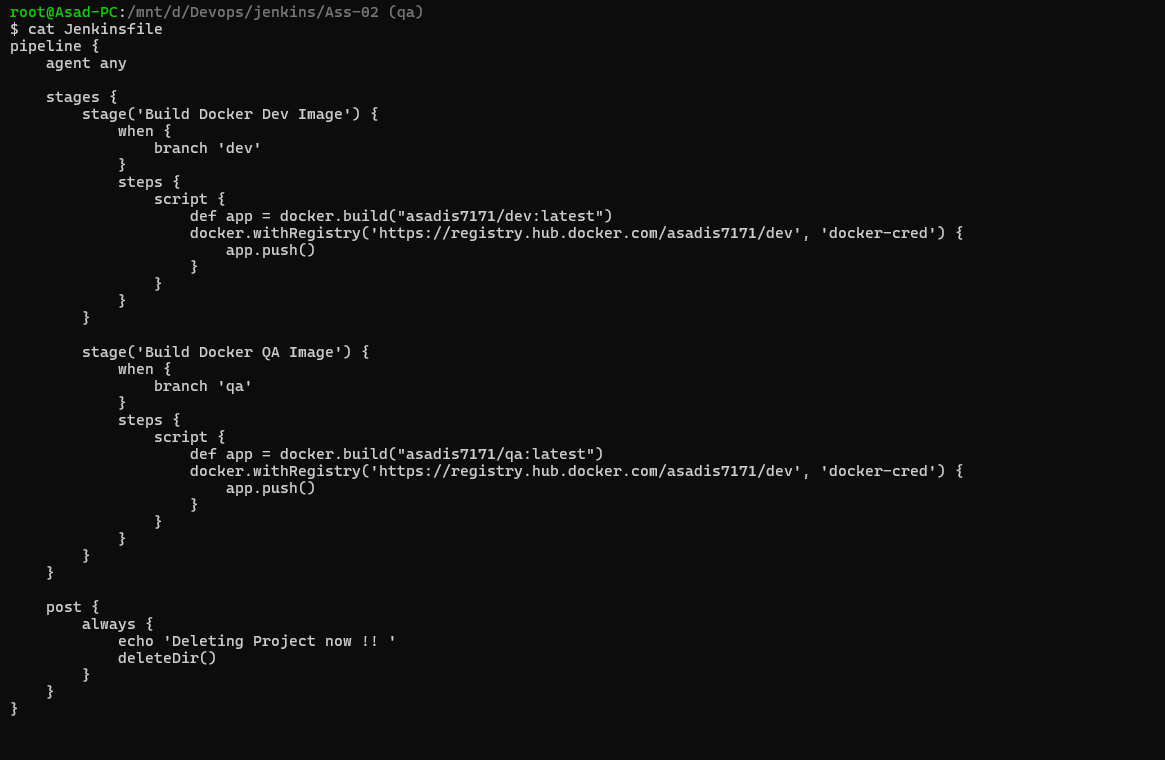
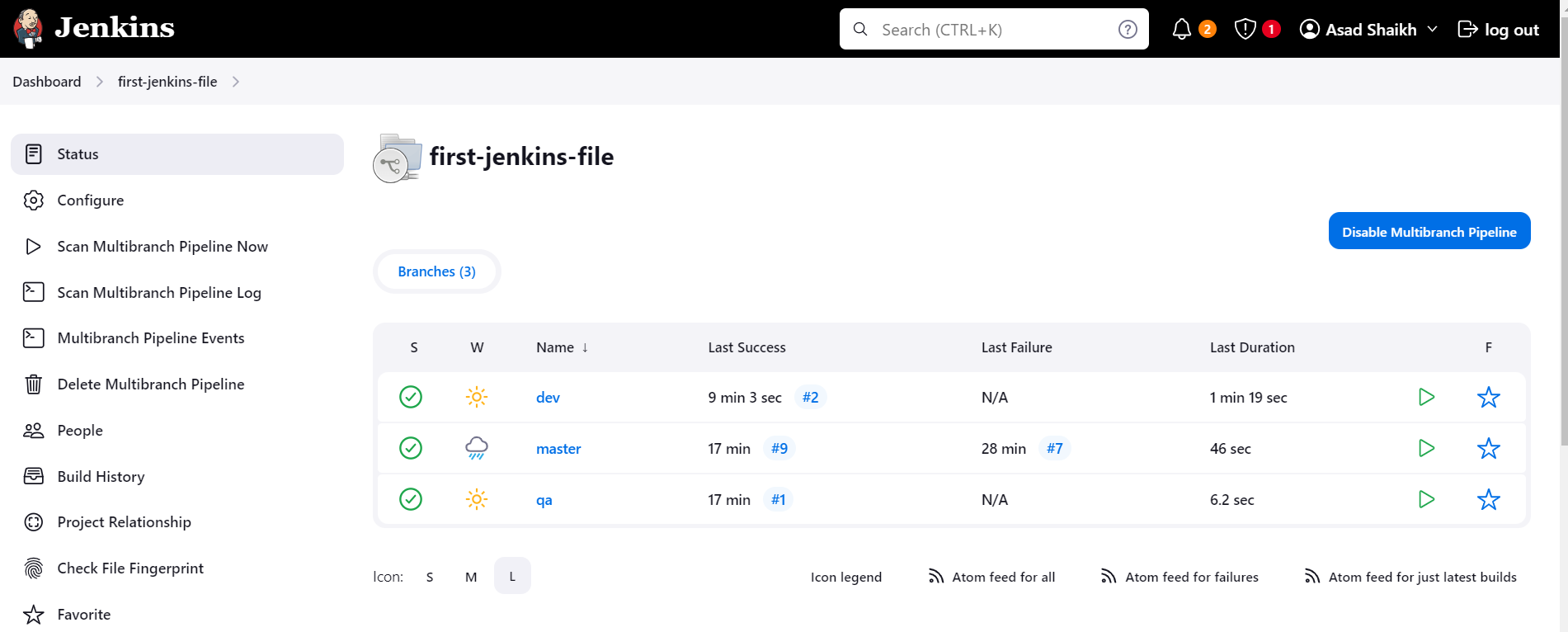
1. Create a Jenkins file to construct and upload Docker images to your Docker-hub registries. Ensure that when the branch is 'dev', the image is constructed and uploaded to the DEV Docker-hub registry. Similarly, when the branch is 'QA', it should be sent to the QA Docker-hub registry.

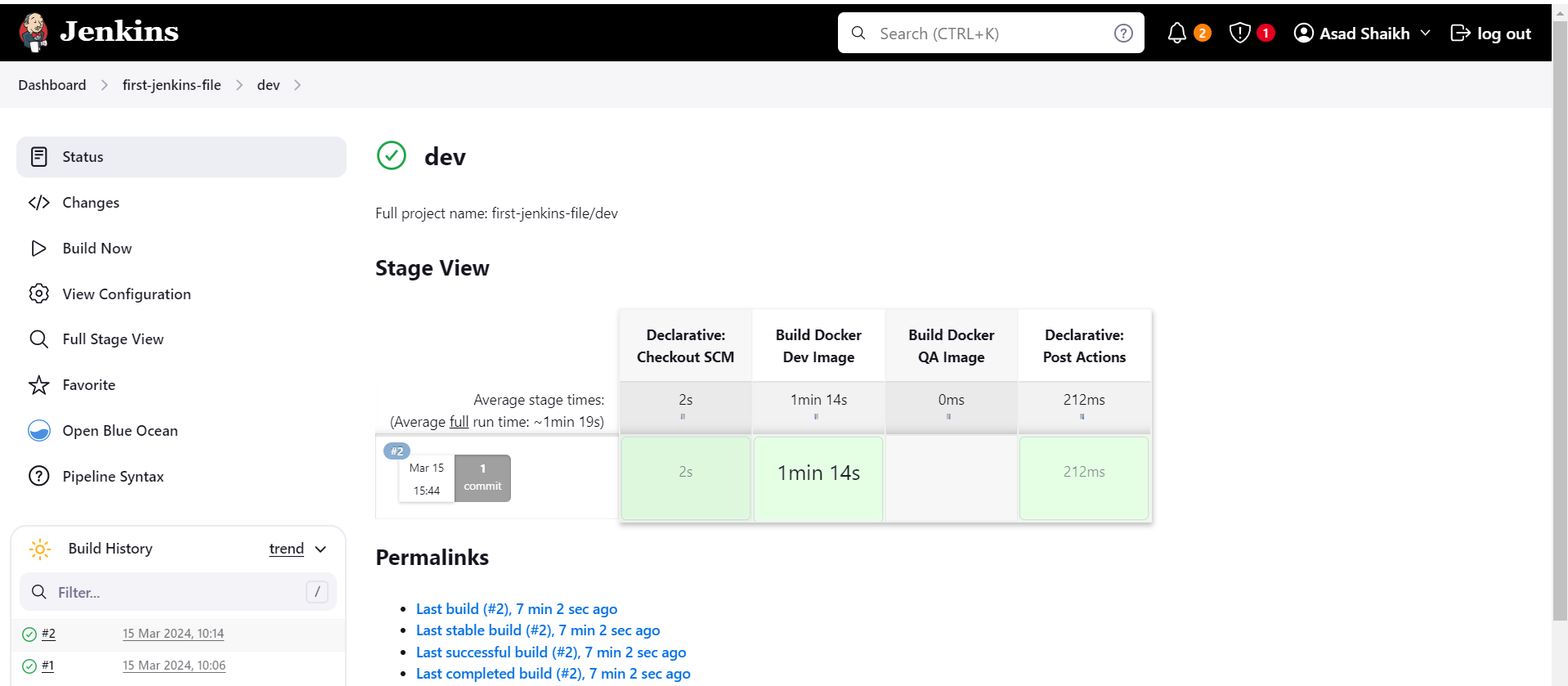
🡪



Multi branch pipeline



Running dev branch pipeline



Below are the logs

Started by user [Asad Shaikh](http://localhost:8080/user/admin)

> git rev-parse --resolve-git-dir /var/jenkins\_home/caches/git-f85fc5c1fab0c581be53f123d6c0a4e0/.git # timeout=10

Setting origin to <https://github.com/asadis7171/jenkins-ass-02.git>

> git config remote.origin.url <https://github.com/asadis7171/jenkins-ass-02.git> # timeout=10

Fetching origin...

Fetching upstream changes from origin

> git --version # timeout=10

> git --version # 'git version 2.39.2'

> git config --get remote.origin.url # timeout=10

using GIT\_SSH to set credentials my private key

Verifying host key using known hosts file

[You're using 'Known hosts file' strategy to verify ssh host keys, but your known\_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.](https://plugins.jenkins.io/git-client/#plugin-content-ssh-host-key-verification)

> git fetch --tags --force --progress -- origin +refs/heads/\*:refs/remotes/origin/\* # timeout=10

Seen branch in repository origin/dev

Seen branch in repository origin/master

Seen branch in repository origin/qa

Seen 3 remote branches

Obtained Ass-02/Jenkinsfile from 26c2632996ecef819c4dc74ab055e33781bb8ba2

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](http://localhost:8080/computer/(built-in)/) in /var/jenkins\_home/workspace/first-jenkins-file\_dev

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

using credential asad.pem

Cloning the remote Git repository

Cloning with configured refspecs honoured and without tags

Cloning repository <https://github.com/asadis7171/jenkins-ass-02.git>

> git init /var/jenkins\_home/workspace/first-jenkins-file\_dev # timeout=10

Fetching upstream changes from <https://github.com/asadis7171/jenkins-ass-02.git>

> git --version # timeout=10

> git --version # 'git version 2.39.2'

using GIT\_SSH to set credentials my private key

Verifying host key using known hosts file

[You're using 'Known hosts file' strategy to verify ssh host keys, but your known\_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key Verification Configuration' and configure host key verification.](https://plugins.jenkins.io/git-client/#plugin-content-ssh-host-key-verification)

> git fetch --no-tags --force --progress -- <https://github.com/asadis7171/jenkins-ass-02.git> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git config remote.origin.url <https://github.com/asadis7171/jenkins-ass-02.git> # timeout=10

> git config --add remote.origin.fetch +refs/heads/\*:refs/remotes/origin/\* # timeout=10

Avoid second fetch

Checking out Revision 26c2632996ecef819c4dc74ab055e33781bb8ba2 (dev)

> git config core.sparsecheckout # timeout=10

> git checkout -f 26c2632996ecef819c4dc74ab055e33781bb8ba2 # timeout=10

Commit message: " added all files"

> git rev-list --no-walk 9a22f44d853b377f3642a1200b8231e148e11f43 # timeout=10

[Pipeline] }

[Pipeline] // stage

[Pipeline] withEnv

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Build Docker Dev Image)

[Pipeline] script

[Pipeline] {

[Pipeline] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker build -t asadis7171/dev:latest .

#0 building with "default" instance using docker driver

#1 [internal] load build definition from Dockerfile

#1 transferring dockerfile: 603B done

#1 DONE 0.1s

#2 [internal] load metadata for docker.io/jenkins/jenkins:2.440.1-jdk17

#2 DONE 12.4s

#3 [internal] load .dockerignore

#3 transferring context: 2B done

#3 DONE 0.0s

#4 [1/6] FROM docker.io/jenkins/jenkins:2.440.1-jdk17@sha256:01c0b0cf789fa24253090fccea264df223b5e09b14a0ea59f0847c70bdc0f31c

#4 DONE 0.0s

#5 [2/6] RUN apt-get update && apt-get install -y lsb-release

#5 CACHED

#6 [3/6] RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc <https://download.docker.com/linux/debian/gpg>

#6 CACHED

#7 [4/6] RUN echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.asc] <https://download.docker.com/linux/debian> $(lsb\_release -cs) stable" > /etc/apt/sources.list.d/docker.list

#7 CACHED

#8 [5/6] RUN apt-get update && apt-get install -y docker-ce-cli

#8 CACHED

#9 [6/6] RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"

#9 CACHED

#10 exporting to image

#10 exporting layers done

#10 writing image sha256:a4607c273ead09a6af164931502b3d879ef073654d77dff7f7979e2e81ce02fd done

#10 naming to docker.io/asadis7171/dev:latest done

#10 DONE 0.0s

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] withEnv

[Pipeline] {

[Pipeline] withDockerRegistry

$ docker login -u asadis7171 -p \*\*\*\*\*\*\*\* <https://registry.hub.docker.com/asadis7171/dev>

WARNING! Using --password via the CLI is insecure. Use --password-stdin.

WARNING! Your password will be stored unencrypted in /var/jenkins\_home/workspace/first-jenkins-file\_dev@tmp/6de95ba7-be37-4fb9-a97f-5320dc190e3b/config.json.

Configure a credential helper to remove this warning. See

<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

[Pipeline] {

[Pipeline] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker tag asadis7171/dev:latest registry.hub.docker.com/asadis7171/dev:latest

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] isUnix

[Pipeline] withEnv

[Pipeline] {

[Pipeline] sh

+ docker push registry.hub.docker.com/asadis7171/dev:latest

The push refers to repository [registry.hub.docker.com/asadis7171/dev]

9e39961a09b0: Preparing

5bad87ece93a: Preparing

1682bb48f9d2: Preparing

46355de50845: Preparing

05406dc0f097: Preparing

b6731a0d0ccc: Preparing

245d1d0dbfe7: Preparing

72f8bf709949: Preparing

afc00755885d: Preparing

6fd428650ee5: Preparing

b9fe7e6eb007: Preparing

7a50905d05d3: Preparing

0dc9deda4ee6: Preparing

146f222a2562: Preparing

6bdac86cde7b: Preparing

094de07c3af0: Preparing

1a5fc1184c48: Preparing

b9fe7e6eb007: Waiting

7a50905d05d3: Waiting

0dc9deda4ee6: Waiting

245d1d0dbfe7: Waiting

146f222a2562: Waiting

6bdac86cde7b: Waiting

72f8bf709949: Waiting

b6731a0d0ccc: Waiting

094de07c3af0: Waiting

afc00755885d: Waiting

1a5fc1184c48: Waiting

6fd428650ee5: Waiting

46355de50845: Layer already exists

5bad87ece93a: Layer already exists

9e39961a09b0: Layer already exists

05406dc0f097: Layer already exists

1682bb48f9d2: Layer already exists

245d1d0dbfe7: Layer already exists

b6731a0d0ccc: Layer already exists

b9fe7e6eb007: Layer already exists

7a50905d05d3: Layer already exists

72f8bf709949: Layer already exists

afc00755885d: Layer already exists

6fd428650ee5: Layer already exists

094de07c3af0: Layer already exists

6bdac86cde7b: Layer already exists

1a5fc1184c48: Layer already exists

146f222a2562: Layer already exists

0dc9deda4ee6: Layer already exists

latest: digest: sha256:86a064bdcce9cdd21b8e1c27f77a23758a8d6a136a413a20fad82283ebc7d378 size: 3884

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // withDockerRegistry

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // script

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Build Docker QA Image)

Stage "Build Docker QA Image" skipped due to when conditional

[Pipeline] }

[Pipeline] // stage

[Pipeline] stage

[Pipeline] { (Declarative: Post Actions)

[Pipeline] echo

Deleting Project now !!

[Pipeline] deleteDir

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // withEnv

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

Running QA branch pipeline

2. Design a Jenkins file to execute any Terraform code, prompting the user for two inputs: Terraform apply and Terraform destroy. Depending on the provided inputs, execute the corresponding Terraform command accordingly.