

Documentation for learning:

Requirement:

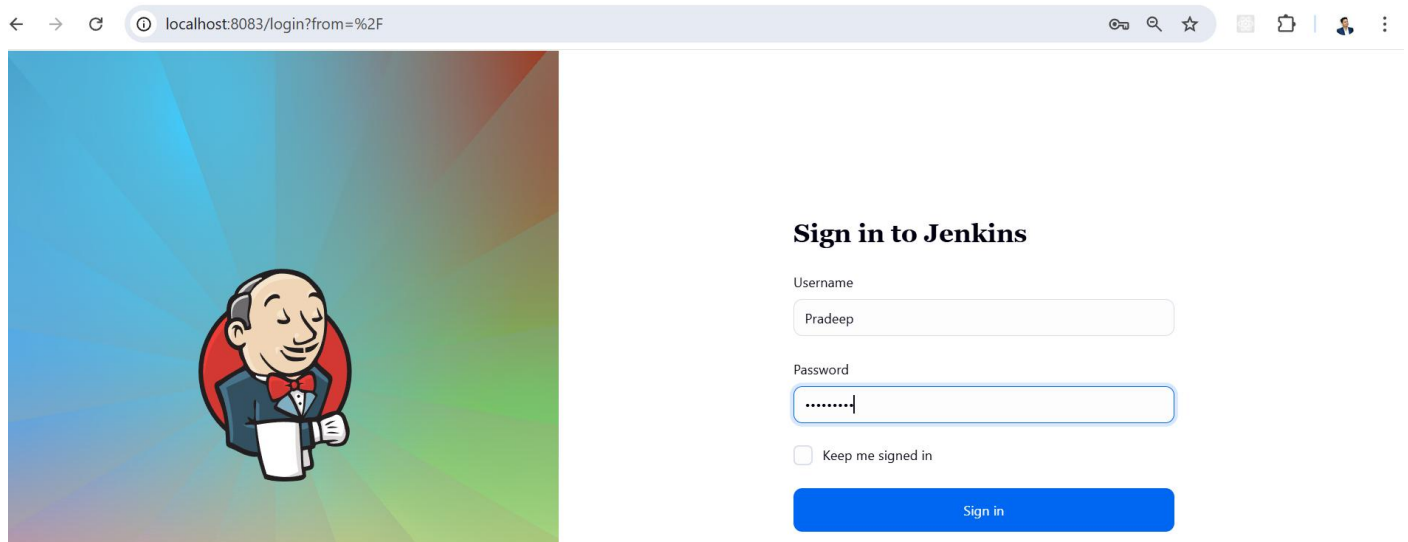
- 1) Create project on git hub
- 2) Create a pipeline in Jenkin
- 3) Do the configuration in Jenkin it execute Jenkin script from project.
- 4) Jenkin pipeline should run in 30 min and also can be run manually.
- 5) Create a docker image of application and Jenkin should deploy the project in Docker.
- 6) Jankin Pipeline will Checkout, Build, SonarQube Analysis, Docker Build, Test and deploy on docker hub.

Prerequisite

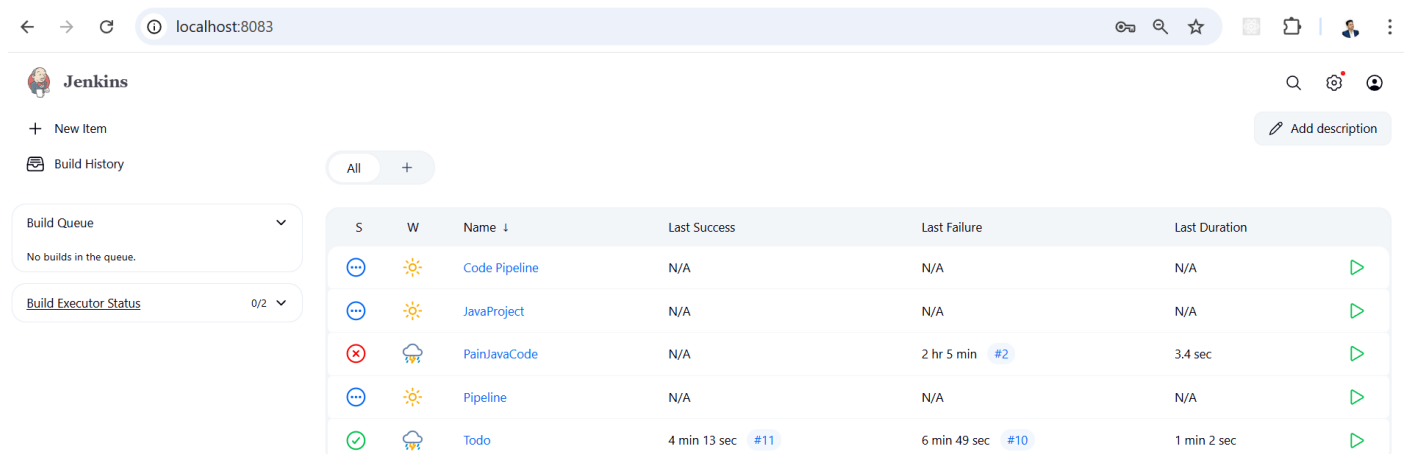
- 1) Create account on Github and push your code to git hub.
- 2) Install docker desktop and create account on docker hub start the docker
- 3) Install sonar and run it locally
- 4) Install Jenking and run it locally

Solution

Login to jenkins[Pradeep/Test@....]



Click on New Item



S	W	Name ↓	Last Success	Last Failure	Last Duration
...	☀	Code Pipeline	N/A	N/A	N/A
...	☀	JavaProject	N/A	N/A	N/A
✖	☁	PainJavaCode	N/A	2 hr 5 min #2	3.4 sec
...	☀	Pipeline	N/A	N/A	N/A
✔	☁	Todo	4 min 13 sec #11	6 min 49 sec #10	1 min 2 sec

Enter name you want to give pipeline and also select pipeline plugin and click on OK

← → ↺

localhost:8083/view/all/newJob

🔍 ☆ 🗂️ 🧑

Jenkins

/ All / New Item


🔍 ⚙️ 👤


### New Item


Enter an item name


todo\_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**

OK

## In Pipeline section

← → ↺

localhost:8083/job/todo\_pipeline/configure

🔍 ☆ 🗂️ 🧑

Jenkins


/ todo\_pipeline / Configuration

🔍 ⚙️ 👤

### Configure

⚙️ General

🕒 Triggers

 Pipeline

🔧 Advanced

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

#### 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/pradeepTechStream/todo-jenkins.git

Credentials ?

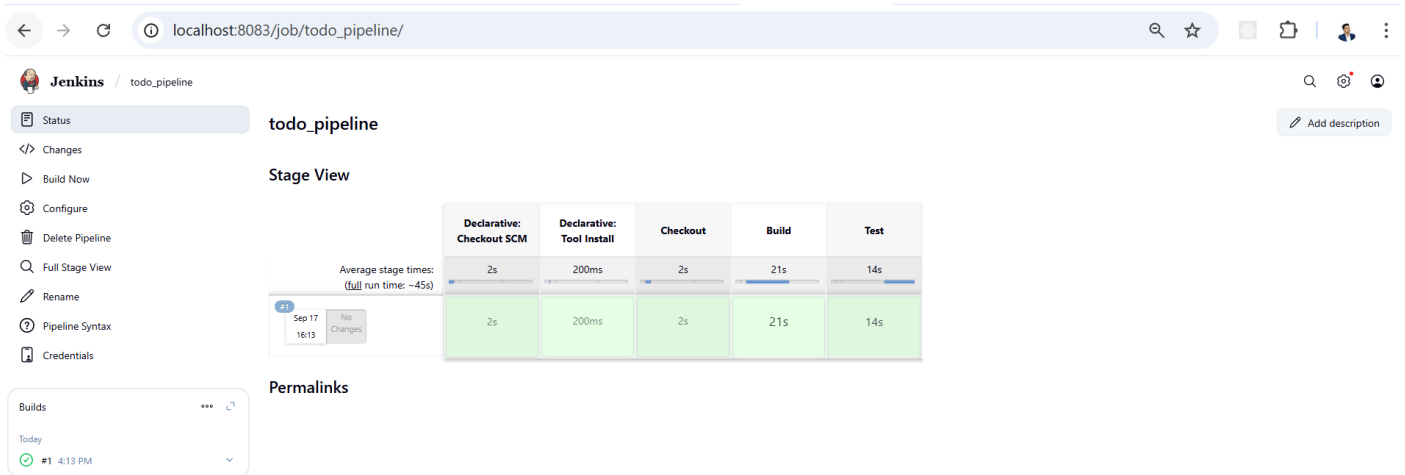
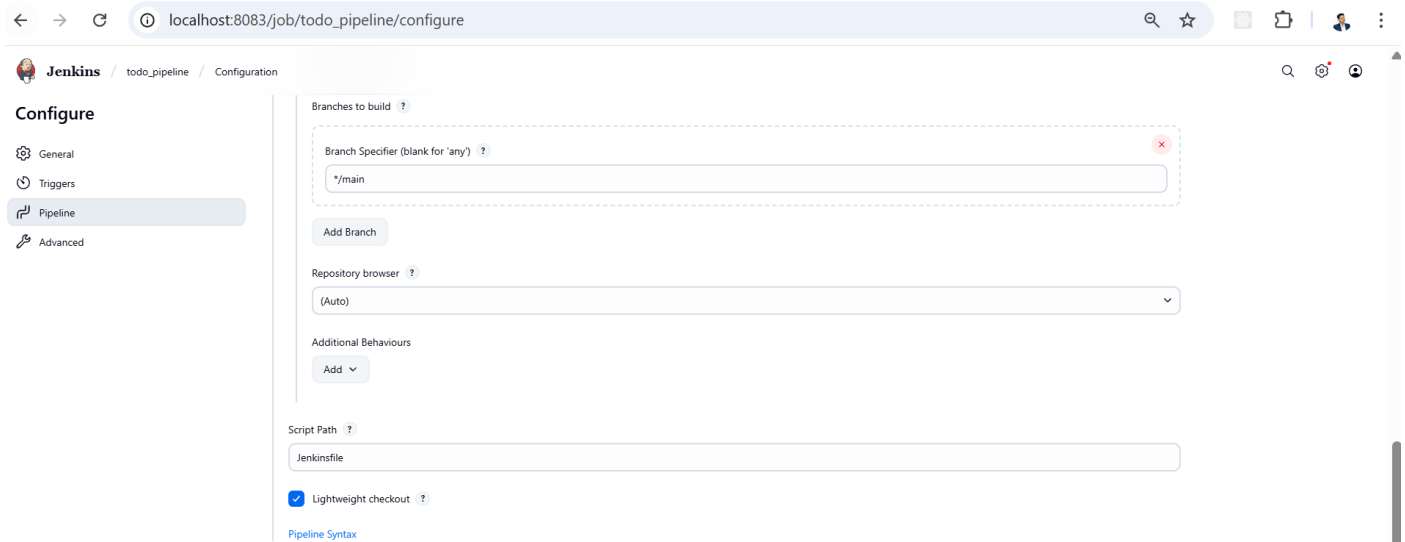
- none -

+ Add

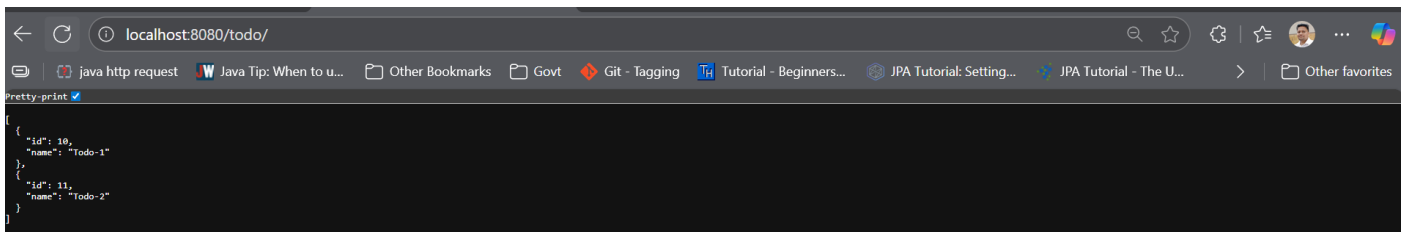
Advanced ▾

Add Repository

Branches to build ?

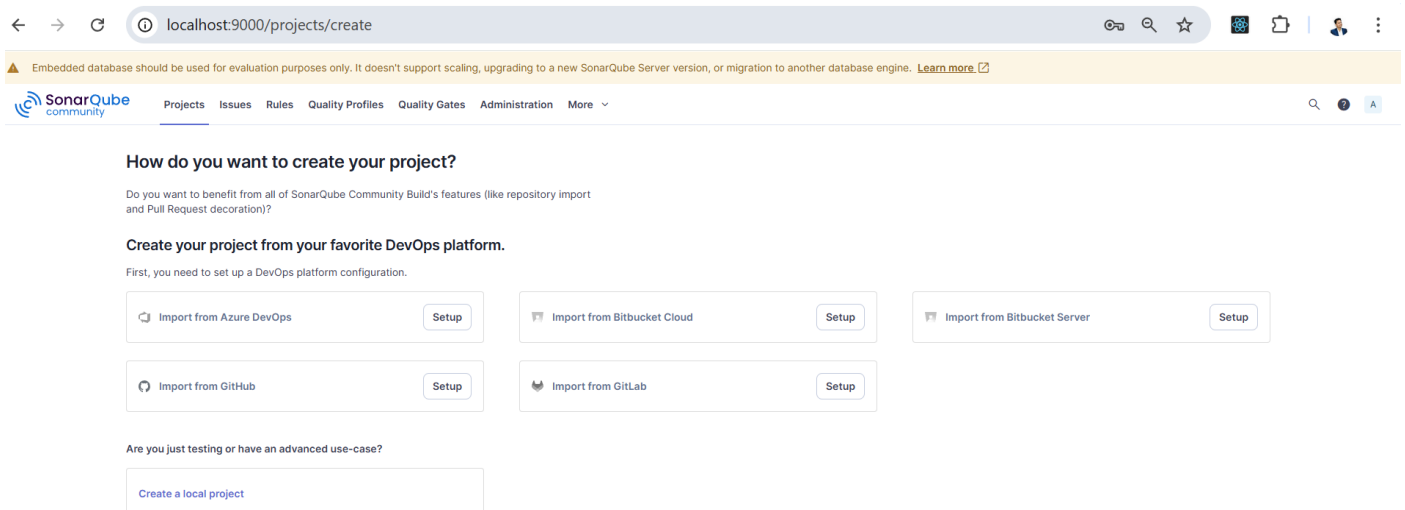


Deployed the code on docker



Install sonar and run command to start: C:\sonarqube\bin\windows-x86-64\StartSonar.bat

After installation default credential is admin / admin but you need to change password [Test@2..1#..3]



Configuration done for sonar as well.

The screenshot shows the SonarQube web interface at `localhost:9000/project/issues?issueStatuses=OPEN%2CCONFIRMED&impactSoftwareQualities=MAINTAINABILITY&id=todo-app`. The interface displays a list of issues for the 'todo-app' project. The left sidebar shows filters for 'My Issues' and 'All', with a 'Clear All Filters' button. The main area shows a list of issues, including 'Remove this unused import 'java.util.List'' and 'Remove this field injection and use constructor injection instead'. The issues are categorized by 'Maintainability' and 'Reliability'.

Todo app on docker hub container

The screenshot shows the Docker Desktop interface. The left sidebar lists 'Containers', 'Images', 'Volumes', 'Builds', 'Dev Environments', 'BETA', 'Docker Scout', 'Extensions', and 'Add Extensions'. The main area shows a table of containers. The 'todo-app' container is running, with a CPU usage of 0.99% and a port of 8080:8080. The 'inspiring\_bhaskara' container is exited (143).

REST API is working for Todo

The screenshot shows a web browser at `localhost:8080/todo/` displaying the REST API response for the Todo app. The response is a JSON array of two todo items:

```
[{"id": 10, "name": "Todo-1"}, {"id": 11, "name": "Todo-2"}]
```

Jenkin will build + Test + Run Sonar + deploy the code on Docker