**Objective**:

To demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

**Tools Used:**

|  |  |
| --- | --- |
| Tools | Description |
| Git Hub | Version control tool and repository |
| Jenkins | To Orchestrate a pipeline |
| Groovy | To write pipeline as a code |
| Docker | To Containerize application |
| Shell Script | Scripting tool used to execute steps sequentially |
|  |  |

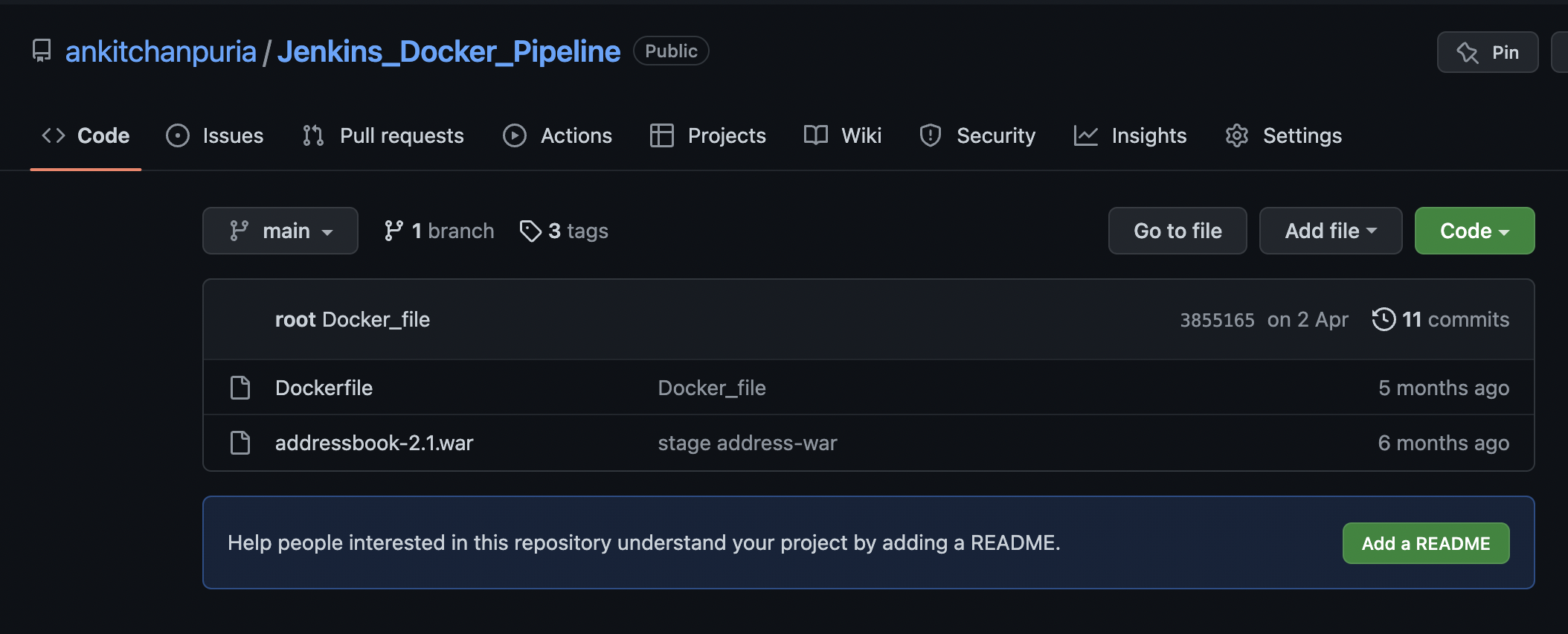
**Steps Involved:**

1. Verify Jenkins is installed and running in targeted VM (Linux/Ubuntu).

Text

Description automatically generated

1. I have stored my Dockerfile and application war file in my git hub repo as below.



repository is available via this public link. Here [Jenkins\_Docker\_Pipeline](https://github.com/ankitchanpuria/Jenkins_Docker_Pipeline.git).

1. Docker file snippet which is used in this project.

Text

Description automatically generated

1. I use a groovy script and execute following steps in my Jenkins pipeline. Groovy script is available here [groovy\_script](https://github.com/ankitchanpuria/Jenkins_Docker_Pipeline/blob/main/groovy_script)

|  |  |
| --- | --- |
| Steps | Short Desc |
| GET SOURCE CODE | We pull application code and docker file from git hub repo. |
| RUN DOCKER | A docker container is created with base image as TOMCAT |
| STAGE APP CODE | We now stage our app code from git hub repo to docker container |
| BUILD DOCKER IMAGE | We now build docker image with base image and application code. |
| BUILD DOCKER IMAGE CONTAINER | We use docker image constructed in step 4 to spawn a new container with our application code. |

1. I have used a wrapper shell script that is responsible for taking this project through each step defined in step 4. Groovy script calls shell script at each stage with argument as per the step defined in the groovy. Wrapper shell script is available here [wrapper\_shell\_script](https://github.com/ankitchanpuria/Jenkins_Docker_Pipeline/blob/main/wrapper.sh)
2. Jenkins docker pipeline is ready.

A screenshot of a computer

Description automatically generated

1. Sample executions are as below

Graphical user interface, application

Description automatically generated