

Name: B Iniyan [24MCR035]

Class: I – MCA – “A”

# DEVOPS TRAINING

## DAY 5 CONFIGURING PIPELINE

Step 1:

Create github repository kubernetes and push the cd kubernetes files as frontend ,backend and k8s

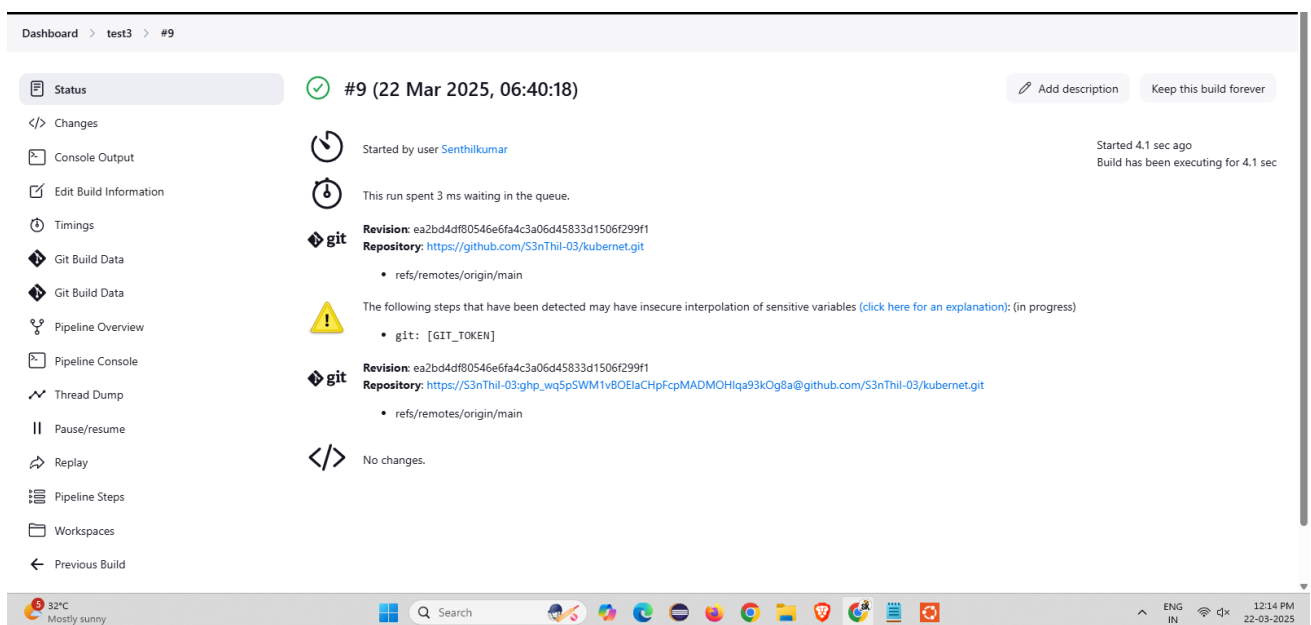
Step 2:

create Jenkinsfile and add the following code and push into the github repo kubernetes

```
Select student@CTS-6: ~/Kubernetes
id https://github.com/S3nThil-03/kubernetes
$ curl -s https://raw.githubusercontent.com/S3nThil-03/kubernetes/main/.devcontainer/devcontainer.json | jq -r '.pipeline' | xargs -I {} bash -c 'cat {}'
{
  "name": "Kubernetes",
  "description": "Kubernetes",
  "image": "mcr.microsoft.com/devcontainers/python:3.10",
  "pipeline": {
    "steps": [
      {
        "name": "Checkout Code",
        "steps": [
          {
            "name": "Checkout Code",
            "command": "git clone https://github.com/S3nThil-03/kubernetes.git"
          }
        ]
      },
      {
        "name": "Build Backend Docker Image",
        "steps": [
          {
            "name": "Build Backend Docker Image",
            "command": "docker build -t s3nthil03/docker-backend:latest -f backend/Dockerfile ."
          }
        ]
      },
      {
        "name": "Build Frontend Docker Image",
        "steps": [
          {
            "name": "Build Frontend Docker Image",
            "command": "docker build -t s3nthil03/docker-frontend:latest -f frontend/Dockerfile ."
          }
        ]
      },
      {
        "name": "Login to Docker Registry",
        "steps": [
          {
            "name": "Login to Docker Registry",
            "command": "docker login --username s3nthil03 --password $(cat /dev/urandom | tr -dc 'a-z0-9' | fold -w 40 | tr -d '\\n' | xargs echo)"
          }
        ]
      },
      {
        "name": "Push Docker Images",
        "steps": [
          {
            "name": "Push Docker Images",
            "command": "docker push s3nthil03/docker-backend:latest"
          },
          {
            "name": "Push Docker Images",
            "command": "docker push s3nthil03/docker-frontend:latest"
          }
        ]
      },
      {
        "name": "Stop & Remove Existing Containers",
        "steps": [
          {
            "name": "Stop & Remove Existing Containers",
            "command": "docker rm -f $(docker ps -q)"
          }
        ]
      }
    ]
  }
}
```

### Step 3:

Open Jenkins create a item in pipeline and click ok and go to configure add the repo url and credentials and click build



### Step 4 :

Open console output and check build is complete or not.

The screenshot shows the Jenkins 'Console Output' for a pipeline named 'test3' (build #9). The output details the process of obtaining the Jenkinsfile from a GitHub repository, checking out the code, and running a series of Git commands to fetch and checkout the latest version of the Kubernetes installation script. The pipeline concludes with a successful checkout of the revision 'ea2bd4df80546e6fa4c3a06d45833d1506f299f1'.

```
Started by user Senthilkumar
Obtained Jenkinsfile from git https://github.com/S3nThil-03/kubernetes.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/test3
[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 github-senthil
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/test3/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/S3nThil-03/kubernetes.git # timeout=10
Fetching upstream changes from https://github.com/S3nThil-03/kubernetes.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/S3nThil-03/kubernetes.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision ea2bd4df80546e6fa4c3a06d45833d1506f299f1 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f ea2bd4df80546e6fa4c3a06d45833d1506f299f1 # timeout=10
```

## Step 5 :

Go to dashboard > manage Jenkins > plugins and install the Kubernetes once it all download success will shown.

The screenshot displays the 'Manage Jenkins > Plugins' interface. On the left, the 'Download progress' tab is selected. The main area shows a list of plugins being installed, all of which have a green checkmark indicating success. The plugins listed are: Kubernetes Client API, Authentication Tokens API, Kubernetes Credentials, Kubernetes, and Loading plugin extensions. Below the list, there is a link to 'Go back to the top page' and a checkbox for 'Restart Jenkins when installation is complete and no jobs are running'.

Plugin	Status
Kubernetes Client API	Success
Authentication Tokens API	Success
Kubernetes Credentials	Success
Kubernetes	Success
Loading plugin extensions	Success