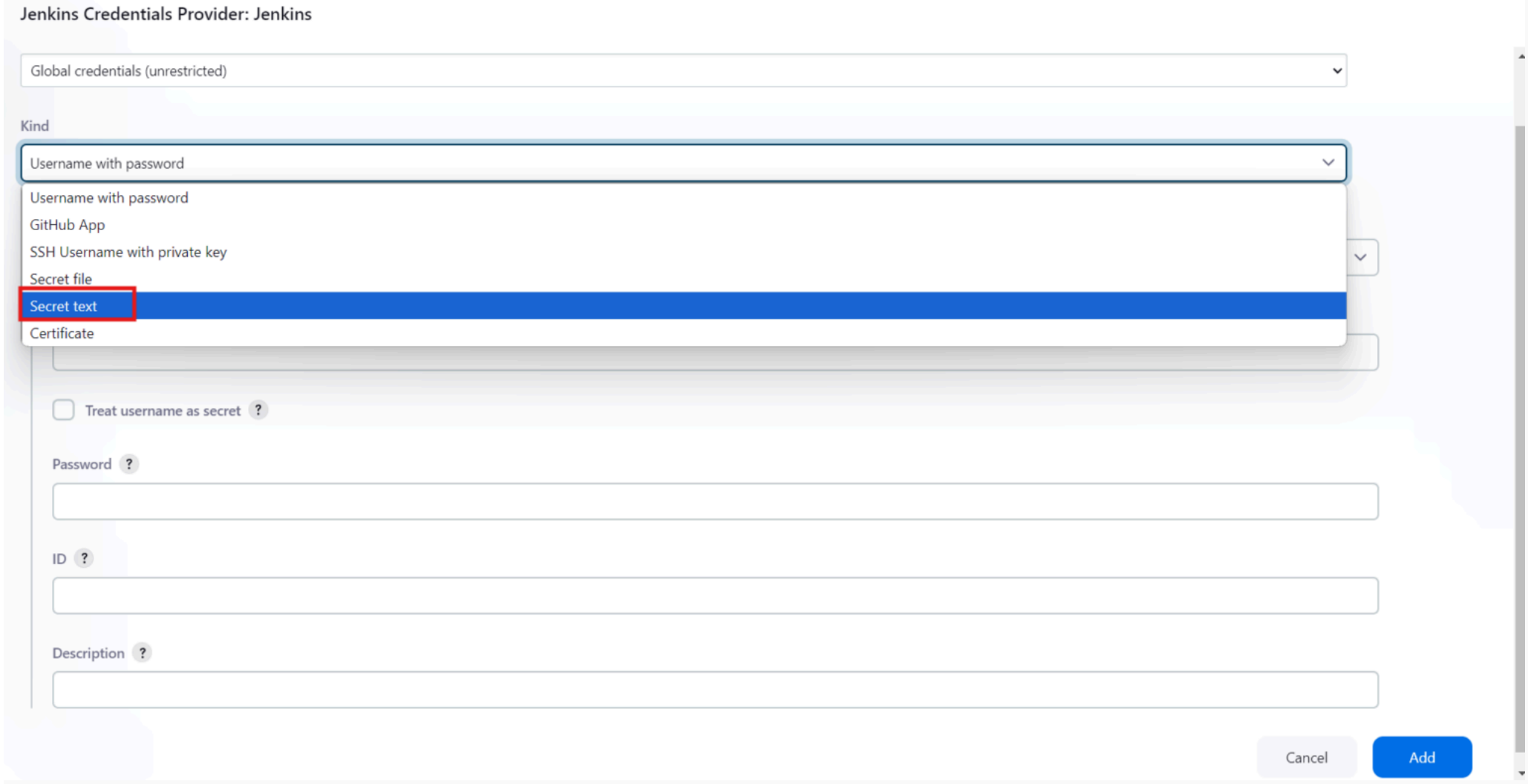


Step 4: Add GitHub PAT to Jenkins

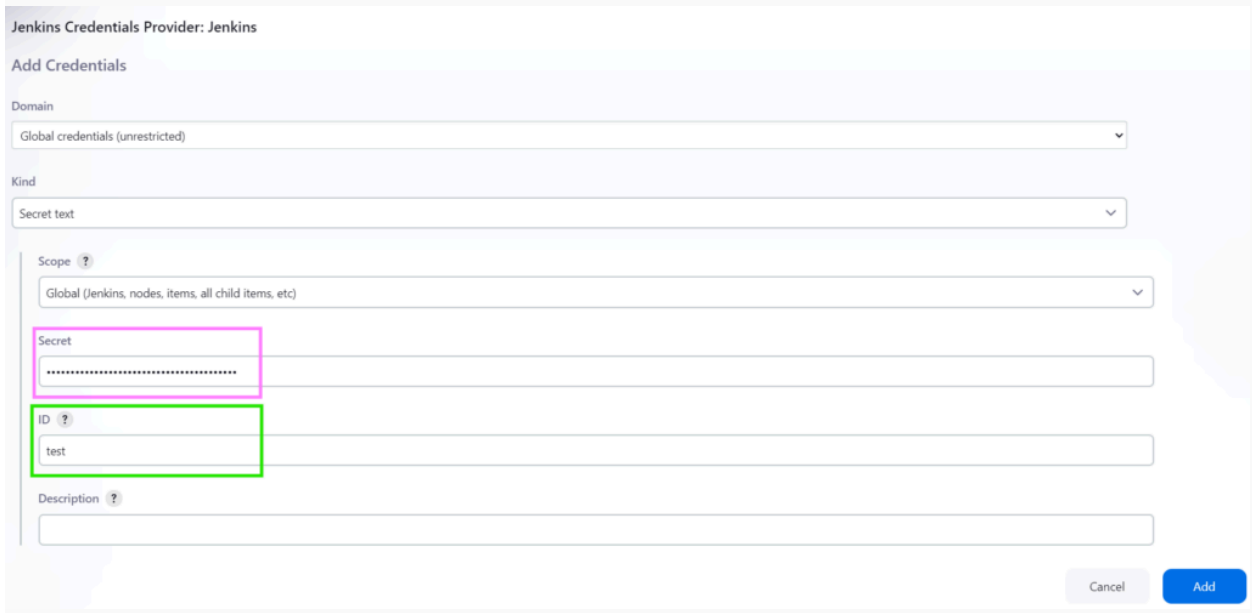




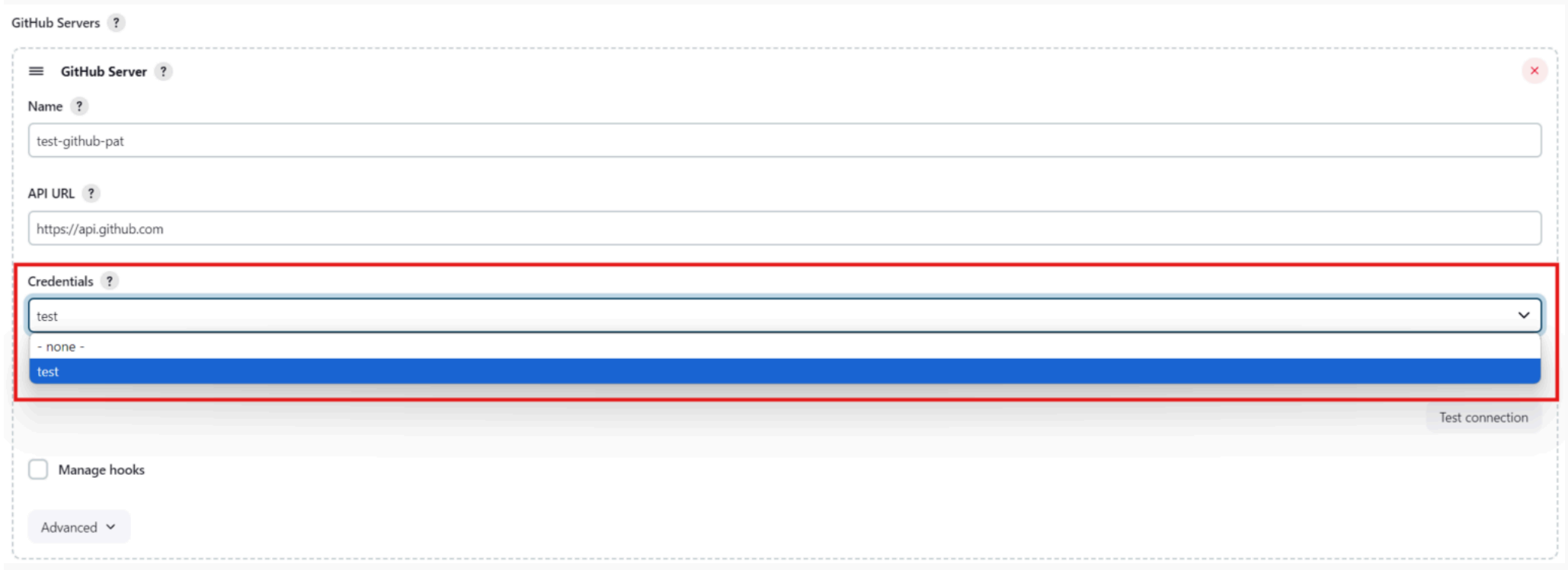
Green highlight section can be any name you want. Click the red highlight button to create a credential with a GitHub user's PAT.



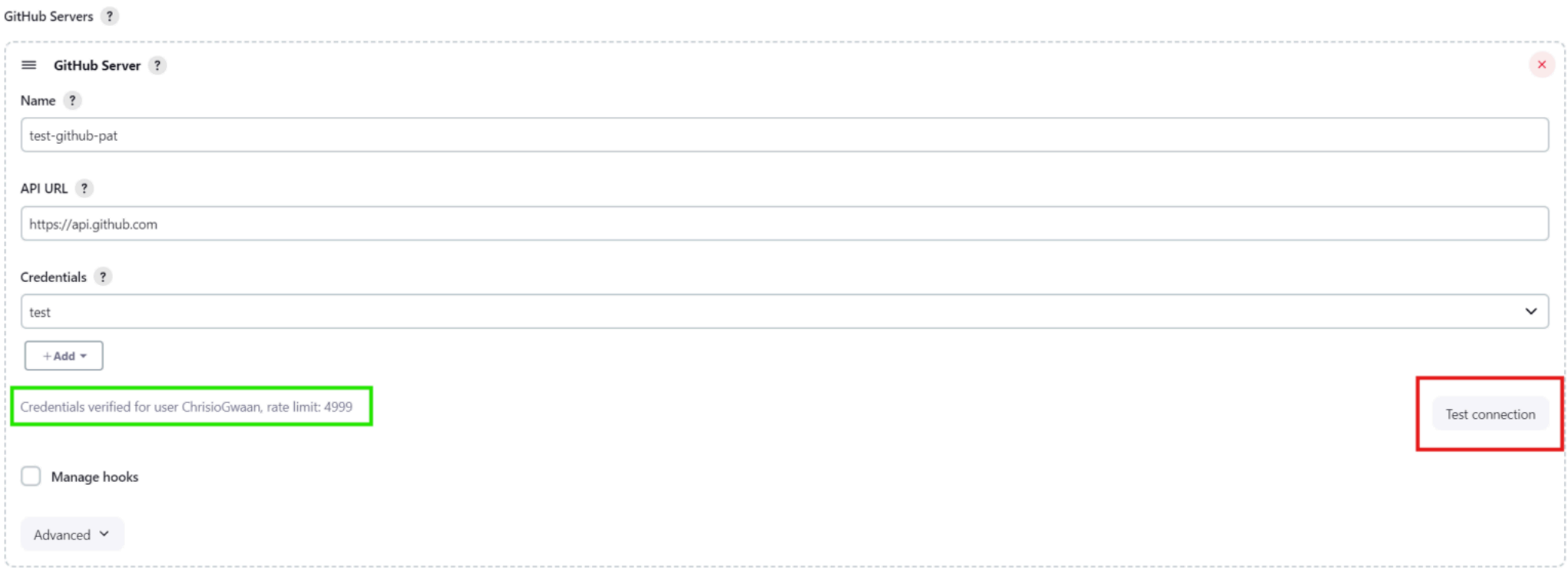
Now you can put a user's PAT to the pink highlight section. You can also give the name of this PAT to the green highlight section.



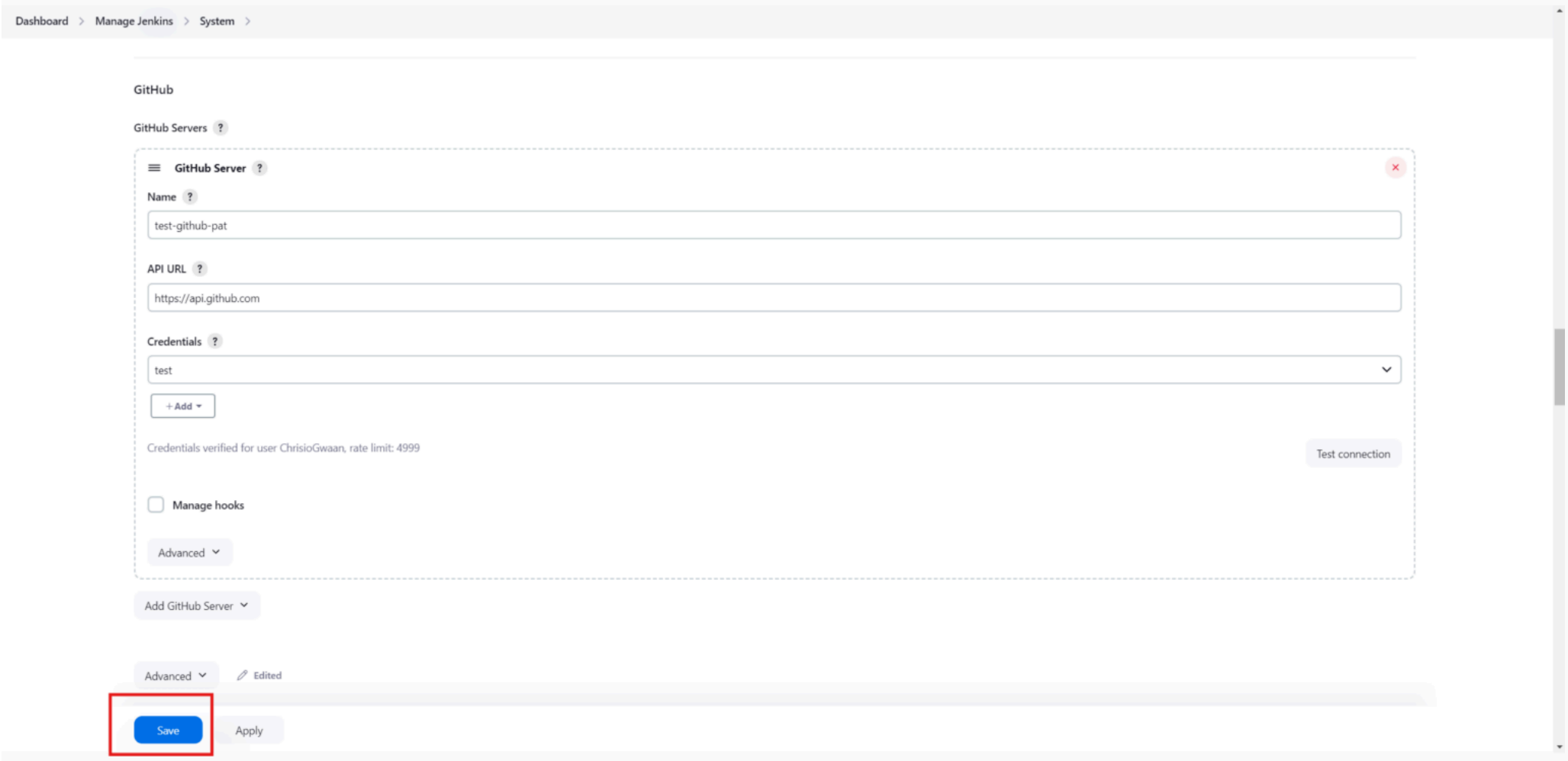
After a credential is created, you can see an option from `credentials` section.



To check if the PAT is working, simply just click the red highlight button, and you will see a status message with the GitHub userID of the PAT you provide from the green highlight section.



Now, click save.



Step 5: Generate SSH key

1. Connect to your EC2 instance.
2. Generate an ED25519 ssh key on your instance terminal, and replace the `your_email@example.com` with the one that belongs to the GitHub user who generated the PAT from the previous step.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
# Keep clicking 'Enter' as choosing default
```

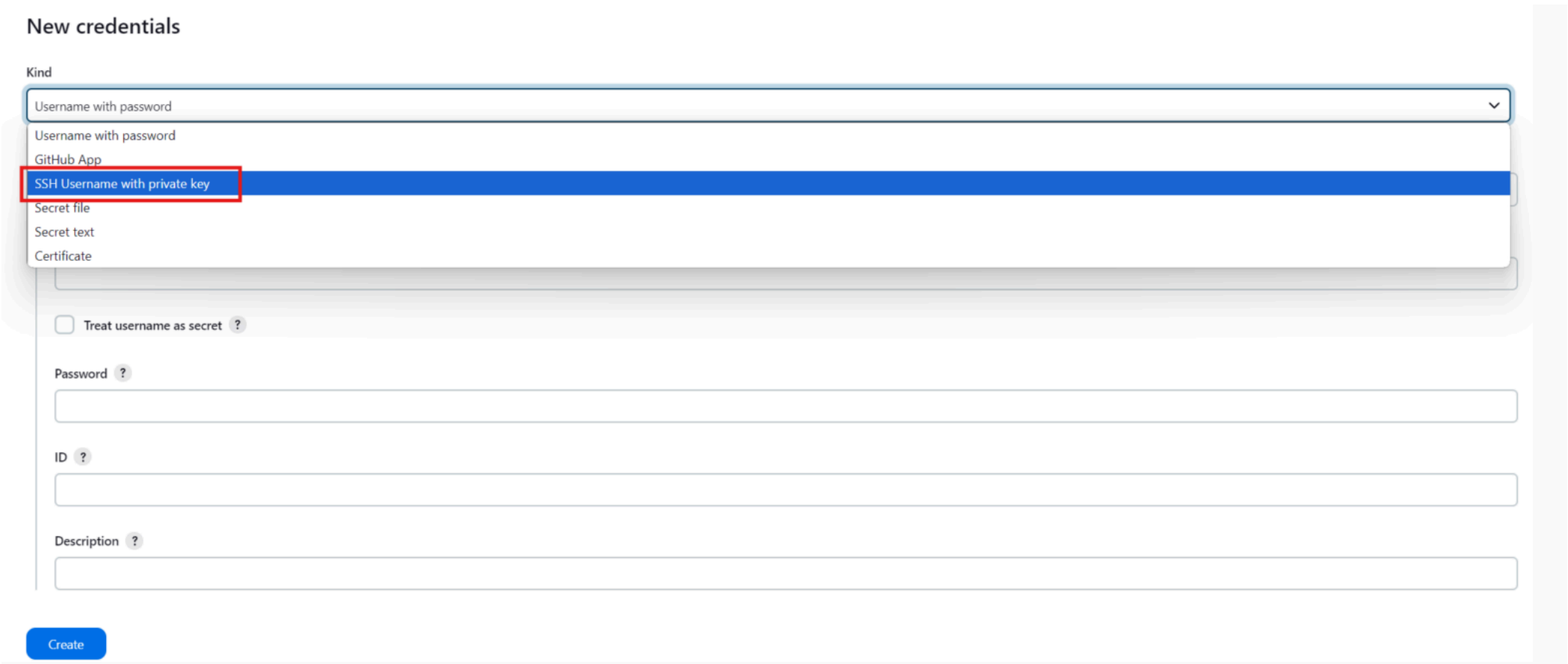
```
ubuntu@ip-172-31-8-50:~$ ssh-keygen -t ed25519 -C "[redacted]"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
[redacted]
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|             . . |
|            . o  |
|         . .o  . |
|       .So+*o . o|
|       .+.B**o.+o|
|        .=.O**..o|
|         oO+*.+o.|
|         E*B .+o |
+----[SHA256]-----+
```

3. Print out the `private key`, and copy it.

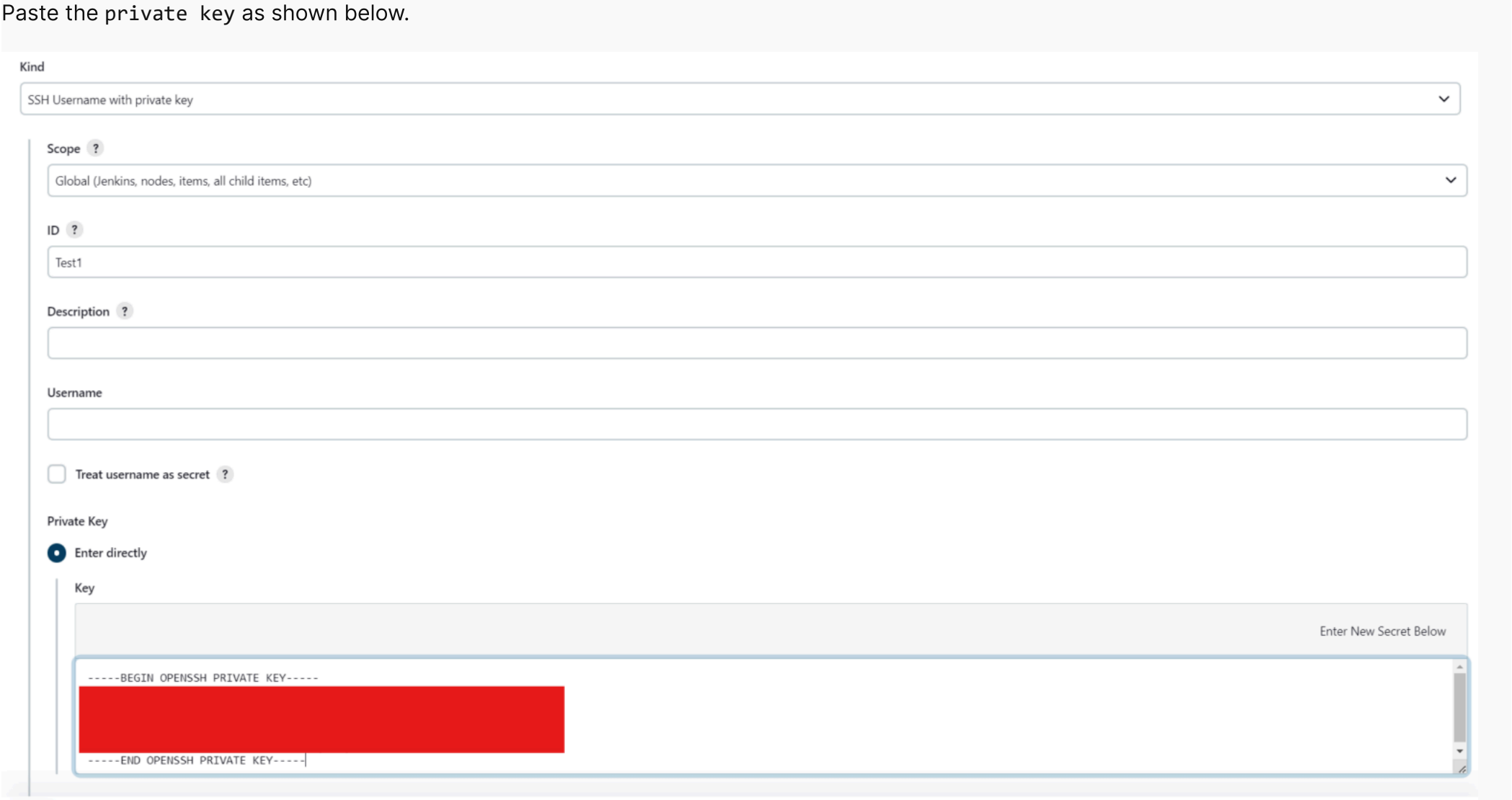
`cat ~/.ssh/id_ed25519`

```
ubuntu@ip-172-31-8-50:~$ cat ~/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----
```

4. Go to `Jenkins` → `Manage Jenkins` → `Security` → `Credentials`



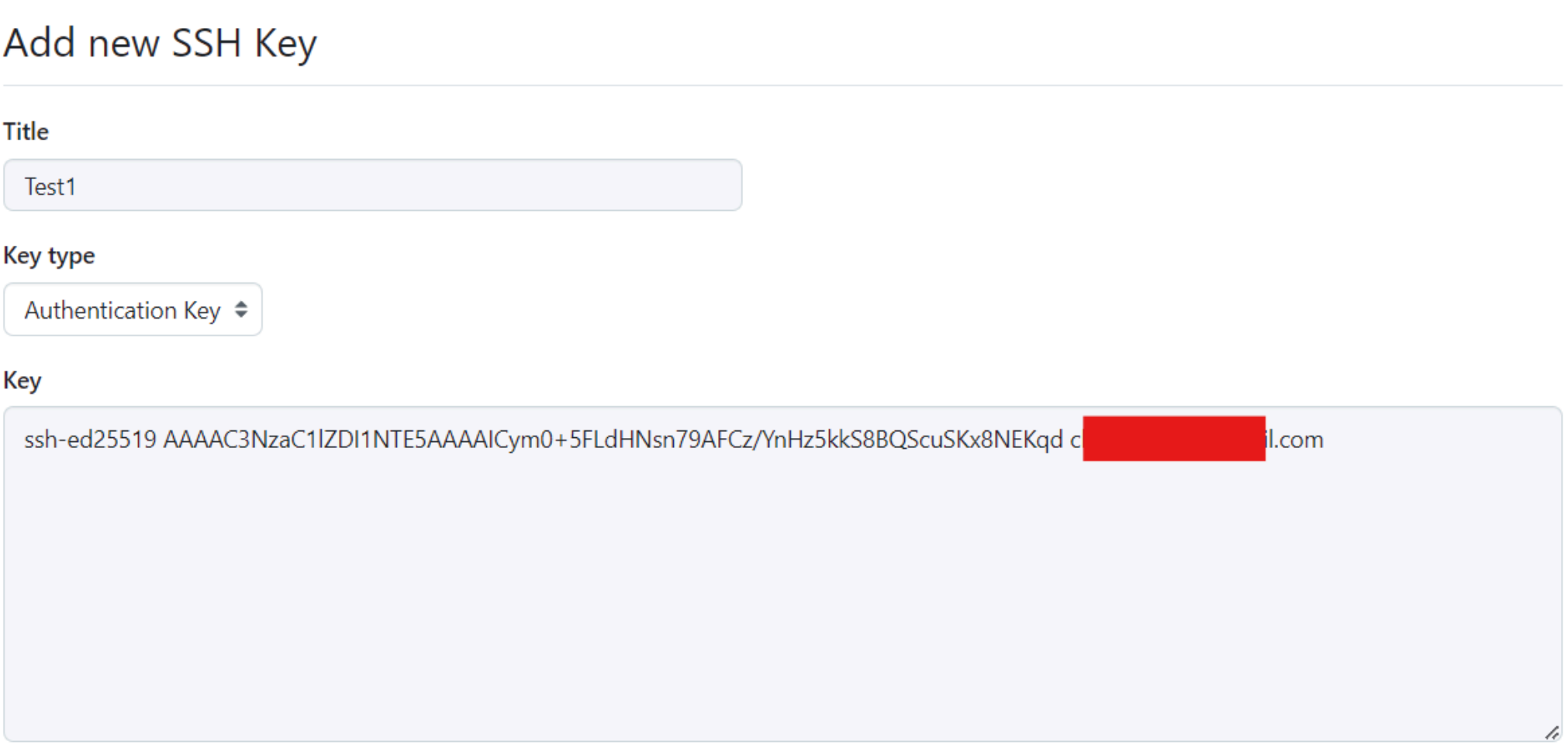
Paste the `private key` as shown below.



5. Go back to the instance terminal, and print out the `public key` from the SSH key, then copy the entire `public key`.

```
cat ~/.ssh/id_ed25519.pub
```

6. Go to `GitHub profile` → `Settings` → `SSH and GPG keys` → Click `New SSH key` → Paste the `public key` to the key section.



Step 6: Grant Jenkins access to SSH key

1. Create the `.ssh` directory in the Jenkins folder.

```
# If the .ssh directory does not exist, create it and set the proper permissions
sudo mkdir /var/lib/jenkins/.ssh
sudo chown jenkins:jenkins /var/lib/jenkins/.ssh
sudo chmod 700 /var/lib/jenkins/.ssh
```

2. Copy the SSH Key Files.

```
sudo cp ~/.ssh/id_ed25519 /var/lib/jenkins/.ssh/
sudo cp ~/.ssh/id_ed25519.pub /var/lib/jenkins/.ssh/
```

3. Set Proper Permissions.

```
# Make sure the Jenkins user owns the SSH key files and that permissions are set correctly
sudo chown jenkins:jenkins /var/lib/jenkins/.ssh/id_ed25519 /var/lib/jenkins/.ssh/id_ed25519.pub
sudo chmod 600 /var/lib/jenkins/.ssh/id_ed25519
sudo chmod 644 /var/lib/jenkins/.ssh/id_ed25519.pub
```

4. Switch to the Jenkins User.

```
sudo -u jenkins -s
```

5. Add GitHub's Host Key.

```
# Use ssh-keyscan to fetch GitHub's public key and add it to the known_hosts file
ssh-keyscan github.com >> ~/.ssh/known_hosts

# This command appends the GitHub host key to the Jenkins user's known_hosts file.
```

6. Exit the Jenkins User Shell.

```
exit
```

7. Test the SSH Connection.

```
sudo -u jenkins ssh -T git@github.com
```

```
ubuntu@ip-172-31-8-50:~$ sudo -u jenkins ssh -T git@github.com
Hi ChrisioGwaan! You've successfully authenticated, but GitHub does not provide shell access.
```

8. (Optional) Restart Jenkins

```
sudo systemctl restart jenkins
```

Step 7: Grant Jenkins access to use Docker command

1. Add Jenkins User to Docker Group.

```
sudo usermod -aG docker jenkins
```

2. Verify Group Membership.

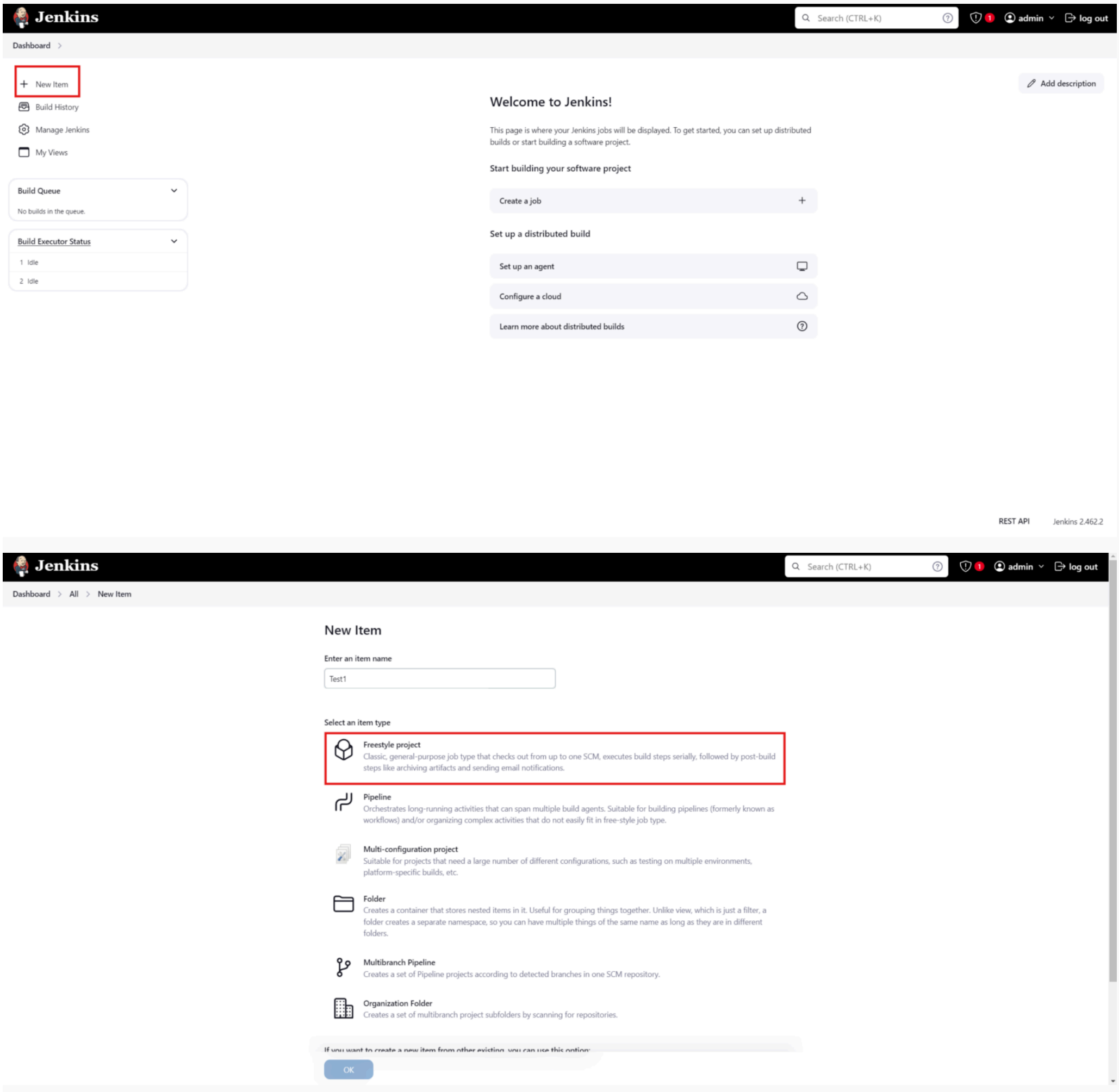
```
groups jenkins
```

3. Restart Jenkins.

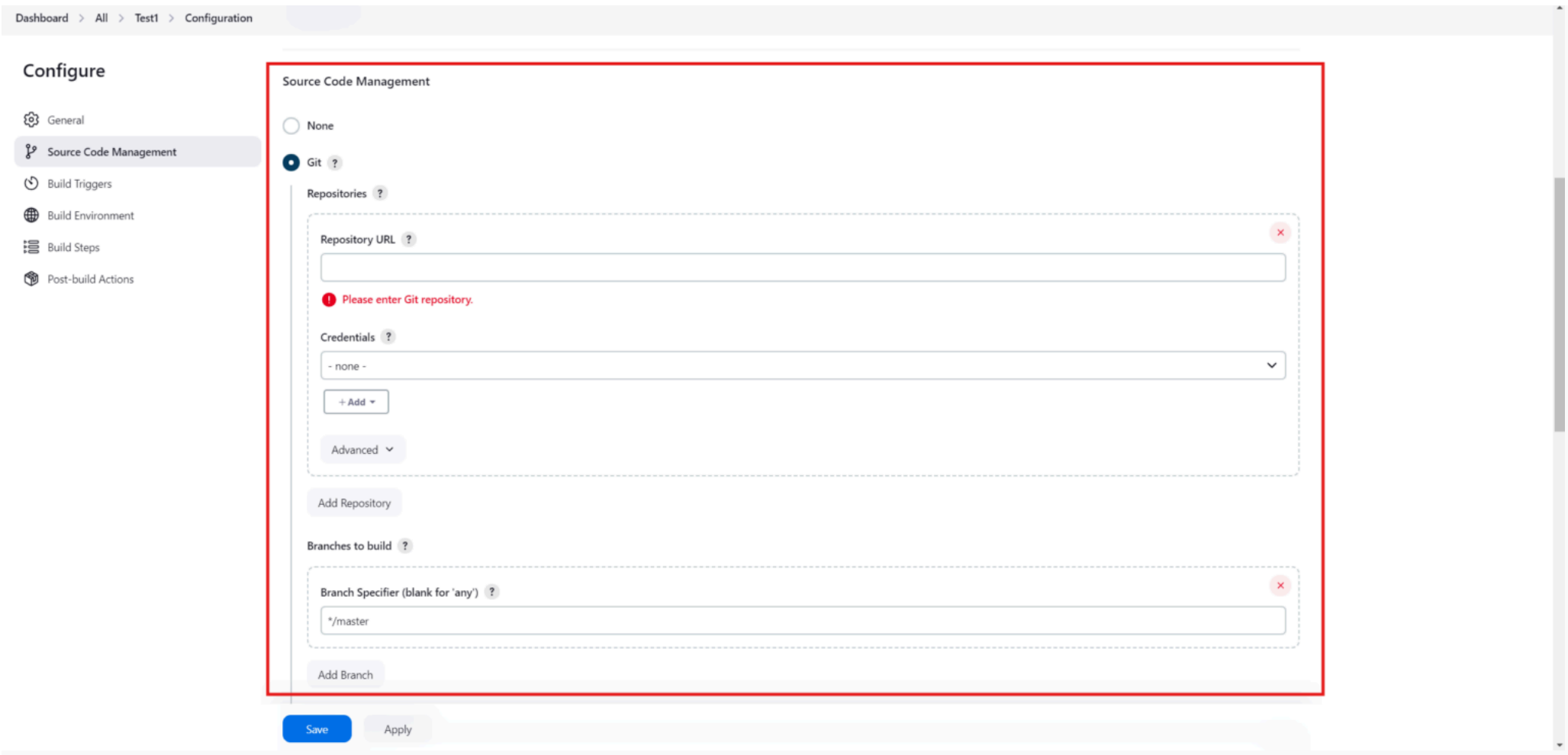
```
sudo systemctl restart jenkins
```

Step 8: Create a job for a project on Jenkins

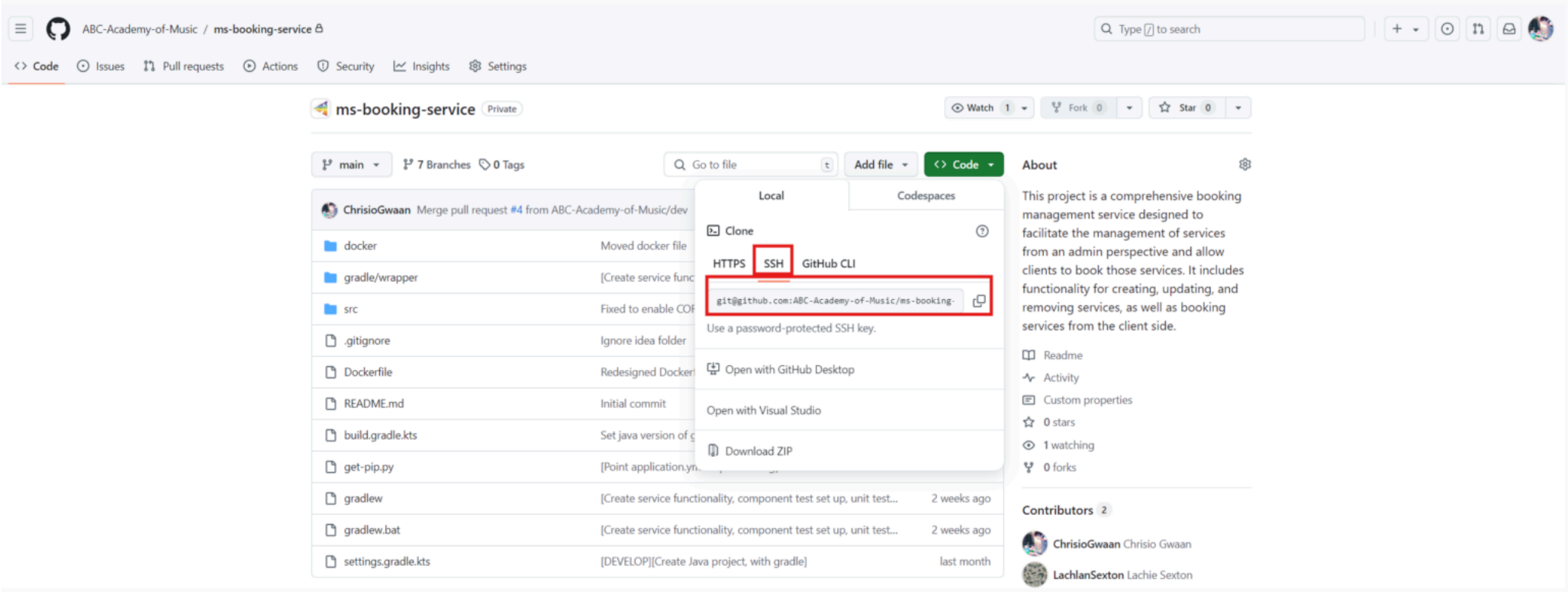
1. Create a job.



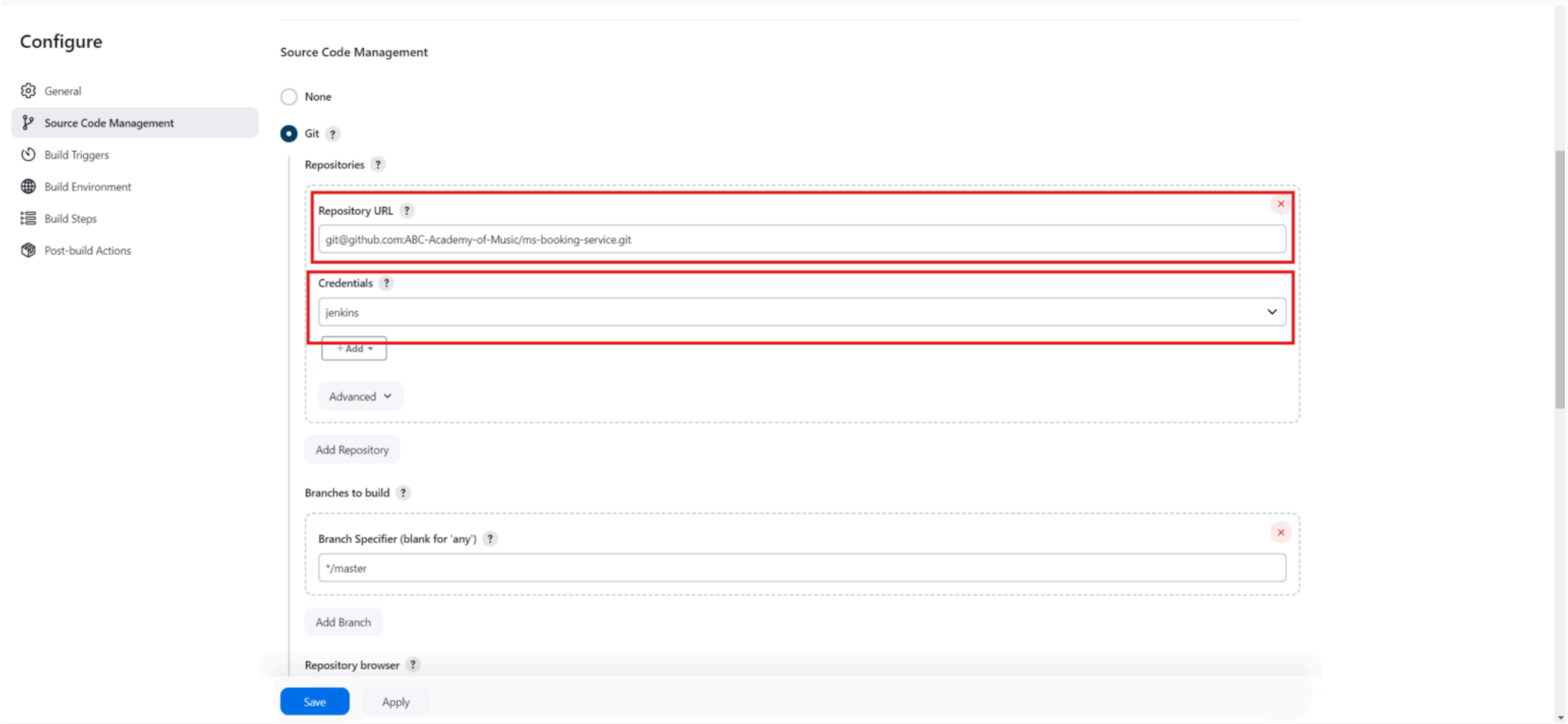
2. Go to `Source Code Management`.



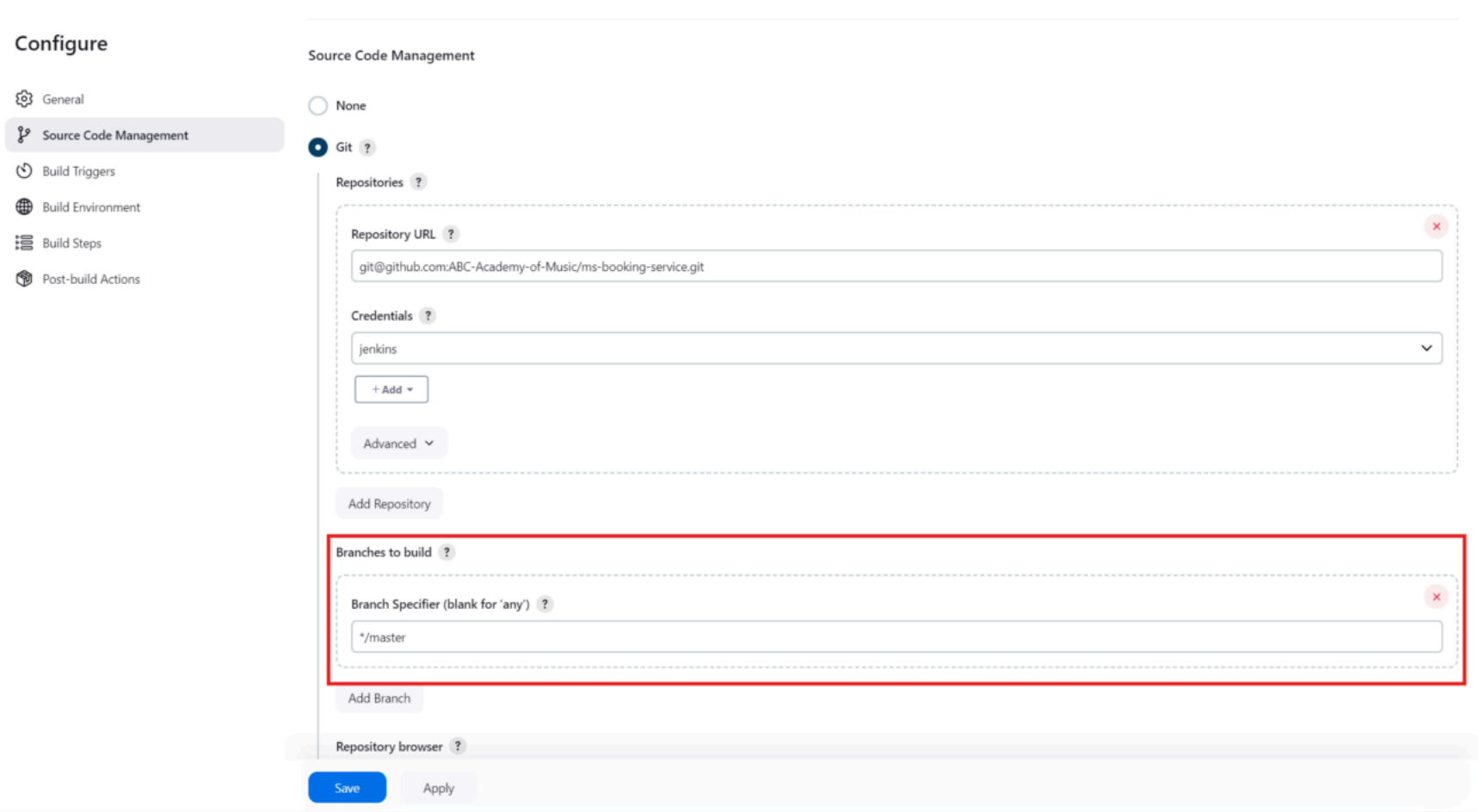
3. Go to your GitHub repository and copy the SSH link of the repo.



4. Go back to `Jenkins Source Code Management` and paste the link to `Repository URL`. Then choose the credential that is created previously.

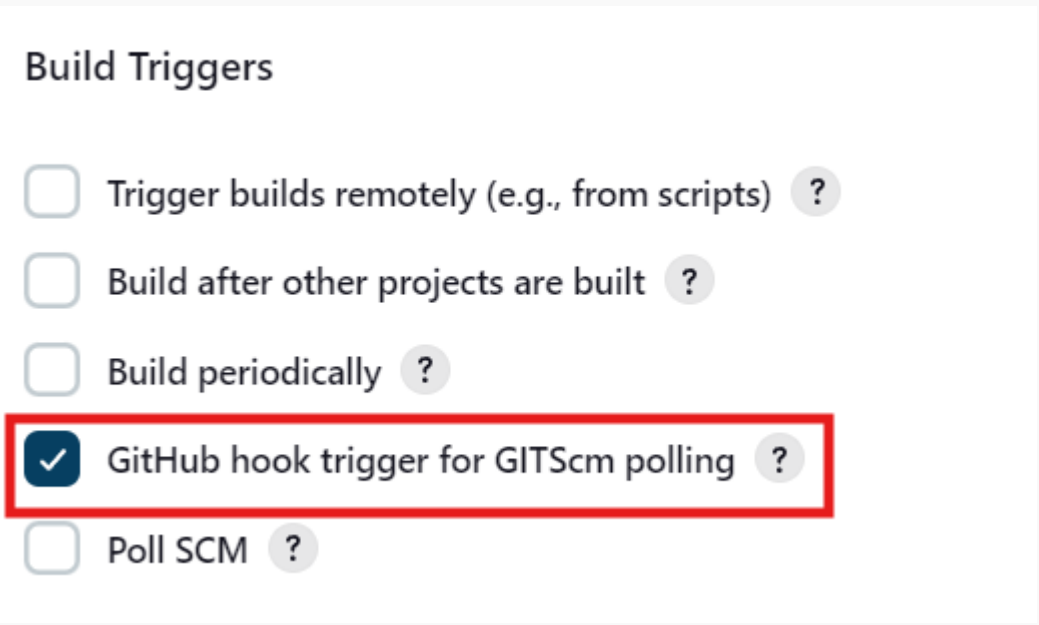


5. Change the branch to the specific one you want for deployment.

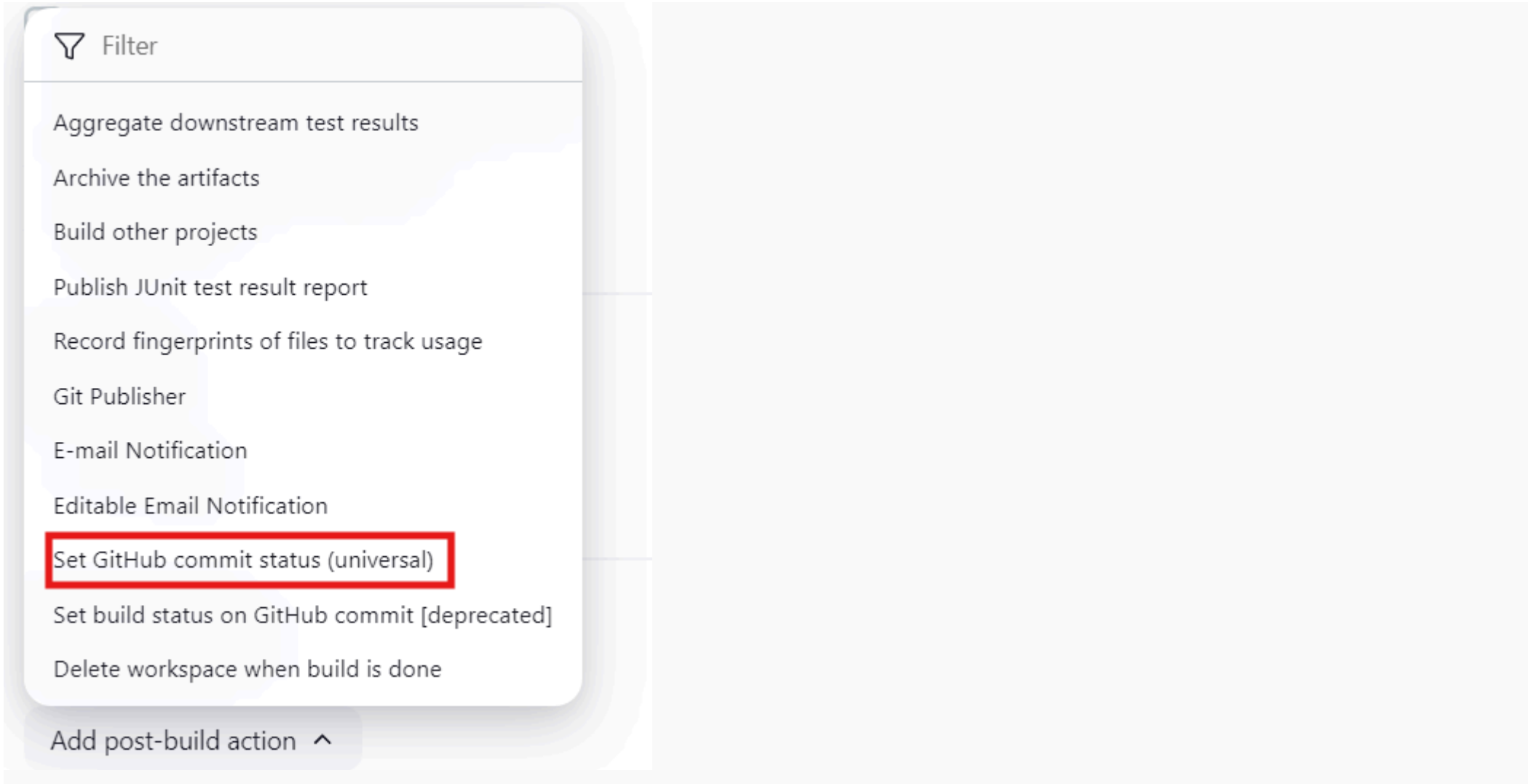


6. Enable GitHub hook trigger for GITScm polling.

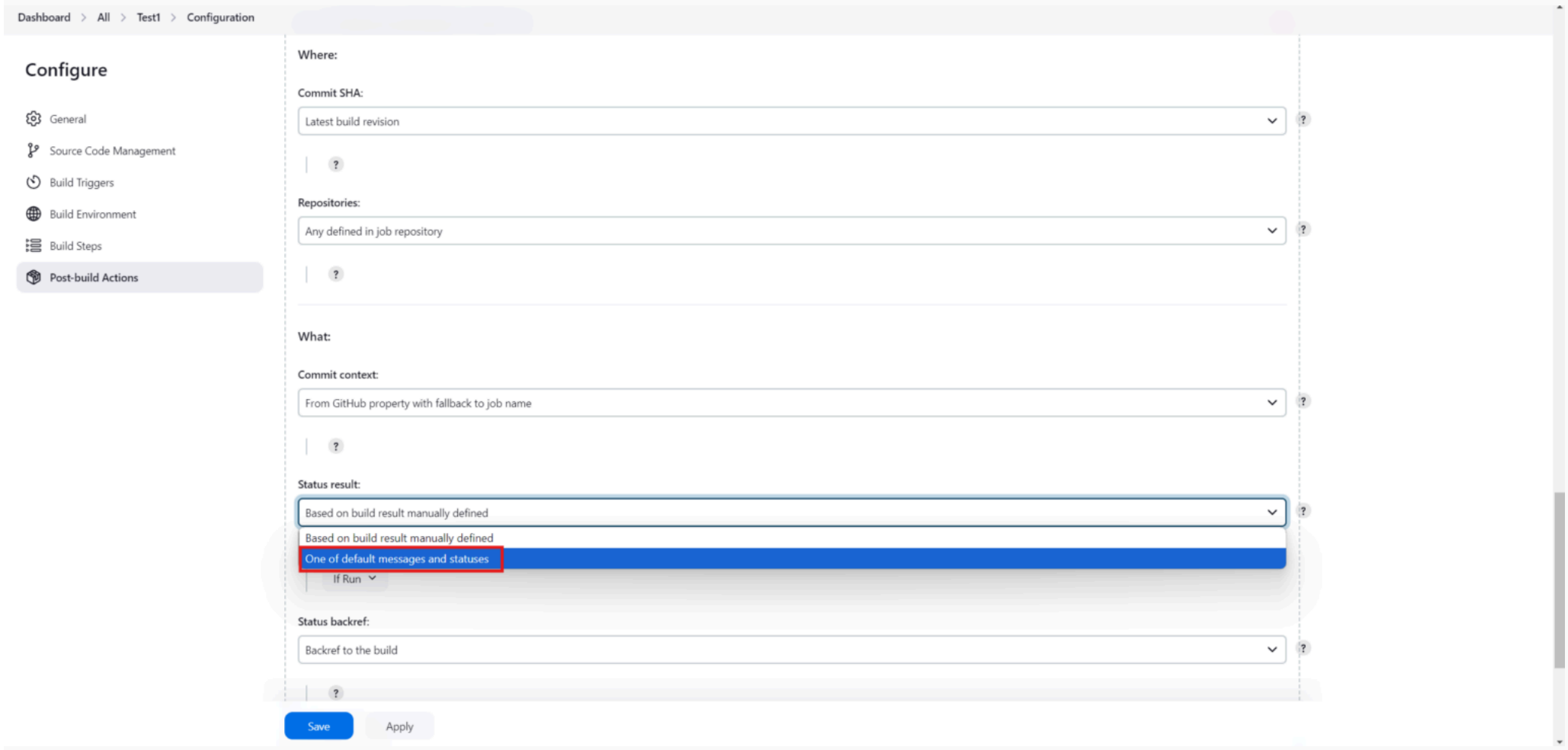
When Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository which matches the Git repository defined in SCM/Git section of this job. If they match and this option is enabled, GitHub Plugin triggers a one-time polling on GITScm. When GITScm polls GitHub, it finds that there is a change and initiates a build. The last sentence describes the behavior of Git plugin, thus the polling and initiating the build is not a part of GitHub plugin.



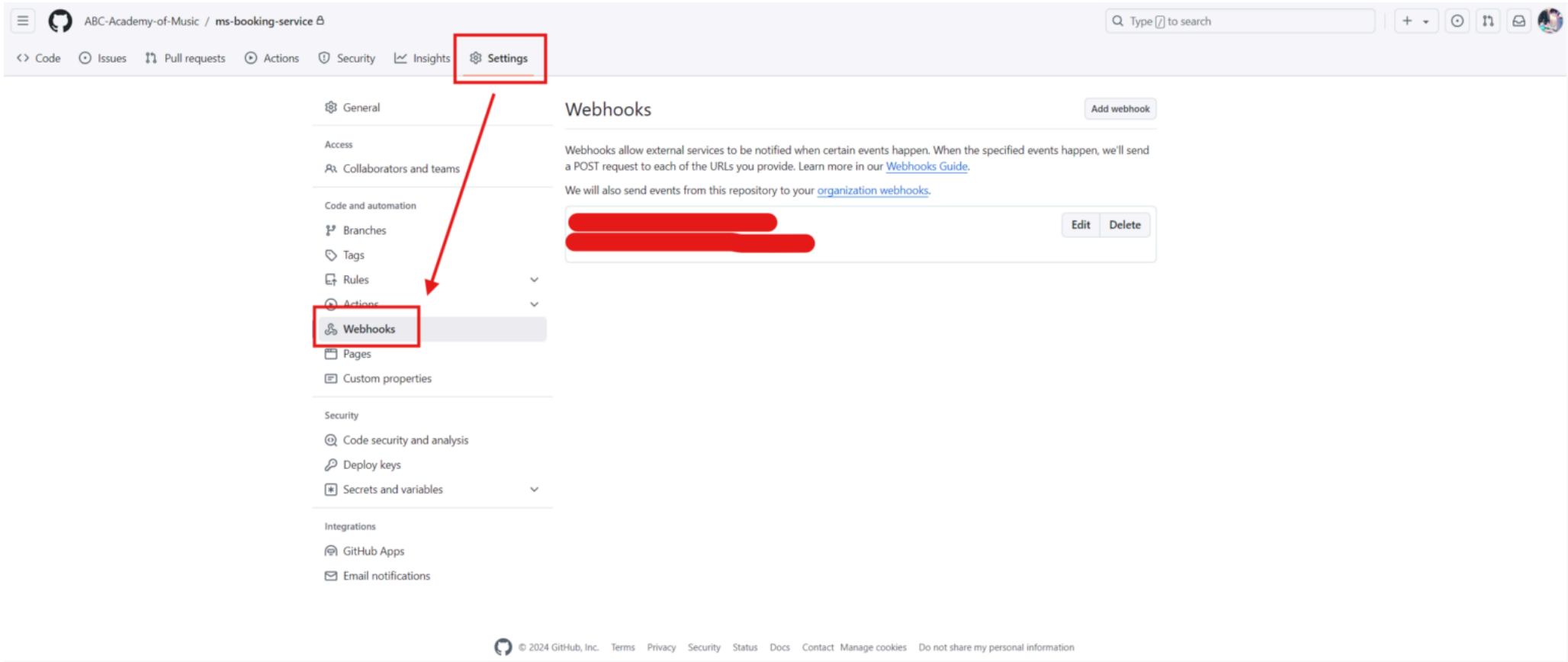
7. Set commit status to send it to GitHub.

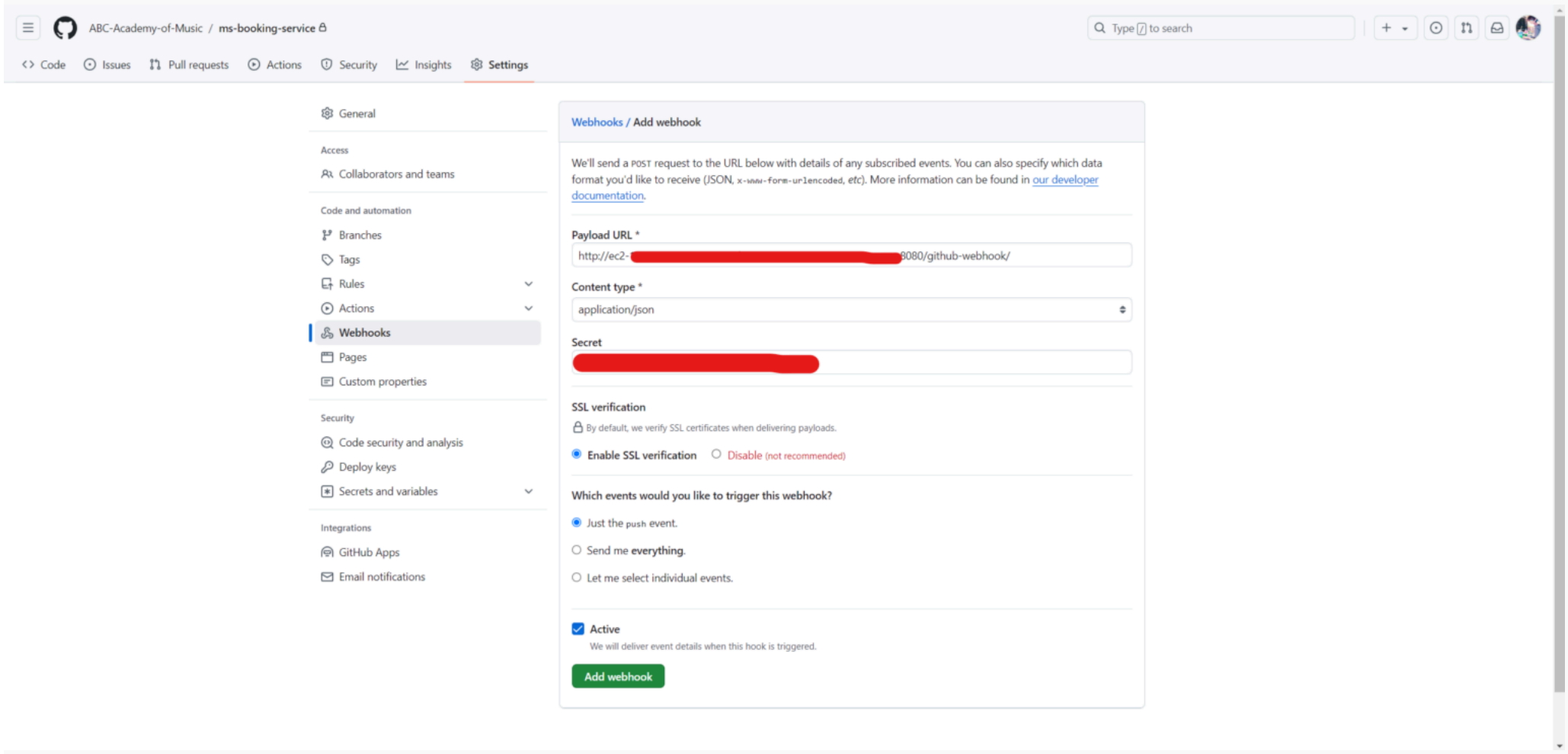


Keep the rest of settings the same, just change to the one with red colour highlight.



7. Go to `GitHub Repository → Settings → Webhooks → Add webhook`

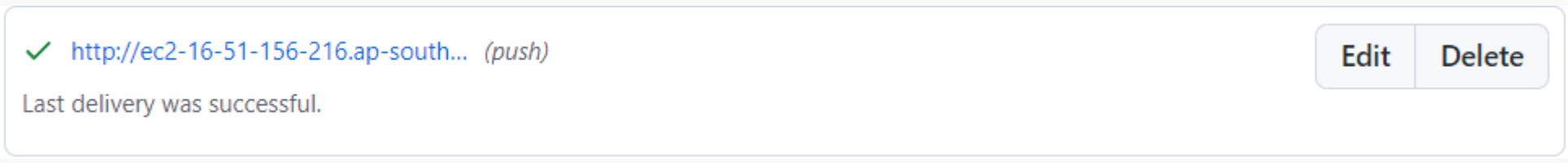




Put PAT to the `Secret` section. And put the Jenkins URL, then write an extention path on it as shown below.

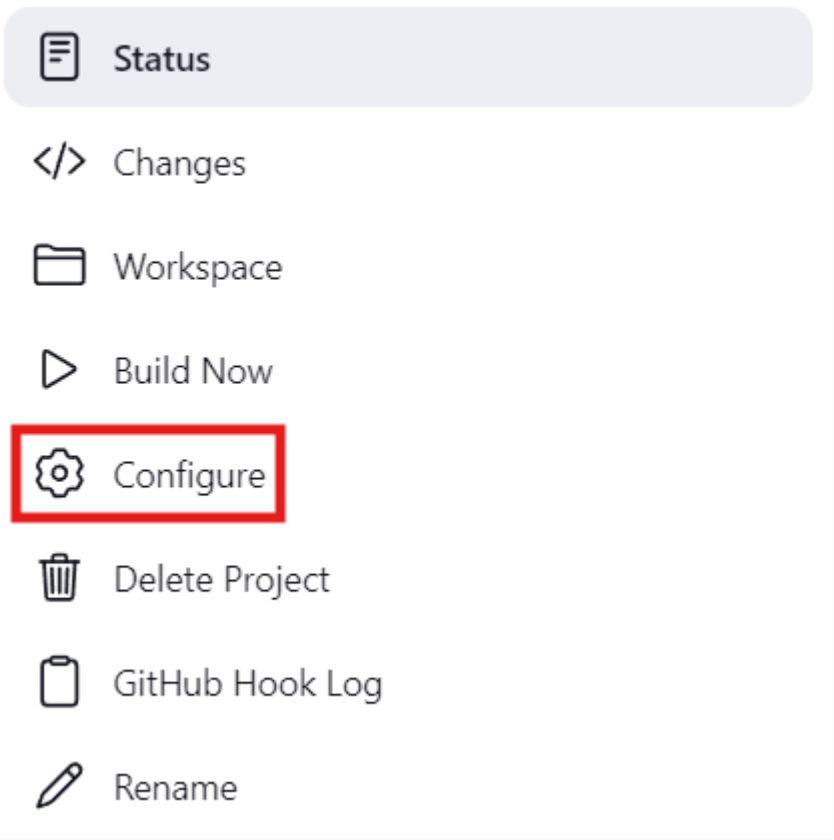
`http://<jenkins_url>/github-webhooks/`

GitHub will send a request to test if the provided url is working. If successful, it should be like the following picture.



Step 9: Create a building, testing, and deploying script for your project

1. Go back to Jenkins project, and click `Configure`.



2. Create a build step with the `execute shell`. Here is an **example** of Hinna booking service project's build script.

- First, we build the Spring Boot application to obtain the execution files.
- Second, we remove the existing container and image from the previous build. If that's the first time, those steps will be skipped as there's no previous container or image.
- Third, we build an image with the Dockerfile.
- Fourth, we push the image into a container with a specific port.

Build Steps

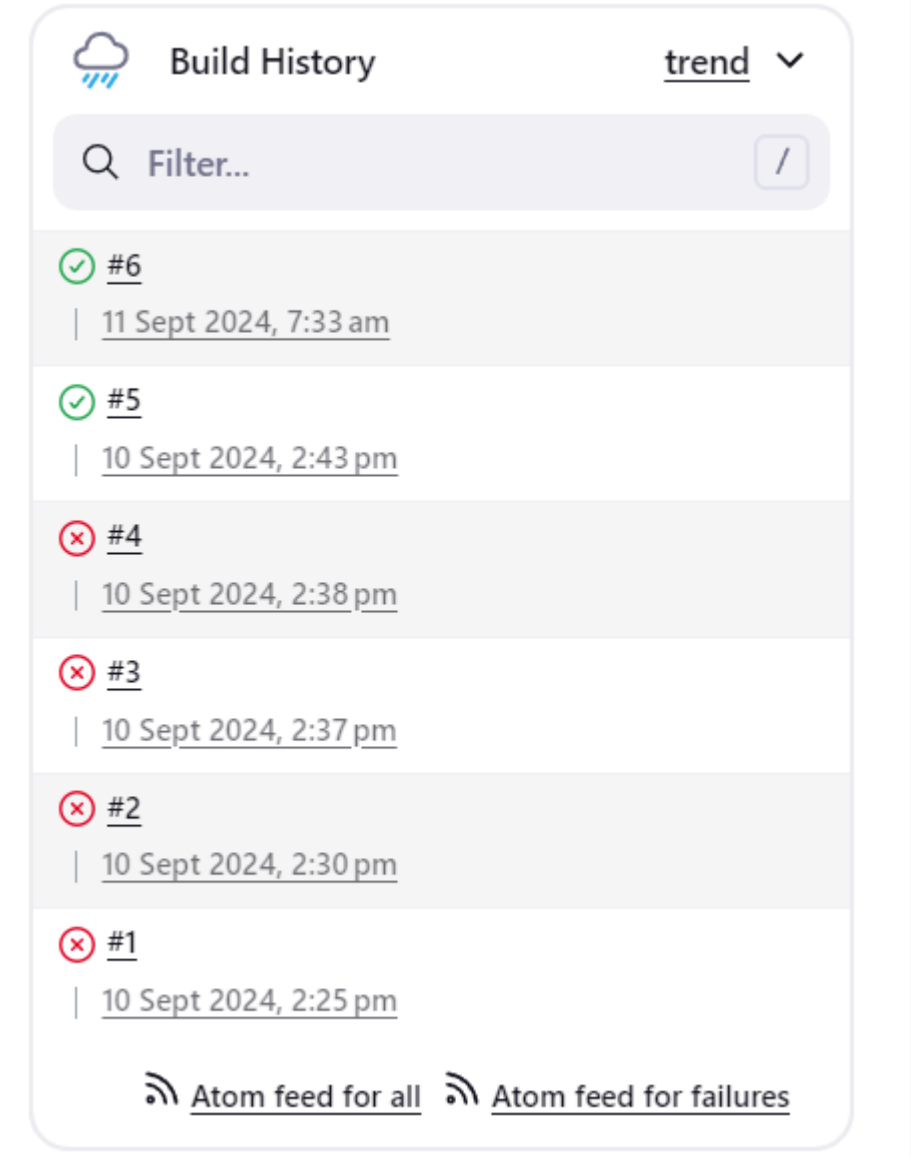
Execute shell

Command

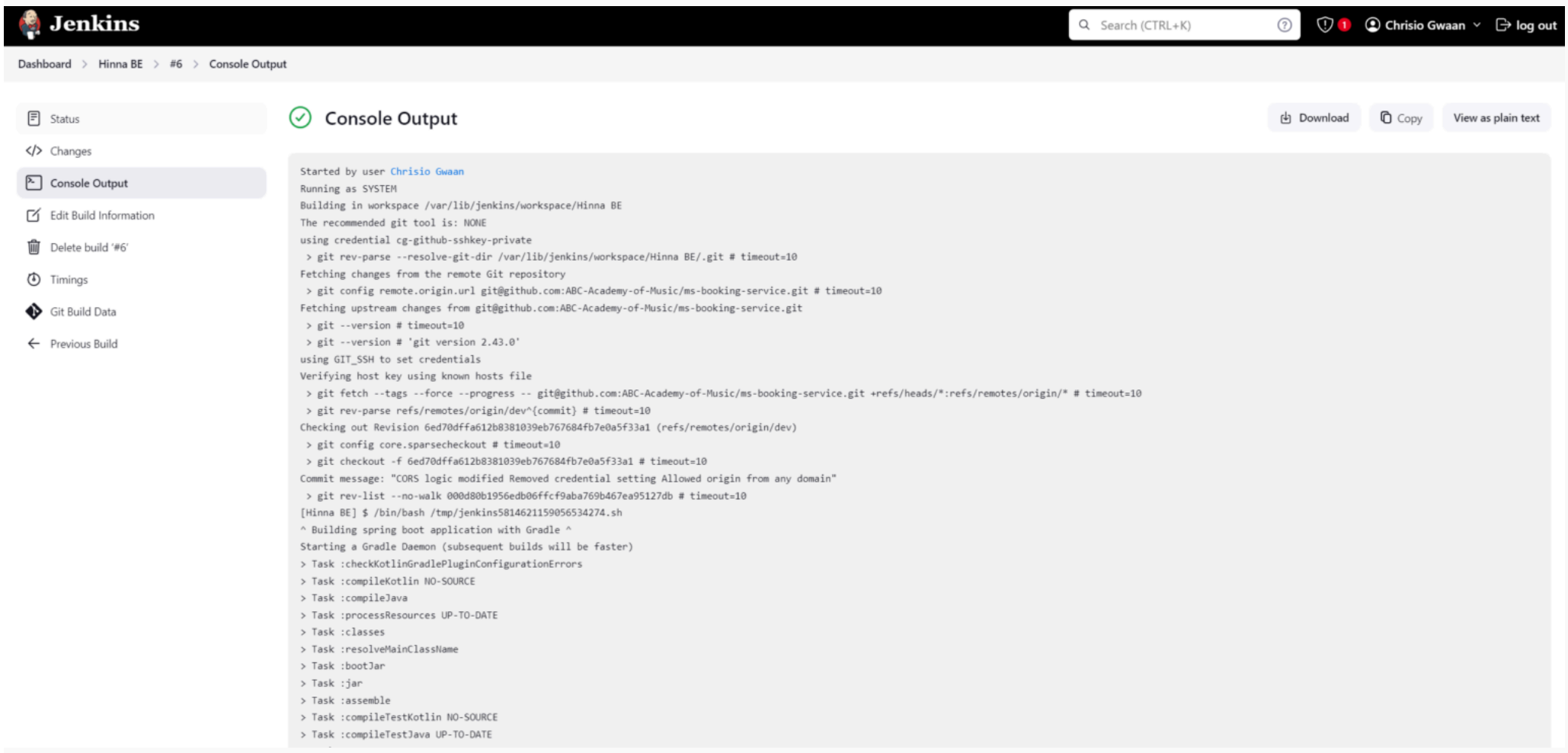
See the list of available environment variables

```
#!/bin/bash

echo "^ Building spring boot application with Gradle ^"
./gradlew build

echo "^ Stopping and removing container if it exists ^"
docker rm -f hinna-ms-booking 2>/dev/null || true

echo "^ Removing image if it exists ^"
docker rmi hinna-ms-booking 2>/dev/null || true

echo "^ Building an image using Dockerfile ^"
docker build -t hinna-ms-booking .

echo "^ Running the image to a container using specific port ^"
docker run -d --restart=always -p 8081:8081 -v /data/dump:/tmp --name hinna-ms-booking hinna-ms-booking
```

Now, everything has been set up properly. The Jenkins will show the record of every build.



For each build, they provide an option to see a real time console output.



Resources

1. Docker installation guide for Ubuntu
2. Jenkins installation guide for Linux
3. Jenkins GitHub Integration

Help us improve this article

Submit Feedback