



UNIVERSITY OF SCIENCE - VNUHCM  
FACULTY OF INFORMATION TECHNOLOGY

---

## Project 03 CI/CD

---

Implementing CI & CD using Git, Jenkins, and Docker

*Author*

Thanh Nhan PHU  
Minh Nhat Hung PHAN  
Hai Tuyen NGUYEN

January 2, 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Team members . . . . .	1
<b>2</b>	<b>Git deployment</b>	<b>2</b>
<b>3</b>	<b>Github deployment</b>	<b>4</b>
<b>4</b>	<b>Jenkins deployment</b>	<b>5</b>
4.1	Jenkins installation using Docker . . . . .	5
4.2	Jenkins startup customization . . . . .	5
4.3	Jenkins pipeline creation . . . . .	5
<b>5</b>	<b>Ngrok installation and deployment</b>	<b>14</b>
5.1	Ngrok installation . . . . .	14
<b>6</b>	<b>Jenkins and GitHub connection deployment</b>	<b>17</b>
6.1	Github webhook creation . . . . .	17
<b>7</b>	<b>Jenkins stage view</b>	<b>19</b>
<b>8</b>	<b>Docker deployment</b>	<b>22</b>
<b>9</b>	<b>References</b>	<b>28</b>

# **Chapter 1**

## **Introduction**

### **1.1 Team members**

- Thanh Nhan Phu - 21127382
- Minh Nhat Hung Phan - 21127614
- Hai Tuyen Nguyen - 21127474

# Chapter 2

## Git deployment

To init a git repository, we use the command **git init** to initialize a git repository in the current directory. After that, we use **git add** to add files to the staging area. Finally, we use **git commit** to commit the changes to the repository. The following figures show the process of initializing a git repository, adding files to the staging area, and committing the changes to the repository.

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test
$ git init
Initialized empty Git repository in D:/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test/.git/
```

Figure 2.1. Git initialization phase

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git commit -m 'Update'
[main (root-commit) b7d7f69] Update
 39 files changed, 18389 insertions(+)
 create mode 100644 about.html
 create mode 100644 css/bootstrap.css
 create mode 100644 css/font-awesome.min.css
 create mode 100644 css/responsive.css
 create mode 100644 css/style.css
 create mode 100644 css/style.css.map
 create mode 100644 css/style.scss
 create mode 100644 fonts/fontawesome-webfont.ttf
 create mode 100644 fonts/fontawesome-webfont.woff
 create mode 100644 fonts/fontawesome-webfont.woff2
 create mode 100644 images/about-img.jpg
 create mode 100644 images/about-img.png
 create mode 100644 images/client1.jpg
 create mode 100644 images/client2.jpg
 create mode 100644 images/favicon.png
 create mode 100644 images/hero-bg.png
 create mode 100644 images/next.png
 create mode 100644 images/prev.png
 create mode 100644 images/s1.png
 create mode 100644 images/s2.png
 create mode 100644 images/s3.png
 create mode 100644 images/slider-bg.jpg
 create mode 100644 images/slider-img.png
 create mode 100644 images/team-1.jpg
 create mode 100644 images/team-2.jpg
 create mode 100644 images/team-3.jpg
 create mode 100644 images/team-4.jpg
```

Figure 2.2. Git add phase

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git add .
warning: in the working copy of 'css/bootstrap.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'css/font-awesome.min.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'css/style.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'css/style.css.map', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'js/bootstrap.js', LF will be replaced by CRLF the next time Git touches it
```

Figure 2.3. Git commit phase

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git config user.name "21127382-21127614-21127474"

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git config credential.username "21127382-21127614-21127474"
```

Figure 2.4. Git config phase

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 45, done.
Counting objects: 100% (45/45), done.
Delta compression using up to 6 threads
Compressing objects: 100% (45/45), done.
Writing objects: 100% (45/45), 1.99 MiB | 1.11 MiB/s, done.
Total 45 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/21127382-21127614-21127474/Jenkins-CICD.git
 * [new branch]      main -> main
```

Figure 2.5. Git push phase

# Chapter 3

## Github deployment

To create a new repository on GitHub, we use the command **git remote add origin** to add a remote repository to the local repository. After that, we use **git push -u origin master** to push the local repository to the remote repository.

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

<b>Owner *</b>	<b>Repository name *</b>
 21127382-21127614-21127474	/ Jenkins-CICD

Your new repository will be created as Jenkins-CICD-.  
The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

Great repository names are short and memorable. Need inspiration? How about [laughing-eureka](#) ?

**Description (optional)**

 **Public**  
Anyone on the internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

**Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Figure 3.1. Creating a new repository

# Chapter 4

## Jenkins deployment

### 4.1 Jenkins installation using Docker

To install Jenkins using Docker, we first create a bridge network using the command:  
**docker network create jenkins**.

After that, we run a Docker container using the command:

```
docker run --name jenkins-docker -rm -detach -privileged -network jenkins -network-alias docker -env DOCKER_HOST=tcp://docker:2376 -env DOCKER_CERT_PATH=/certs/client -env DOCKER_TLS_VERIFY=1 -volume jenkins-docker-certs:/certs/client -volume jenkins-data:/var/jenkins_home -publish 2376:2376 docker:dind
```

Finally, we build a Docker image using the command:

```
docker build -t myjenkins-blueocean:1.1 .
```

And run a Docker container using the command:

```
docker run --name jenkins-blueocean -rm -detach -network jenkins -env DOCKER_HOST=tcp://docker:2376 -env DOCKER_CERT_PATH=/certs/client -env DOCKER_TLS_VERIFY=1 -volume jenkins-data:/var/jenkins_home -volume jenkins-docker-certs:/certs/client:ro -publish 8080:8080 -publish 50000:50000 myjenkins-blueocean:1.1
```

The following figures show the process of creating a bridge network, running a Docker container, building a Docker image, and running a Docker container.

```
C:\Users\Hai_Tuyen>docker network create jenkins
3067766ac201f5ea2e2f752317c282af5d3462f0e5b329da29b4f99f54f0137

C:\Users\Hai_Tuyen>
```

Figure 4.1. Docker bridge creation

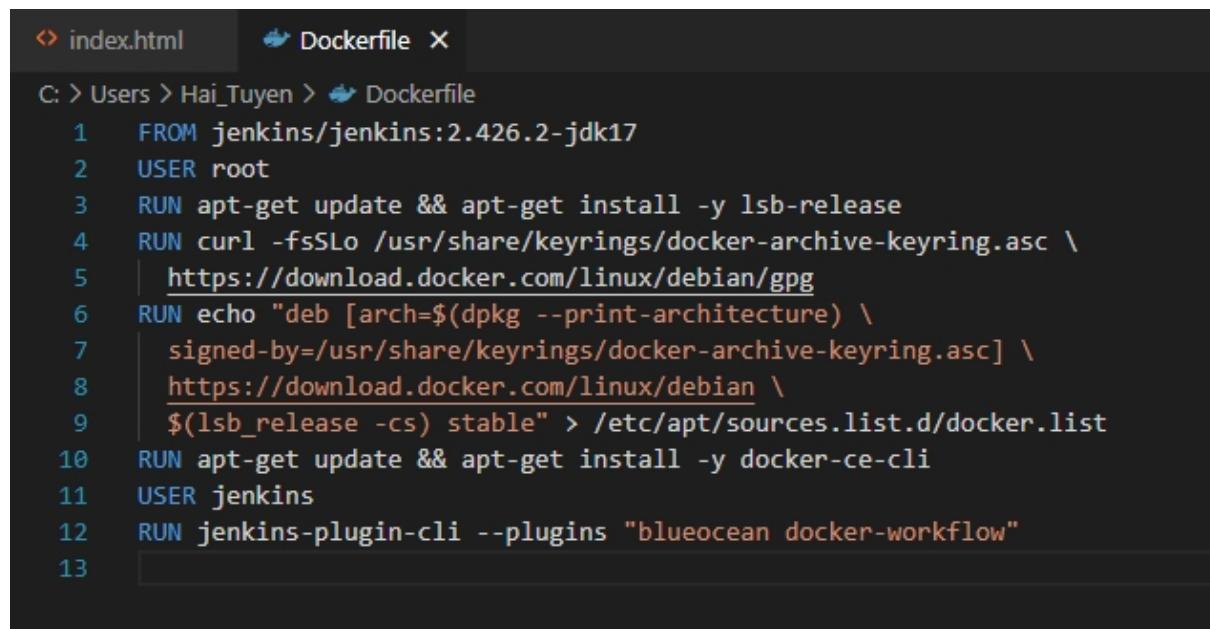
### 4.2 Jenkins startup customization

### 4.3 Jenkins pipeline creation

```
C:\Users\Hai_Tuyen>docker run --name jenkins-docker --rm --detach ^
More?   --privileged --network jenkins --network-alias docker ^
More?   --env DOCKER_TLS_CERTDIR=/certs ^
More?   --volume jenkins-docker-certs:/certs/client ^
More?   --volume jenkins-data:/var/jenkins_home ^
More?   --publish 2376:2376 ^
More?   docker:dind
Unable to find image 'docker:dind' locally
dind: Pulling from library/docker
661ff4d9561e: Pull complete
b94b6133cd82: Pull complete
4f4fb700ef54: Pull complete
de090124fc79: Pull complete
f04b53a848df: Pull complete
69c523561410: Pull complete
64a7d03e6f4c: Pull complete
ace0118efca4: Pull complete
80554da1d9db: Pull complete
361bd5a92831: Pull complete
33610589f742: Pull complete
fd848af64097: Pull complete
a4753775b651: Pull complete
1c63dcbb22ce0: Pull complete
47ce6921876b: Pull complete
Digest: sha256:1b9844d846ce3a6a6af7013e999a373112c3c0450aca49e155ae444526a2c45e
Status: Downloaded newer image for docker:dind
5721d633047c1292acaa5c6a5fdf43b6ecd5590415aa7e3961d28abc109bddcc

C:\Users\Hai_Tuyen>
```

Figure 4.2. Docker dind running



The screenshot shows a code editor window with two tabs: 'index.html' and 'Dockerfile'. The 'Dockerfile' tab is active and displays the following Dockerfile content:

```
C: > Users > Hai_Tuyen > Dockerfile X
1  FROM jenkins/jenkins:2.426.2-jdk17
2  USER root
3  RUN apt-get update && apt-get install -y lsb-release
4  RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
5    https://download.docker.com/linux/debian/gpg
6  RUN echo "deb [arch=$(dpkg --print-architecture) \
7    signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
8    https://download.docker.com/linux/debian \
9    $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
10 RUN apt-get update && apt-get install -y docker-ce-cli
11 USER jenkins
12 RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
13
```

Figure 4.3. Docker file creation

```
[+] Building 50.7s (11/11) FINISHED
=> [internal] load .dockerignore
=> > transferring context: 2B
=> [internal] load build definition from Dockerfile
=> > transferring dockerfile: 615B
=> [internal] load metadata for docker.io/jenkins/jenkins:2.426.2-jdk17
=> [auth] Jenkins/pull token for registry-1.docker.io
=> [1/6] FROM docker.io/jenkins/jenkins:2.426.2-jdk17@sha256:186a48ae298e34a21b27fe737bf0a854c3e73421ce858c4d40c403802589e23f
=> [2/6] RUN apt-get update && apt-get install -y lsb-release
=> [3/6] RUN curl -fsSlo /usr/share/keyrings/docker-archive-keyring.asc https://download.docker.com/linux/debian/gpg
=> [4/6] RUN echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.asc] https://download.docker.com/linux
=> [5/6] RUN apt-get update && apt-get install -y docker-ce-cli
=> [6/6] RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
=> exporting to image
=> exporting layers
=> writing image sha256:041404764e2a23d92b4aeed37c21a035d643c10a23477d965bc12a7b2f395647
=> naming to docker.io/library/myjenkins-blueocean:2.426.2-1
C:\Users\Hai_Tuyen>
```

Figure 4.4. Docker image build

```
C:\Users\Hai_Tuyen>
C:\Users\Hai_Tuyen>docker run --name jenkins-blueocean --restart=on-failure --detach ^
More? --network jenkins --env DOCKER_HOST=tcp://docker:2376 ^
More? --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 ^
More? --volume jenkins-data:/var/jenkins_home ^
More? --volume jenkins-docker-certs:/certs/client:ro ^
More? --publish 8080:8080 --publish 50000:50000 myjenkins-blueocean:2.426.2-1
3785d2da33dda52afab2007964fcfbfd0549f43833c286b551b519adc7f7111d

C:\Users\Hai_Tuyen>
```

Figure 4.5. Creating a container from newly build image

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
3785d2da33dd	myjenkins-blueocean:2.426.2-1	"/usr/bin/tini -- /u..."	About a minute ago	Up About a minute	0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp
5721d633047c	docker:dind	"dockerd-entrypoint..."	9 minutes ago	Up 9 minutes	2375/tcp, 0.0.0.0:2376->2376/tcp

Figure 4.6. Jenkins container running

```
C:\Users\Hai_Tuyen>docker exec -it jenkins-blueocean bash
jenkins@3785d2da33dd:/$ cat /var/jenkins_home/secrets/initialAdminPassword
154ebba0fb9e4ecc8f2316990f84114e
jenkins@3785d2da33dd:/$
```

Figure 4.7. Remembering the password to access Jenkins

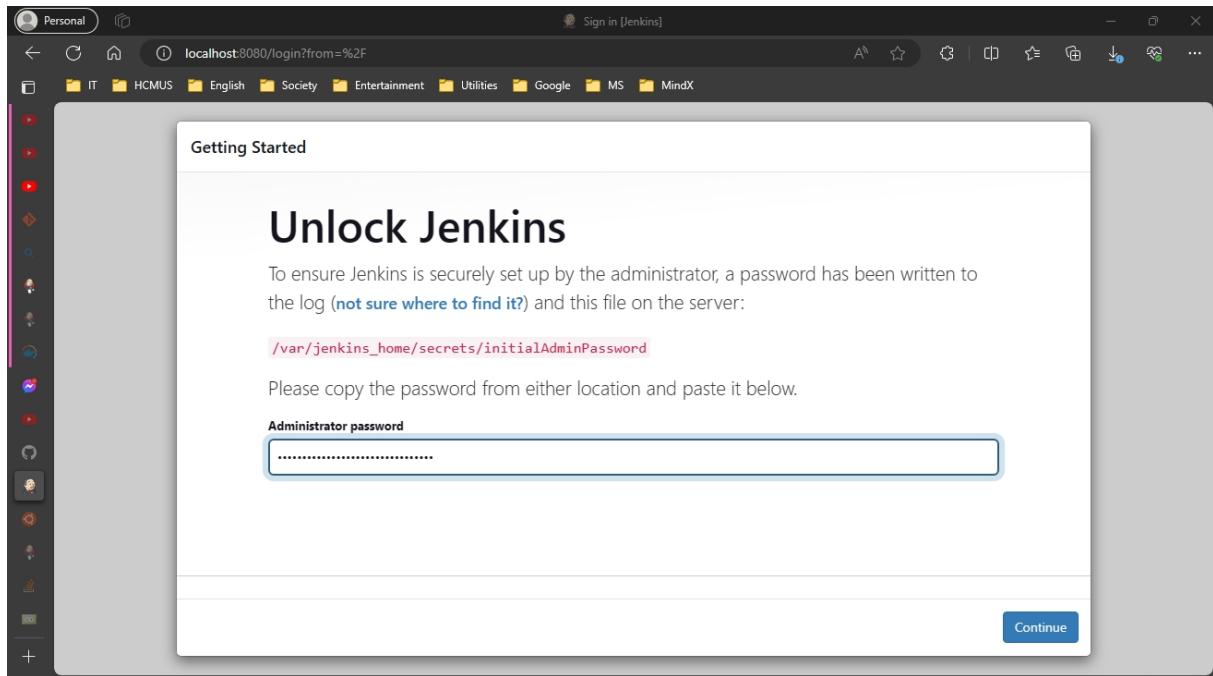


Figure 4.8. Unlocking Jenkins using the password provided in the previous step

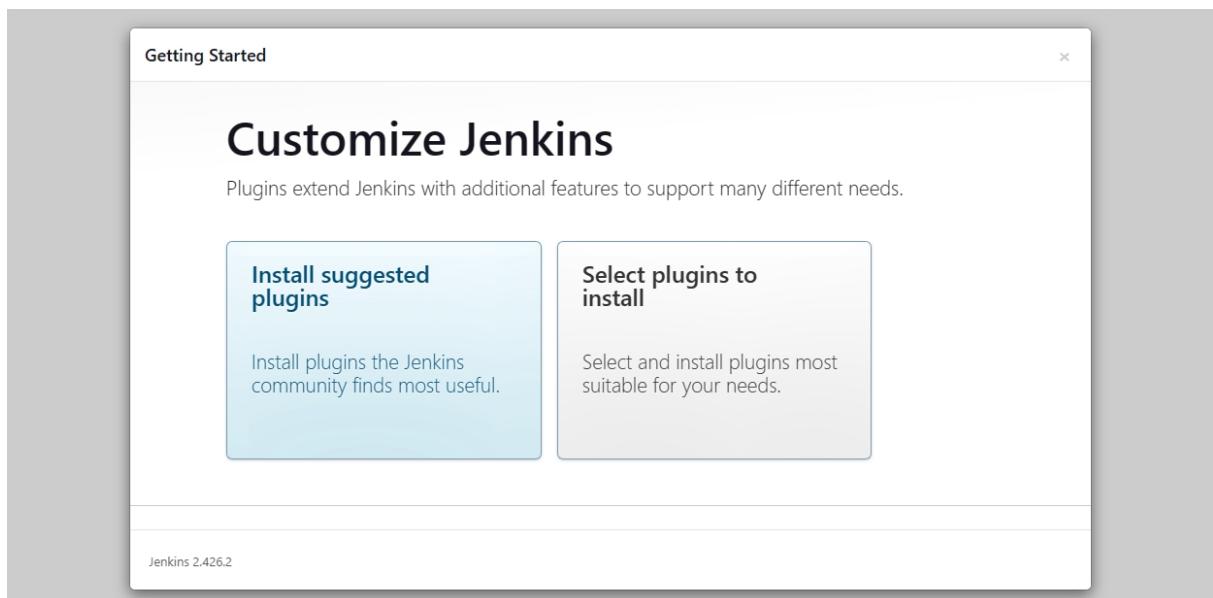


Figure 4.9. Jenkins installation and plugin selection

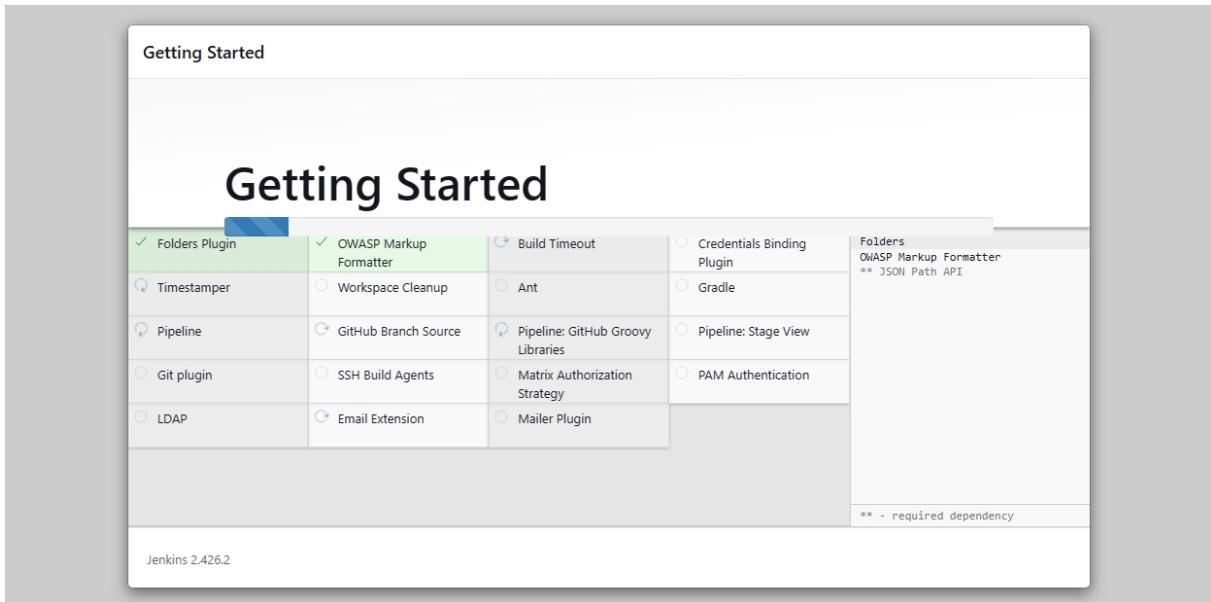


Figure 4.10. Jenkins installation in progress

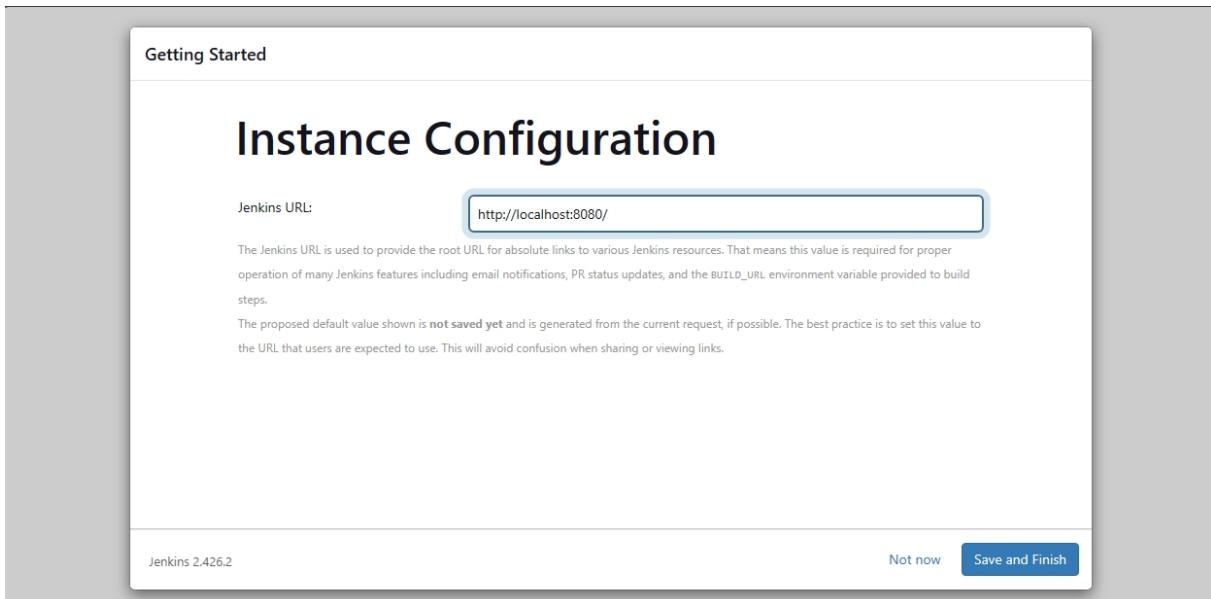


Figure 4.11. Jenkins Instance configuration URL

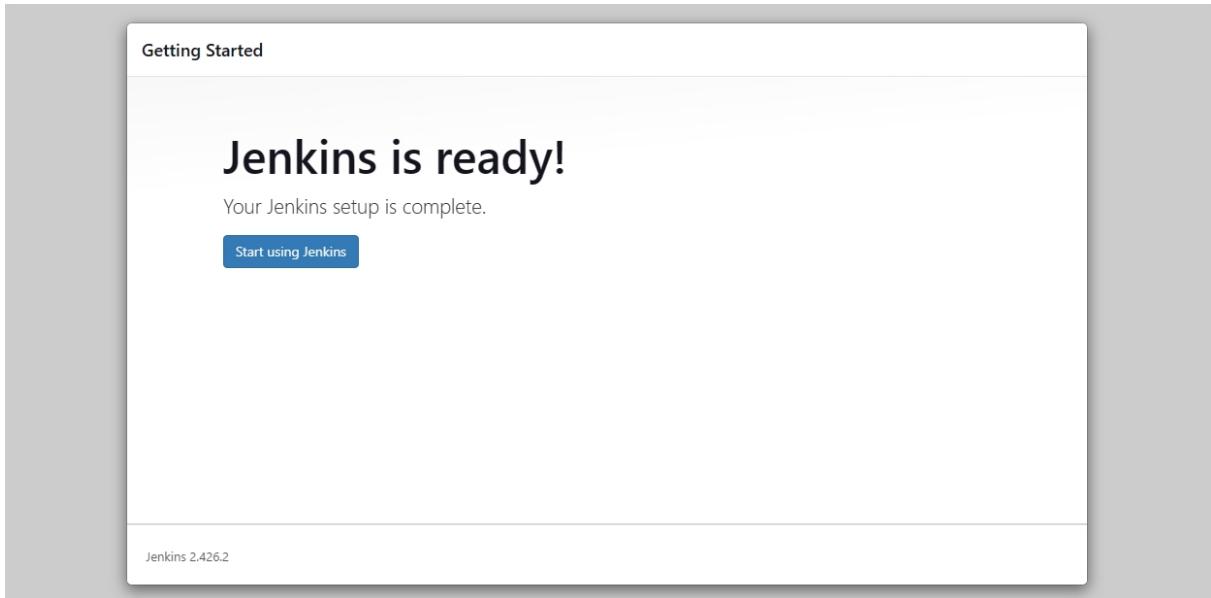


Figure 4.12. Jenkins installation completed

A screenshot of the Jenkins dashboard. The top navigation bar includes the Jenkins logo, a search bar, and user information (21127382-21127614-21127474). Below the header, there's a sidebar with links like '+ New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' button, and sections for 'Build Queue' (empty), 'Build Executor Status' (1 Idle, 2 Idle), and 'Set up a distributed build' (links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds').

Figure 4.13. An overview of Jenkins dashboard

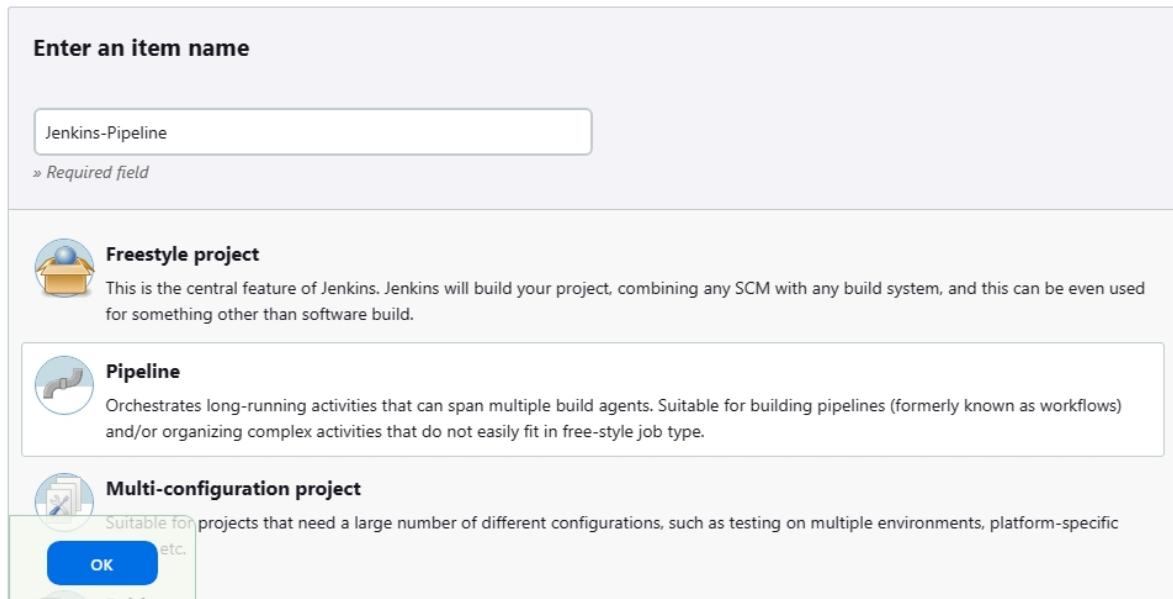


Figure 4.14. Creating a new pipeline

## Build Triggers

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

When Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository which matches the Git repository defined in SCM/Git section of this job. If they match and this option is enabled, GitHub Plugin triggers a one-time polling on GITScm. When GITScm polls GitHub, it finds that there is a change and initiates a build. The last sentence describes the behavior of Git plugin, thus the polling and initiating the build is not a part of GitHub plugin.  
(from [GitHub plugin](#))

Poll SCM ?

Quiet period ?

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

Figure 4.15. Preferring to use GitHub hook trigger for GITScm polling

## Pipeline

### Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

Credentials ?

- none -

+ Add ▾

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

Script Path ?

Lightweight checkout ?

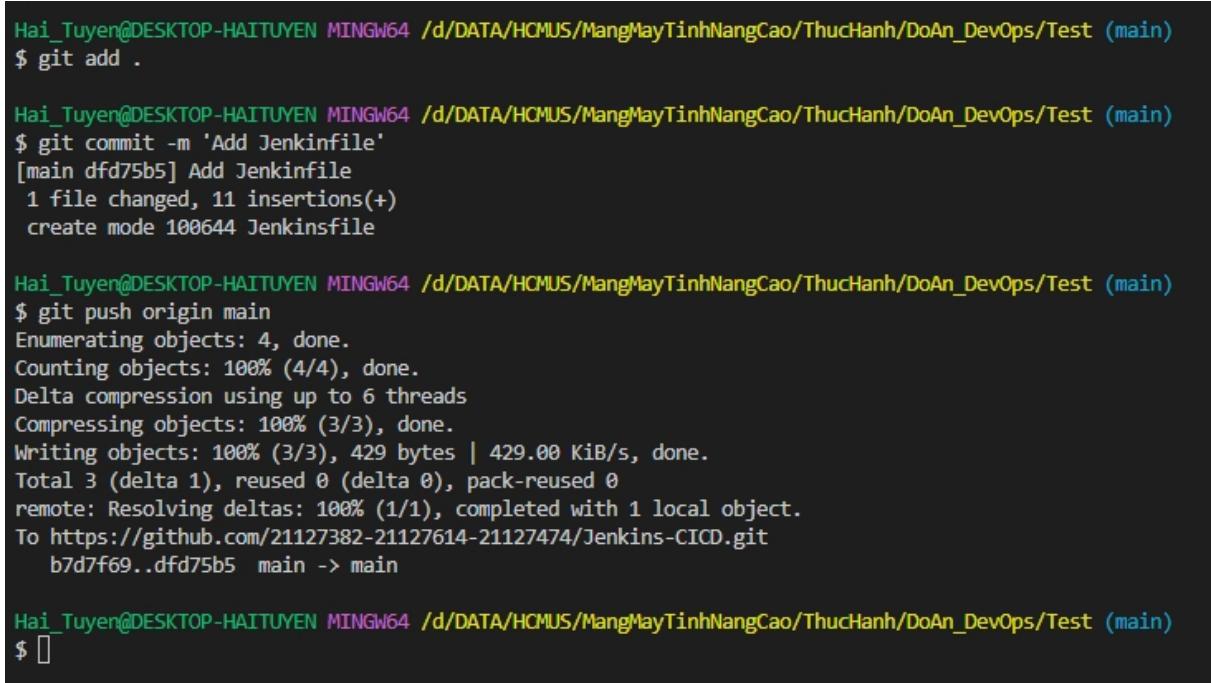
Figure 4.16. Pipeline script from GitHub repository



```
index.html Jenkinsfile

Jenkinsfile
1 pipeline {
2     agent any
3
4     stages {
5         stage('Pull') {
6             steps {
7                 git branch: 'main', url: 'https://github.com/21127382-21127614-21127474/Jenkins-CICD.git'
8             }
9         }
10    }
11 }
12 }
```

Figure 4.17. An example of a Jenkinsfile



```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git add .

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git commit -m 'Add Jenkinfile'
[main dfd75b5] Add Jenkinfile
 1 file changed, 11 insertions(+)
 create mode 100644 Jenkinsfile

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 429 bytes | 429.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/21127382-21127614-21127474/Jenkins-CICD.git
 b7d7f69..dfd75b5  main -> main

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ 
```

Figure 4.18. Pushing the Jenkinsfile to GitHub repository

# Chapter 5

## Ngrok installation and deployment

### 5.1 Ngrok installation

To install Ngrok, we first download the Ngrok executable file from the official website. After that, we add the authentication token to the default configuration file. Finally, we forward the localhost:8080 to a public URL using the command:

**./ngrok http 8080**

The following figures show the process of downloading Ngrok, adding the authentication token to the default configuration file, and forwarding the localhost:8080 to a public URL.

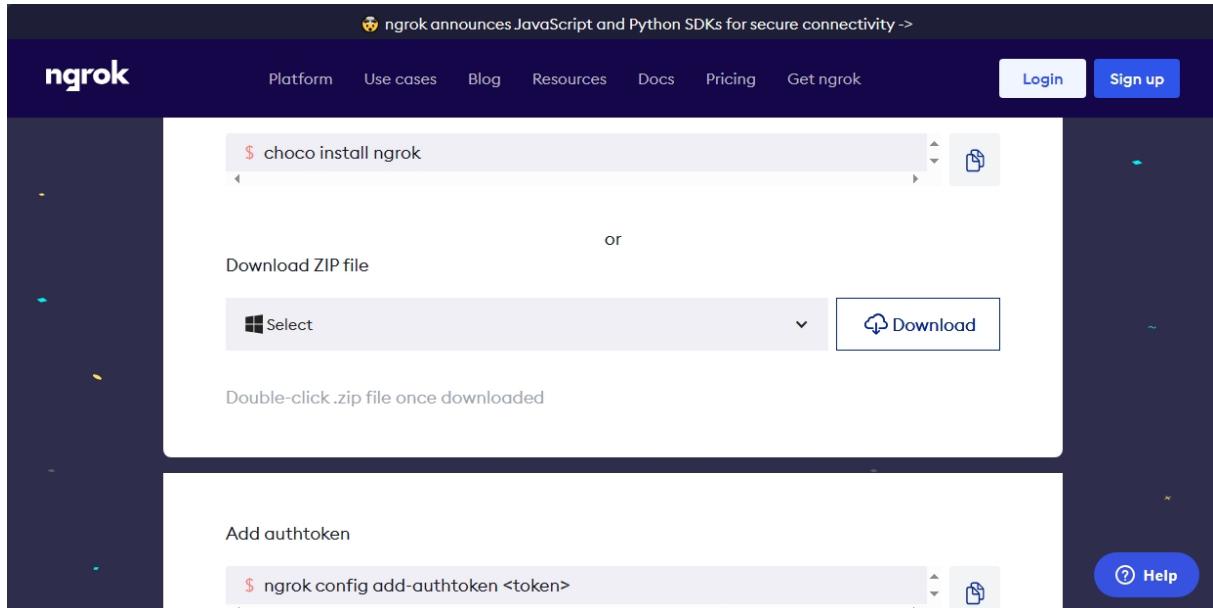


Figure 5.1. Downloading Ngrok

```
C:\Users\Hai_Tuyen>ngrok config add-authtoken 2aFfW8uTUYjlxFqPC5ImDivWi4F_4TArVJVm7CWDfk5kAiDES
Authtoken saved to configuration file: C:\Users\Hai_Tuyen\AppData\Local/ngrok/ngrok.yml

C:\Users\Hai_Tuyen>
```

Figure 5.2. Adding Ngrok authentication token to default configuration file

ID	Region	URL	Edge	Created
ep-YlNIAr	GLOBAL	<a href="https://8b93-2402-800-6374-6a50-31ad-1a8c-ef13-6a8c.ngrok-free.app">https://8b93-2402-800-6374-6a50-31ad-1a8c-ef13-6a8c.ngrok-free.app</a>	Agent Initiated	<1m ago

Figure 5.3. Forwarding localhost:8080 to a public URL

```
ngrok
Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status          online
Account                 21127382-21127614-21127474 (Plan: Free)
Version                3.5.0
Region                 Asia Pacific (ap)
Latency                57ms
Web Interface          http://127.0.0.1:4040
Forwarding             https://manatee-endless-mayfly.ngrok-free.app -> http://localhost:8080

Connections            ttl     opn      rt1      rt5      p50      p90
                        0       0       0.00    0.00    0.00    0.00
```

Figure 5.4. Ngrok forwarding localhost:8080 to a public URL

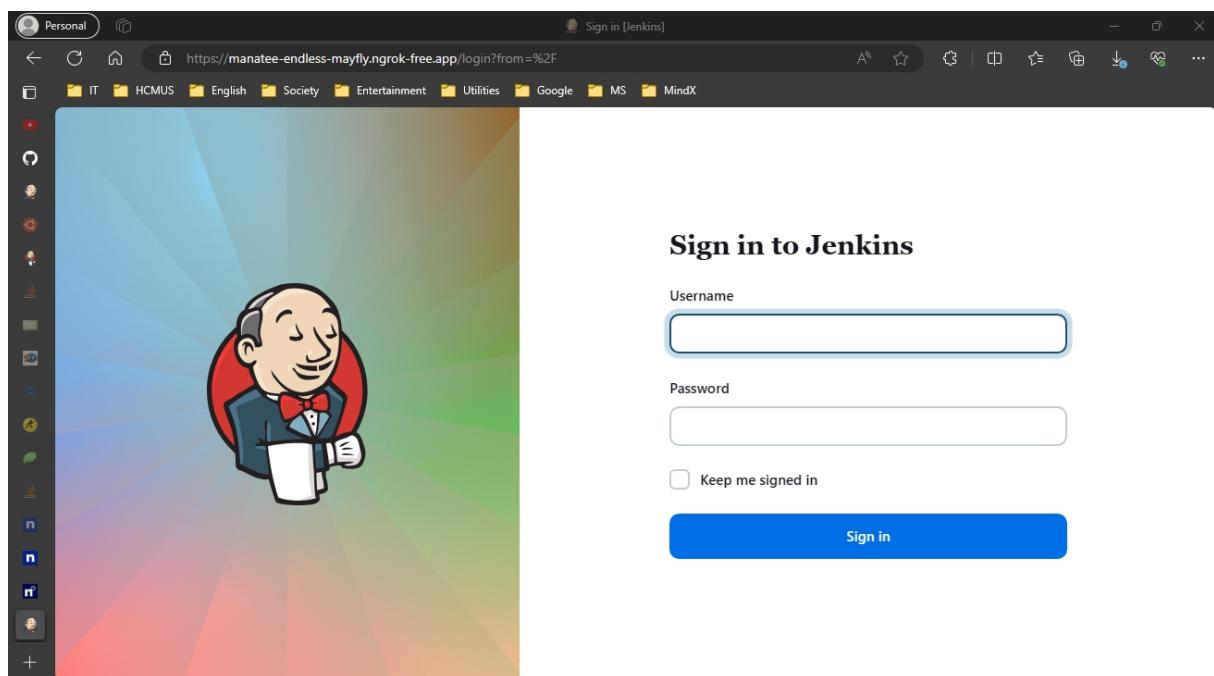


Figure 5.5. Localhost:8080 forwarded to a public URL and now we can access Jenkins from anywhere

# Chapter 6

## Jenkins and GitHub connection deployment

### 6.1 Github webhook creation

To create a webhook on GitHub, we first go to the repository settings and select Webhooks. After that, we add a new webhook and configure it to send a POST request to the Ngrok public URL. Finally, we select the events that we want to trigger the webhook. The following figures show the process of creating a webhook on GitHub.

**Webhooks / Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

`https://manatee-endless-mayfly.ngrok-free.app/github-webhook/`

**Content type**

`application/x-www-form-urlencoded`

**Secret**

[Redacted]

**SSL verification**

By default, we verify SSL certificates when delivering payloads.

**Enable SSL verification**    **Disable (not recommended)**

**Which events would you like to trigger this webhook?**

Just the push event.  
 Send me everything.  
 Let me select individual events.

**Active**  
We will deliver event details when this hook is triggered.

**Add webhook**

Figure 6.1. Adding a webhook to GitHub repository

# Chapter 7

## Jenkins stage view

By pushing a new file to the GitHub repository, we can see that the Jenkins pipeline is triggered and the stage view is updated. The following figures show the process of pushing a new file to the GitHub repository and the stage view is updated.

```
C:\Users\Hai_Tuyen>docker container exec -it jenkins-blueocean bash
jenkins@3785d2da33dd:/$ cd /var/jenkins_home/workspace/Jenkins-Pipeline
jenkins@3785d2da33dd:~/workspace/Jenkins-Pipeline$ ls
Jenkinsfile  about.html  css  fonts  images  index.html  js  service.html  team.html  why.html
jenkins@3785d2da33dd:~/workspace/Jenkins-Pipeline$
```

Figure 7.1. Listing all files in the current Jenkins workspace

## Stage View



Figure 7.2. Jenkins stage view with SCM and Build stages

```
Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git add .

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git commit -m 'Add README.txt'
[main 88b44f8] Add README.txt
 1 file changed, 3 insertions(+)
 create mode 100644 README.txt

Hai_Tuyen@DESKTOP-HAITUYEN MINGW64 /d/DATA/HCMUS/MangMayTinhNangCao/ThucHanh/DoAn_DevOps/Test (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 447 bytes | 447.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/21127382-21127614-21127474/Jenkins-CICD.git
 dfd75b5..88b44f8 main -> main
```

Figure 7.3. Pushing a new readme file to GitHub repository

```
C:\Users\Hai_Tuyen>docker container exec -it jenkins-blueocean bash
jenkins@3785d2da33dd:/ $ cd /var/jenkins_home/workspace/Jenkins-Pipeline
jenkins@3785d2da33dd:~/workspace/Jenkins-Pipeline$ ls
Jenkinsfile about.html css fonts images index.html js service.html team.html why.html
jenkins@3785d2da33dd:~/workspace/Jenkins-Pipeline$ ls
Jenkinsfile README.txt about.html css fonts images index.html js service.html team.html why.html
jenkins@3785d2da33dd:~/workspace/Jenkins-Pipeline$
```

Figure 7.4. A new ReadMe file is now in the Jenkins workspace

## Stage View



Figure 7.5. A new stage is added to the Jenkins stage view

# Chapter 8

## Docker deployment

An access token is added to Docker Hub to allow Jenkins to automatically build a new image, deploy and cleanup the image. A new user/password credential is added to Jenkins to allow Jenkins to push the newly built image to Docker Hub. The following figures show the process of adding an access token to Docker Hub and a new user/password credential to Jenkins.

The screenshot shows a web-based form for creating a new access token. At the top, the title "New Access Token" is displayed. Below it, a descriptive text states: "A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)". The form includes fields for "Access Token Description \*", which is currently empty, and "Access permissions", which is set to "Read, Write, Delete". A note below the permissions dropdown explains: "Read, Write, Delete tokens allow you to manage your repositories." At the bottom right of the form are two buttons: "Cancel" and "Generate".

Figure 8.1. Adding a new access token on Docker Hub

Access Tokens						<a href="#">New Access Token</a>
	Description	Source	Scope	Last Used	Created	
<input type="checkbox"/>	Generated by Docker De...	AUTO-GENERATED	Read, Write, Delete	Dec 30, 2023 23:36:38	Dec 30, 2023 13:08:11	<a href="#">⋮</a>
<input type="checkbox"/>	JenkinsPipeline	MANUAL	Read, Write, Delete	Never	Dec 31, 2023 01:49:33	<a href="#">⋮</a>

Figure 8.2. A new access token is added to Docker Hub

## New credentials

Kind

Username with password

Scope ?

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

Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

211273822112761421127474

Treat username as secret ?

Password ?

.....

ID ?

dockerhub

Description ?

Figure 8.3. Adding a new user/password credential to Jenkins

Global credentials (unrestricted)				<a href="#">+ Add Credentials</a>
ID	Name	Kind	Description	
	dockerhub	dockerhub	Username with password	dockerhub 
Icon:	S M L			

Figure 8.4. A new user/password credential for Docker Hub is added to Jenkins

```
Jenkinsfile
1 pipeline {
2     agent any
3
4     environment {
5         dockerImage = ''
6         registry = '211273822112761421127474/crypto-site-nginx'
7         registryCredential = 'dockerhub'
8     }
9     stages {
10         stage('Pull') {
11             steps {
12                 git branch: 'main', url: 'https://github.com/21127382-21127614-21127474/Jenkins-CI_CD.git'
13             }
14         }
15         stage('Build') {
16             steps {
17                 script {
18                     dockerImage = docker.build registry
19                 }
20             }
21         }
22         stage('Push') {
23             steps {
24                 script {
25                     docker.withRegistry( '', registryCredential ) {
26                         dockerImage.push()
27                     }
28                 }
29             }
30         }
31     }
32 }
```

Figure 8.5. Pushing the new Jenkinsfile to GitHub repository

## Stage View

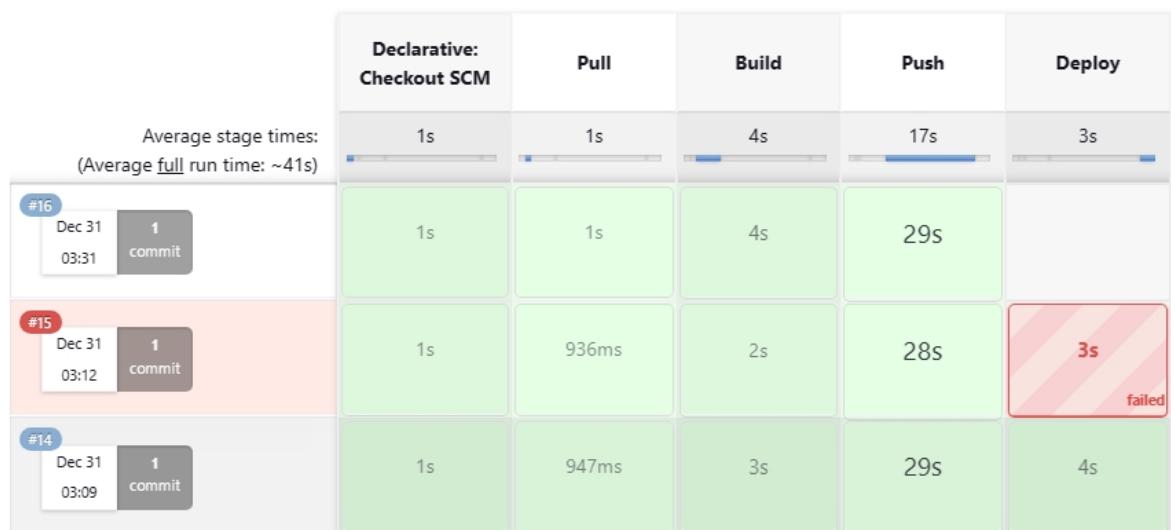


Figure 8.6. New stages are added to the Jenkins stage view

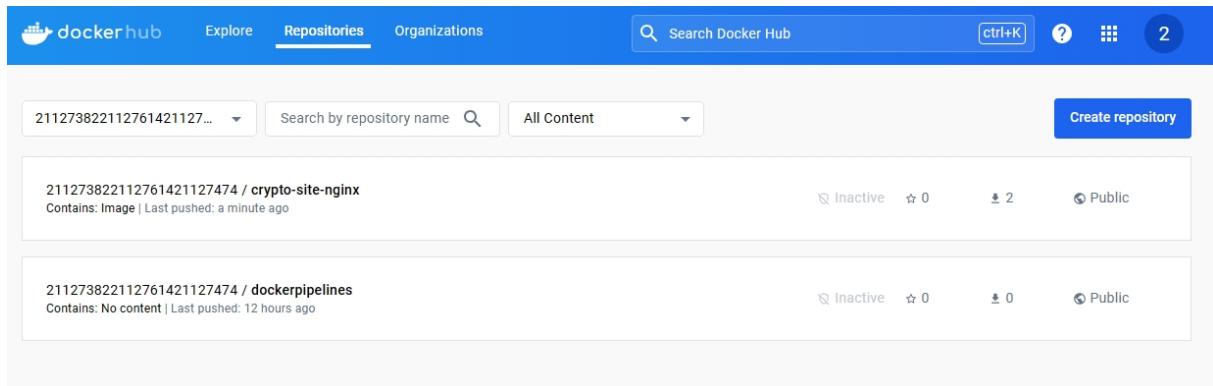


Figure 8.7. Docker Hub is now automatically building a new image

```
stage('Deploy') {
    steps {
        withDockerRegistry(credentialsId: 'dockerhub', url: 'https://index.docker.io/v1/') {
            sh 'docker run -d -p 4000:80 --name my-nginx-container 211273822112761421127474/crypto-site-nginx'
        }
    }
}
```

Figure 8.8. Adjusting the Jenkinsfile to deploy the newly built image

## Stage View

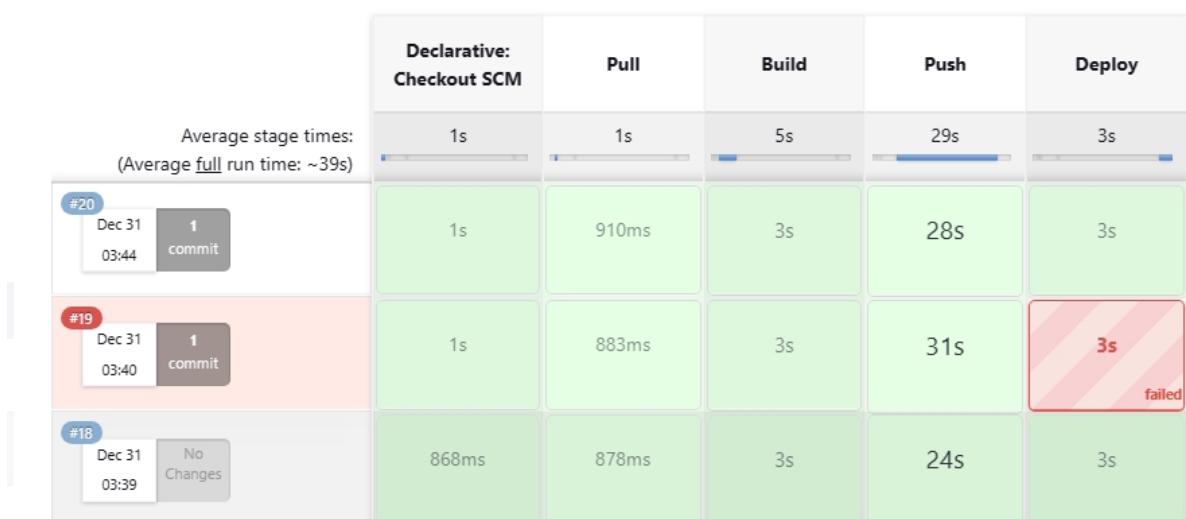


Figure 8.9. Result after pushing the new Jenkinsfile to GitHub repository

```
root@424dafe9ff20:/# cd /usr/share/nginx/html/
root@424dafe9ff20:/usr/share/nginx/html# ls
50x.html Jenkinsfile about.html fonts index.html service.html why.html
Dockerfile README.txt css images js team.html
root@424dafe9ff20:/usr/share/nginx/html# cd README.txt
bash: cd: README.txt: Not a directory
root@424dafe9ff20:/usr/share/nginx/html# cat README.txt
21127382 - Phù Thành Nhân - ptnhan21@clc.fitus.edu.vn - 0913744887
21127614 - Phan Minh Nhật Hưng - pmnhung21@clc.fitus.edu.vn - 0775591708
21127474 - Nguyễn Hải Tuyên - nhtuyen@clc.fitus.edu.vn - 0971172165
root@424dafe9ff20:/usr/share/nginx/html#
```

Figure 8.10. Before pushing the new Jenkinsfile to GitHub repository

```

index.html Jenkinsfile Dockerfile README.txt
① README.txt
1 21127382 - Phù Thành Nhân - ptnhan21@clc.fitus.edu.vn - 0913744887
2 21127614 - Phan Minh Nhật Hưng - pmnhung21@clc.fitus.edu.vn - 0775591708
3 21127474 - Nguyễn Hải Tuyên - nhtuyen@clc.fitus.edu.vn - 0971172165
4 21127474 - Nguyễn Hải Tuyên - nhtuyen@clc.fitus.edu.vn - 0971172165

```

Figure 8.11. Before pushing the new Jenkinsfile to GitHub repository

```

C:\Users\Hai_Tuyen>docker container exec -it jenkins-blueocean bash
jenkins@3785d2da33dd:/$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
 NAMES
424daf9ff20 fe77b1100c9c "/docker-entrypoint...." 8 hours ago Up 8 hours 0.0.0.0:4000->
80/tcp my-nginx-container
jenkins@3785d2da33dd:/$ docker ps
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
e7167dc14263 211273822112761421127474/crypto-site-nginx "/docker-entrypoint...." 21 seconds
ago Up 20 seconds 0.0.0.0:4000->80/tcp my-nginx-container
jenkins@3785d2da33dd:$

```

Figure 8.12. After pushing the new Jenkinsfile to GitHub repository

```

jenkins@3785d2da33dd:$ docker ps
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
e7167dc14263 211273822112761421127474/crypto-site-nginx "/docker-entrypoint...." 21 seconds
ago Up 20 seconds 0.0.0.0:4000->80/tcp my-nginx-container
jenkins@3785d2da33dd:$ docker container exec -it my-nginx-container bash
root@e7167dc14263:/# cd usr/share/nginx/html/
root@e7167dc14263:/usr/share/nginx/html# ls
50x.html Jenkinsfile about.html fonts index.html service.html why.html
Dockerfile README.txt css images js team.html
root@e7167dc14263:/usr/share/nginx/html# cat README.txt
21127382 - Phù Thành Nhân - ptnhan21@clc.fitus.edu.vn - 0913744887
21127614 - Phan Minh Nhật Hưng - pmnhung21@clc.fitus.edu.vn - 0775591708
21127474 - Nguyễn Hải Tuyên - nhtuyen@clc.fitus.edu.vn - 0971172165
21127474 - Nguyễn Hải Tuyên - nhtuyen@clc.fitus.edu.vn - 0971172165
root@e7167dc14263:/usr/share/nginx/html#

```

Figure 8.13. Checking the content of the ReadMe file

## Stage View



Figure 8.14. The pipeline is now successfully deployed

## Stage View

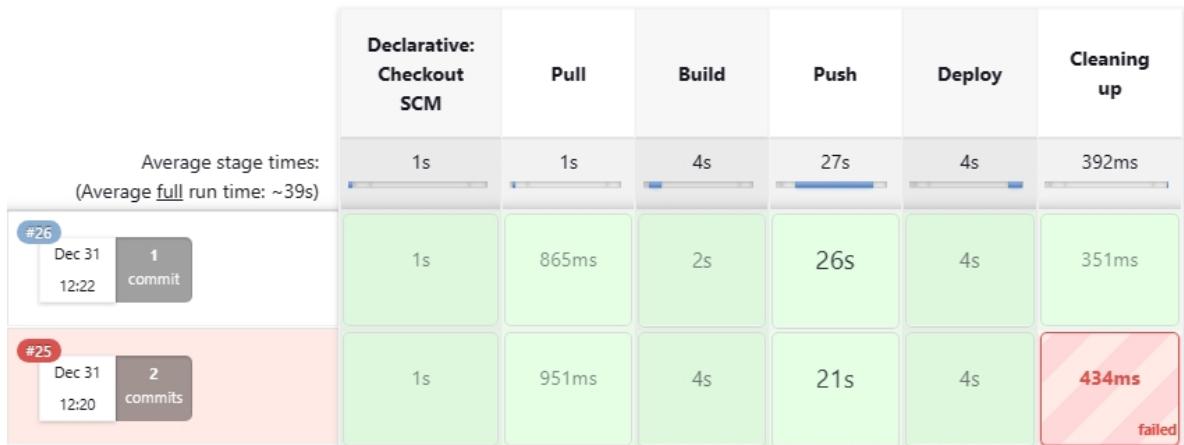


Figure 8.15. Adding a new stage to clean up the workspace

```
+ docker rmi 211273822112761421127474/crypto-site-nginx:26
Untagged: 211273822112761421127474/crypto-site-nginx:26
Untagged: 211273822112761421127474/crypto-site-
nginx@sha256:37875b5a8663f8a71adefea500d50773231689626d7e4d324dfa470240eb11c
Deleted: sha256:a24fff12b671283d2645ce54cc186091af01d320c4928d27da849188a36f597f
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Figure 8.16. Checking the log of the cleanup stage

# Chapter 9

## References

[1], [2], [3], [4], [5]

# Bibliography

- [1] GitHub. “CI/CD pipeline with GitHub”. 2017. URL: <https://resources.github.com/devops/fundamentals/ci-cd/ci-cd-with-github-and-jenkins/> (page 28).
- [2] Jenkins. “Jenkins Documentation”. 2021. URL: <https://www.jenkins.io/doc/> (page 28).
- [3] Docker. “Docker Documentation”. 2021. URL: <https://docs.docker.com/> (page 28).
- [4] GitHub. “GitHub Documentation”. 2021. URL: <https://docs.github.com/en> (page 28).
- [5] SysopsMicro. “CI/CD pipeline”. 2021. URL: <https://medium.com/the-programmer/building-ci-cd-pipeline-with-jenkins-kubernetes-github-part-2-cbb6c366aa41> (page 28).