Install Jenkins on rocky

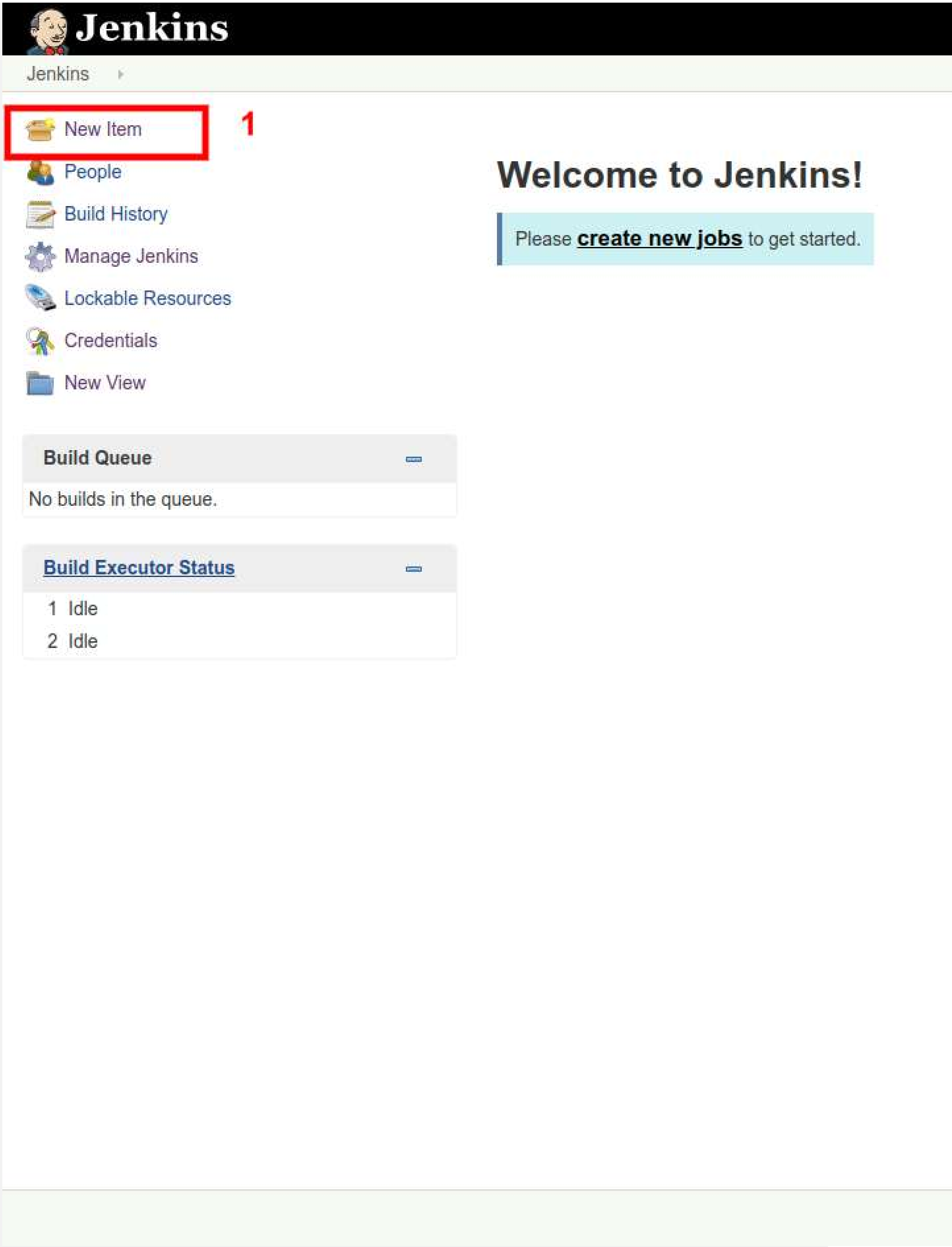
https://sysadminxpert.com/install-jenkins-on-rocky-linux-8-or-centos-8/

CI/CD Pipeline Using Jenkins, Git and Maven

appfleet team
Follow
Mar 18, 2020 · 6 min read

Step 1: Run Your First Jenkins Job

Once you are through with the installation and configuration of Jenkins, now let us create your first Jenkins Job. Login into Jenkins dashboard, find *New Item* as shown below:



Welcome to Jenkins!

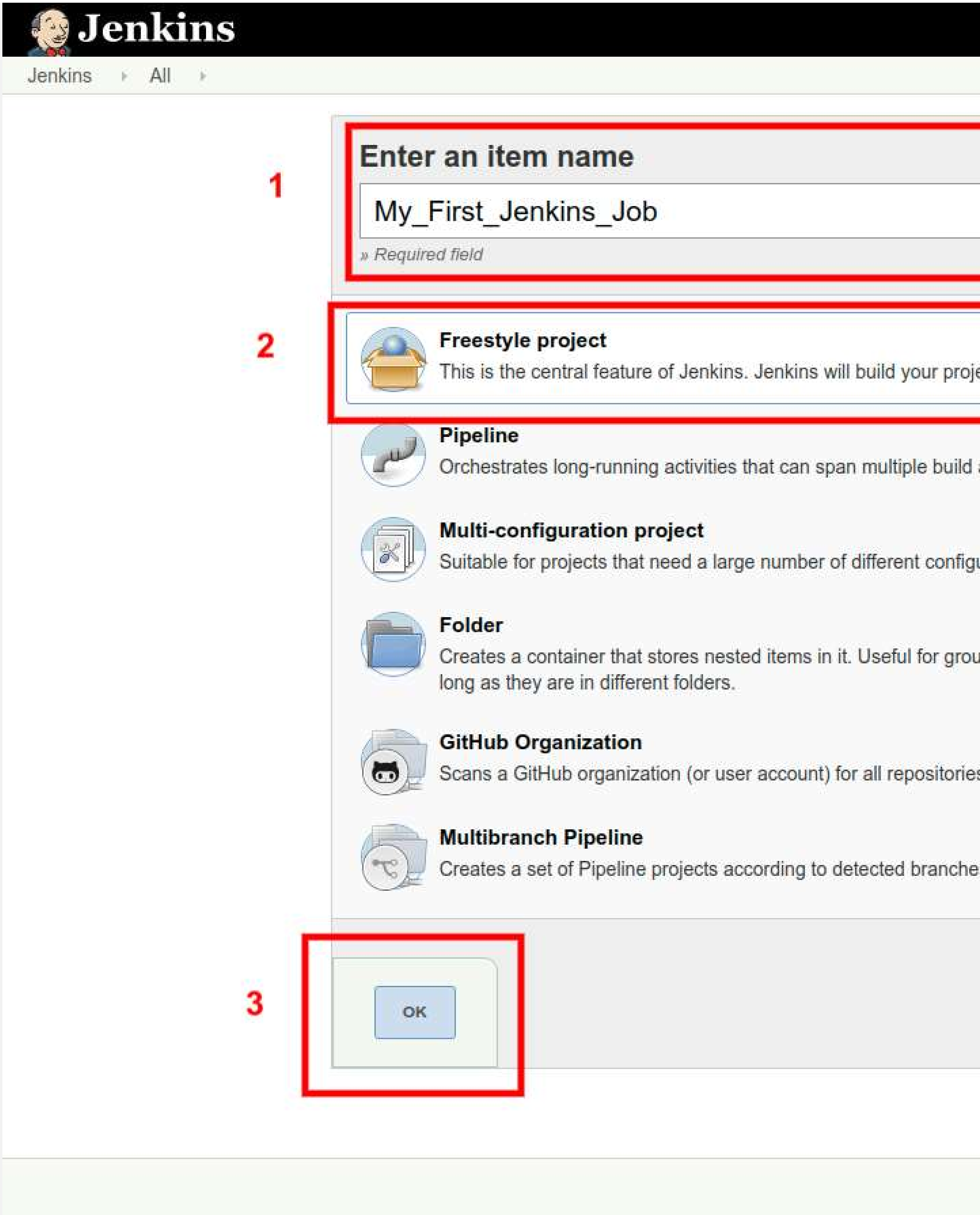
Please **create new jobs** to get started.

No builds in the queue.

1 Idle

2 Idle

In the next page, enter the item name **My_First_Jenkins_Job**, select *Freestyle project* and click on *OK*, as shown below:



Enter an item name

My_First_Jenkins_Job

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your proje

Pipeline
Orchestrates long-running activities that can span multiple build a

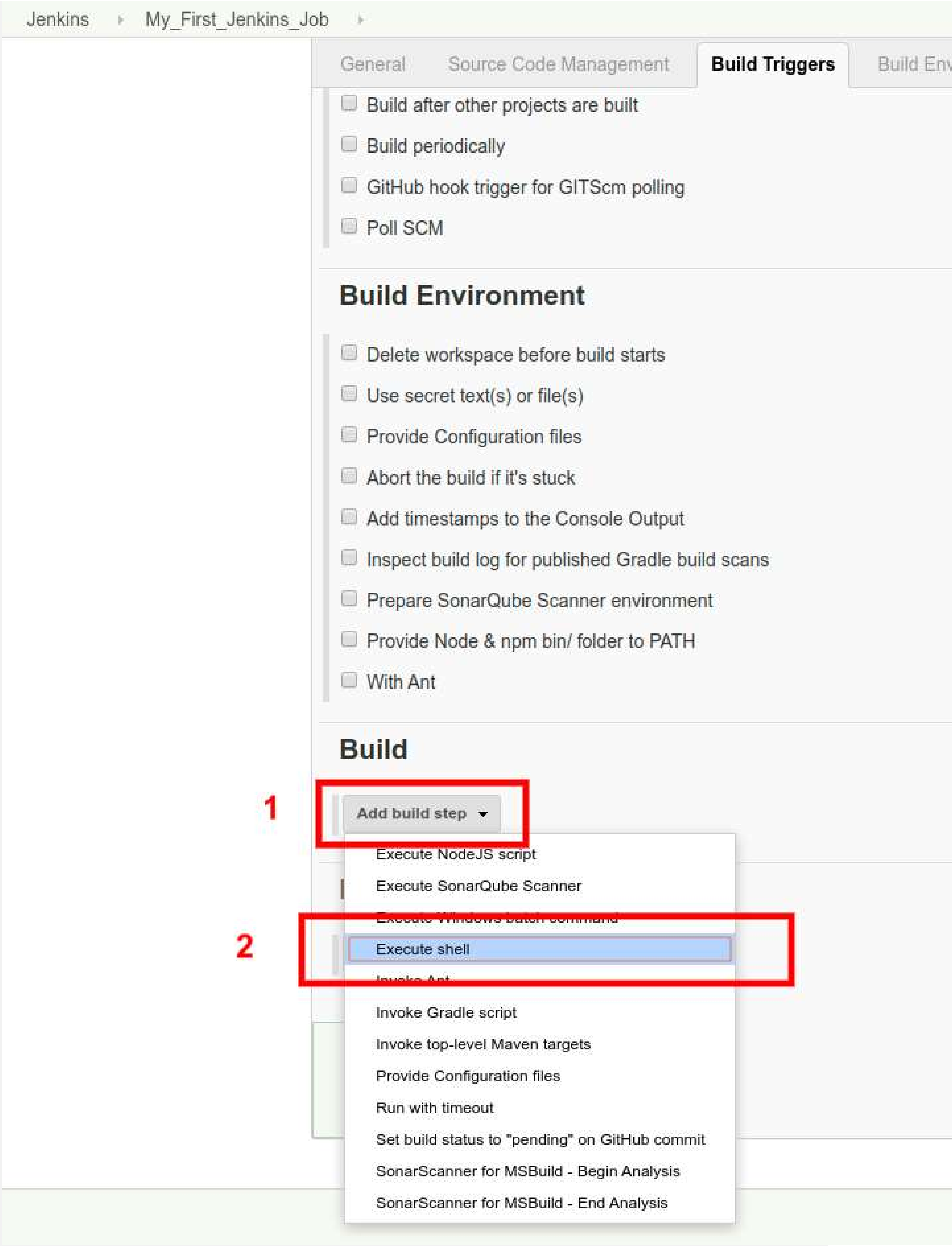
Multi-configuration project
Suitable for projects that need a large number of different configu

Folder
Creates a container that stores nested items in it. Useful for grou long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branche

Under `Jenkins > My_First_Jenkins_Job`, enter a short description into the text field for future reference. As of now, we are going to leave the *Source Code Management* field to *None*. Next, go to *Build* and select *Execute Shell* from the drop-down menu, as shown below:



☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

Build Environment

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Provide Configuration files

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ Prepare SonarQube Scanner environment

☐ Provide Node & npm bin/ folder to PATH

☐ With Ant

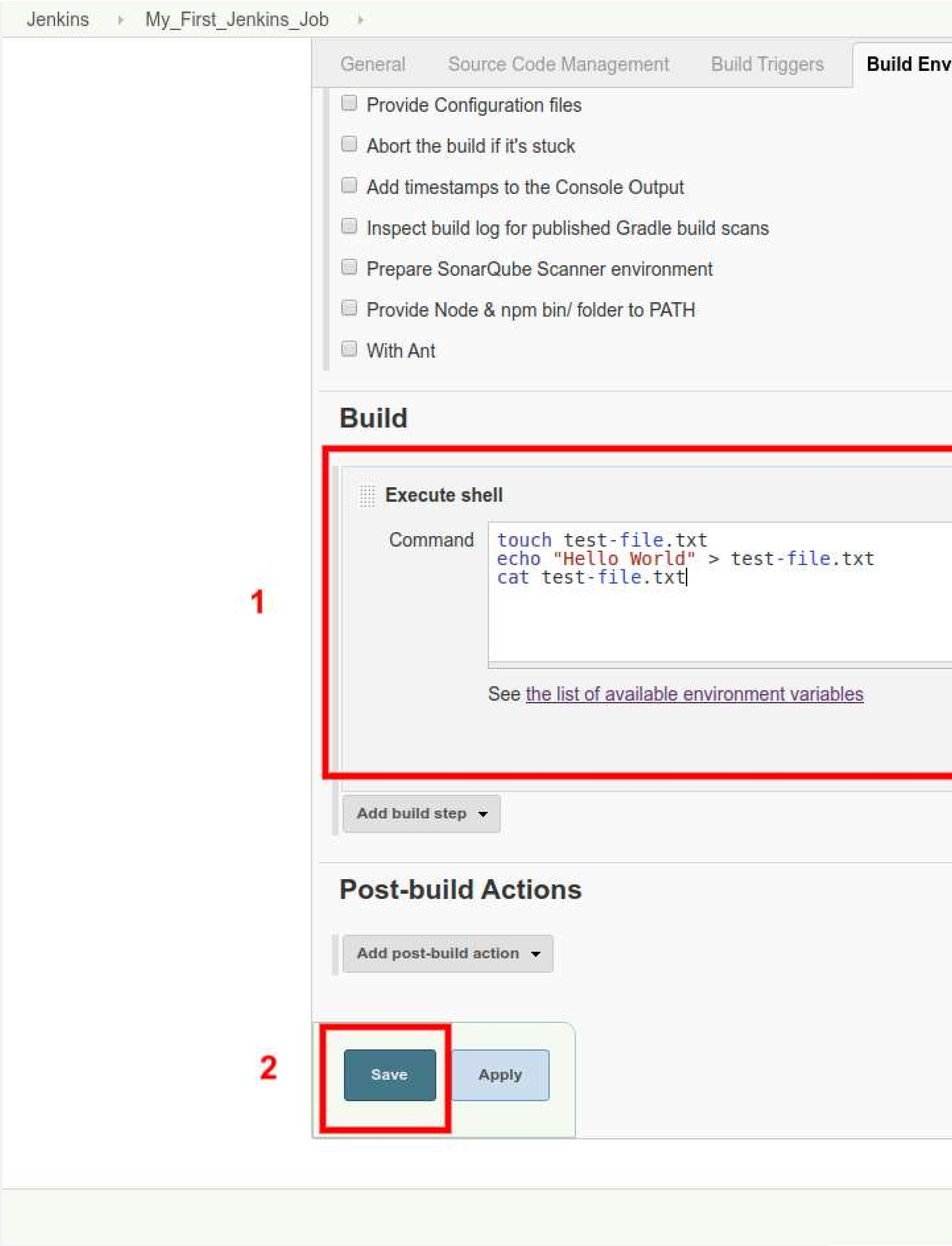
Build

1
Add build step
Execute NodeJS script
Execute SonarQube Scanner
Execute Windows batch command
2
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Provide Configuration files
Run with timeout
Set build status to "pending" on GitHub commit
SonarScanner for MSBuild - Begin Analysis
SonarScanner for MSBuild - End Analysis

Next, let us try to run simple Shell commands:

```
touch test-file.txt
echo "Hello World" > test-file.txt
cat test-file.txt
```

Enter the above-mentioned commands in the text box, and save the configuration, as shown in the figure:



□ Provide Configuration files

□ Abort the build if it's stuck

□ Add timestamps to the Console Output

□ Inspect build log for published Gradle build scans

□ Prepare SonarQube Scanner environment

□ Provide Node & npm bin/ folder to PATH

□ With Ant

Build

Execute shell

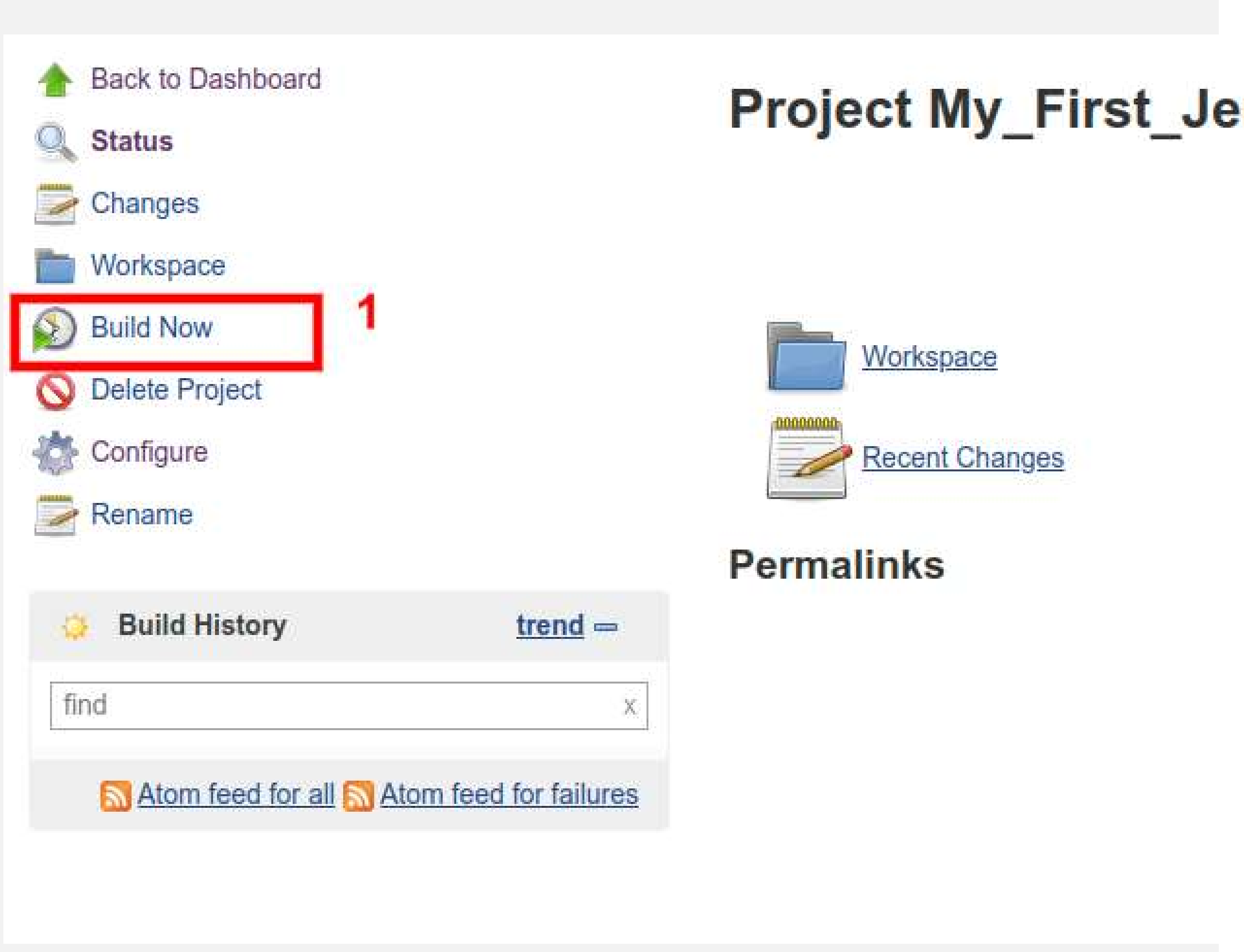
Command

```
touch test-file.txt
echo "Hello World" > test-file.txt
cat test-file.txt
```

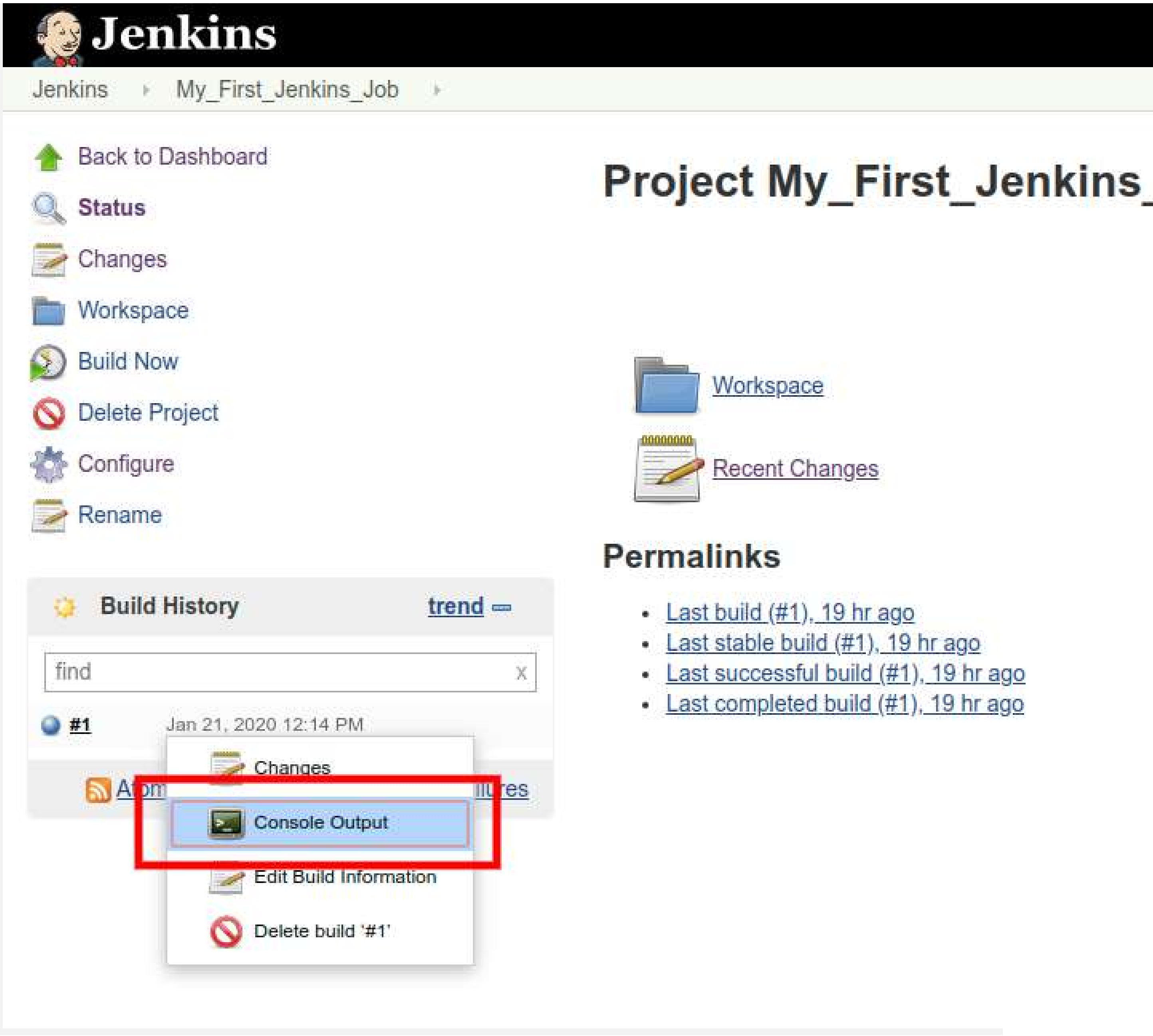
See the list of available environment variables

Post-build Actions

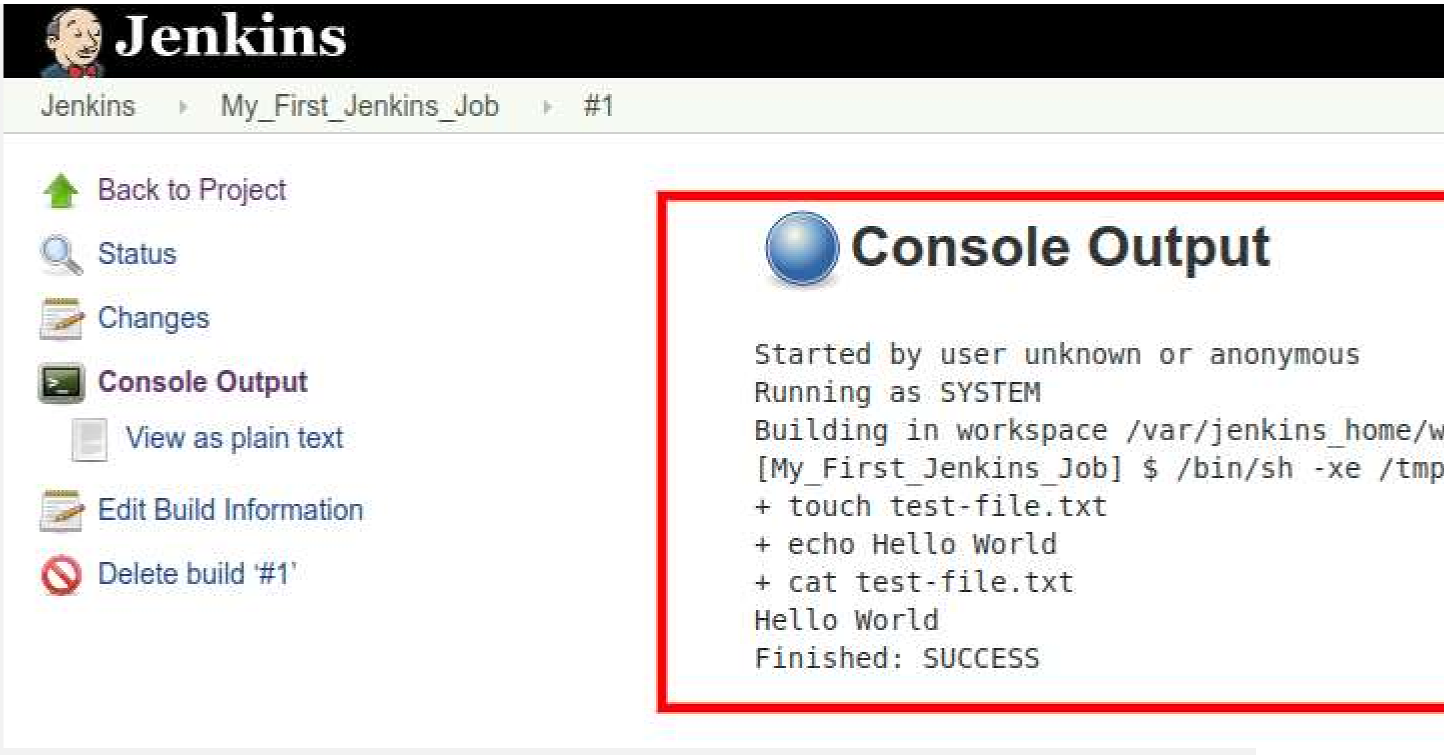
It’s time to build your project **My_First_Jenkins_Job**. On the left-hand side menu, click the *Build Now* option as shown below:



To see the console output of your first build, follow the screenshot below:



On the next page, you will see the following output:



```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace /var/jenkins_home/w
[My_First_Jenkins_Job] $ /bin/sh -xe /tmp
+ touch test-file.txt
+ echo Hello World
+ cat test-file.txt
Hello World
Finished: SUCCESS
```

You can see the executed command output, and in the last line, `Success` indicates that our job finished successfully. This is how you run a Jenkins job.