Linux Academy

Hands-on Lab

Configure a Jenkins Freestyle Project

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Live! Lab page
- Username: linuxacademy
- Password: 123456
- Root Password: 123456

Related Courses

Certified Jenkins Engineer

Related Videos

Setting Up GitHub

Freestyle Project Configuration

Source Code Management and Git Plugin

Git Hooks and Other Build Triggers

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we use Docker to spin up Jenkins and create an simple Jenkins freestyle project, using GitHub to trigger builds.

Set Up a Docker Jenkins Master

Add the Docker repository:

```
[linuxacademy@ip] sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
[linuxacademy@ip] sudo yum clean all
```

Install the version of Docker CE we want to download:

```
[linuxacademy@ip] sudo yum install docker-ce
```

You will be prompted to confirm GPG keys.

Start the Docker services:

```
[linuxacademy@ip] sudo systemctl start docker
```

We now want to pull down the Jenkins Docker image:

```
[linuxacademy@ip] sudo docker pull jenkins:2.19.4
```

Create a directory for the Jenkins container to share on the host:

```
[linuxacademy@ip] sudo mkdir /var/jenkins_home
```

Run the container:

```
[linuxacademy@ip] sudo docker run -d -u root -p 8080:8080 -p 50000:50000
-v /var/jenkins_home:/var/jenkins_home jenkins:2.19.4
```

`-d` denotes that we are running the task in the background, while the `-p` flag shows which ports we are using. The `-v` maps the `/var/jenkins_home` directories between the two hosts.

To confirm the process is successful, navigate to your web browser and input the IP address of our host, followed by `:8080` to access the "Getting Started" Jenkins page. Copy the directory path provided, so we can determine the administrative password, then `cat` the file from your terminal to view:

```
[linuxacademy@ip] sudo cat /var/jenkins_home/secrets/
```

```
initialAdminPassword
```

Use that password to log in to Jenkins from your web browser. **Install suggested plugins**, then wait for the installation process to finish. We are now prompted to create our first admin user. Set the credentials to whatever you wish, but be sure to remember the username and password. **Save and Finish**. **Start using Jenkins**.

Forking a GitHub Repo

In your web browser, access and log in to github.com. Navigate to the repository for this lab, located at the following URL: https://github.com/linuxacademy/content-jenkins-freestyle

Press the **Fork** button in the upper right corner to fork it to your account.

Configure Git

Return to your lab server. We need to generate an SSH key. Use `docker ps` to retrieve your container ID:

```
[linuxacademy@ip] sudo docker ps
```

Now use `docker exec` to generate the key, inputting your container ID in the defined place:

```
[linuxacademy@ip] sudo docker exec -it <CONTAINTERID> bash
```

This drop us into a Bash shell inside of the Docker container. Generate the SSH key, using the defaults if you so desire:

```
root@dockerip: cd ~
root@dockerip: ssh-keygen
```

Copy your public key to add to GitHub:

```
root@dockerip: cat .ssh/id_rsa.pub
```

Return to your fork of the Jenkins Freestyle project on GitHub and press **Settings**, then **Deploy keys**. Press **Add deploy key**. Copy your public key into the text box. Give the key a title. Check *Allow write access*. **Add key**.

Press **Integrations & services** on the left menu. Under **Services**, select **Add service**, then find *Jenkins (Git plugin)*. For the Jenkins URL input the server's IP address, including the port 8080, then press Add Service.

Return to the **Jenkins Dashboard** and click on **New Item**. Give it an item name of *Lab Freestyle*, then select **Freestyle project**. Click **OK**.

Check the *GitHub project* checkbox, and input the GitHub URL for your fork of the project. Under **Source Code Management**, select *Git*, then input the repository's clone with SSH link, found under the green **Clone or download** button on the repository's page. Press **Add**, **Jenkins** besides the **Credentials** option. For **Kind** select *SSH Username with private key*. Set the **Username** to *root*, then set **Private Key** to *From the Jenkins master ~/.ssh*. Press **Add**. Now select the key from the credentials dropdown.

For **Repository Browser**, select *githubweb* and set the URL to the main project URL. **Save** the changes.

Adding Build Steps and a Trigger

Under the project page, select **Configure** from the left menu. Scroll down to the **Build Triggers** section and check *Poll SCM*. We need to do this to get the GitHub plugin to work. Leave the schedule box empty.

Find the **Build** section and under the dropdown menu select *Execute shell*. We want to set some environment variable interpolation. Input:

```
echo "${JOB_NAME} [${BUILD_NUMBER}]"
```

Add a second *Execute shell* with the following to compile the Java code:

```
echo "Building Java Code"
javac -d . src/*.java
echo Main-Class: Rectangulator > MANIFEST.MF
jar -cmf MANIFEST.MF rectangle.jar *.class
```

Add another *Execute shell* to execute the rectangle file:

```
java -jar rectangle.jar 2 3
```

Save.

Open your local workstation terminal (ideally already configured to use git), and clone the freestyle project locally:

```
[user@workstation] git clone https://github.com/USERNAME/content-
jenkins-freestyle.git
```

Navigate into the directory and make some changes to the `README.md` file:

```
[user@workstation] cd content-jenkins-freestyle/
```

```
[user@workstation] echo "My rectangle Java code" >> README.md
```

Commit the changes:

```
[user@workstation] git commit -am "Added subtitle to the README.md file"
```

Push the changes to master:

```
[user@workstation] git push origin master
```

Return to the Jenkins console and wait for the job to start (under **Build History** in the left menu). Click the build, then go to **Console Output** to view the steps we input in the Build section of our configuration. We can also see the results of the script being output.

This lab is now complete!