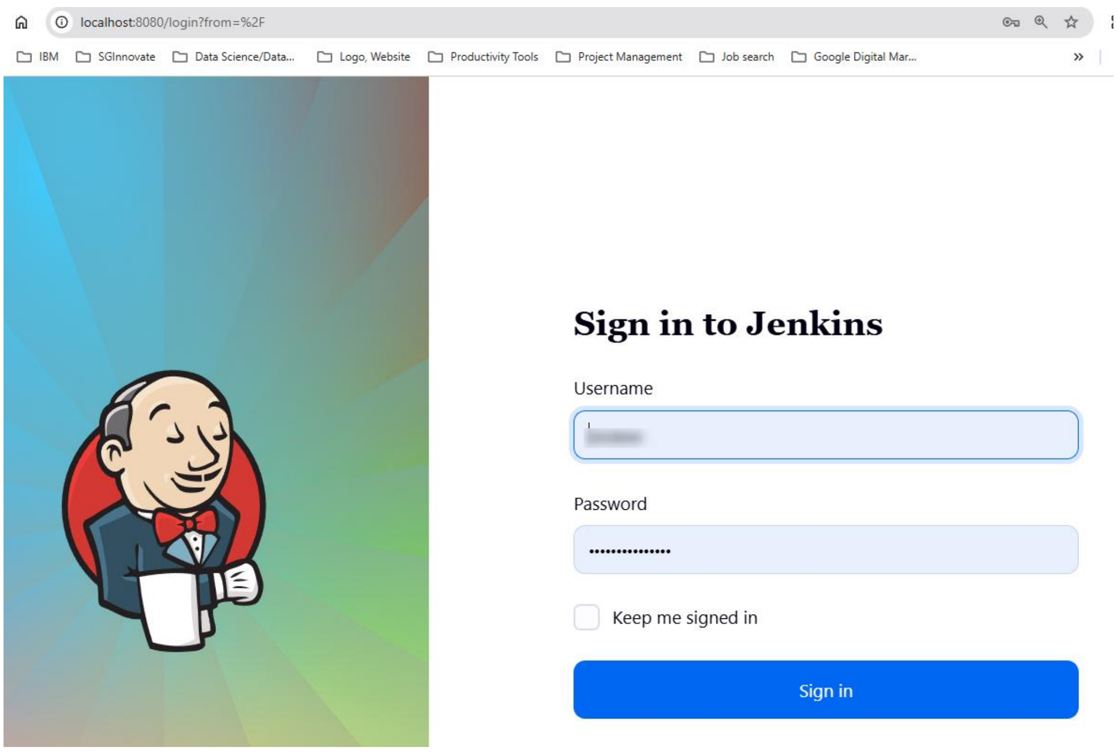
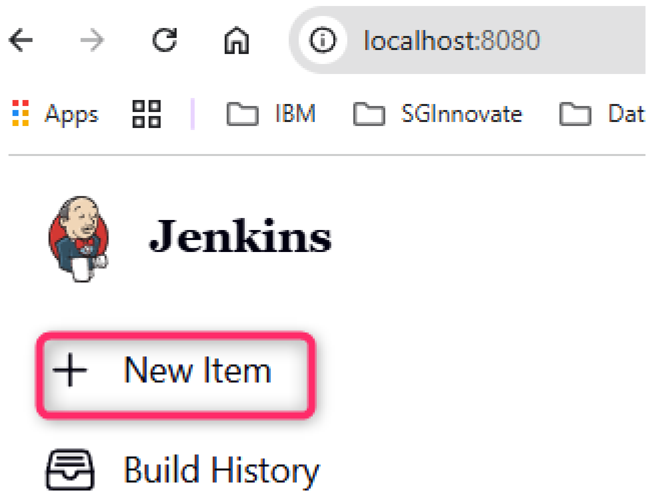
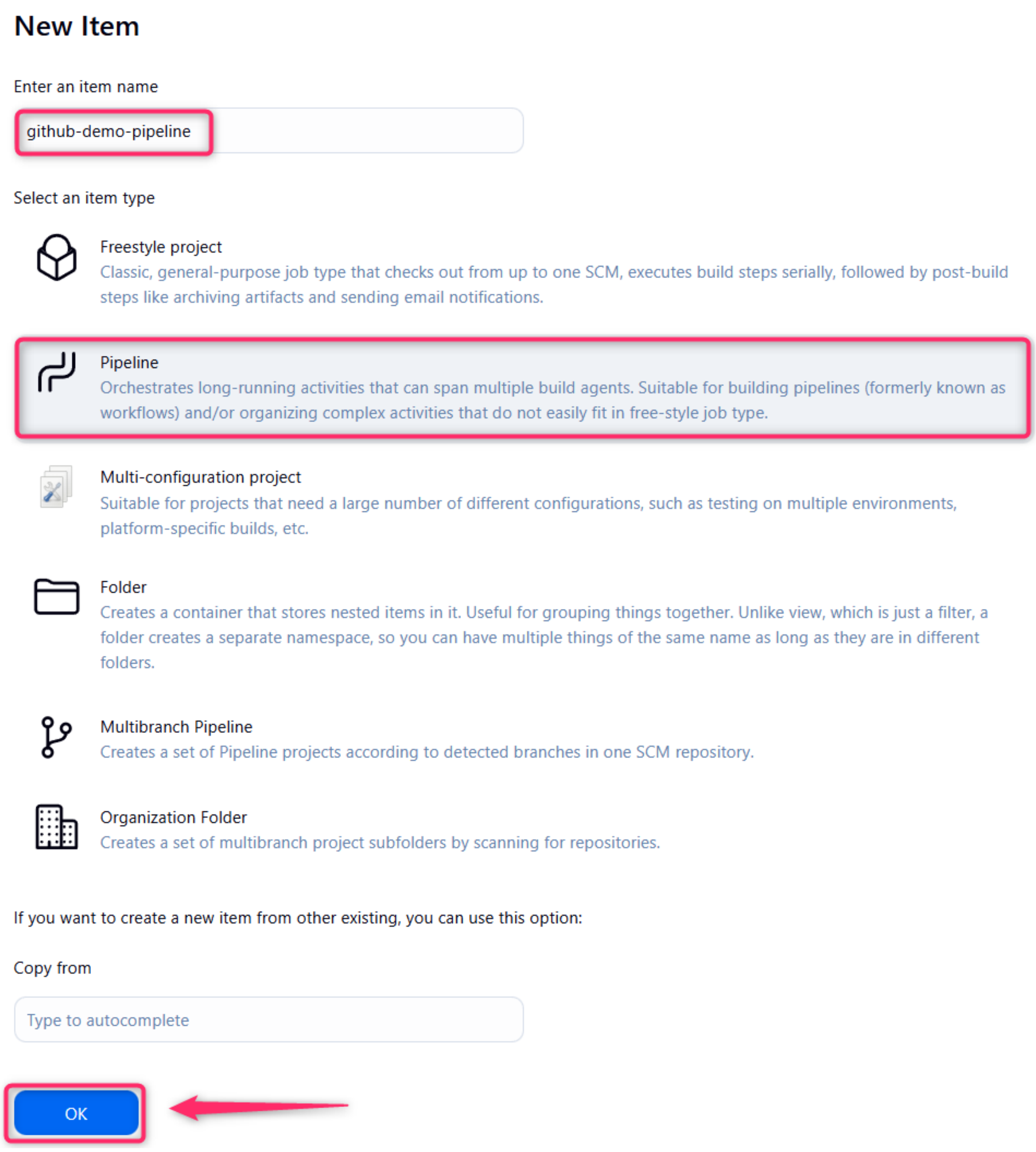
1. Go to your localhost:8080 and login to Jenkins



2. Click on "New Item" or "Create a new job"



3. Give your item a descriptive name, like github-demo-pipeline. Select "Pipeline" as the project type. Click "OK" to create the new job.



New Item

Enter an item name

github-demo-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.

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.

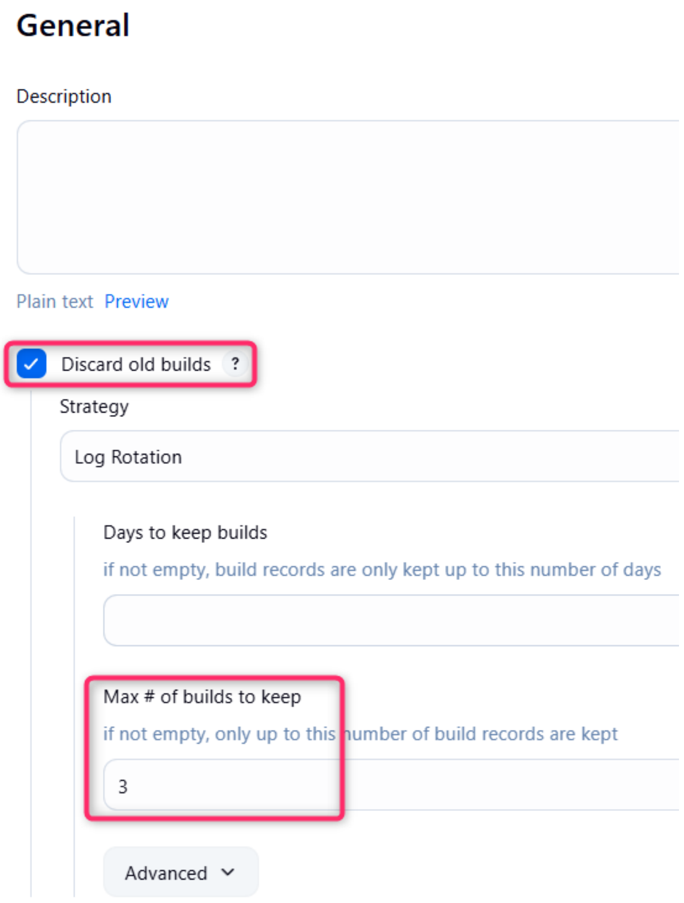
Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

Type to autocomplete

4. Configure the Pipeline to Use GitHub.
   Select discard old builds in General section.



General

Description

Plain text Preview

☑ Discard old builds

Strategy

Log Rotation

Days to keep builds

if not empty, build records are only kept up to this number of days

Max # of builds to keep

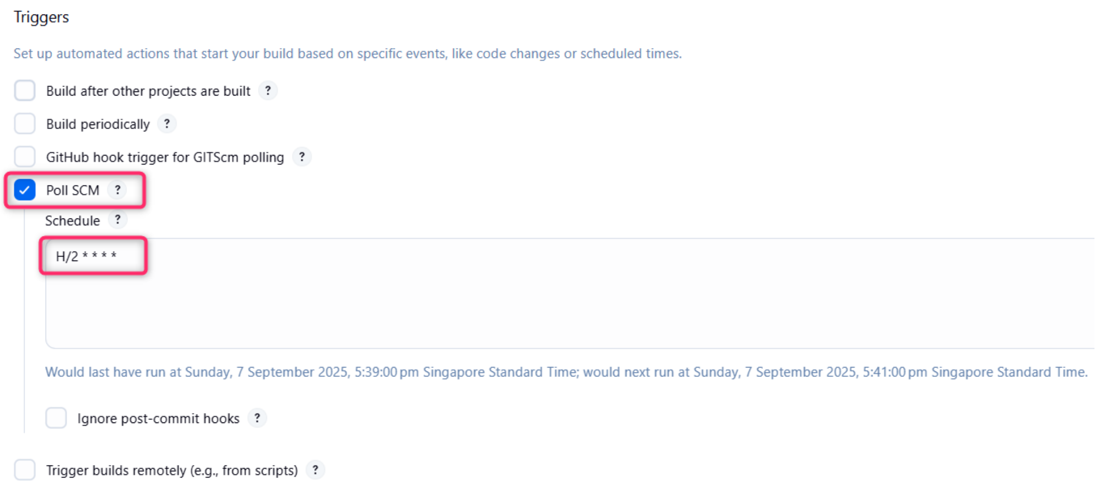
if not empty, only up to this number of build records are kept

3

Advanced

For this exercise, select Max # of builds = 3. This means Jenkins will only keep the last 3 builds and discard the rest.

Select the Poll SCM in the Triggers section. In the Schedule, put down: H/2 * * * * (make sure to copy exactly, with the space between 2 and the *). This means Jenkins will pull from the GitHub every 2 minutes to see if there are any new changes to trigger the build.



Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

☐ Build after other projects are built

☐ Build periodically

☐ GitHub hook trigger for GITScm polling

☑ Poll SCM

Schedule

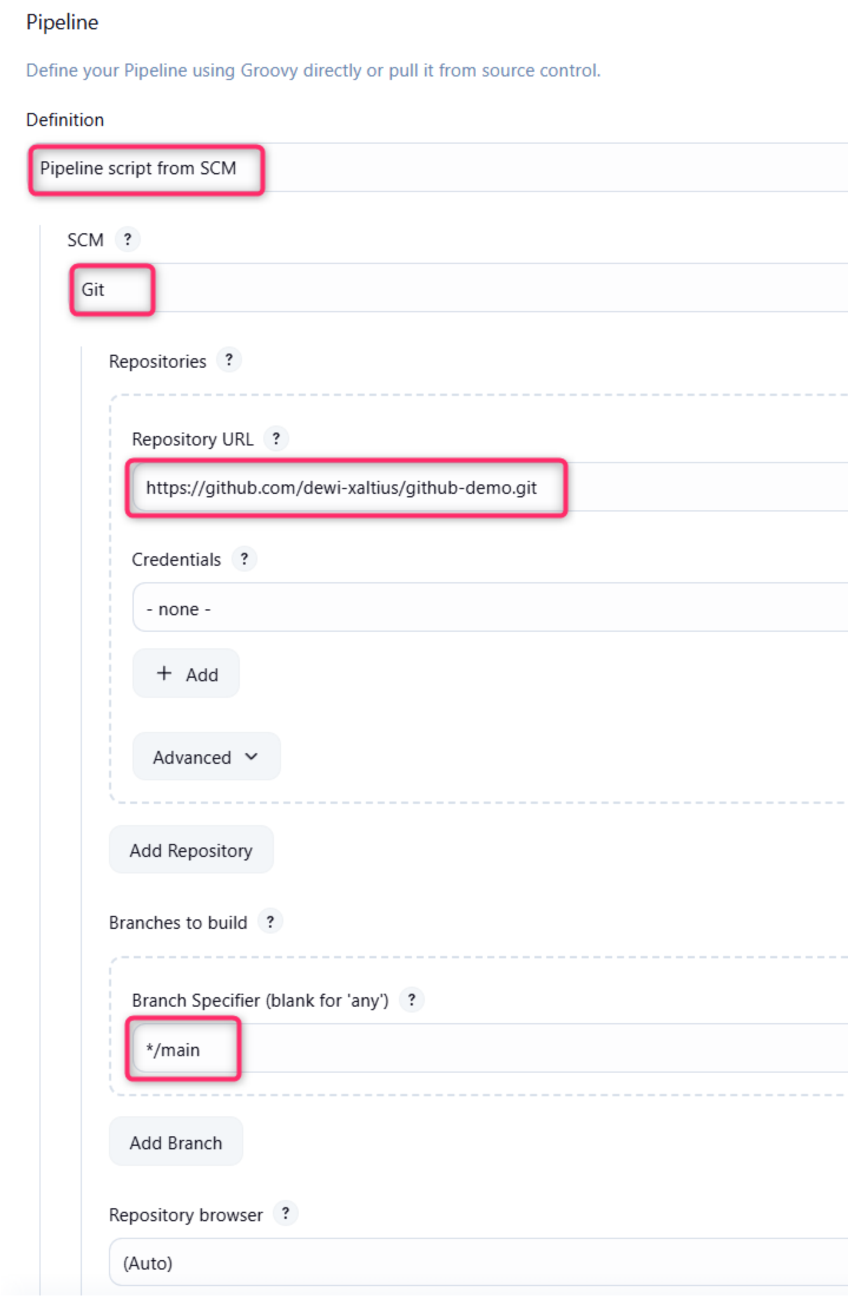
H/2 * * * *

Would last have run at Sunday, 7 September 2025, 5:39:00 pm Singapore Standard Time; would next run at Sunday, 7 September 2025, 5:41:00 pm Singapore Standard Time.

☐ Ignore post-commit hooks

☐ Trigger builds remotely (e.g., from scripts)

For the Definition, select “Pipeline script from SCM”. Once you have selected this, a new set of options will appear. For the SCM field, select Git. Enter your repository URL. Ensure the Branch Specifier is set to */main.



Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/dewi-xaltius/github-demo.git

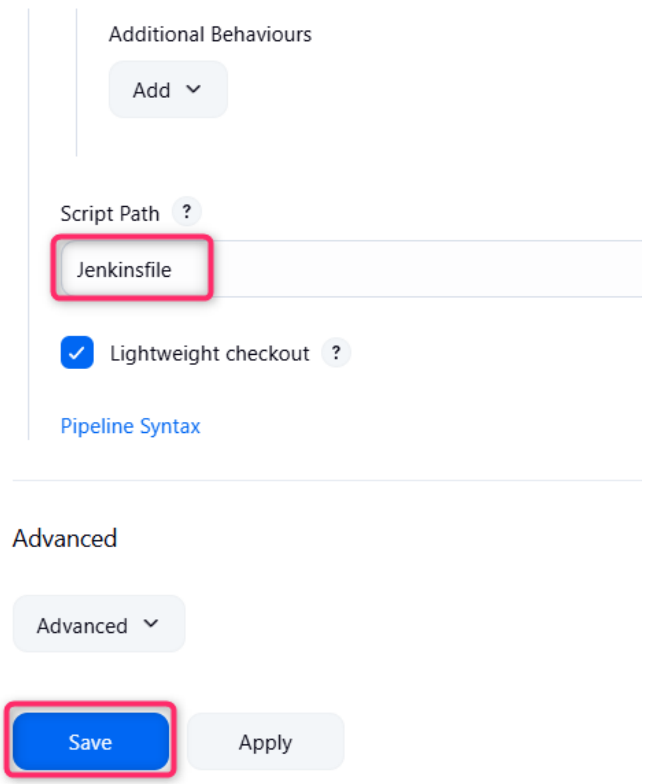
Credentials: - none -

Branches to build

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

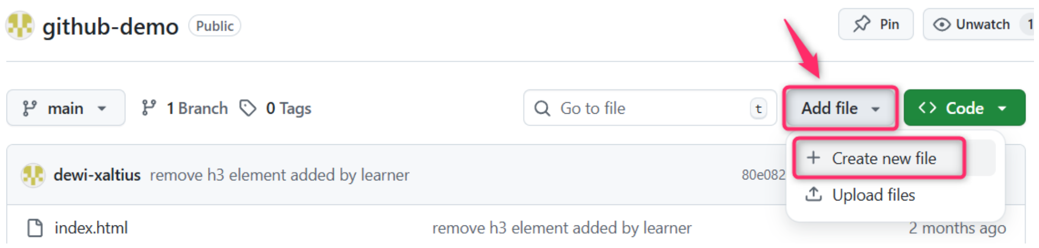
Repository browser: (Auto)

Leave the Script Path as the default Jenkinsfile.

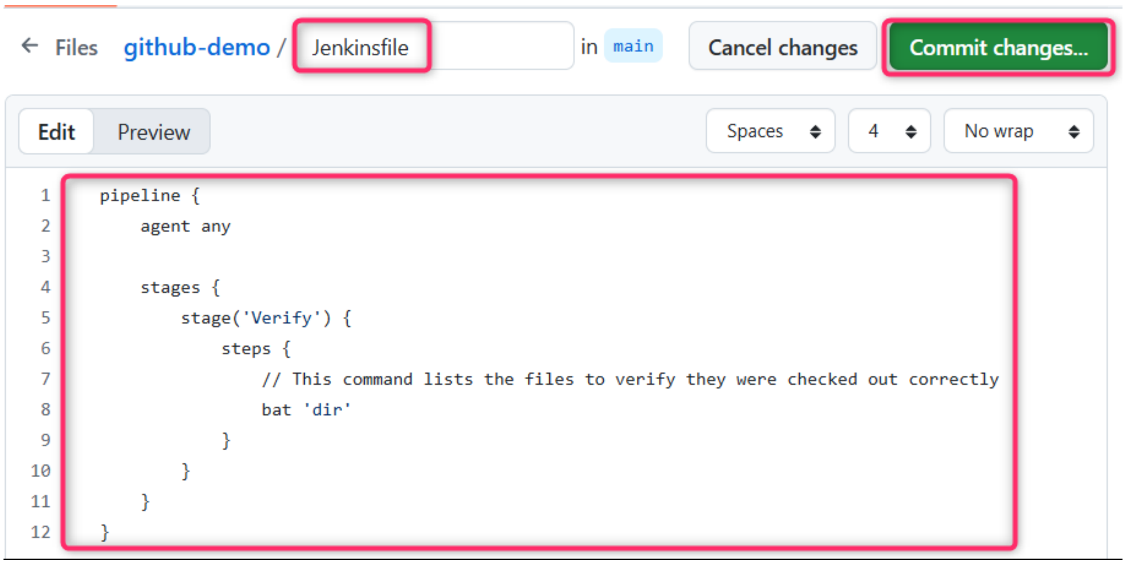


Now that your Jenkins job is configured, it needs a set of instructions to follow. These instructions are stored in a file named Jenkinsfile at the root of your GitHub repository. Think of it as the recipe for your CI/CD pipeline.

5. Create the Jenkinsfile
   The Jenkinsfile is written in a Groovy-based Domain Specific Language (DSL) and defines the stages of your pipeline.



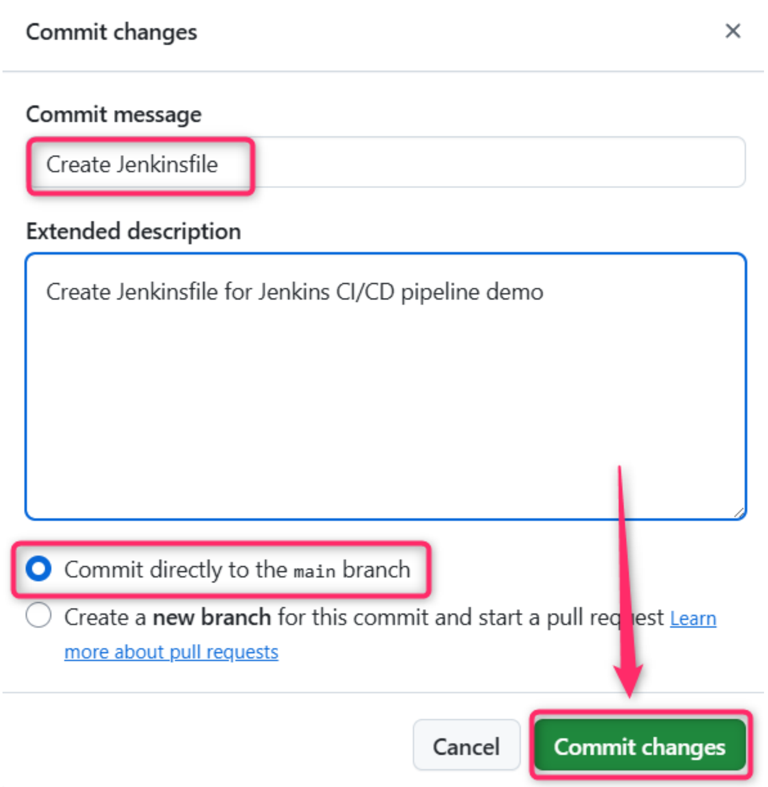
Jenkins looks for this specific file name by default in the root of the repository. Commit your changes to your main branch



```
pipeline {
    agent any

    stages {
        stage('Verify') {
            steps {
                // This command lists the files to verify they were checked out correctly
                bat 'dir'
            }
        }
    }
}
```

Note: if you are using mac, you need to change bat to sh:

```
stages {
    stage('Verify') {
        steps {
            // This command lists the files on a macOS/Linux machine
            sh 'ls -al'
        }
    }
}
```



Commit message

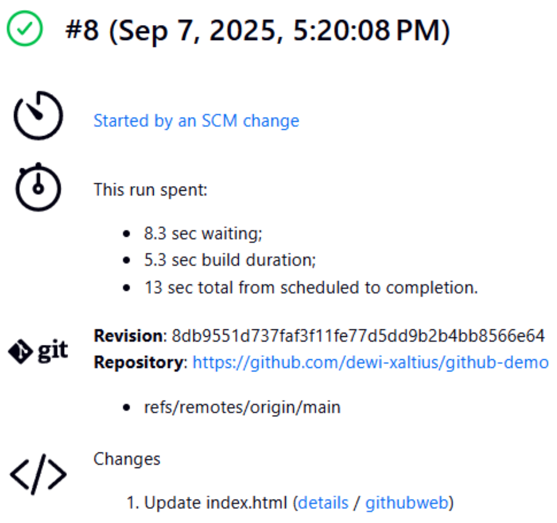
Create Jenkinsfile

Extended description

Create Jenkinsfile for Jenkins CI/CD pipeline demo

● Commit directly to the main branch
○ Create a new branch for this commit and start a pull request Learn more about pull requests
Cancel
Commit changes

6. Check Jenkins for your build



#8 (Sep 7, 2025, 5:20:08 PM)

Started by an SCM change

This run spent:

- 8.3 sec waiting;
- 5.3 sec build duration;
- 13 sec total from scheduled to completion.

Revision: 8db9551d737faf3f11fe77d5dd9b2b4bb8566e64
Repository: https://github.com/dewi-xaltius/github-demo

- refs/remotes/origin/main

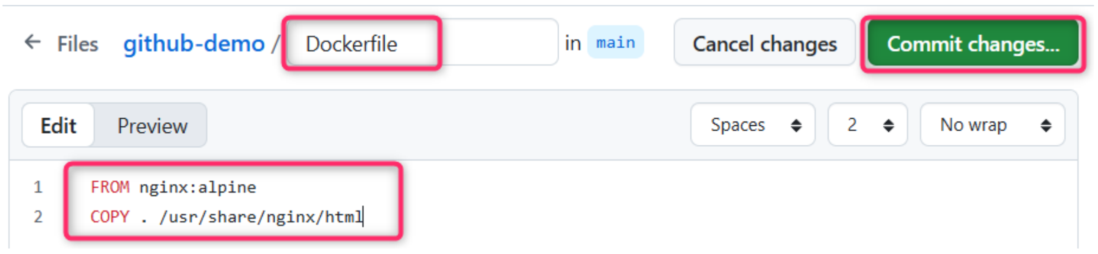
Changes

1. Update index.html (details / githubweb)

It should have a green check when your build is successful.

7. Create a few changes in your GitHub repo, commit the changes to your main, check Jenkins for the build. Do this a few times so you can see that Jenkins is only keeping the last 3 builds.

8. Create a Dockerfile in the root of your GitHub repo



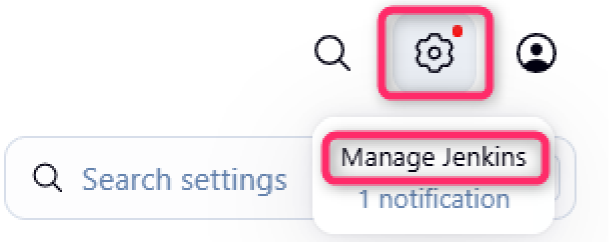
```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

9. Update the Jenkinsfile

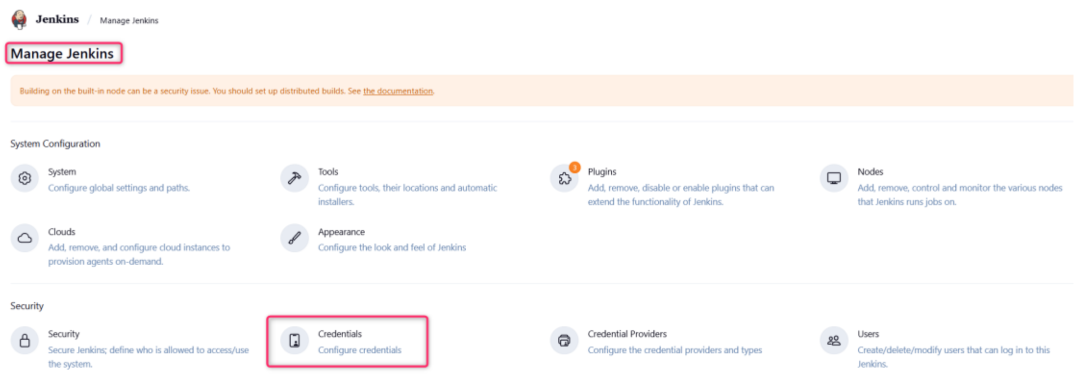
```
pipeline {
  agent any

  stages {
    stage('Check Docker Version') {
      steps {
        bat 'docker --version'
      }
    }

    stage('Verify') {
      steps {
        bat 'dir'
      }
    }
```

```
            stage('Build Docker Image') {
              steps {
                bat 'docker build -t your-docker-hub-username/github-demo .'
              }
            }

            stage('Push Docker Image') {
              steps {
                withCredentials([usernamePassword(credentialsId: 'docker-hub-
        credentials', passwordVariable: 'DOCKER_PASSWORD', usernameVariable:
        'DOCKER_USERNAME')]) {
                  bat 'docker login -u %DOCKER_USERNAME% -p
        %DOCKER_PASSWORD%'
                  bat 'docker push your-docker-hub-username/github-demo'
                }
              }
            }
          }
}
```

10. Go to Jenkins settings -> Manage Jenkins



Select Credentials.


Jenkins / Manage Jenkins

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

Security

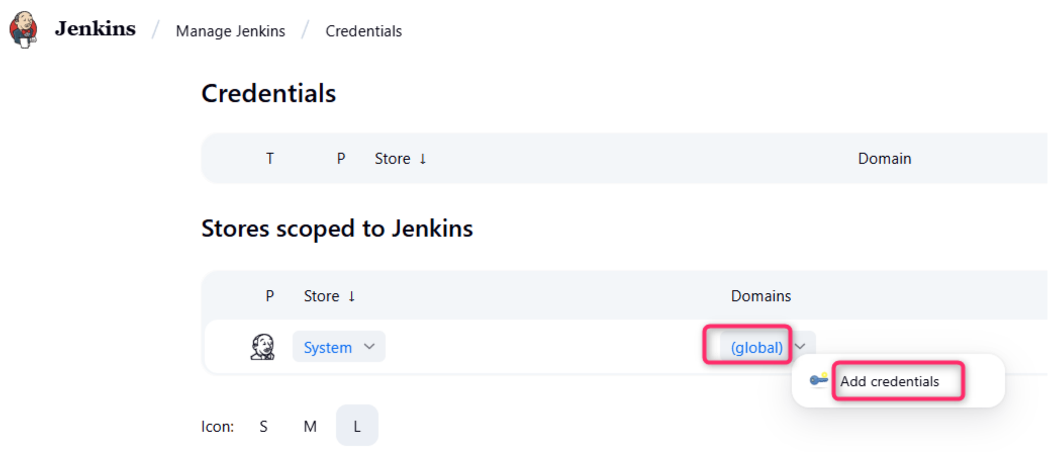
Security
Secure Jenkins; define who is allowed to access/use the system.

Credentials
Configure credentials

Credential Providers
Configure the credential providers and types

Users
Create/delete/modify users that can log in to this Jenkins.

Select global -> Add credentials



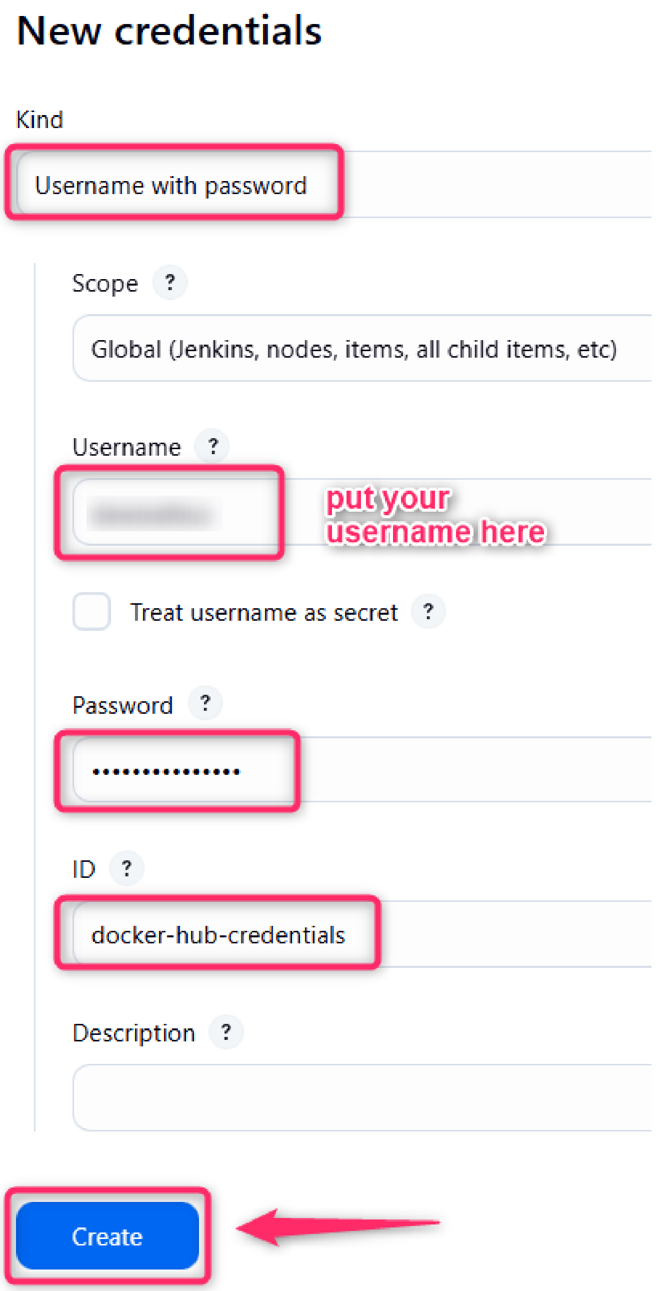
Jenkins / Manage Jenkins / Credentials

Credentials

T	P	Store ↓	Domain

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)



New credentials

Kind: Username with password

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

Username: [redacted]

Treat username as secret

Password: •••••••••••••••

ID: docker-hub-credentials

Description:

11. When you are still setting up the credentials, because you have created the changes in your GitHub, it will trigger a build and the build will fail until you completed the credentials. Run the build again.

12. Check that you can see the docker image in your Docker Desktop

13. Run the docker container

    ```
    docker run -d -p 8081:80 --name github-demo-app <your-docker-hub-
    username>/github-demo
    ```

    We are running the docker container at port 8081 because Jenkins is running at 8080.

14. View your web app at http://localhost:8081

15. Clean up: Once you are done, please stop and remove the container to free up resources with these commands:

    ```
    docker stop github-demo-app
    docker rm github-demo-app
    ```

Note: When you make changes to your GitHub repo, Jenkins will create a new build and a new image with those changes. Jenkins will push that new image to Docker Hub. In order for you to see the changes, you have to run a new container that will pull from the image from the latest changes. Then when you go to http://localhost:8081, you will see your new changes.