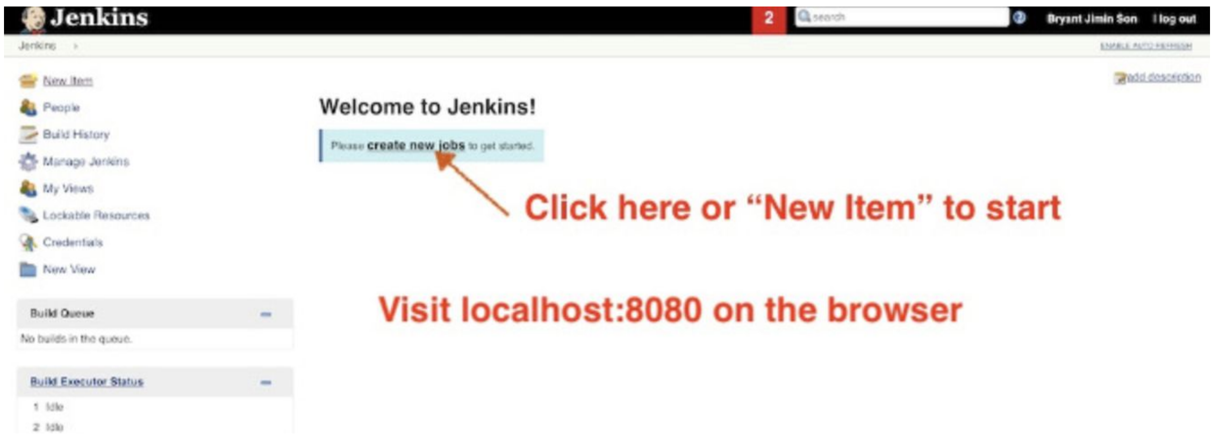
Create Jobs and Pipeline

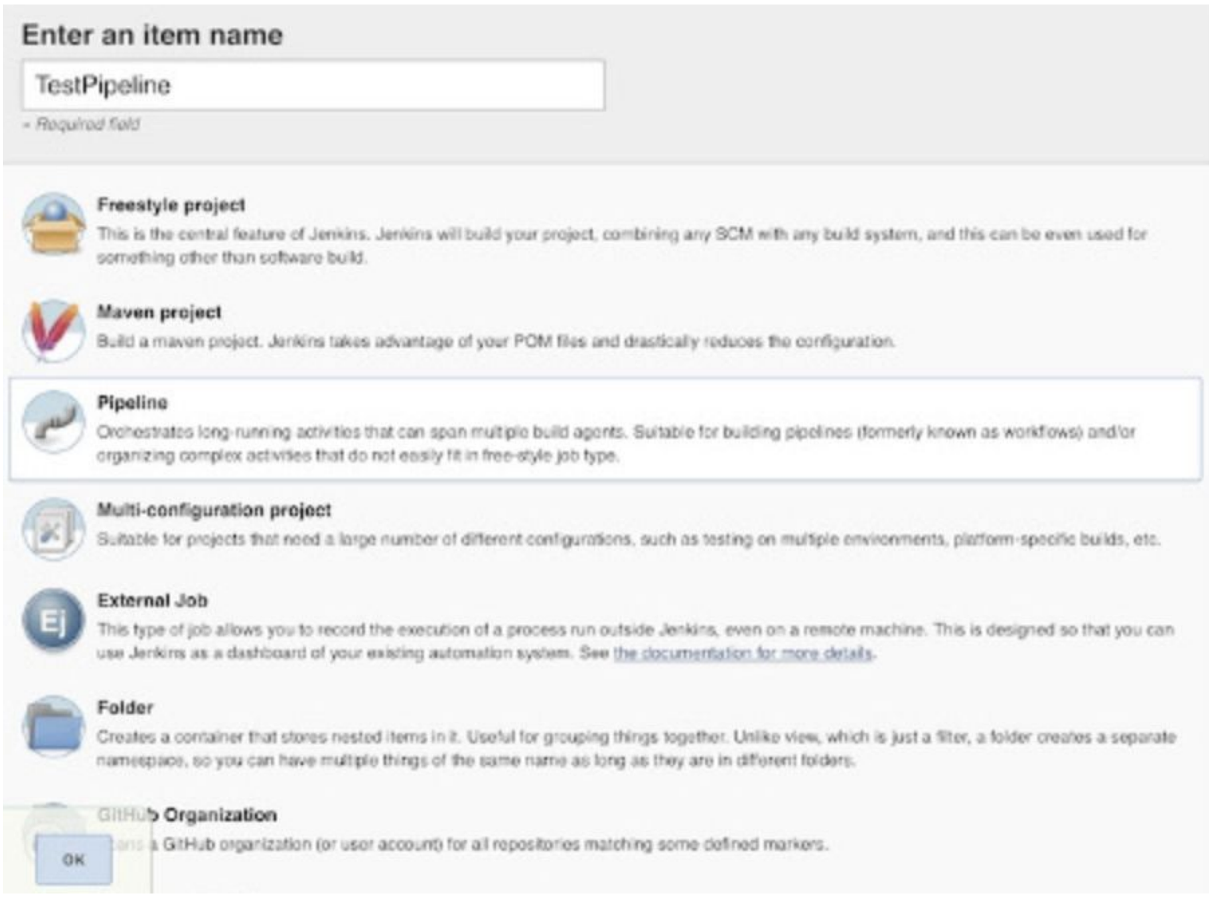
Here are some tutorials that introducing how to create jobs and pipeline:
https://www.youtube.com/watch?v=m0a2CzgLNsc
https://opensource.com/article/19/9/intro-building-cicd-pipelines-jenkins

Again, you just need to follow the following main steps we extracted from the external resources.

Step 1: Create a new Jenkins job
Open a web browser and navigate to **localhost:8080**. Unless you have a previous Jenkins installation, it should go straight to the Jenkins dashboard. Click **Create New Jobs**. You can also click **New Item** on the left.



Step 2: Create a pipeline job
In this step, you can select and define what type of Jenkins job you want to create. Select **Pipeline** and give it a name (e.g., TestPipeline). Click **OK** to create a pipeline job.


Enter an item name

TestPipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See the documentation for more details.

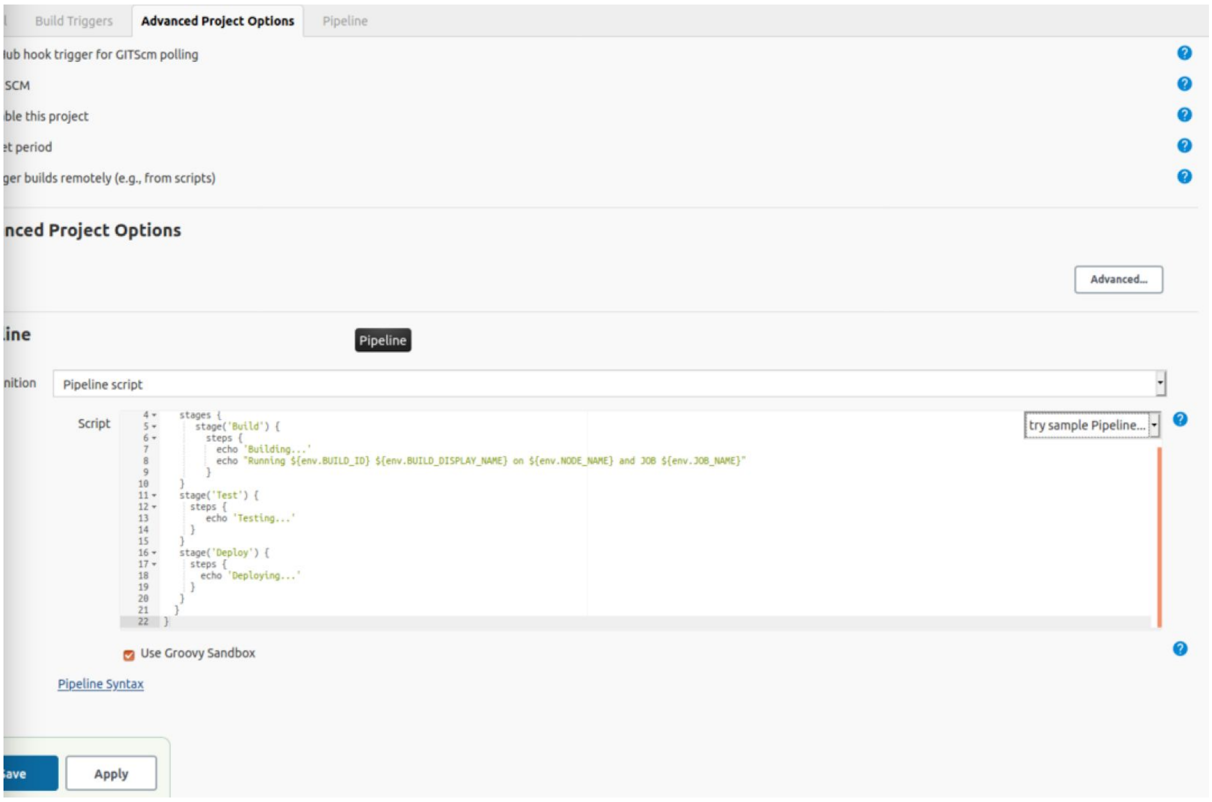
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

You will see a Jenkins job configuration page. Scroll down to find Pipeline section. There are two ways to execute a Jenkins pipeline. One way is by directly writing a pipeline script on Jenkins, and the other way is by retrieving the Jenkins file from SCM (source control management). We will go through both ways in the next two steps.

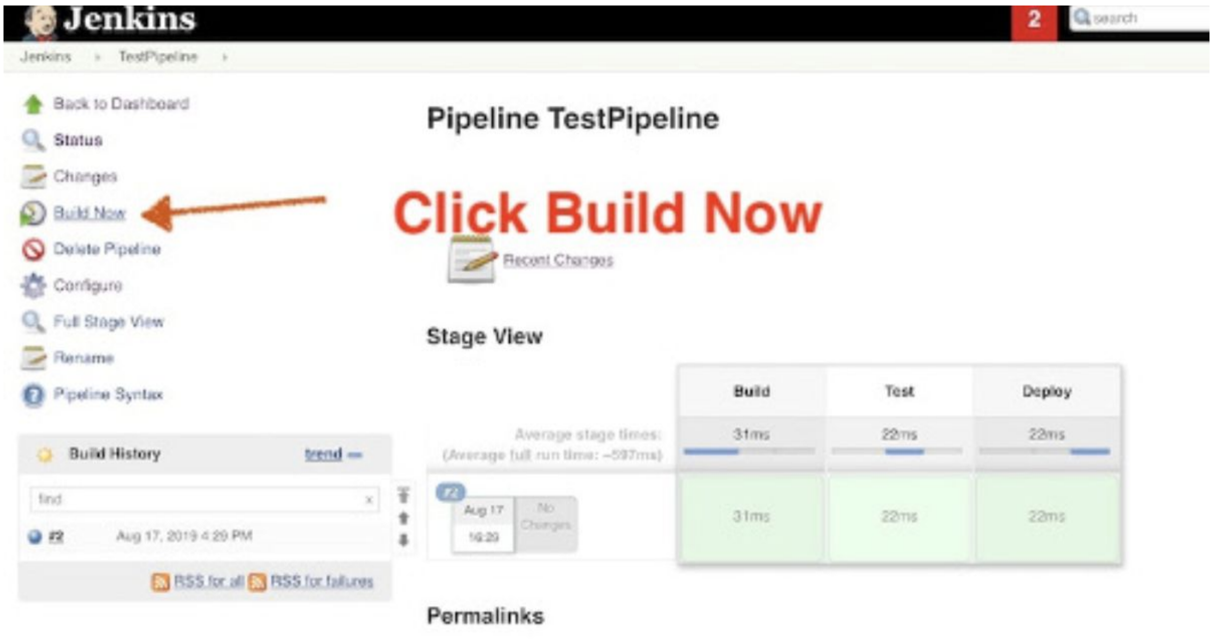
Step 3: Configure and execute a pipeline job through a direct script
To execute the pipeline with a direct script, begin by copying the contents of the sample Jenkinsfile from GitHub (https://github.com/bryantson/CICDPractice/blob/master/Jenkinsfile). Choose **Pipeline script** as the **Destination** and paste the **Jenkinsfile** contents in **Script**. Spend a little time studying how the Jenkins file is structured. Notice that there are three Stages: Build, Test, and Deploy, which are arbitrary and can be anything. Inside each Stage, there are Steps; in this example, they just print some random messages.

Click **Save** to keep the changes, and it should automatically take you back to the Job Overview.

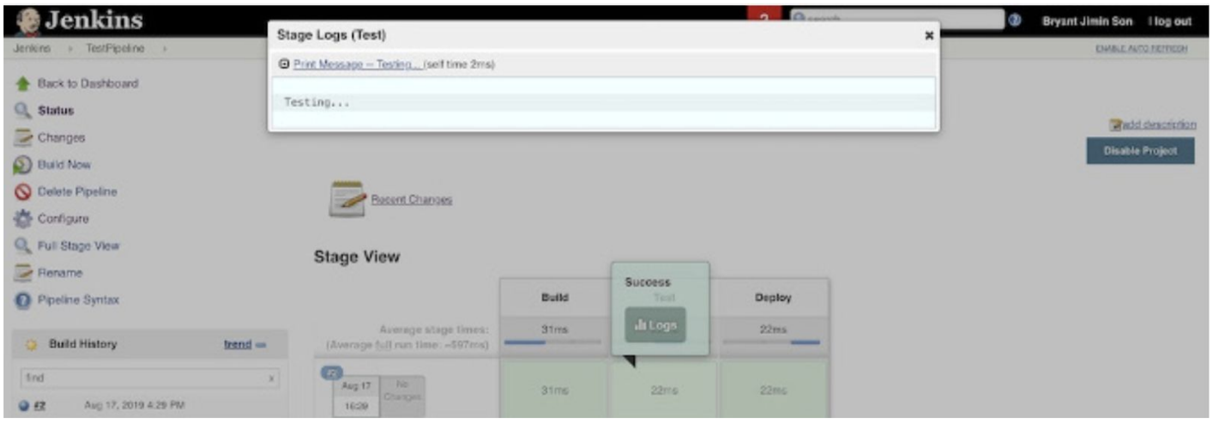


```
  stages {
    stage('Build') {
      steps {
        echo 'Building...'
        echo "Running ${env.BUILD_ID} ${env.BUILD_DISPLAY_NAME} on ${env.NODE_NAME} and JOB ${env.JOB_NAME}"
      }
    }
    stage('Test') {
      steps {
        echo 'Testing...'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying...'
      }
    }
  }
}
```

To start the process to build the pipeline, click **Build Now**. If everything works, you will see your first pipeline (like the one below).



To see the output from the pipeline script build, click any of the Stages and click **Log**. You will see a message like this.



Stage Logs (Test)

Print Message -- Testing... (self time 2ms)

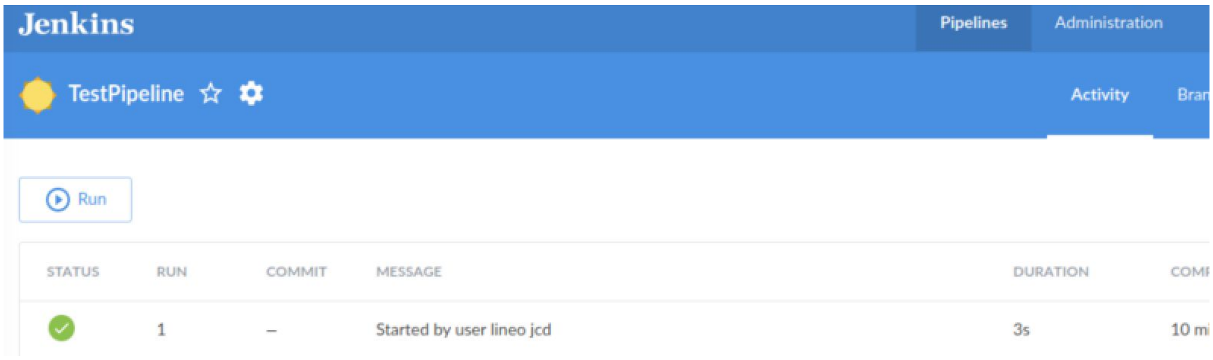
```
Testing...
```

As I mentioned before, Blue Ocean provides better UI for Jenkins. You can also see this pipeline in Blue Ocean.

Accessing Blue Ocean

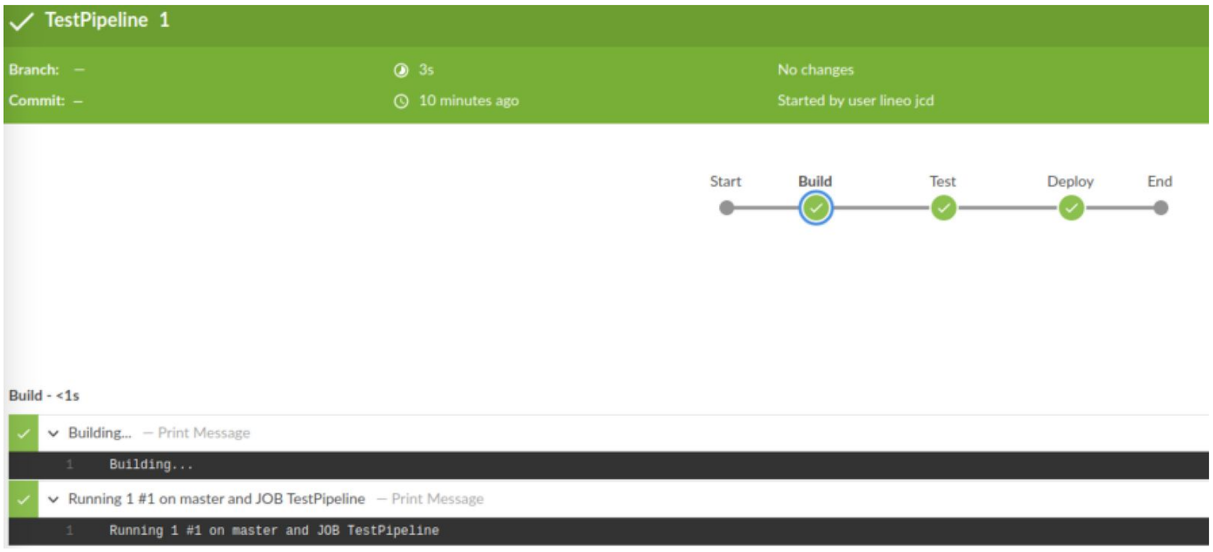
Once a Jenkins environment has Blue Ocean installed, after logging in to the Jenkins classic UI, you can access the Blue Ocean UI by clicking Open Blue Ocean on the left.

You will see the pipeline that has been run, click on the record.



STATUS	RUN	COMMIT	MESSAGE	DURATION	COMI
✓	1	–	Started by user lineo jcd	3s	10 mi

You will see the advanced pipeline visualisations


TestPipeline 1
Branch: –
Commit: –
3s
10 minutes ago
No changes
Started by user lineo jcd
Start
Build
Test
Deploy
End
Build - <1s
Building... – Print Message
Running 1 #1 on master and JOB TestPipeline – Print Message

```
Building...
```

```
Running 1 #1 on master and JOB TestPipeline
```

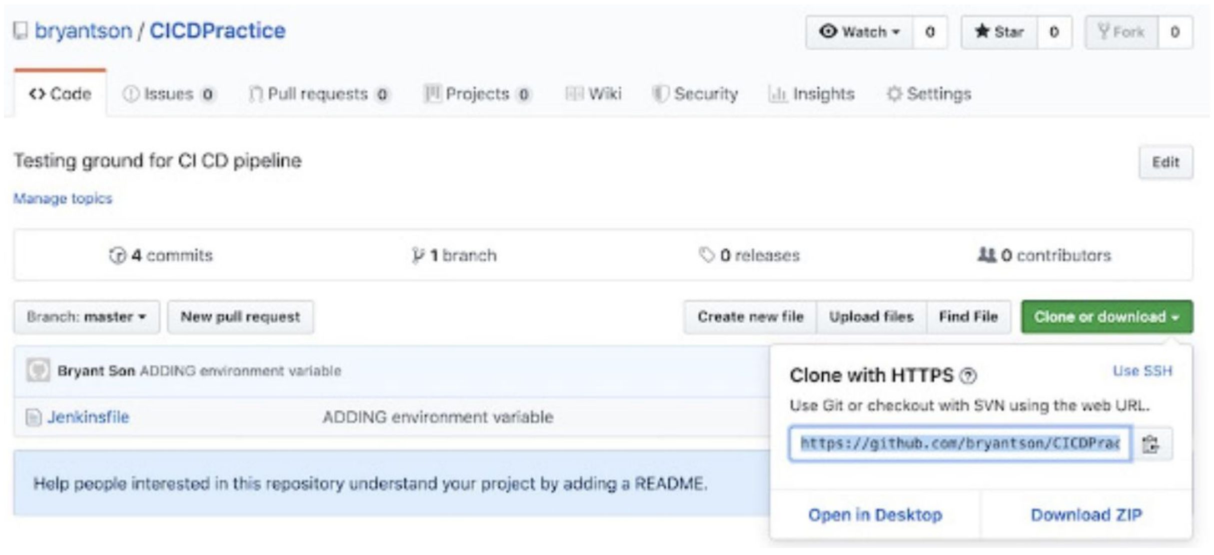
Switching to the classic UI

Blue Ocean does not support some legacy or administrative features of Jenkins that are necessary to some users.
If you need to leave the Blue Ocean user experience to access these features, click the Go to classic icon at the top of common section of Blue Ocean's **navigation bar**.

Clicking this button takes you to the equivalent page in the Jenkins classic UI, or the most relevant classic UI page that parallels the current page in Blue Ocean.

Step 4: Configure and execute a pipeline job with SCM

Now, switch gears: In this step, you will Deploy the same Jenkins job by copying the **Jenkinsfile** from a source-controlled GitHub. In the same GitHub repository(https://github.com/bryantson/CICDPractice.git), pick up the repository URL by clicking **Clone or download** and copying its URL.


bryantson / CICDPractice

Watch 0 · Star 0 · Fork 0

Code · Issues 0 · Pull requests 0 · Projects 0 · Wiki · Security · Insights · Settings

Testing ground for CI CD pipeline

Manage topics

4 commits · 1 branch · 0 releases · 0 contributors

Branch: master · New pull request · Create new file · Upload files · Find File · Clone or download

Bryant Son ADDING environment variable

Jenkinsfile ADDING environment variable

Help people interested in this repository understand your project by adding a README.

Clone with HTTPS

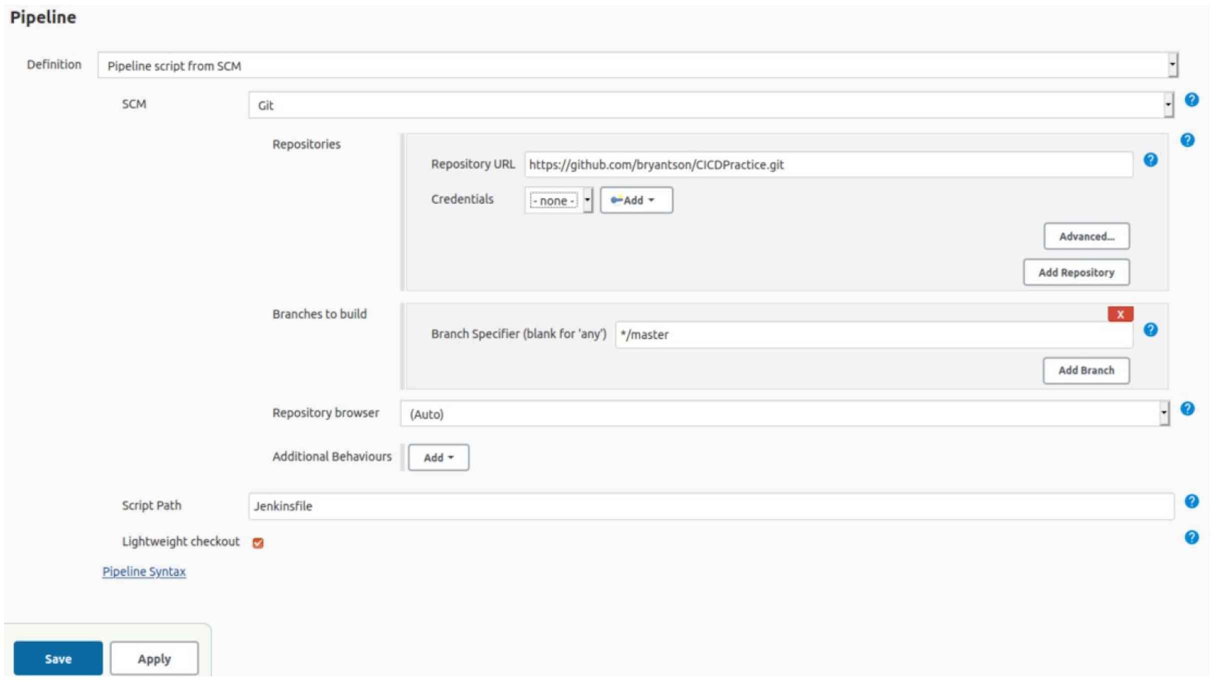
Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/bryantson/CICDPrac

Open in Desktop · Download ZIP

Click **Configure** to modify the existing job. Scroll to the **Advanced Project Options** setting, but this time, select the **Pipeline script from SCM** option in the **Destination** dropdown. Paste the GitHub repo's URL in the **Repository URL**, and type **Jenkinsfile** in the **Script Path**. Save by clicking the **Save** button.


Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/bryantson/CICDPractice.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */master

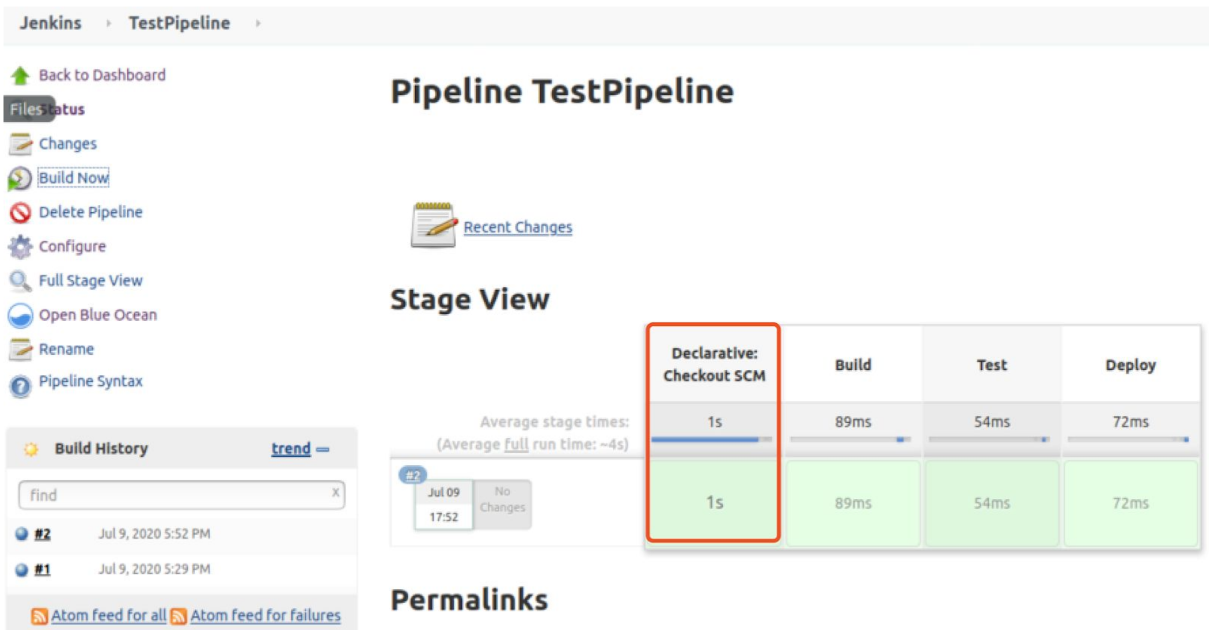
Repository browser: (Auto)

Additional Behaviours: Add

Script Path: Jenkinsfile

Lightweight checkout: ✓

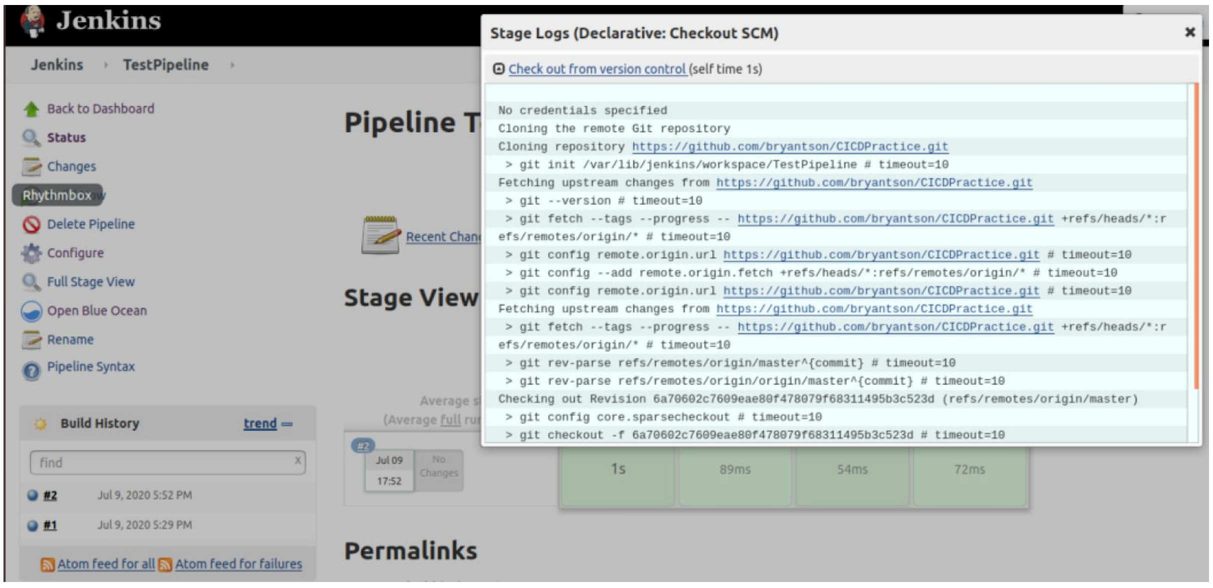
To build the pipeline, once you are back to the Task Overview page, click **Build Now** to execute the job again. The result will be the same as before, except you have one additional stage called **Declaration: Checkout SCM**.



	Declarative: Checkout SCM	Build	Test	Deploy
Average stage times: (Average full run time: ~4s)	1s	89ms	54ms	72ms
#2 Jul 09 17:52 No Changes	1s	89ms	54ms	72ms

Build	Date
#2	Jul 9, 2020 5:52 PM
#1	Jul 9, 2020 5:29 PM

To see the pipeline's output from the SCM build, click the Stage and view the **Log** to check how the source control cloning process went.

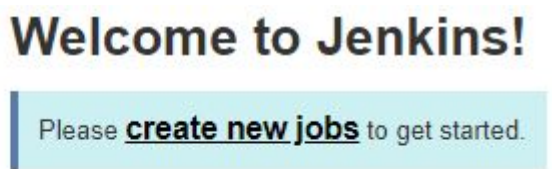


```
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/bryantson/CICDPractice.git
 > git init /var/lib/jenkins/workspace/TestPipeline # timeout=10
Fetching upstream changes from https://github.com/bryantson/CICDPractice.git
 > git --version # timeout=10
 > git fetch --tags --progress -- https://github.com/bryantson/CICDPractice.git +refs/heads/*:r
efs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/bryantson/CICDPractice.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/bryantson/CICDPractice.git # timeout=10
Fetching upstream changes from https://github.com/bryantson/CICDPractice.git
 > git fetch --tags --progress -- https://github.com/bryantson/CICDPractice.git +refs/heads/*:r
efs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 6a70602c7609eae80f478079f68311495b3c523d (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 6a70602c7609eae80f478079f68311495b3c523d # timeout=10
```

Congratulations! You've built your first Jenkins pipeline!

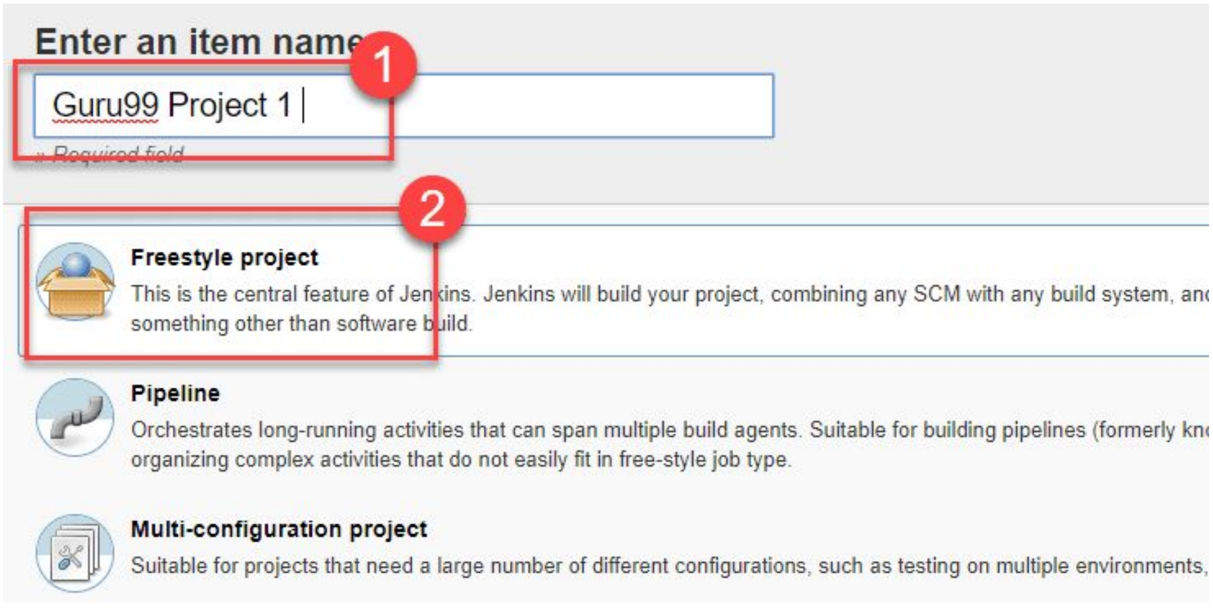
Another way to Integrating Jenkins with GitHub
The details can be found here: *https://www.guru99.com/jenkins-github-integration.html*

Step 1) Create a new job in Jenkins


Welcome to Jenkins!

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

Step 2) Enter the item name, select job type and click **OK**. We shall create a Freestyle project as an example.


Enter an item name

Guru99 Project 1

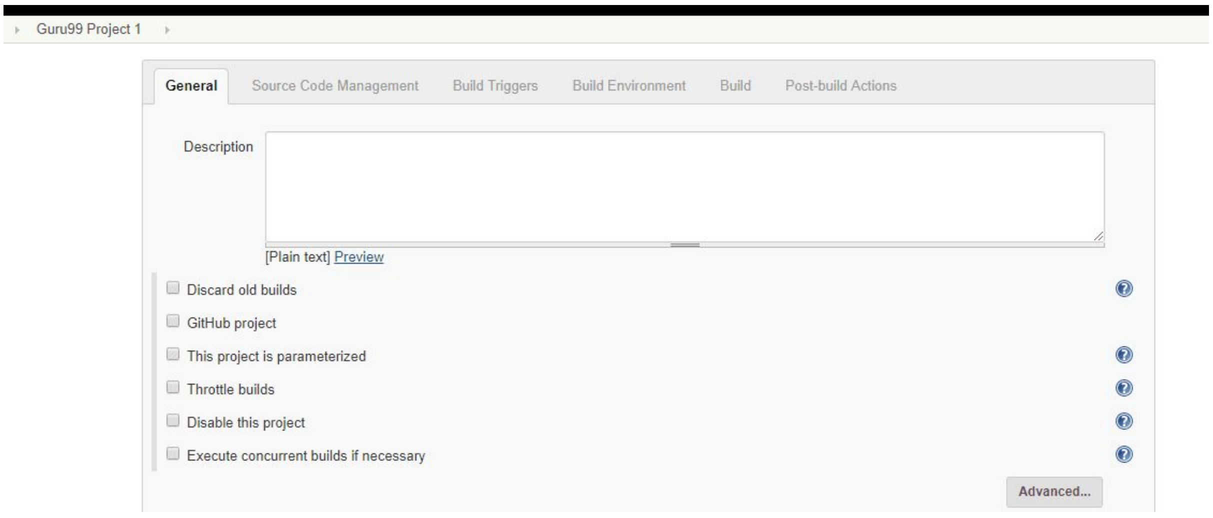
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, anc
something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly kno
organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments,

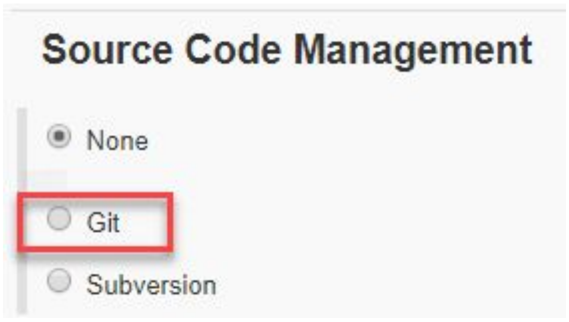
Step 3) Once you click **OK**, the page will be redirected to its project form. Here you will need to enter the project information:



Description

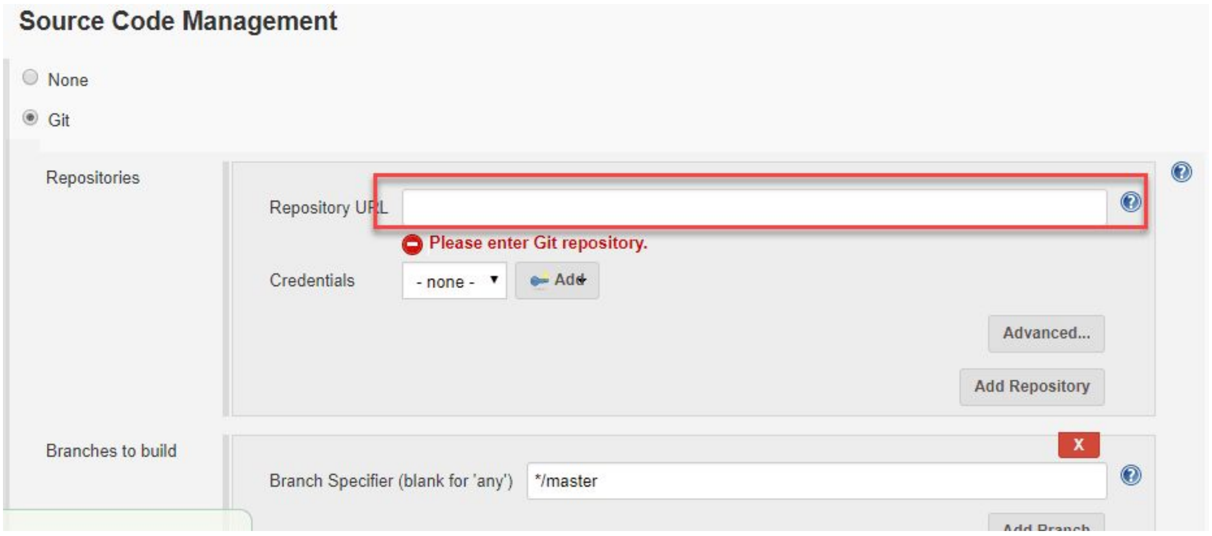
- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

Step 4) You will see a **Git** option under **Source Code Management** if your Git plugin has been installed in Jenkins:

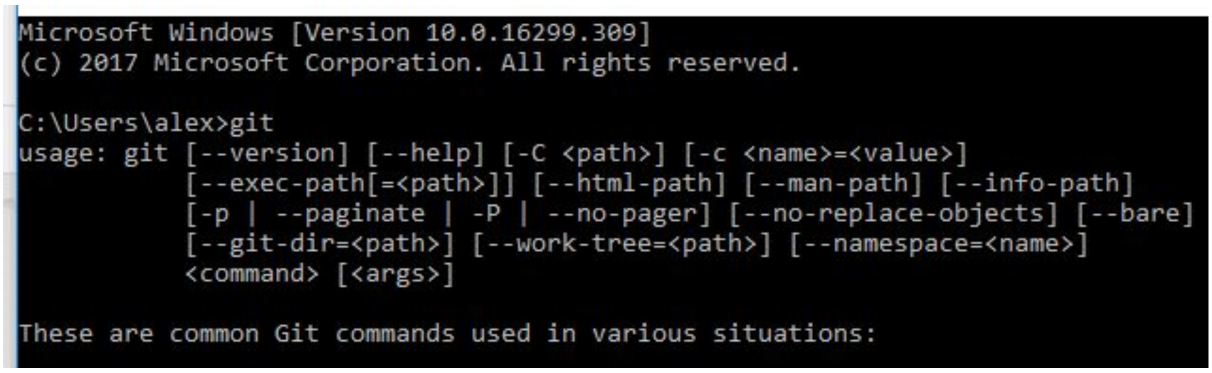


NOTE: If the Git option does not appear, try re-installing the plugins, followed by a restart and a re-login into your Jenkins dashboard. You will now be able to see the Git option as mentioned above.

Step 5) Enter the Git repository URL to pull the code from GitHub.



Step 6) You can execute Git repositories in your Jenkins once Git has been installed on your machine. To check if it has been successfully installed onto your system, open your **command prompt**, type "Git" and press enter. You should see different options come up for Git:



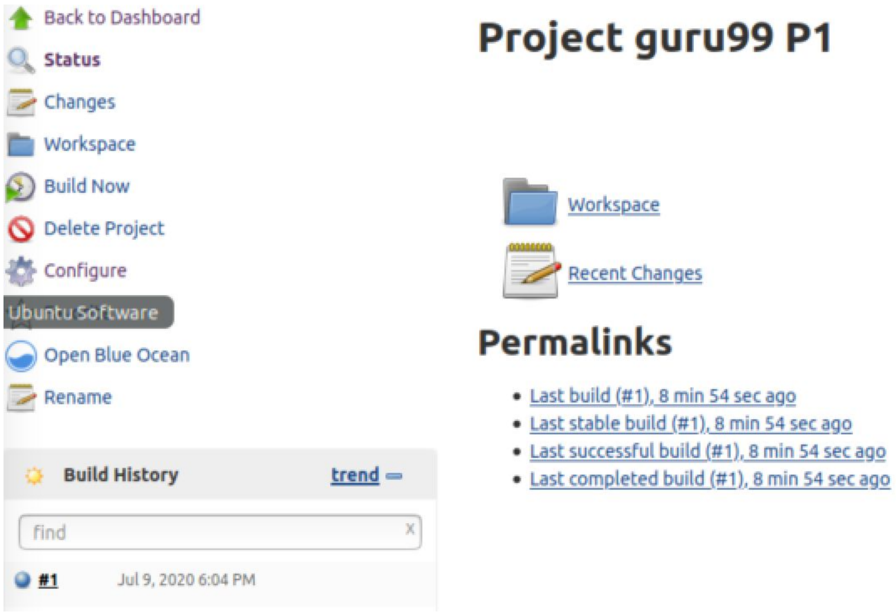
```
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\alex>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:
```

This means that Git has been installed in your system.

Step 7) Click **Build Now** to execute the job , and see the result in Blue Ocean.



Project guru99 P1

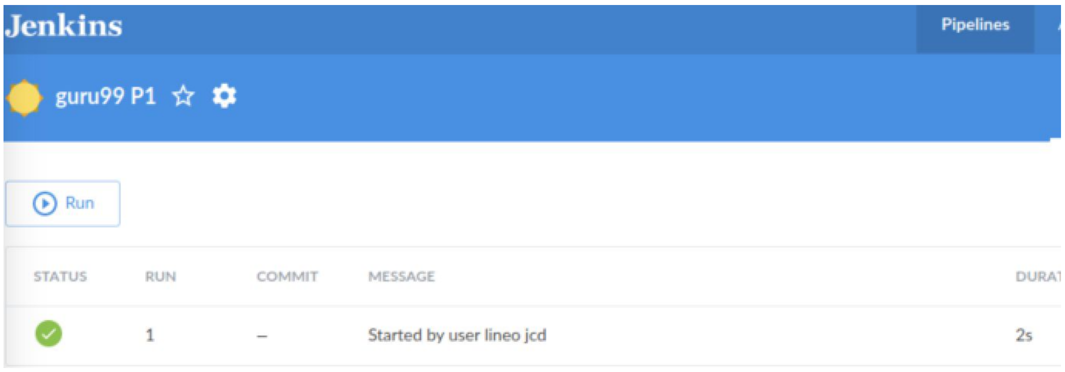
Workspace

Recent Changes

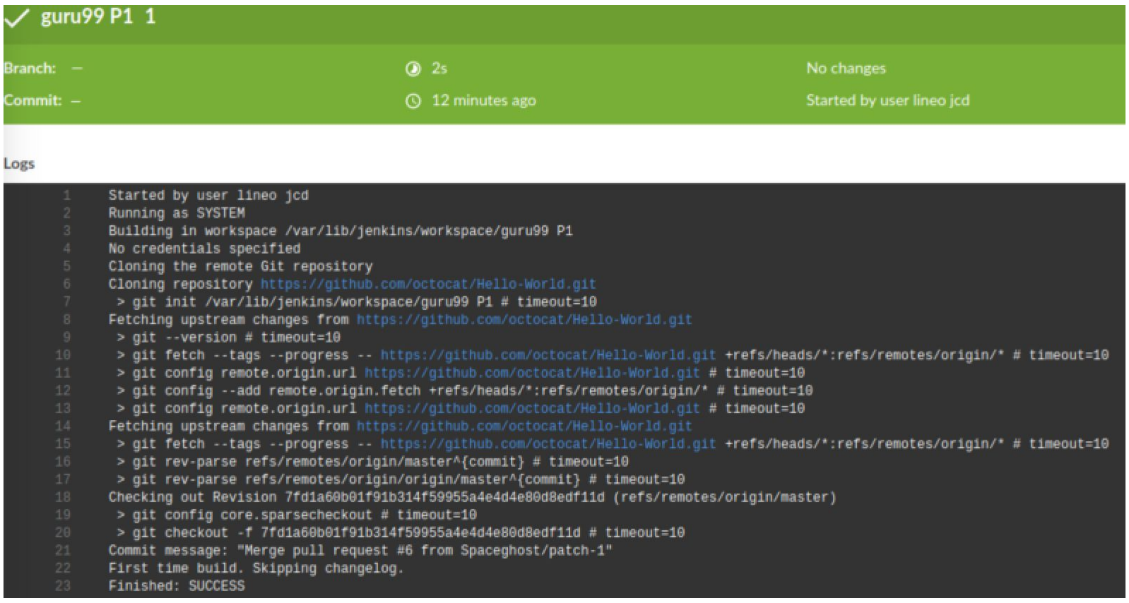
Permalinks

- Last build (#1), 8 min 54 sec ago
- Last stable build (#1), 8 min 54 sec ago
- Last successful build (#1), 8 min 54 sec ago
- Last completed build (#1), 8 min 54 sec ago

#1 Jul 9, 2020 6:04 PM



STATUS	RUN	COMMIT	MESSAGE	DURAT
✓	1	–	Started by user lineo jcd	2s



guru99 P1 1

Branch: –
Commit: –
2s
12 minutes ago
No changes
Started by user lineo jcd

Logs

```
Started by user lineo jcd
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/guru99 P1
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/octocat/Hello-World.git
 > git init /var/lib/jenkins/workspace/guru99 P1 # timeout=10
Fetching upstream changes from https://github.com/octocat/Hello-World.git
 > git --version # timeout=10
 > git fetch --tags --progress -- https://github.com/octocat/Hello-World.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/octocat/Hello-World.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/octocat/Hello-World.git # timeout=10
Fetching upstream changes from https://github.com/octocat/Hello-World.git
 > git fetch --tags --progress -- https://github.com/octocat/Hello-World.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 7fd1a60b01f91b314f59955a4e4d4e80d8edf11d (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 7fd1a60b01f91b314f59955a4e4d4e80d8edf11d # timeout=10
Commit message: "Merge pull request #6 from Spaceghost/patch-1"
First time build. Skipping changelog.
Finished: SUCCESS
```