Lab 3: Writing Pipeline as a Code

Author: Gourav Shah

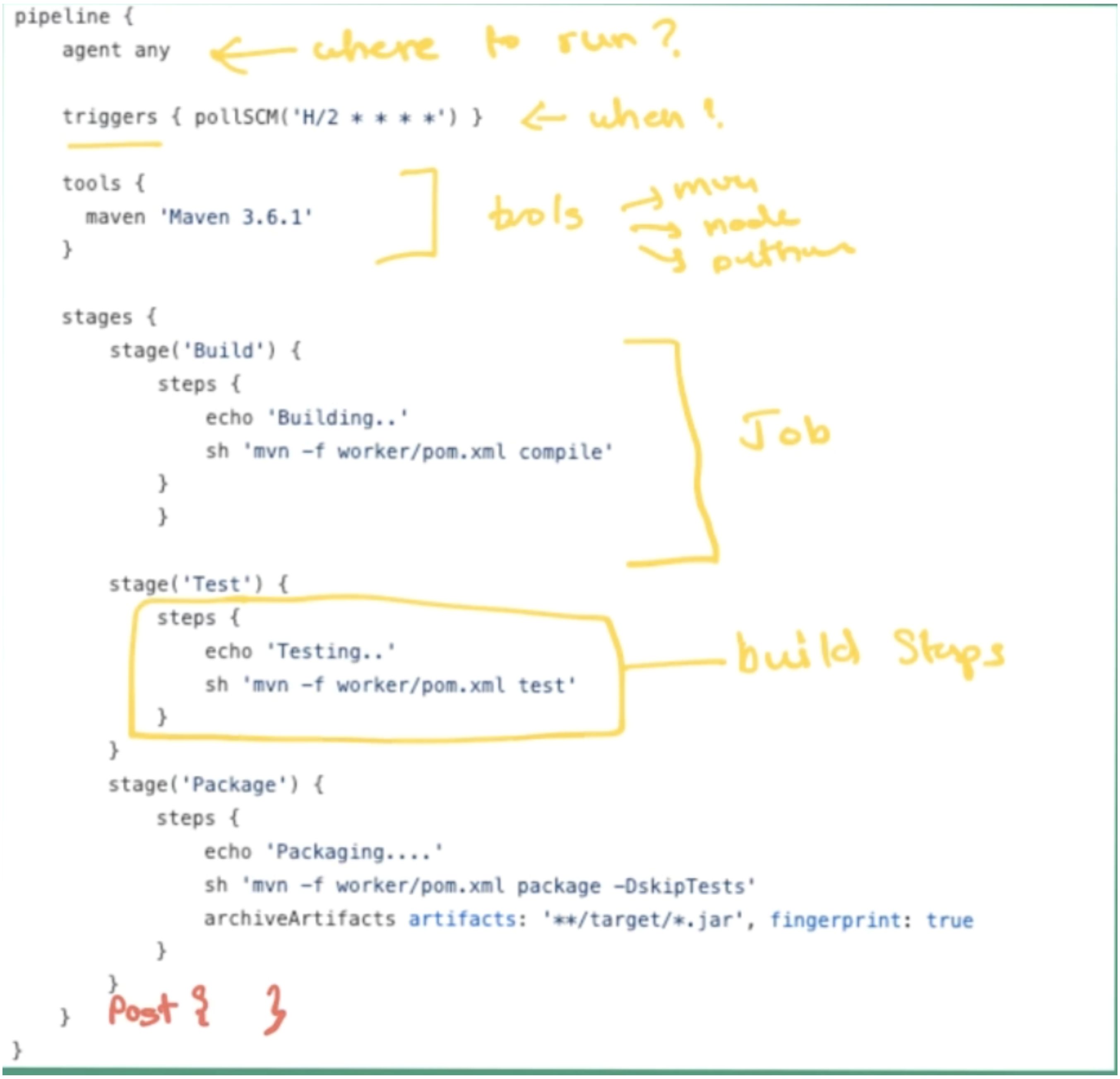
Publisher: School of Devops

Version : v2024.05.01.01

Reading List:

- Declarative Pipeline Syntax
- Declarative Pipeline Steps

Following is an example pipeline code. Examine it to learn how to create a Jenkinsfile and define the jobs as a code.



```
pipeline {
    agent any

    triggers { pollSCM('H/2 * * * *') }

    tools {
      maven 'Maven 3.6.1'
    }

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
                sh 'mvn -f worker/pom.xml compile'
            }
            }

        stage('Test') {
            steps {
                echo 'Testing..'
                sh 'mvn -f worker/pom.xml test'
            }
        }
        stage('Package') {
            steps {
                echo 'Packaging....'
                sh 'mvn -f worker/pom.xml package -DskipTests'
                archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
            }
        }
    }
}
```

Some of the important directives are,

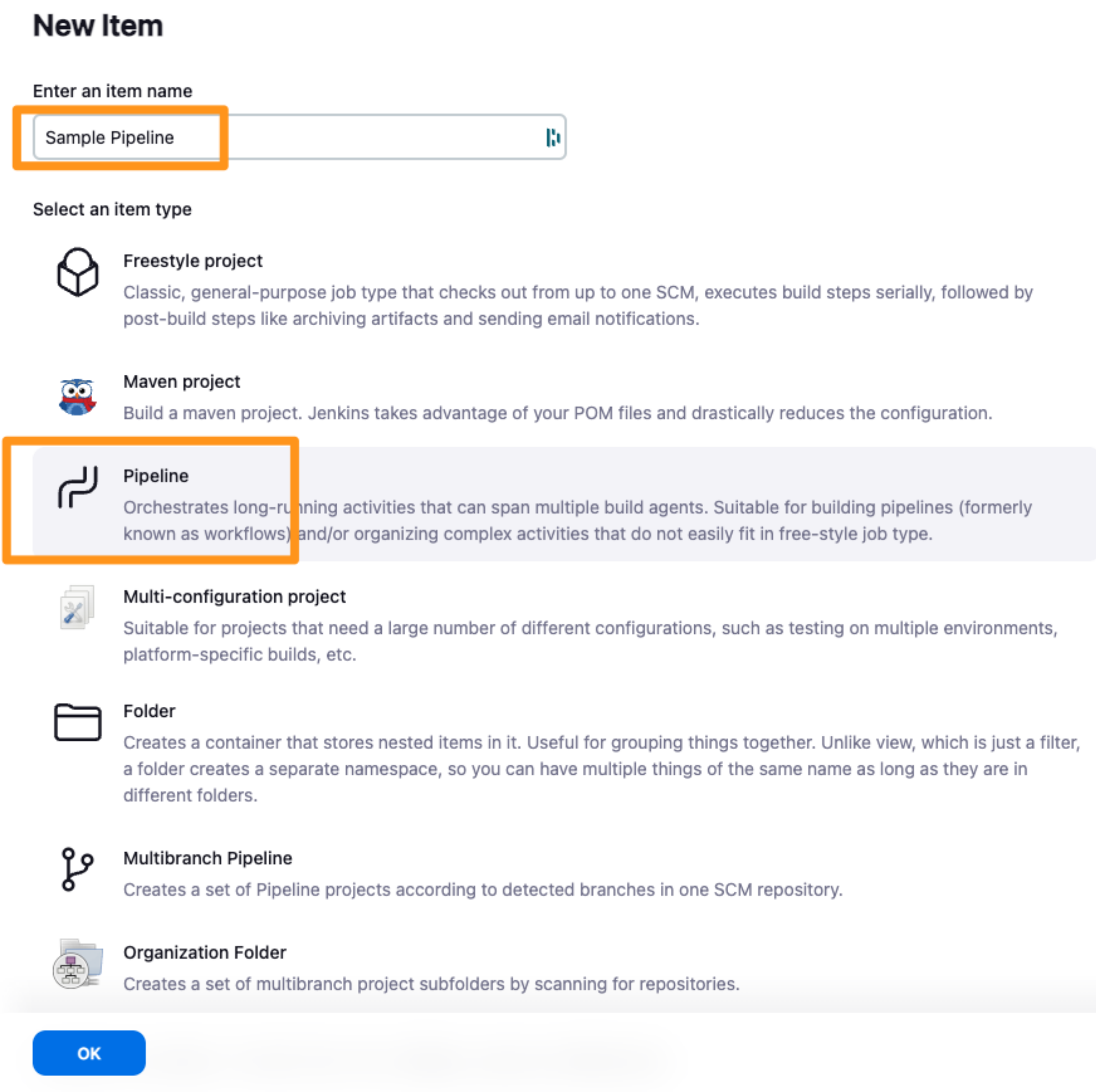
- Pipeline
- Agent
- Tools
- Stages | Stage | Steps
- Post

You could remember those as **PAT3SP**.

Creating a sample declarative pipeline

In this sub section you will learn how to create and execute a declarative pipeline.

Begin by creating a sample pipeline job. To create it, go to jenkins top page and select new item. Provide name as `Sample Pipeline` and select `pipeline` as the item type.



New Item

Enter an item name

Sample Pipeline

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

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.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

goto job configuration page, Now you are going to write declarative pipeline, visit declarative pipeline for your reference.

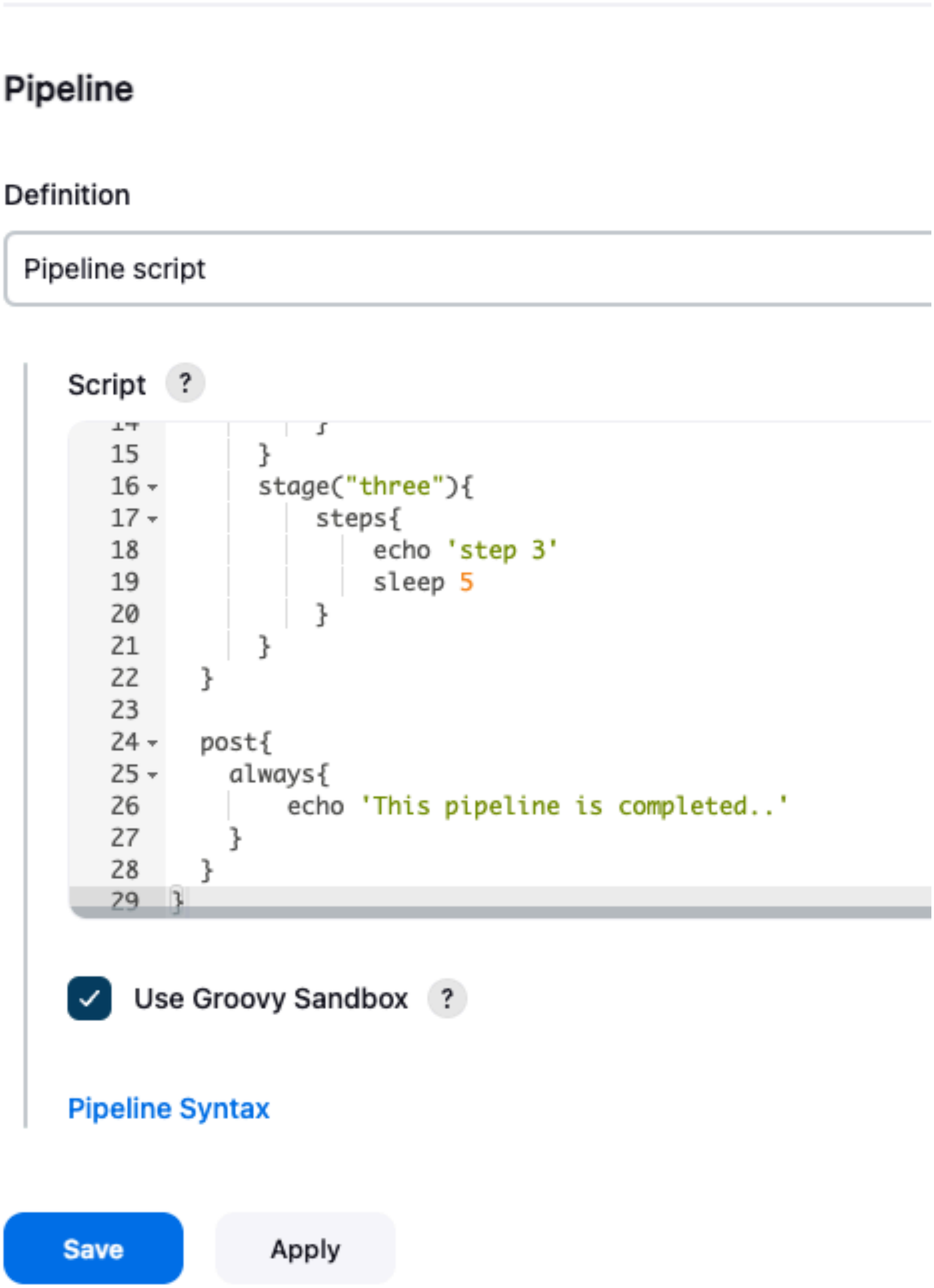
Paste the following code in the **Pipeline Script** section to configure your pipeline job.

```
pipeline {
  agent  stages{
      stage("one"){
          steps{
              echo 'step 1'
              sleep 3
          }
      }
      stage("two"){
          steps{
              echo 'step 2'
              sleep 9
          }
      }
      stage("three"){
          steps{
              echo 'step 3'
              sleep 5
          }
      }
  }

  post{
    always{
        echo 'This pipeline is completed..'
    }
  }
}
```

Source: Sample Jenkins Pipeline · GitHub

e.g.



Pipeline

Definition

Pipeline script

Script ?

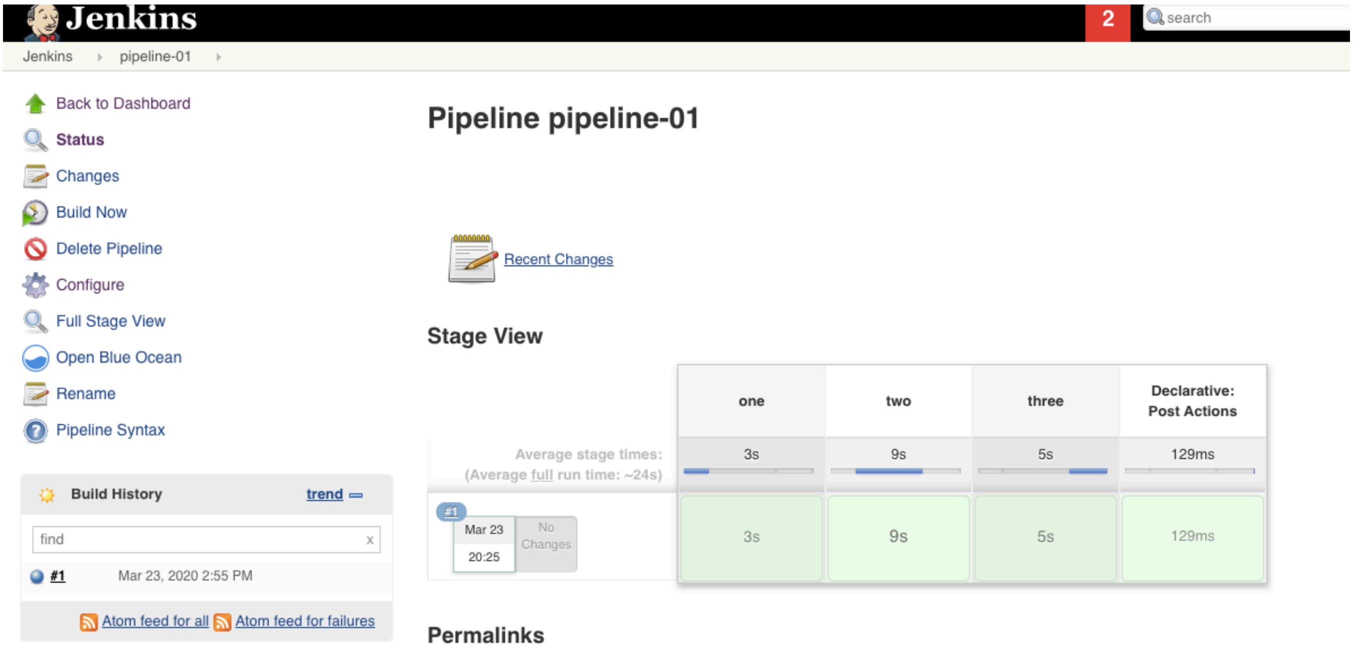
```
          }
      }
      stage("three"){
          steps{
              echo 'step 3'
              sleep 5
          }
      }
  }

  post{
    always{
        echo 'This pipeline is completed..'
    }
  }
}
```

☑ Use Groovy Sandbox ?

Pipeline Syntax

After configuring the job, build it and view the stages to know how the pipeline works and how much time it will take to complete the job.



Pipeline pipeline-01

Stage View

	one	two	three	Declarative: Post Actions
Average stage times: (Average full run time: ~24s)	3s	9s	5s	129ms
#1 Mar 23 20:25 No Changes	3s	9s	5s	129ms

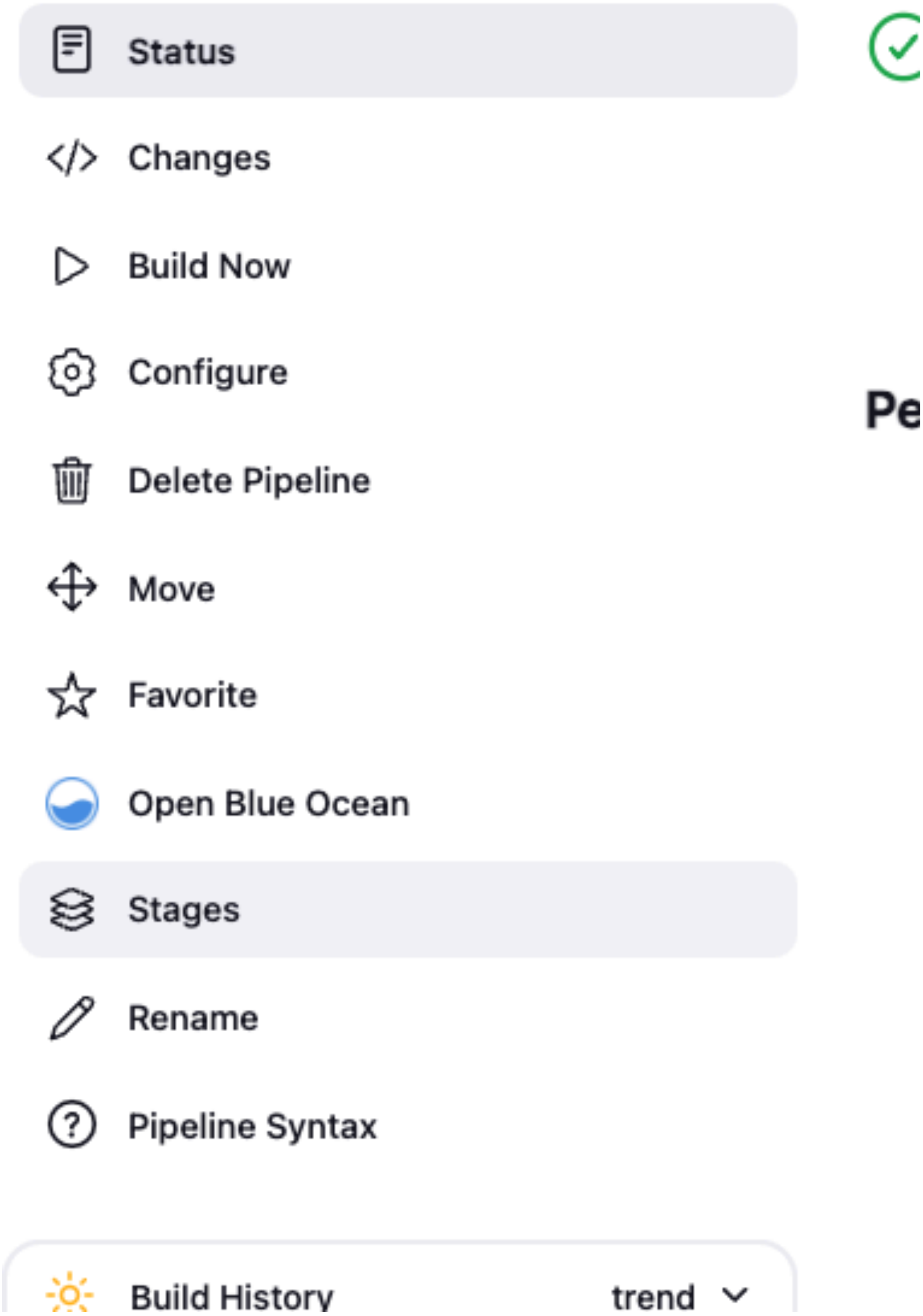
Build History trend

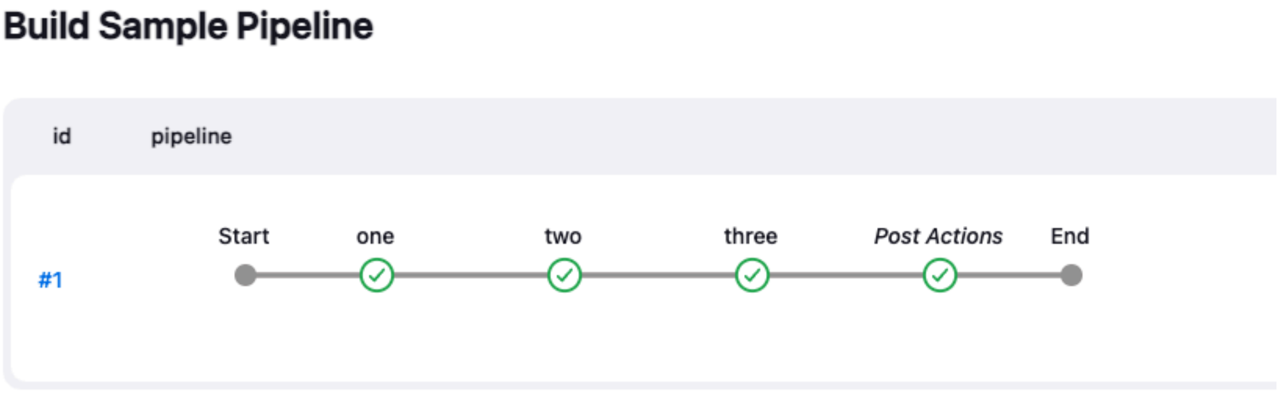
find x

#1 Mar 23, 2020 2:55 PM

Atom feed for all Atom feed for failures

You could also explore the Stages tag on the left to dive into each stage


Status
Changes
Build Now
Configure
Delete Pipeline
Move
Favorite
Open Blue Ocean
Stages
Rename
Pipeline Syntax
Build History
trend
Pe



Build Sample Pipeline

you could see the time interval between each steps in stage view and select to see a specific stage logs.

Writing a Jenkinsfile for the Maven App

As part of the earlier lab, you may have already created a fork of the sysfoo repository GitHub - udbc/sysfoo: Sample java webapp with maven which prints system info. Switch to your own fork and clone it to your workspace.

warning: make sure to replace the repository with your own fork

```
git clone https://github.com/xxxx/sysfoo.git
cd sysfoo
```

Now, start writing the Declarative Pipeline code for the maven app as follows,

File: sysfoo/Jenkisfile

```
pipeline{

   agent any

   tools{
       maven 'Maven 3.9.6'
    }

    stages{
        stage('build'){
            steps{
                echo 'compile maven app'
                sh 'mvn compile'
            }
        }
        stage('test'){
            steps{
                echo 'test maven app'
                sh 'mvn clean test'
            }
        }
        stage('package'){
            steps{
                echo 'package maven app'
                sh 'mvn package -DskipTests'
            }
        }
    }

}
```

Once written, commit in the Jenkinsfile and push the changes to your repository.

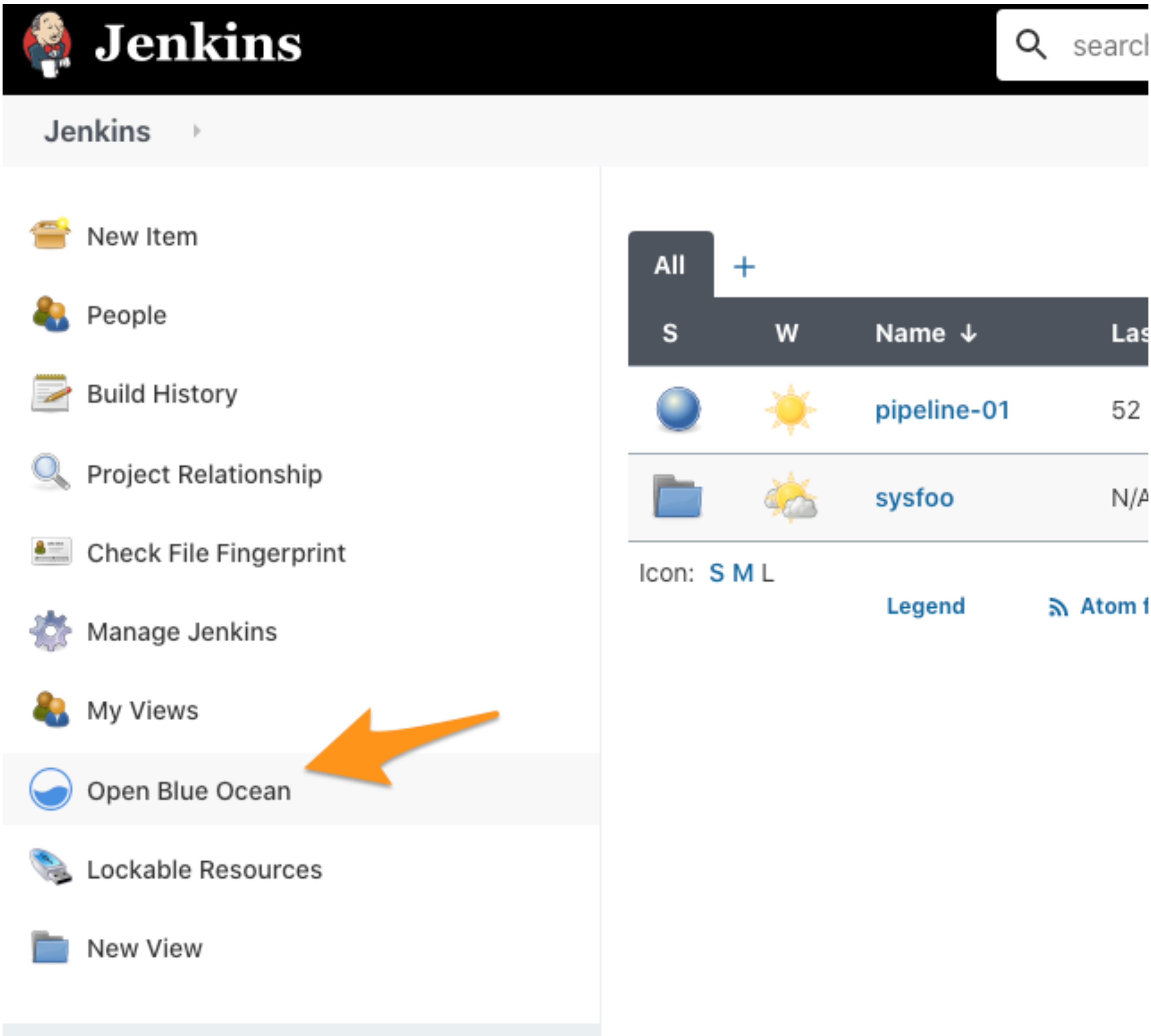
```
git add Jenkinsfile
git commit -am "adding Jenkinsfile for sysfoo"
git push origin master
```

Creating a Pipeline using Blue Ocean

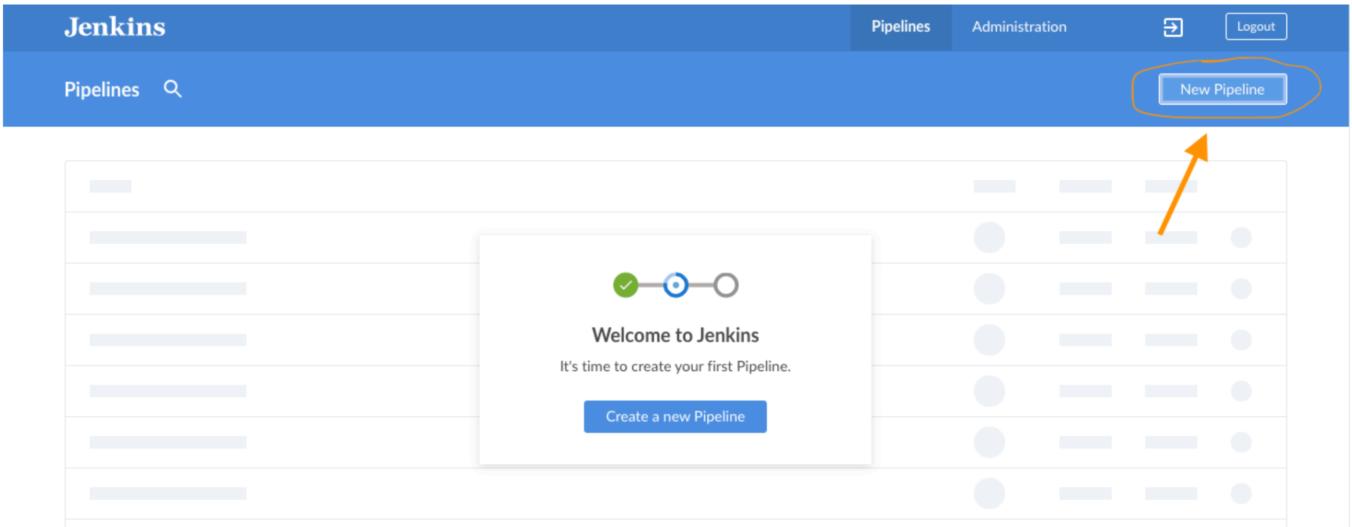
Note: Make Sure to use Blue Ocean UI to create Pipeline. Else you will not be able to edit it later from the UI.

First broke to the Jenkins top page and then head over the Jenkins console and select `Open Blue`

`Ocean` from the main page.



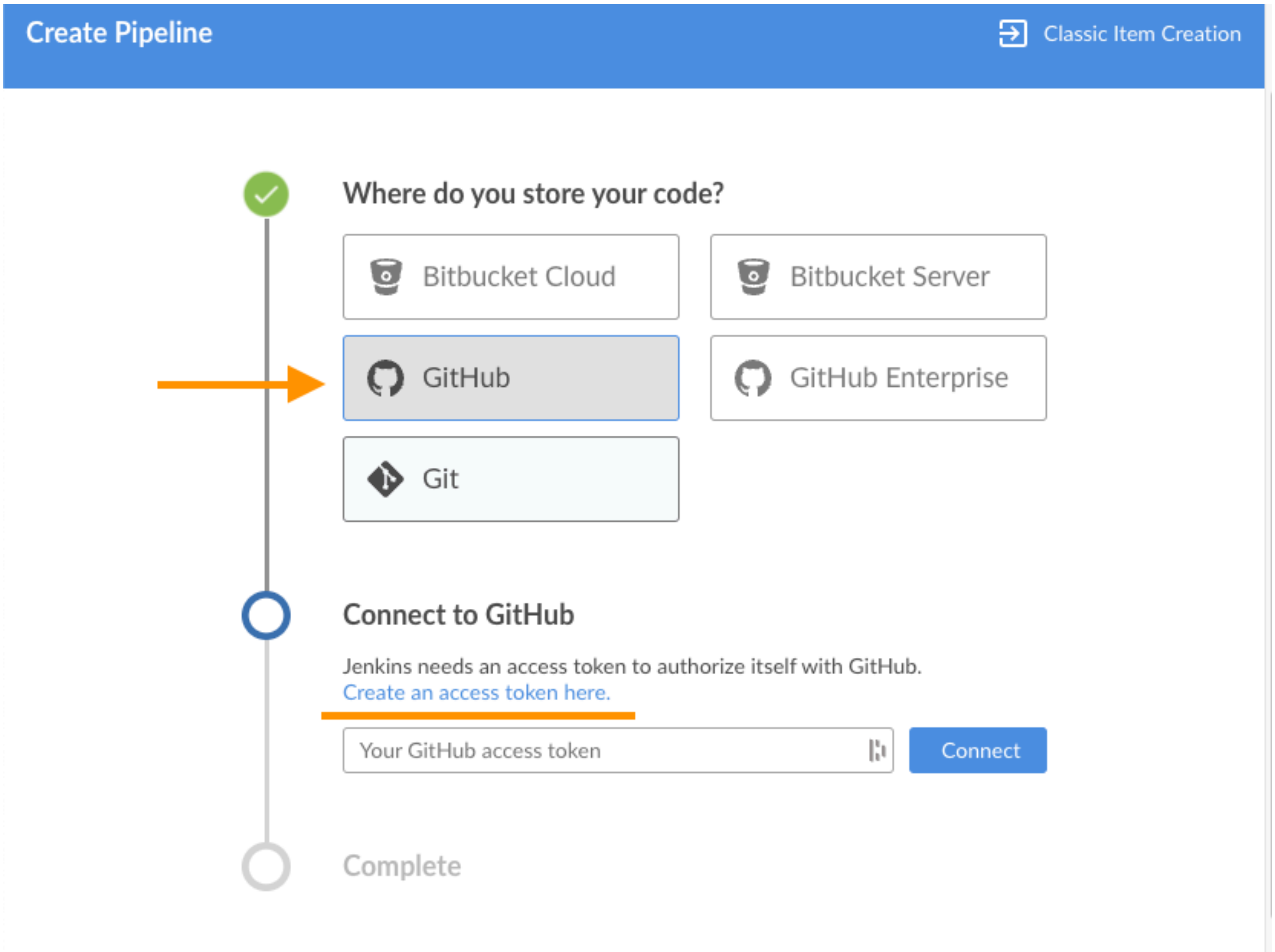
Once the new UI opens, click on `New Pipeline` from the top right of the page.



Welcome to Jenkins

It's time to create your first Pipeline.

Select your Source Code Management system. In this case, it would be GitHub



Where do you store your code?

Connect to GitHub

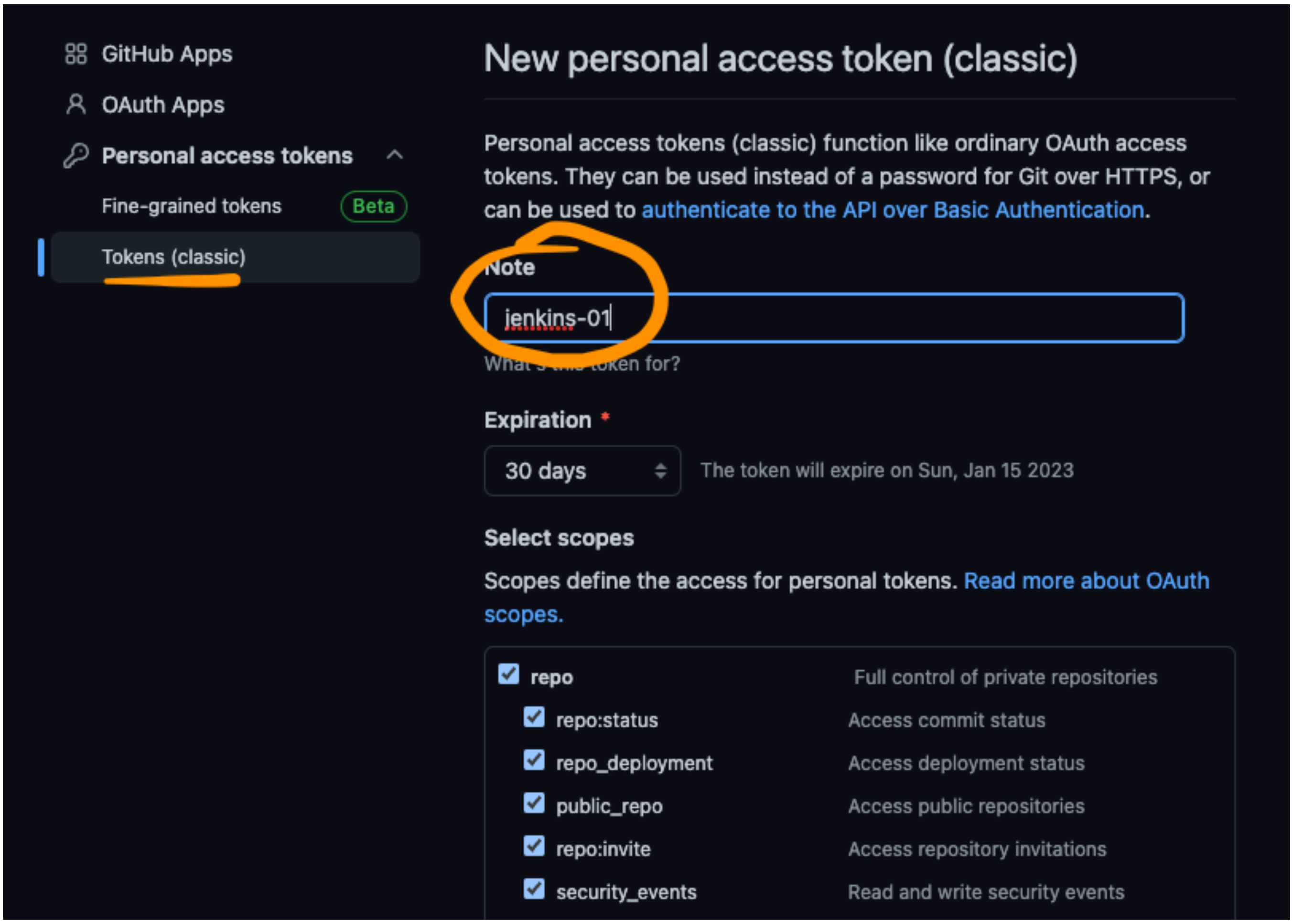
Jenkins needs an access token to authorize itself with GitHub. Create an access token here.

You would need to connect Jenekins/Blue Ocean with GitHub. The way to do that is to generate an access token.

If you are already logged into GitHub from the browser, you could simply click on the link which reads

“Create an access token here”. It will have you verify your credentials and take you directly to the token generation page.

You could provide a name for the token



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

jenkins-01

What's this token for?

Expiration *

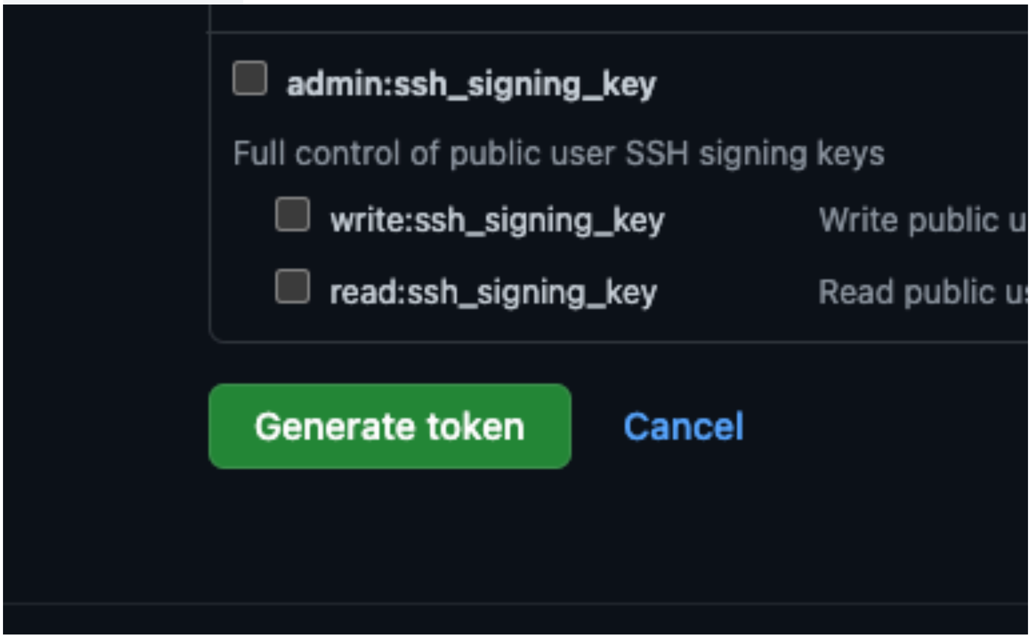
30 days — The token will expire on Sun, Jan 15 2023

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

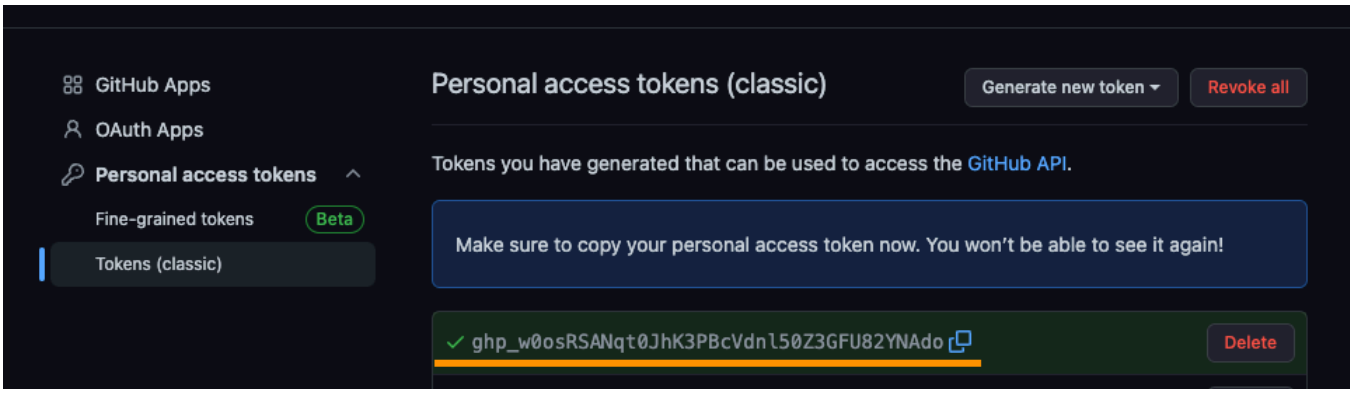
- ☑ repo — Full control of private repositories
  - ☑ repo:status — Access commit status
  - ☑ repo_deployment — Access deployment status
  - ☑ public_repo — Access public repositories
  - ☑ repo:invite — Access repository invitations
  - ☑ security_events — Read and write security events

Ensure that `Tokens(Classic)` option is selected, and provide a name. Scroll down to click on `Generate Token`



- □ admin:ssh_signing_key — Full control of public user SSH signing keys
  - □ write:ssh_signing_key — Write public u
  - □ read:ssh_signing_key — Read public us

Once generated, you get only once chance to copy it over. Keep this page open until configure this token on Jenkins.



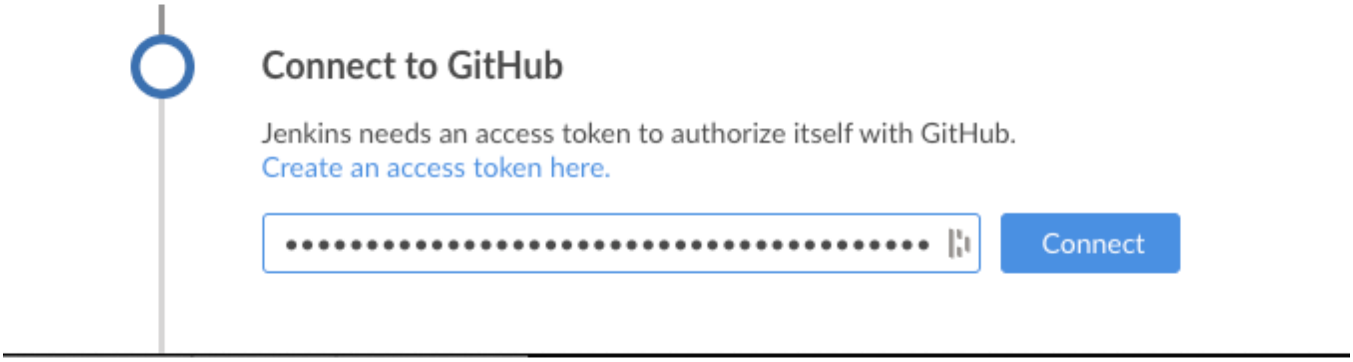
Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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

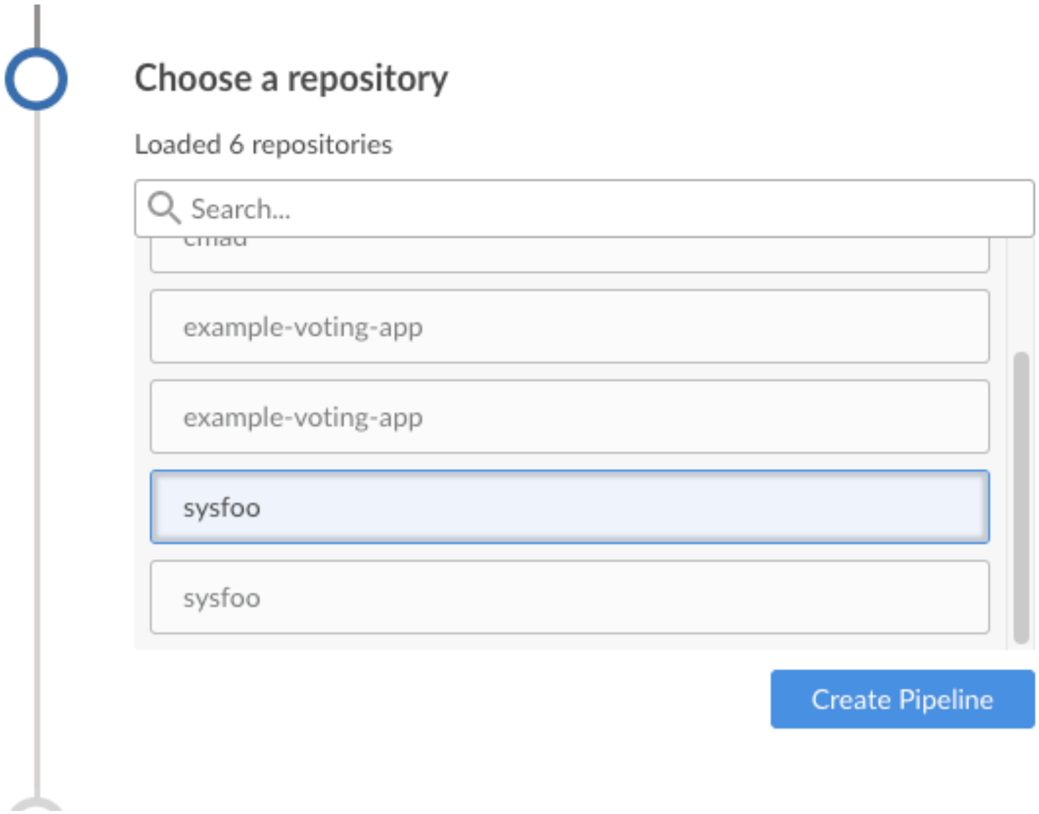
Copy the token, go back to Blue Ocean UI and paste it there. Click on Connect to proceed.



Connect to GitHub

Jenkins needs an access token to authorize itself with GitHub.
Create an access token here.

Once connected, it would scan your account. Select the organisation/user and the relevant repository.

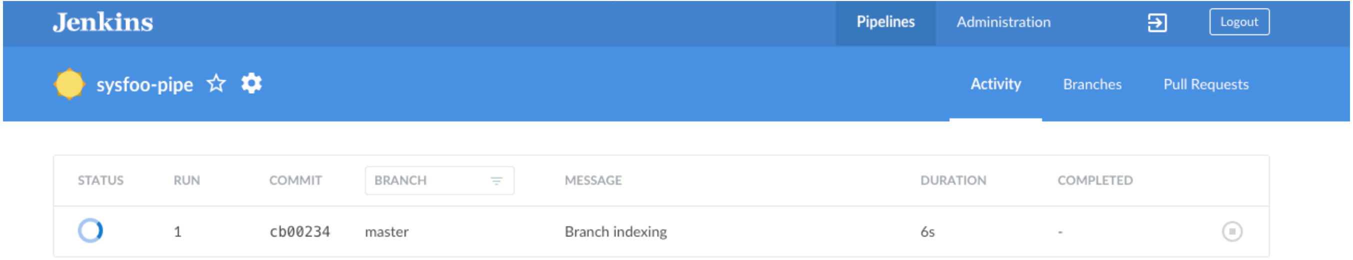


Choose a repository

Loaded 6 repositories

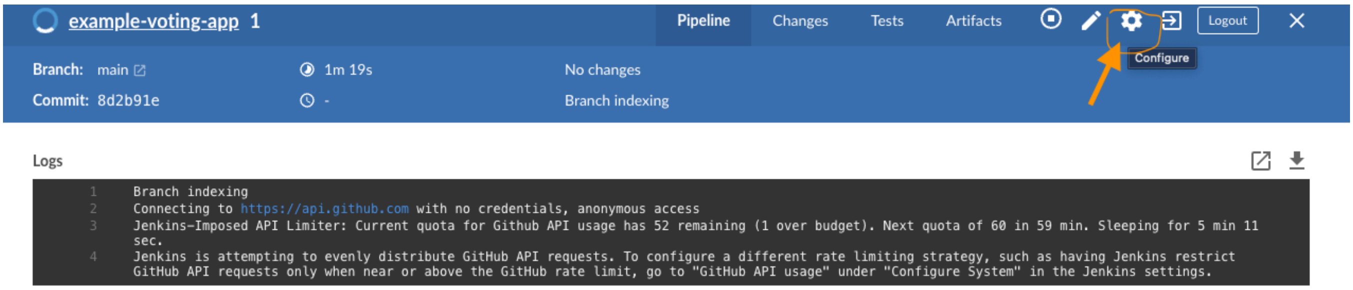
Proceed to `Create Pipeline`.

It will create a pipeline and take you to the page such as follows.



STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
○	1	cb00234	master	Branch indexing	6s	-

If you click on the `main/master` branch, and see the following message, you need to provide GitHub credentials to pull the code.



Branch: main

Commit: 8d2b91e

1m 19s

-

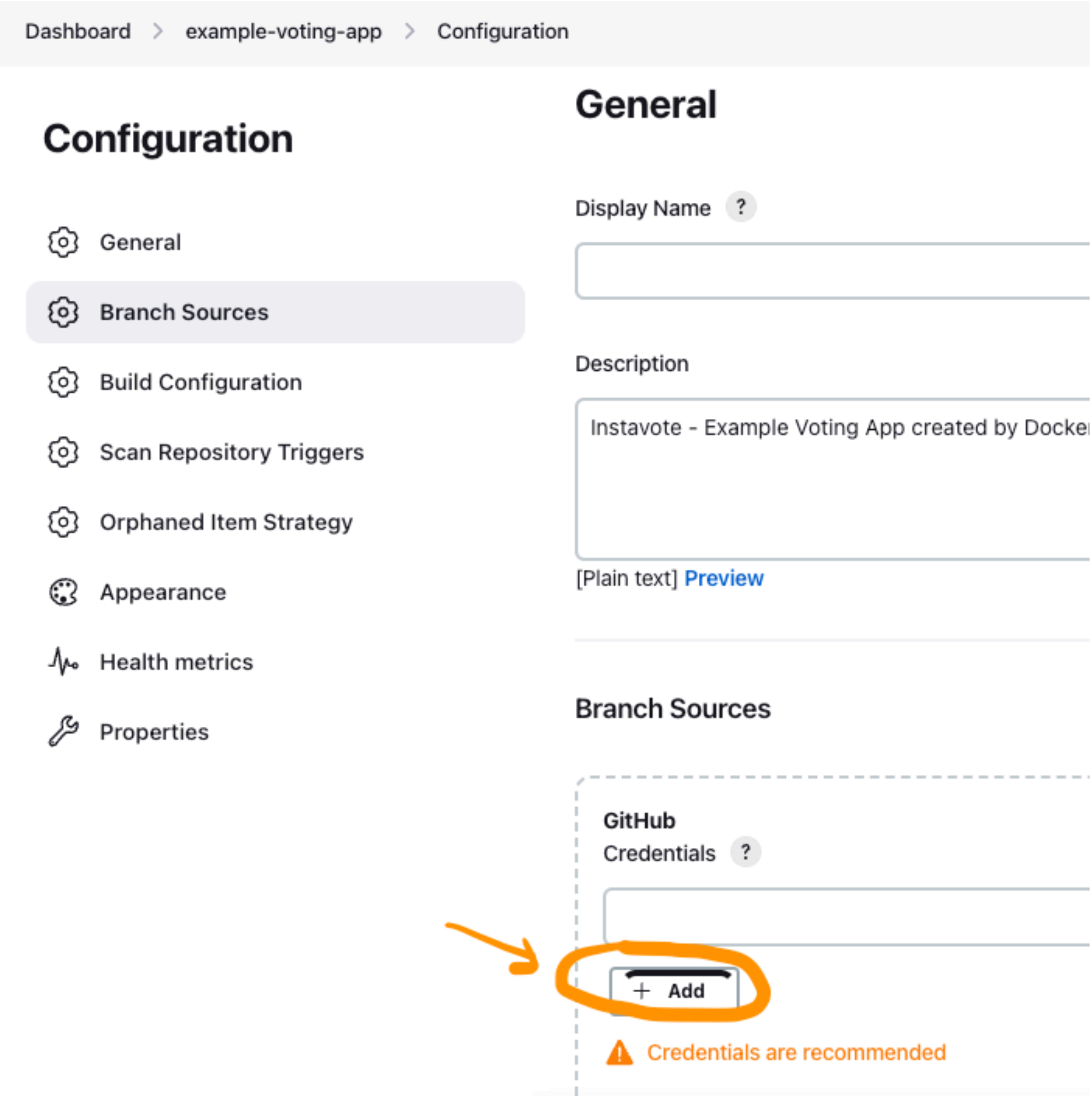
No changes

Branch indexing

Logs

```
Branch indexing
Connecting to https://api.github.com with no credentials, anonymous access
Jenkins-Imposed API Limiter: Current quota for Github API usage has 52 remaining (1 over budget). Next quota of 60 in 59 min. Sleeping for 5 min 11
sec.
Jenkins is attempting to evenly distribute GitHub API requests. To configure a different rate limiting strategy, such as having Jenkins restrict
GitHub API requests only when near or above the GitHub rate limit, go to "GitHub API usage" under "Configure System" in the Jenkins settings.
```

To do so, click on the `Gear` icon from the top right menu. This will open the job configuration page as follows,



Dashboard > example-voting-app > Configuration

Configuration

- General
- Branch Sources
- Build Configuration
- Scan Repository Triggers
- Orphaned Item Strategy
- Appearance
- Health metrics
- Properties

General

Display Name ?

Description

Instavote - Example Voting App created by Docke

[Plain text] Preview

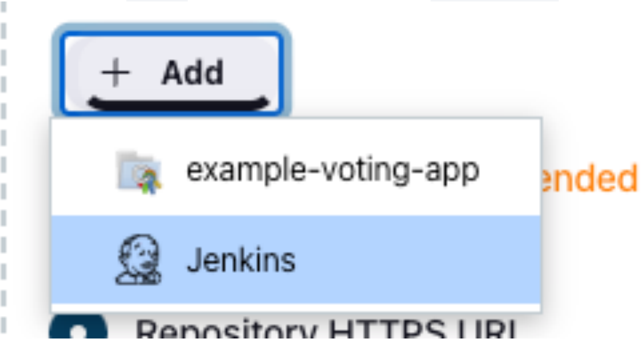
Branch Sources

GitHub

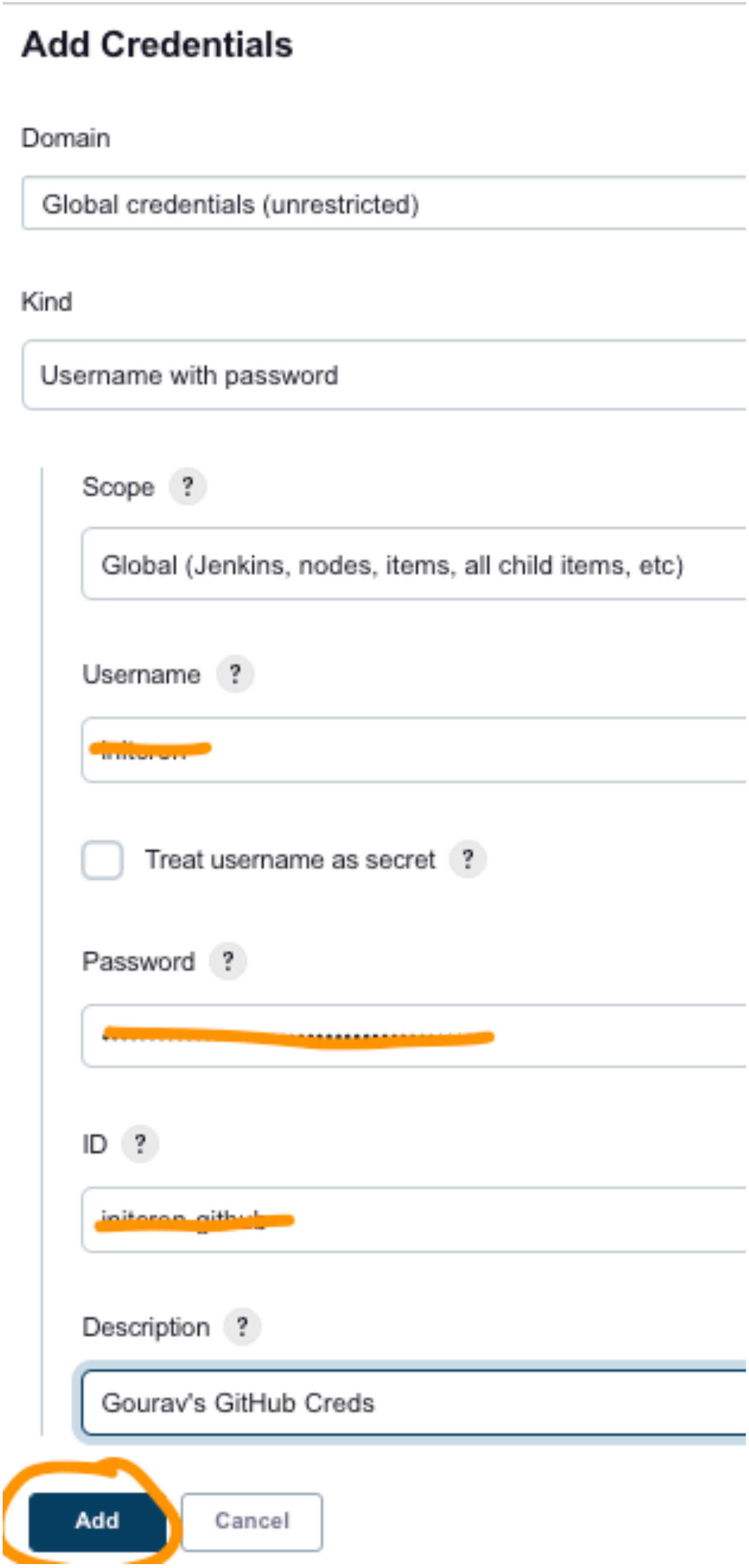
Credentials ?

⚠ Credentials are recommended

Click on the `Add` button and select `Jenkins` as the credentials provider.



Using the `Add Credentials` pop up form provide your GitHub credentials. You could use the Token instead of the password here (Same token created earlier, or create a new one with repository access).



Add Credentials

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

[redacted]

□ Treat username as secret ?

Password ?

[redacted]

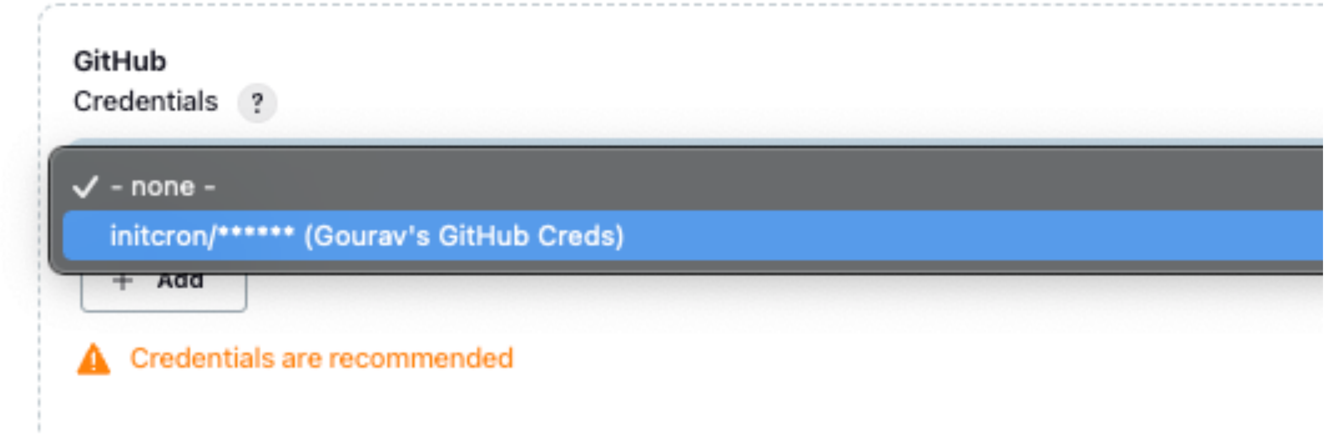
ID ?

[redacted]

Description ?

Gourav's GitHub Creds

Once added, select those credentials using the drop down as follows,



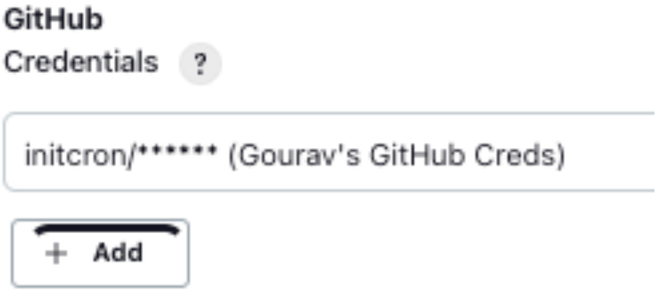
GitHub

Credentials ?

✓ - none -

initcron/****** (Gourav's GitHub Creds)

⚠ Credentials are recommended

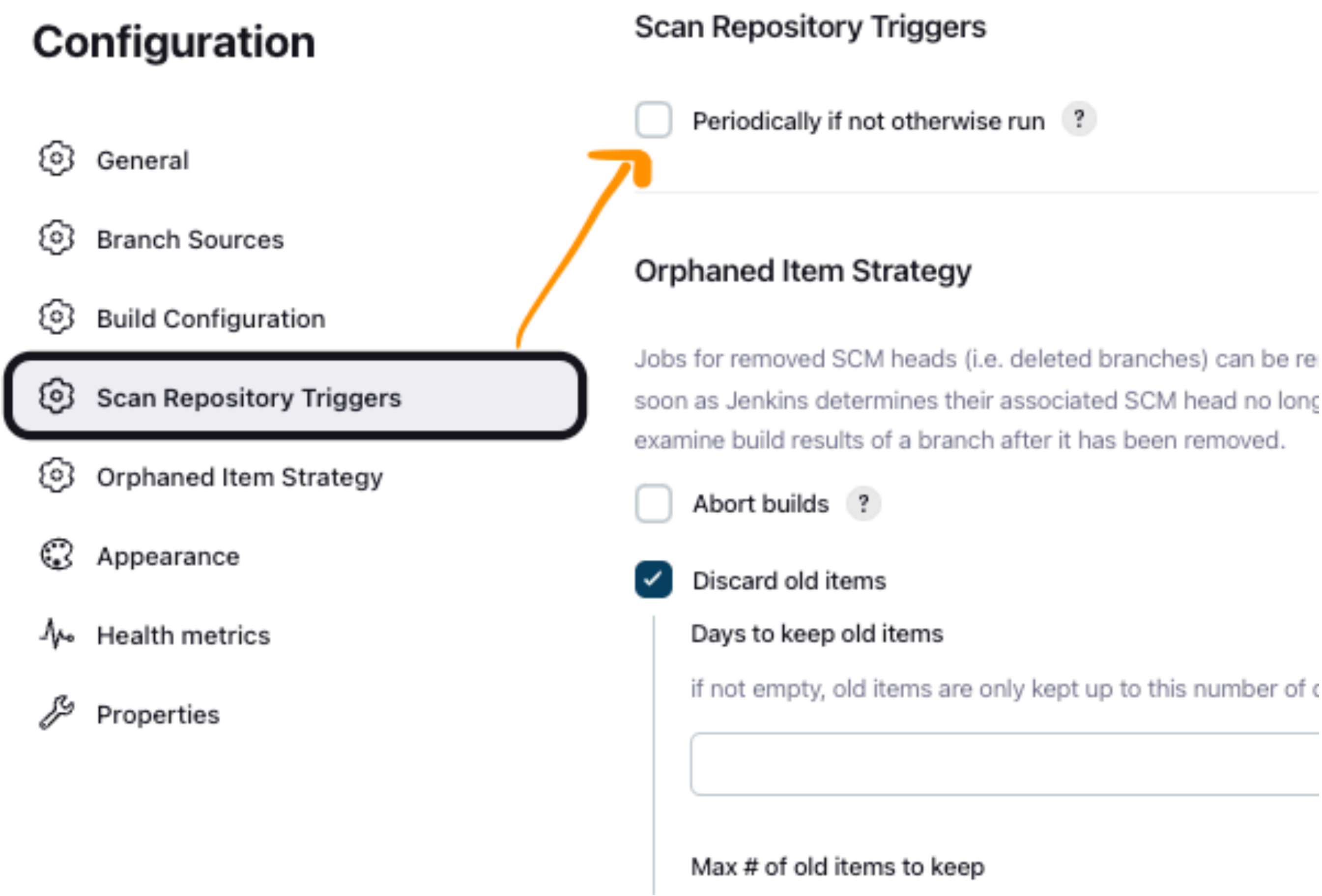


GitHub

Credentials ?

initcron/****** (Gourav's GitHub Creds)

Also, scroll down to `Scan Repository Triggers` and select the check box , which will open a drop down.



Scan Repository Triggers

□ Periodically if not otherwise run ?

Orphaned Item Strategy

Jobs for removed SCM heads (i.e. deleted branches) can be re
soon as Jenkins determines their associated SCM head no long
examine build results of a branch after it has been removed.

□ Abort builds ?

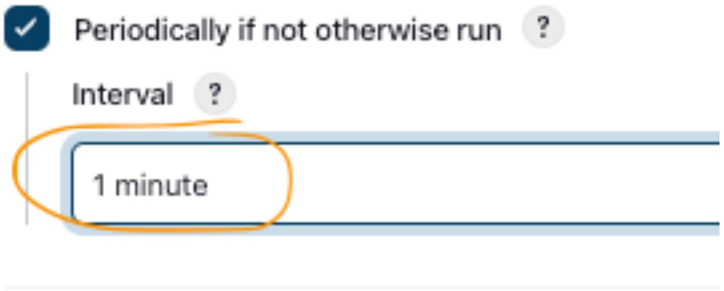
☑ Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of

Max # of old items to keep

Choose an interval of `1 minute` from the drop down.

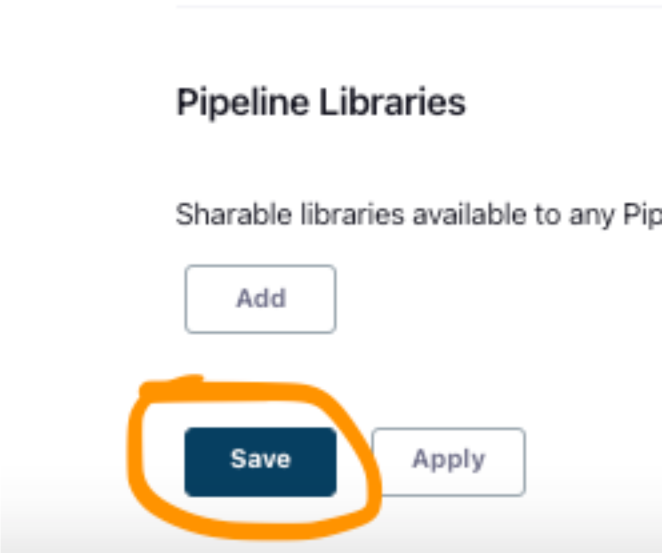


☑ Periodically if not otherwise run ?

Interval ?

1 minute

Once done, scroll down to save the pipeline job.

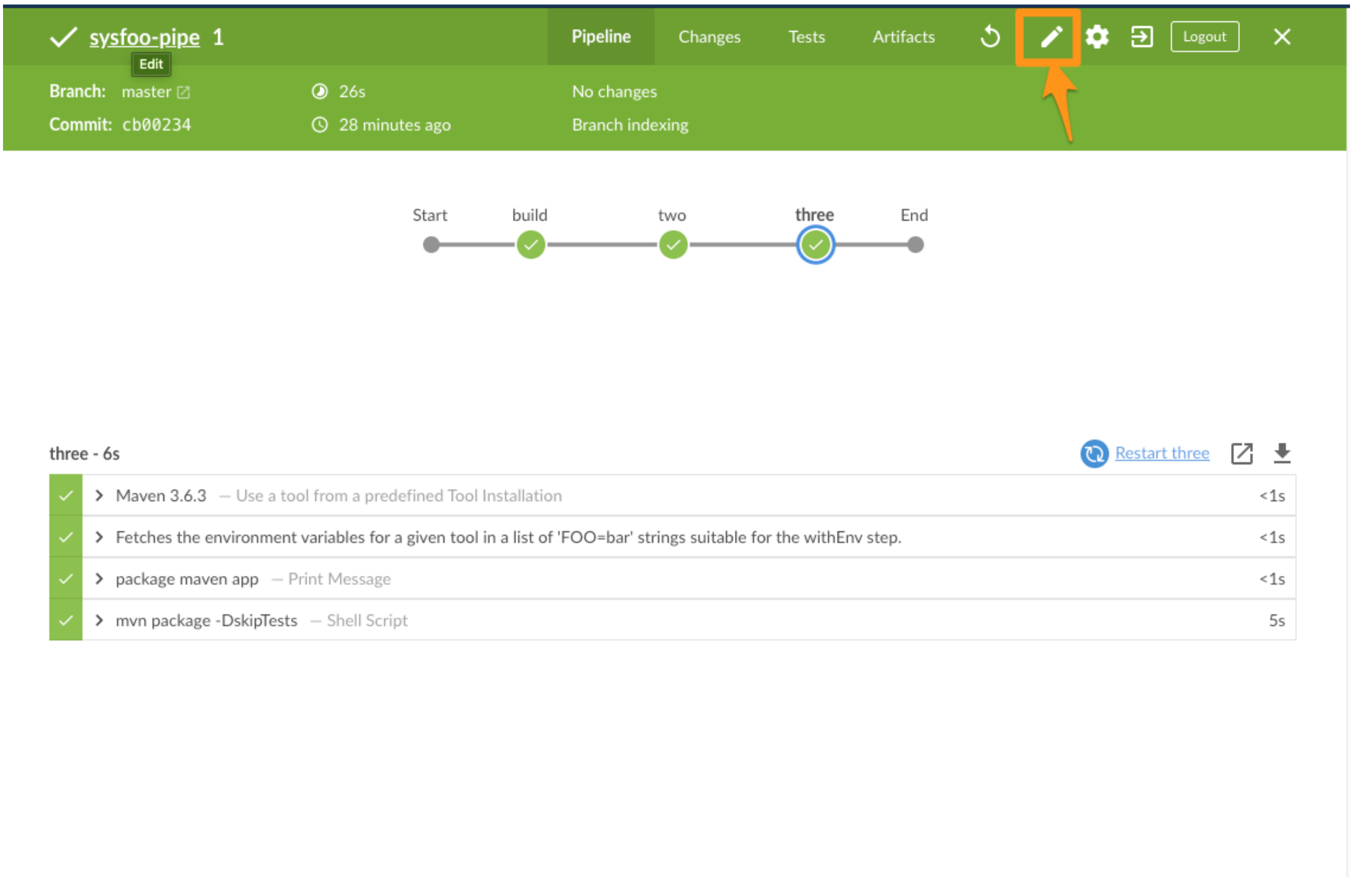


Now using blue ocean UI, go back to your pipeline job, where you could examine it progressing through the pipeline stages e.g.

Modifying Pipeline Code with Blue Ocean

The easiest and the most intuitive way to update pipeline code is to use the graphical interface that Blue Ocean provides.

Lets update the existing pipeline job to add artefact archival step. Head over to the Blue Ocean configuration page for the pipeline and click on the edit button represented by a pencil icon.



Branch: master

Commit: cb00234

26s

28 minutes ago

No changes

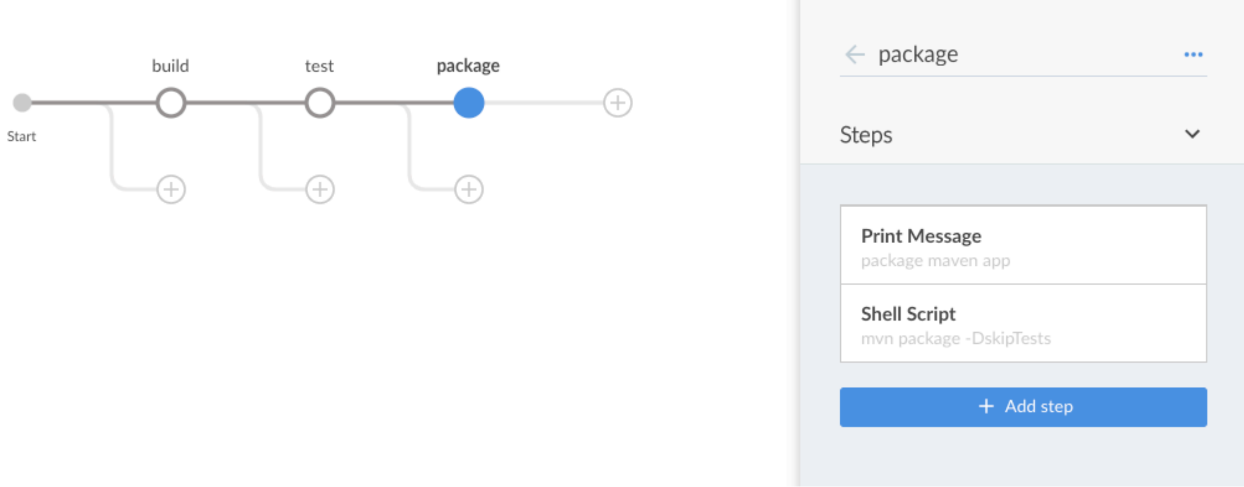
Branch indexing

three - 6s

✓	› Maven 3.6.3 — Use a tool from a predefined Tool Installation	<1s
✓	› Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	› package maven app — Print Message	<1s
✓	› mvn package -DskipTests — Shell Script	5s

Warning: You will see the edit option with pencil icon only if you have created the pipeline with Blue Ocean. If you had created it from classic Jenkins UI, you would not be able to proceed, so go back and create it again by following the steps in this guide.

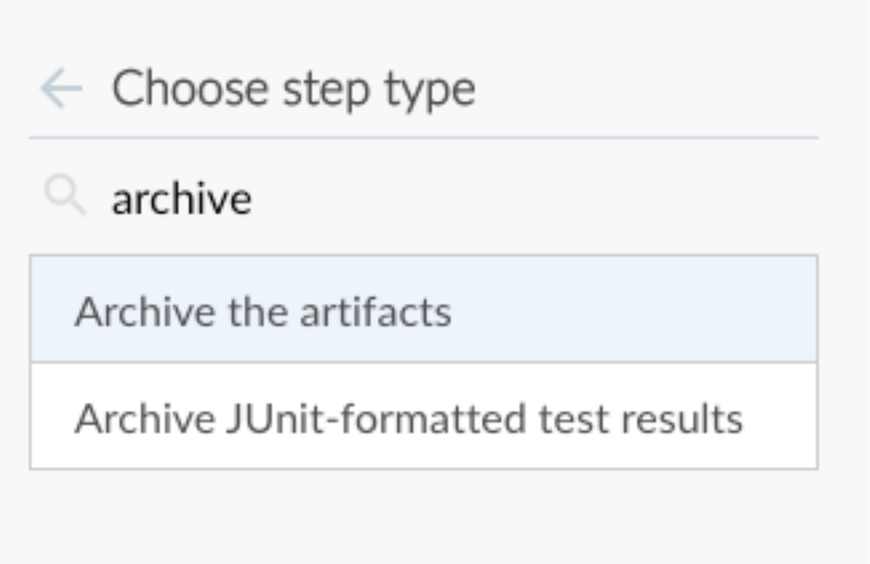
Select **package** stage and from the configuration on the right, click on **Add Step**



Print Message
package maven app

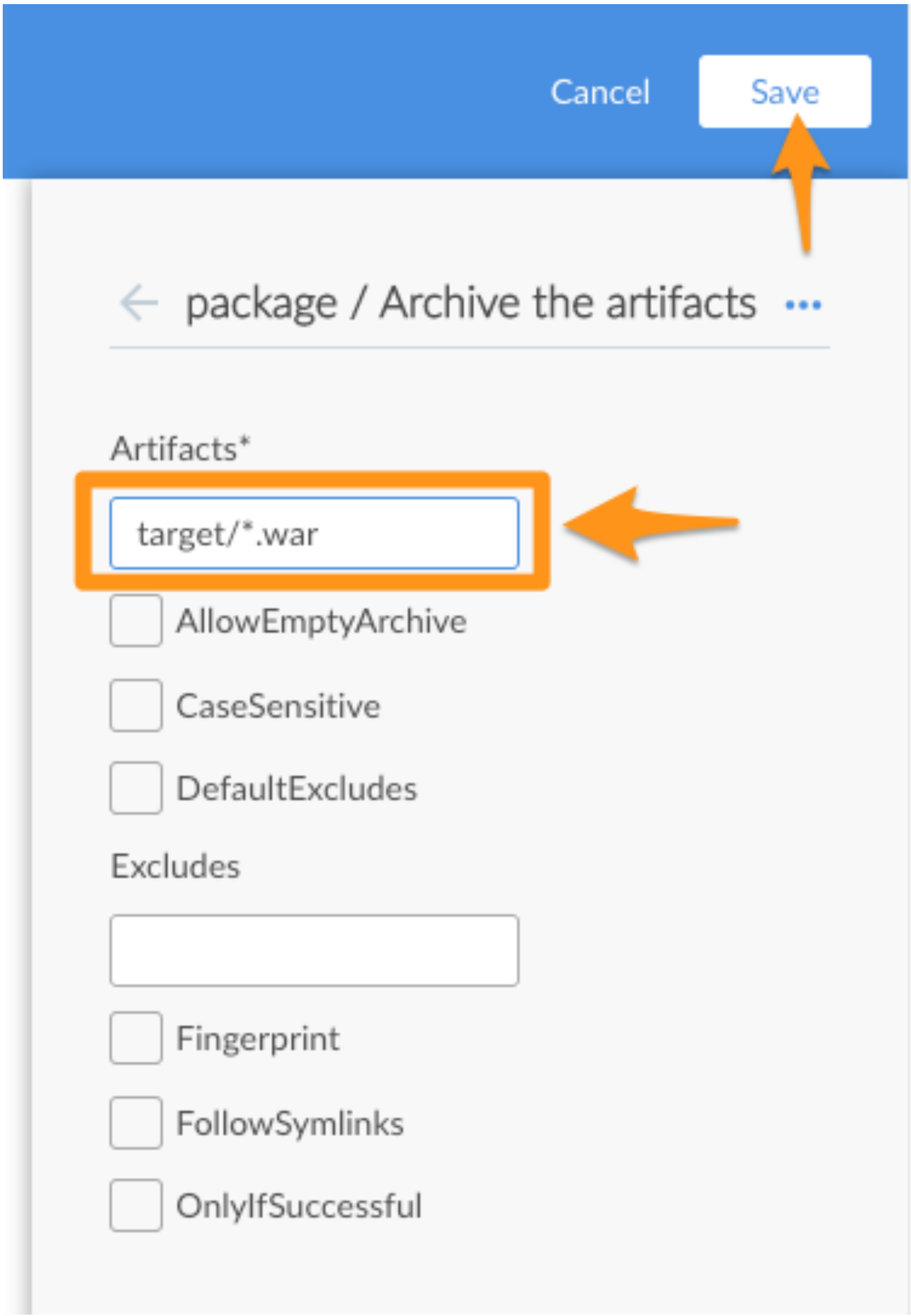
Shell Script
mvn package -DskipTests

Search for **archive** in the list, and you should see the desired step listed i.e. **Archive the Artifact**. Select it.



Provide the artefact path as `**/target/*.jar` and save.

TODO



package / Archive the artifacts

Artifacts*

target/*.war

□ AllowEmptyArchive

□ CaseSensitive

□ DefaultExcludes

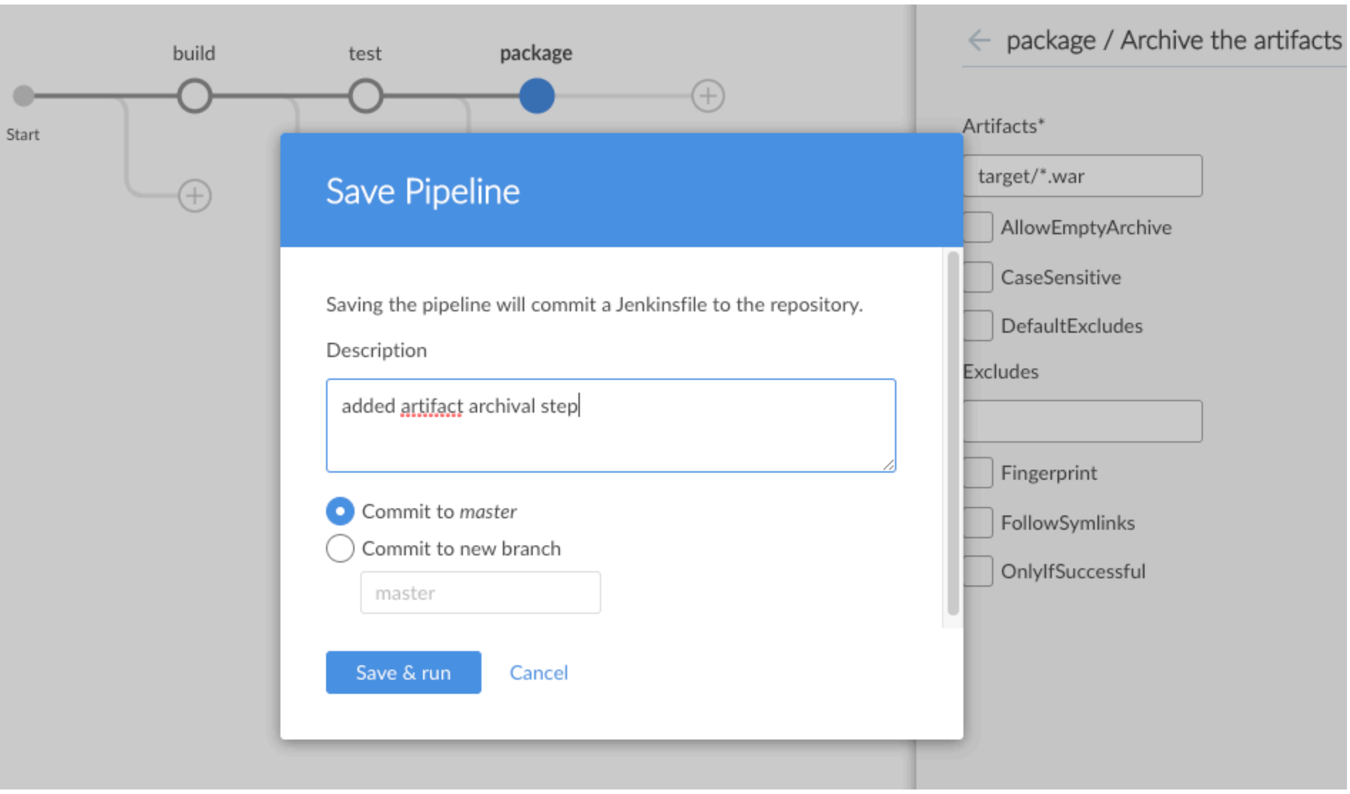
Excludes

□ Fingerprint

□ FollowSymlinks

□ OnlyIfSuccessful

Clicking on save opens a **Save Pipeline** box. Provide a description and the branch (e//g master) to commit these changes to. Click on **Save and Run**



package / Archive the artifacts

Artifacts*

target/*.war

☐ AllowEmptyArchive

☐ CaseSensitive

☐ DefaultExcludes

Excludes

☐ Fingerprint

☐ FollowSymlinks

☐ OnlyIfSuccessful

Save Pipeline

Saving the pipeline will commit a Jenkinsfile to the repository.

Description

added artifact archival step

● Commit to *master*

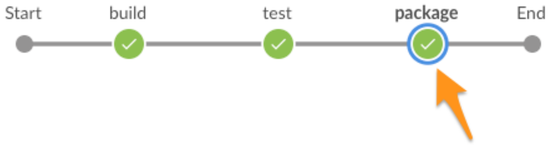
○ Commit to new branch

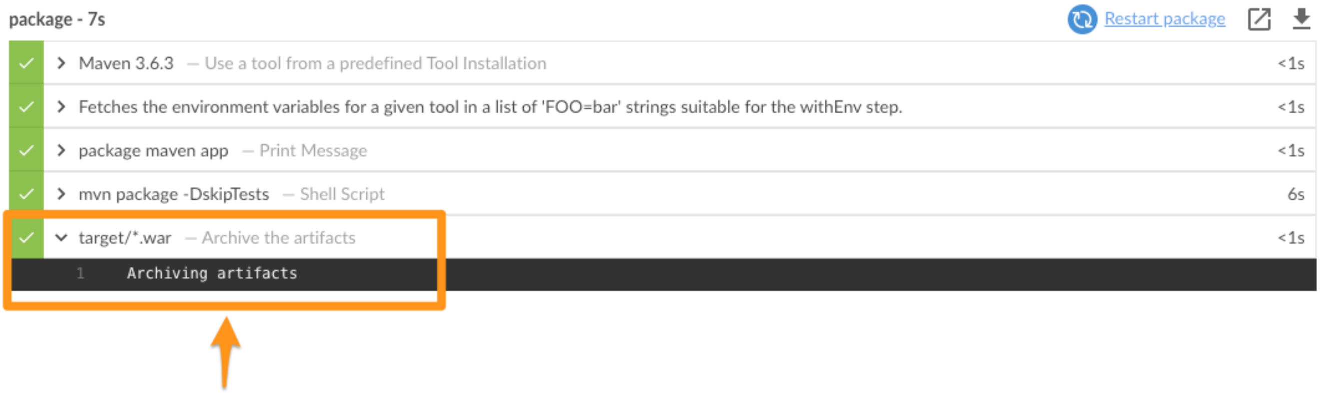
master

This will do two things,

1. Commit the changes to your GitHub repo.
2. Trigger a new pipeline run.

If you observe the pipeline run, it shows up the **Archiving artifacts** message for the package job.



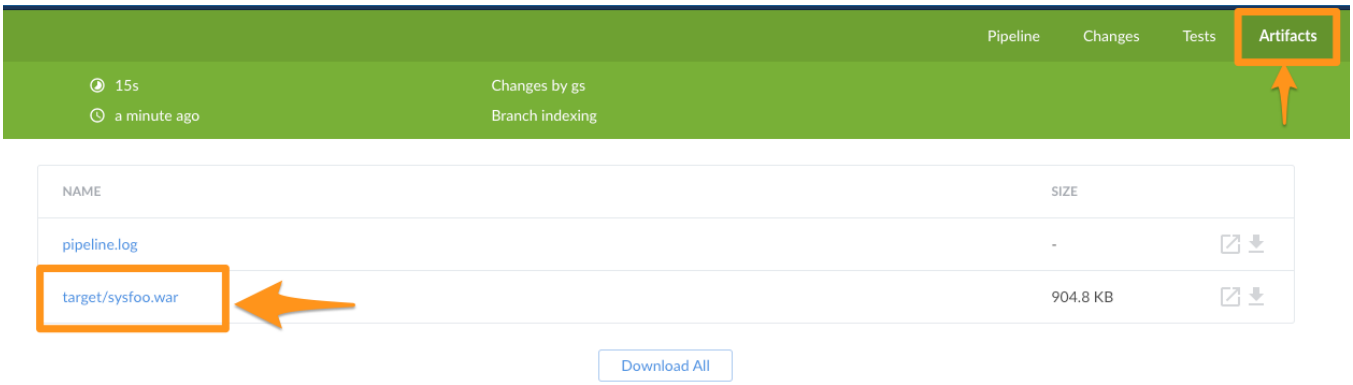


package - 7s

✓	Maven 3.6.3 — Use a tool from a predefined Tool Installation	<1s
✓	Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	package maven app — Print Message	<1s
✓	mvn package -DskipTests — Shell Script	6s
✓	target/*.war — Archive the artifacts	<1s

```
Archiving artifacts
```

You could also validate the artefact is archived by visiting the **artifacts** tab.



15s
a minute ago

Changes by gs
Branch indexing

NAME	SIZE
pipeline.log	-
target/sysfoo.war	904.8 KB

Try This

What you have created is a multi branch pipeline with two way integration with GitHub. You should also try the following to understand how it works,

- Observe the GitHub commits and find out how Jenkins is sending updates back to GitHub for every commit.
- Create a new branch on GitHub, observe Jenkins automatically tracking it and launching a new pipeline. When you delete the branch from GitHub, the pipeline will be deleted from Jenkins side as well. Now, Isn't that fantastic. .?

#cicd/labsv3