

# DevOps

## #Create GitHub repo

### STEP 1:

"Add and commit all files to the remote repository."

```
student@cacc1-6:~/kubernetes-days5/day5$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/student/kubernetes-days5/day5/.git/
student@cacc1-6:~/kubernetes-days5/day5$ git branch -M main
student@cacc1-6:~/kubernetes-days5/day5$ git add .
student@cacc1-6:~/kubernetes-days5/day5$ git commit "Initial commit"
error: pathspec 'Initial commit' did not match any file(s) known to git
student@cacc1-6:~/kubernetes-days5/day5$ git commit -m "Initial commit"
[master (root-commit) 262f6c0] Initial commit
13 files changed, 177 insertions(+)
create mode 100644 README.md
create mode 100644 backend/app.py
create mode 100644 backend/dockerfile
create mode 100644 backend/products.csv
create mode 100644 backend/requirements.txt
create mode 100644 commands-to-stop-instances
create mode 100644 frontend/dockerfile
create mode 100644 frontend/index.html
create mode 100644 k8s/allow-all.yaml
create mode 100644 k8s/backend-deployment.yaml
create mode 100644 k8s/configmap.yaml
create mode 100644 k8s/frontend-deployment.yaml
create mode 100644 k8s/service.yaml
student@cacc1-6:~/kubernetes-days5/day5$ git remote add origin https://github.com/iamJABASTIN/DevOps-day-5.git
```

### STEP 2: Push local changes to the remote repository

```
student@cacc1-6:~/kubernetes-days5/day5$ git push origin main
Username for 'https://github.com': iamJABASTIN
Password for 'https://iamJABASTIN@github.com':
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 16 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (18/18), 2.71 MiB | 2.71 MiB/s, done.
Total 18 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/iamJABASTIN/DevOps-day-5.git
 * [new branch]      main -> main
```

### STEP 3: Create a Jenkins config file **Jenkinsfile**

A **Jenkinsfile** is a script that defines a **Jenkins pipeline**. It outlines how Jenkins should build, test, and deploy your code. Instead of manually configuring Jenkins jobs through the web interface, you can store the entire pipeline as code in your repository

```
student@cacc1-6:~/kubernetes-days5/day5$ nano Jenkinsfile
student@cacc1-6:~/kubernetes-days5/day5$ cat Jenkinsfile
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = "iamjabastin/docker-app:latest" // Change this to your registry
        CONTAINER_NAME = "docker-running-app"
        REGISTRY_CREDENTIALS = "docker-hub-jabastin" // Jenkins credentials ID
    }

    stages {
        stage('Checkout Code') {
            steps {
                withCredentials([UsernamePassword(credentialsId: 'github-jabastin', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
                    git url: "https://$GIT_USER:$GIT_TOKEN@github.com/iamJABASTIN/DevOps-day-5.git", branch: 'main'
                }
            }
        }

        stage('Build Docker Image') {
            steps {

```

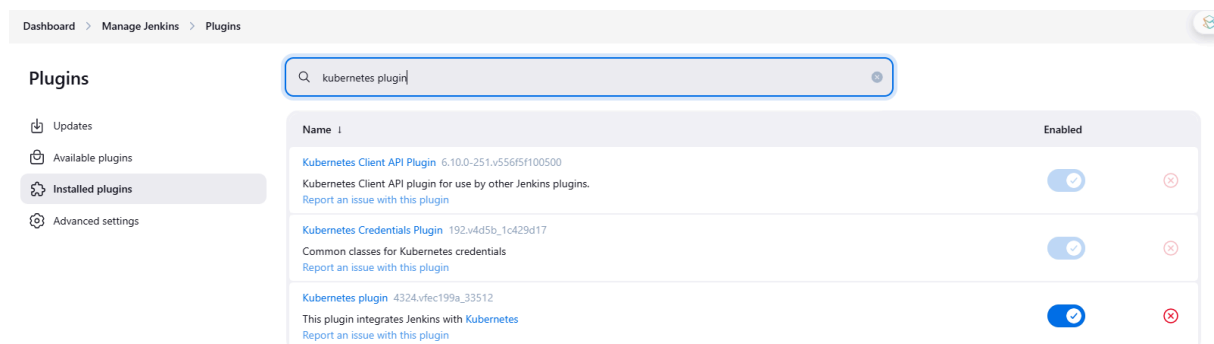
## STEP 4: Enable and Start the Jenkins

```
student@mcacc1-6:~/kubernetes-days/day5$ sudo systemctl enable jenkins
[sudo] password for student:
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
student@mcacc1-6:~/kubernetes-days/day5$ sudo systemctl start jenkins
student@mcacc1-6:~/kubernetes-days/day5$ |
```

Vist the Jenkins in Port 8080

## STEP 5: JENKINS PIPELINE

1. Create a Jenkins Pipeline
2. Push the Jenkins file to the Remote Repo
3. Install **Kubernetes plugin** in Jenkins

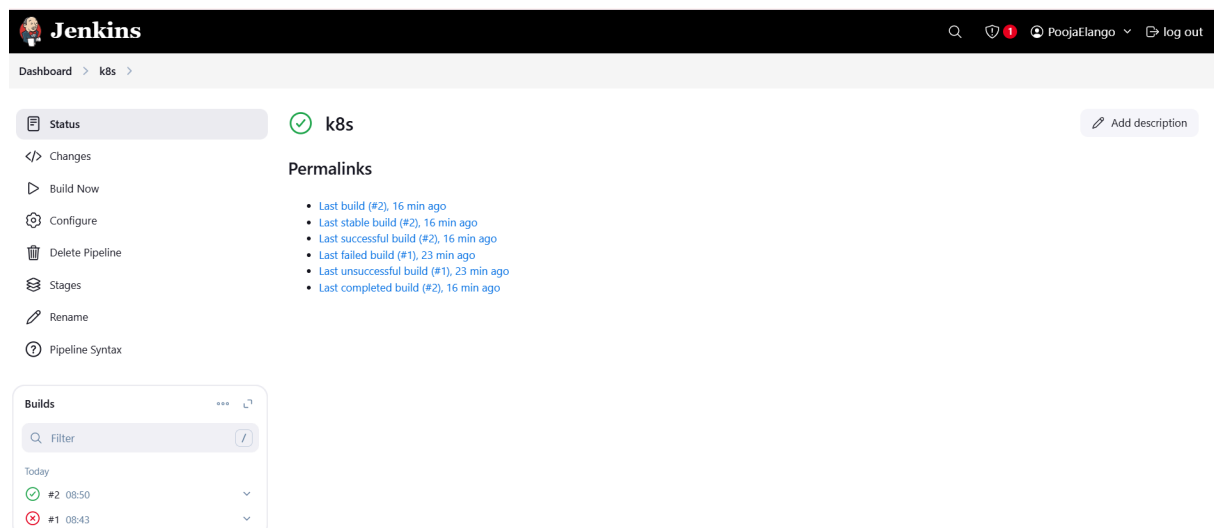


The screenshot shows the Jenkins 'Plugins' page. The breadcrumb navigation is 'Dashboard > Manage Jenkins > Plugins'. A search bar contains 'kubernetes plugin'. On the left sidebar, 'Installed plugins' is selected. The main table lists three installed plugins, all with 'Enabled' status (blue toggle switch and green checkmark icon).

Name	Enabled
<b>Kubernetes Client API Plugin</b> 6.10.0-251.v556f5f100500 Kubernetes Client API plugin for use by other Jenkins plugins. <a href="#">Report an issue with this plugin</a>	Enabled
<b>Kubernetes Credentials Plugin</b> 192.v4d5b_1c429d17 Common classes for Kubernetes credentials <a href="#">Report an issue with this plugin</a>	Enabled
<b>Kubernetes plugin</b> 4324.vfec199a_33512 This plugin integrates Jenkins with Kubernetes <a href="#">Report an issue with this plugin</a>	Enabled

## STEP 6:

Build the Jenkins

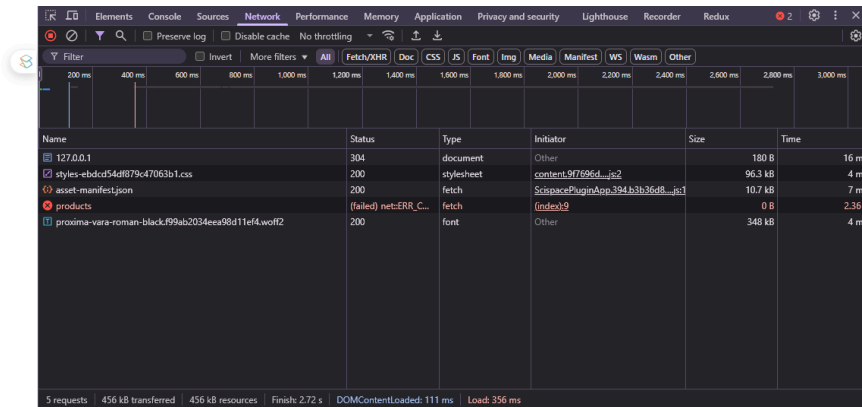


The screenshot shows the Jenkins Pipeline view for a pipeline named 'k8s'. The top bar includes the Jenkins logo, search, shield, user 'PoojaElango', and 'log out'. The breadcrumb is 'Dashboard > k8s'. On the left, the 'Status' tab is selected, showing a green checkmark icon and the name 'k8s'. Below this is a list of actions: 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Stages', 'Rename', and 'Pipeline Syntax'. The 'Permalinks' section lists several links for the last build (#2), last stable build (#2), last successful build (#2), last failed build (#1), last unsuccessful build (#1), and last completed build (#2). At the bottom, the 'Builds' section shows a filter bar and a list of builds for 'Today': build #2 at 08:50 (successful, green checkmark) and build #1 at 08:43 (failed, red X).



# Welcome to Our Store

Loading...



**Note:** We expect this kind of output because we are running this frontend on localhost.

— COMPLETED —