

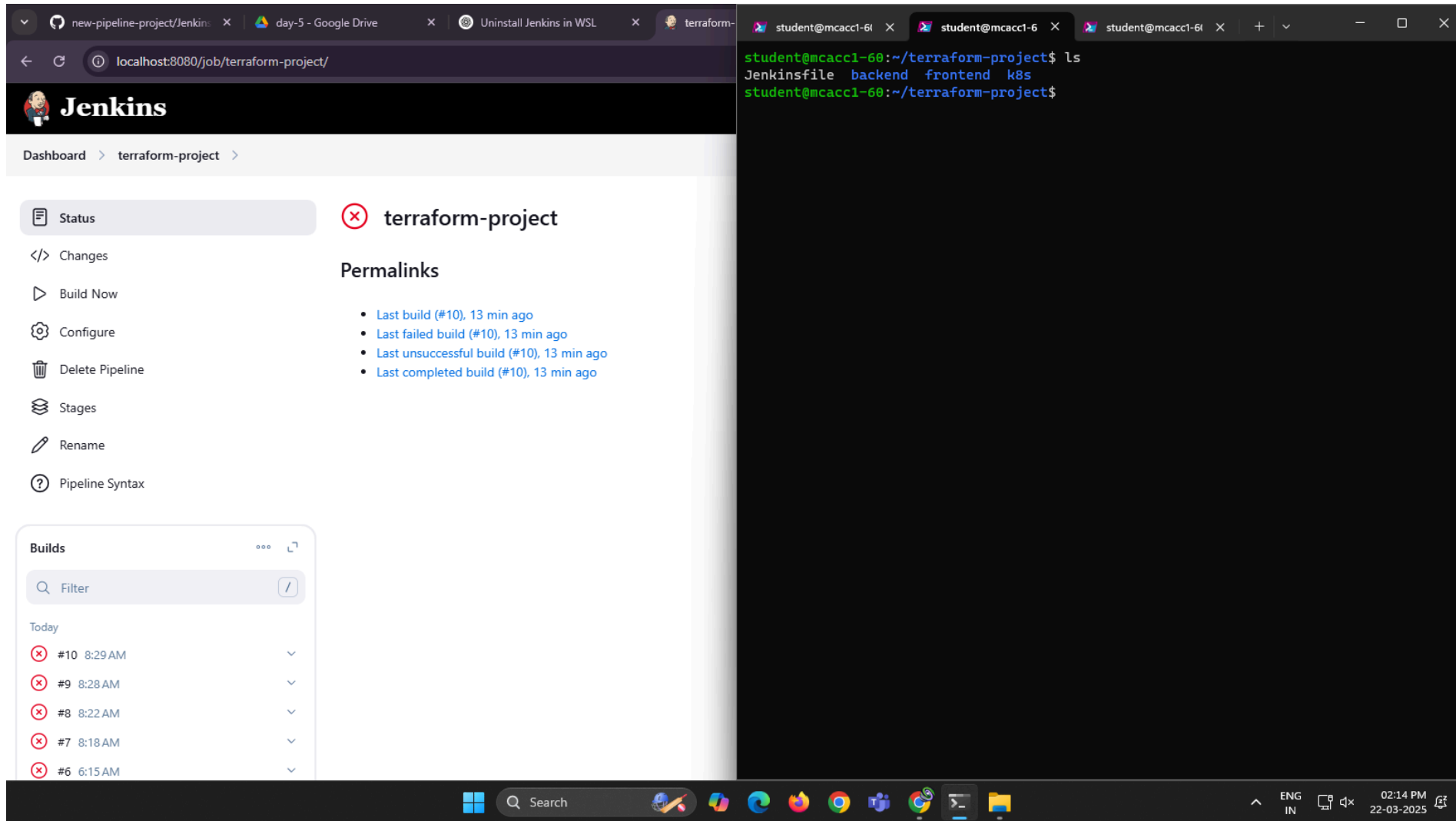
# Setting up minikube and kubectl and deploying a project in kubernetes

Day - 3 and 4

24MCR029

Harikrishnan N

# Project Structure



The image shows a Jenkins dashboard on the left and a terminal window on the right, both within a Windows environment.

**Jenkins Dashboard:**

- URL: `localhost:8080/job/terraform-project/`
- Page Title: **terraform-project** (with a red 'x' icon)
- Left Sidebar:
  - Status
  - Changes
  - Build Now
  - Configure
  - Delete Pipeline
  - Stages
  - Rename
  - Pipeline Syntax
- Main Content:
  - Permalinks:
    - Last build (#10), 13 min ago
    - Last failed build (#10), 13 min ago
    - Last unsuccessful build (#10), 13 min ago
    - Last completed build (#10), 13 min ago
- Builds List (Today):
  - #10 8:29 AM
  - #9 8:28 AM
  - #8 8:22 AM
  - #7 8:18 AM
  - #6 6:15 AM

**Terminal Window:**

```
student@mcacc1-60:~/terraform-project$ ls
Jenkinsfile  backend  frontend  k8s
student@mcacc1-60:~/terraform-project$
```

The Windows taskbar at the bottom shows the Start button, a search bar, and several application icons. The system tray on the right indicates the language is English (IN), the date is 22-03-2025, and the time is 02:14 PM.

# Configuring git in new Jenkins pipeline project: Step 1

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/job/terraform-project/configure`. The page title is "Configure" and the breadcrumb navigation shows "Dashboard > terraform-project > Configuration".

On the left sidebar, there are four tabs: "General", "Triggers", "Pipeline" (which is selected and highlighted), and "Advanced".

The main content area is titled "Configure" and contains a section "Definition" with a dropdown menu set to "Pipeline script from SCM". Below this, there is a section "SCM" with a dropdown menu set to "Git".

Under the "SCM" section, there is a "Repositories" section. It contains a "Repository URL" field with the value `https://github.com/SysSyncer/terraform-project.git`. Below the URL field, there is a red error message: "Please enter Git repository." Below the error message, there is a "Credentials" dropdown menu set to "- none -". At the bottom of the "Repositories" section, there are two buttons: "+ Add" and "Advanced" (with a dropdown arrow).

At the bottom of the configuration page, there are two buttons: "Save" and "Apply".

The Windows taskbar is visible at the bottom of the screen, showing the Start button, a search bar, and several application icons. The system tray on the right shows the language as "ENG IN", the time as "09:39 AM", and the date as "22-03-2025".

# Configuring git in new Jenkins pipeline project: Step 2

The screenshot shows the Jenkins web interface in a browser window. The address bar displays `localhost:8080/job/terraform-project/configure`. The breadcrumb navigation shows `Dashboard > terraform-project > Configuration`. On the left, the 'Configure' sidebar has four options: 'General', 'Triggers', 'Pipeline' (which is selected and highlighted), and 'Advanced'. The main content area is titled 'Configure' and contains the following sections:

- SCM**: A dropdown menu set to 'Git'.
- Repositories**: A dashed box containing:
  - Repository URL**: A text input field with the value `https://github.com/SysSyncer/terraform-project.git`.
  - Credentials**: A dropdown menu set to `SysSyncer/*****`.
  - + Add**: A button to add a new repository.
  - Advanced**: A dropdown menu.
- Add Repository**: A button below the repositories section.
- Branches to build**: A dashed box containing:
  - Branch Specifier (blank for 'any')**: A text input field.

At the bottom of the configuration area are two buttons: 'Save' (in blue) and 'Apply'.

The Windows taskbar at the bottom shows the Start button, a search bar, and several application icons. The system tray on the right indicates the language is 'ENG IN', shows volume and network icons, and displays the time as '09:39 AM' on '22-03-2025'.

# Configuring git in new Jenkins pipeline project: Step 3

Dashboard > terraform-project > Configuration

## Configure

- General
- Triggers
- Pipeline**
- Advanced

**Credentials** ?

SysSyncer/\*\*\*\*\*

+ Add

Advanced ▾

Add Repository

**Branches to build** ?

Branch Specifier (blank for 'any') ?

main

Add Branch

**Repository browser** ?

(Auto)

**Additional Behaviours**

Add ▾

Save Apply

# Configuring Jenkins file

```
student@mcacc1-60: ~/terraform  student@mcacc1-60: ~  +  -
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 2,1 Top

Windows taskbar at the bottom shows the Start button, a search bar, and several application icons including File Explorer, Edge, and various development tools. The system tray on the right indicates the language is English (IN), and the date and time are 09:51 AM on 22-03-2025.

# Configuring Jenkins file: Step 1

```
student@mcacc1-60: ~/terraform  X  student@mcacc1-60: ~  X  +  v
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:lates" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }


    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 4,44 Top  
-- VISUAL -- 20



# Configuring Jenkins file: Step 2

```
student@mcacc1-60: ~/terraform  student@mcacc1-60: ~
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 5,48 Top  
-- VISUAL -- 22

Windows taskbar: Search, ENG IN, 09:51 AM, 22-03-2025



# Configuring Jenkins file: Step 3

```
student@mcacc1-60: ~/terraform  student@mcacc1-60: ~  +  -
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 6,33 Top  
-- VISUAL -- 17

Windows taskbar: Search, File Explorer, Edge, Firefox, Chrome, Teams, OneDrive, Docker Desktop, Jenkins, Task Manager, Settings, Network, Volume, ENG IN, 09:51 AM, 22-03-2025

# Configuring Jenkins file: Step 4

```
student@mcacc1-60: ~/terraform x student@mcacc1-60: ~ x + v
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 12,80 Top  
-- VISUAL -- 14

Windows taskbar: Search, File Explorer, Edge, Firefox, Chrome, VS Code, Docker Desktop, Jenkins, 09:51 AM 22-03-2025

# Configuring Jenkins file: Step 5

```
student@mcacc1-60: ~/terraform  student@mcacc1-60: ~
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 13,101 Top  
-- VISUAL -- 31

Windows taskbar: Search, File Explorer, Edge, Firefox, Chrome, Teams, Docker Desktop, Jenkins, Task Manager, System Tray (ENG IN, 09:51 AM, 22-03-2025)

# Configuring Jenkins file: Step 6

```
student@mcacc1-60: ~/terraform  student@mcacc1-60: ~
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "absurdguy/k8s:latest" // Change this to your registry
    CONTAINER_NAME = "docker-k8s-running-app"
    REGISTRY_CREDENTIALS = "dockerhub-project" // Jenkins credentials ID
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-project', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
          git url: "https://$GIT_USER:$GIT_TOKEN@github.com/SysSyncer/terraform-project.git", branch: 'main'
        }
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t $DOCKER_IMAGE .'
      }
    }

    stage('Login to Docker Registry') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'dockerhub-project', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
        }
      }
    }

    stage('Push to Container Registry') {
      steps {
        sh 'docker push $DOCKER_IMAGE'
      }
    }

    stage('Stop & Remove Existing Container') {
      steps {
        script {

```

Jenkinsfile 26,83 Top  
-- VISUAL -- 17

Windows taskbar: Search, File Explorer, Edge, Firefox, Chrome, Teams, Docker Desktop, Jenkins, Visual Studio Code, 09:52 AM 22-03-2025

# Installing Kubernetes Plugin for Jenkins

The screenshot shows the Jenkins web interface at the URL `localhost:8080/manage/pluginManager/available`. The left sidebar contains navigation links: **Plugins**, **Updates**, **Available plugins** (highlighted), **Installed plugins**, and **Advanced settings**. The main content area displays a search for 'kubernetes' with a table of results. The table has columns for 'Install', 'Name', and 'Released'. The 'Kubernetes' plugin is selected with a blue checkmark. The bottom of the image shows a Windows taskbar with the Start button, search bar, and various application icons.

Install	Name	Released
<input type="checkbox"/>	<b>Kubernetes Client API</b> 6.10.0-251.v556f5f100500 kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	18 days ago
<input type="checkbox"/>	<b>Kubernetes Credentials</b> 192.v4d5b_1c429d17 kubernetes credentials Common classes for Kubernetes credentials	18 days ago
<input checked="" type="checkbox"/>	<b>Kubernetes</b> 4324.vfec199a_33512 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	16 days ago
<input type="checkbox"/>	<b>Kubernetes CLI</b> 1.364.vadef8cb8b823 kubernetes Configure kubectl for Kubernetes	12 hr ago
<input type="checkbox"/>	<b>Kubernetes Credentials Provider</b> 1.276.v99a_de03cb_076 kubernetes credentials Provides a read only credentials store backed by Kubernetes.	14 days ago
<input type="checkbox"/>	<b>Kubernetes :: Pipeline :: DevOps Steps</b> 1.6 pipeline kubernetes	6 yr 1 mo ago
<input type="checkbox"/>	<b>GitLab Credentials - Kubernetes Integration</b> 424.vd4c848a_61813 kubernetes gitlab	27 days ago

# Installing Kubernetes Plugin for Jenkins

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/manage/pluginManager/updates/`. The page title is "Jenkins". The navigation bar shows "Dashboard > Manage Jenkins > Plugins".

**Plugins**

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress**

**Download progress**

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Kubernetes Client API ✓ Success

Authentication Tokens API ✓ Success

Kubernetes Credentials ✓ Success

Kubernetes ✓ Success

Loading plugin extensions ✓ Success

→ [Go back to the top page](#)  
(you can start using the installed plugins right away)

→ ☐ Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.492.2

# Double check if global credentials exists

new-pipeline-project/Jenkins x day-5 - Google Drive x Uninstall Jenkins in WSL x System » Global credentials x Docker Home x Updated Jenkins File x

localhost:8080/manage/credentials/store/system/domain/\_/





Jenkins Student log out

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

## Global credentials (unrestricted)


+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 <a href="#">dockerhub-project</a>	absurdguy/*****	Username with password	
 <a href="#">github-project</a>	SysSyncer/*****	Username with password	

Icon: S M **L**

REST API Jenkins 2.492.2

 ENG IN 02:12 PM 22-03-2025

# Updating the Jenkinsfile: Step 1

The screenshot displays the Jenkins web interface on the left and a code editor on the right. The Jenkins UI shows the 'terraform-project' pipeline with a status of 'Failed' (indicated by a red 'X'). The 'Builds' section lists several failed builds, with the most recent being build #10 at 8:29 AM. The code editor on the right shows the Jenkinsfile content, which defines a pipeline with two stages: 'Checkout Code' and 'Set Minikube Docker Environment'. The 'Checkout Code' stage checks out the code from the repository. The 'Set Minikube Docker Environment' stage sets up the Minikube Docker environment for Jenkins. The 'Build Backend Docker Image' stage is partially visible at the bottom.

**Jenkins UI:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds:
  - Today
  - #10 8:29 AM
  - #9 8:28 AM
  - #8 8:22 AM
  - #7 8:18 AM
  - #6 6:15 AM

**Jenkinsfile:**

```
pipeline {
  agent any
  environment {
    MINIKUBE_HOME = '/home/student/.minikube' // Path to Minikube if it's not de
  }
  stages {
    stage('Checkout Code') {
      steps {
        // Checkout the code from the repository
        git url: "https://github.com/SysSyncer/terraform-project.git", branch
      }
    }
    stage('Set Minikube Docker Environment') {
      steps {
        script {
          // Set up the Minikube Docker environment for Jenkins
          sh '''
            eval $(minikube docker-env)
          '''
        }
      }
    }
    stage('Build Backend Docker Image') {
      steps {
        dir('backend') {
          // Build the Docker image for the backend using Minikube's Docker
          sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
        }
      }
    }
  }
}
```



# Updating the Jenkinsfile: Step 2

The screenshot displays the Jenkins web interface on the left and a code editor on the right. The Jenkins UI shows the 'terraform-project' pipeline with a status of 'Failed' (indicated by a red 'x' icon). The 'Builds' section lists several failed builds, with the most recent being build #10 at 8:29 AM. The code editor on the right shows the Jenkinsfile for this pipeline, which is configured to run on any agent and uses a Docker environment. The pipeline consists of three stages: 'Checkout Code', 'Set Minikube Docker Environment', and 'Build Backend Docker Image'. The 'Checkout Code' stage checks out the code from the repository. The 'Set Minikube Docker Environment' stage sets up the Minikube Docker environment for Jenkins. The 'Build Backend Docker Image' stage builds the Docker image for the backend using Minikube's Docker daemon.

**Jenkins UI:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds:
  - Today
  - #10 8:29 AM
  - #9 8:28 AM
  - #8 8:22 AM
  - #7 8:18 AM
  - #6 6:15 AM

**Jenkinsfile:**

```
pipeline {
  agent any
  environment {
    MINIKUBE_HOME = '/home/student/.minikube' // Path to Minikube if it's not de
  }
  fault
  KUBERNETES_VERSION = '1.20.0' // Optional: Define Kubernetes version if needed
  DOCKER_IMAGE_FRONTEND = "frontend:latest" // Frontend Docker image name
  DOCKER_IMAGE_BACKEND = "backend:latest" // Backend Docker image name
  CONTAINER_NAME_FRONTEND = "docker-k8s-frontend-app" // Frontend container na
  CONTAINER_NAME_BACKEND = "docker-k8s-backend-app" // Backend container name
}

stages {
  stage('Checkout Code') {
    steps {
      // Checkout the code from the repository
      git url: "https://github.com/SysSyncer/terraform-project.git", branch
      : 'main'
    }
  }

  stage('Set Minikube Docker Environment') {
    steps {
      script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
          eval $(minikube docker-env)
        '''
      }
    }
  }

  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
        sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
      }
    }
  }
}
```

# Updating the Jenkinsfile: Step 3

The screenshot displays the Jenkins web interface on the left and a code editor on the right. The Jenkins UI shows the 'terraform-project' pipeline with a status of 'Failed' (indicated by a red 'x' icon). The 'Builds' section lists several failed builds, with the most recent being build #10 at 8:29 AM. The code editor on the right shows the Jenkinsfile for this pipeline, which is configured to run on any agent and uses a Docker environment. The pipeline consists of three stages: 'Checkout Code', 'Set Minikube Docker Environment', and 'Build Backend Docker Image'. The 'Set Minikube Docker Environment' stage is currently selected and highlighted in the editor.

**Jenkins UI (Left Panel):**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax
- Builds: #10 8:29 AM, #9 8:28 AM, #8 8:22 AM, #7 8:18 AM, #6 6:15 AM

**Jenkinsfile (Right Panel):**

```
pipeline {
  agent any
  environment {
    MINIKUBE_HOME = '/home/student/.minikube' // Path to Minikube if it's not de
  }
  fault
  KUBEVERSION = '1.20.0' // Optional: Define Kubernetes version if needed
  DOCKER_IMAGE_FRONTEND = "frontend:latest" // Frontend Docker image name
  DOCKER_IMAGE_BACKEND = "backend:latest" // Backend Docker image name
  CONTAINER_NAME_FRONTEND = "docker-k8s-frontend-app" // Frontend container na
  CONTAINER_NAME_BACKEND = "docker-k8s-backend-app" // Backend container name
}

stages {
  stage('Checkout Code') {
    steps {
      // Checkout the code from the repository
      git url: "https://github.com/SysSyncer/terraform-project.git", branch
    }
  }

  stage('Set Minikube Docker Environment') {
    steps {
      script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
        eval $(minikube docker-env)
        '''
      }
    }
  }

  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
        sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
      }
    }
  }
}
```

# Updating the Jenkinsfile: Step 4

The Jenkins web interface is shown in a browser window. The address bar indicates the URL is `localhost:8080/job/terraform-project/`. The Jenkins logo is at the top left. Below it, the breadcrumb navigation shows `Dashboard > terraform-project >`. On the left sidebar, there are links for `Status` (selected), `Changes`, `Build Now`, `Configure`, `Delete Pipeline`, `Stages`, `Rename`, and `Pipeline Syntax`. The main content area shows the pipeline status for `terraform-project` with a red 'X' icon. Under the heading `Permalinks`, there is a list of build links: 

- Last build (#10), 13 min ago
- Last failed build (#10), 13 min ago
- Last unsuccessful build (#10), 13 min ago
- Last completed build (#10), 13 min ago

 At the bottom left, there is a `Builds` section with a search filter and a list of recent builds: 

- #10 8:29 AM
- #9 8:28 AM
- #8 8:22 AM
- #7 8:18 AM
- #6 6:15 AM

A terminal window displays the content of the `Jenkinsfile`. The file defines a pipeline with several stages and steps. The `Checkout Code` stage checks out the code from a GitHub repository. The `Set Minikube Docker Environment` stage sets up the Minikube Docker environment. The `Build Backend Docker Image` stage builds the Docker image for the backend. The `Build Frontend Docker Image` stage builds the Docker image for the frontend. The terminal output shows the progress of the build, with the `Build Backend Docker Image` stage currently running. The terminal also shows the `Jenkinsfile` content, which is a YAML file defining the pipeline. The terminal output shows the progress of the build, with the `Build Backend Docker Image` stage currently running. The terminal also shows the `Jenkinsfile` content, which is a YAML file defining the pipeline.

```
CONTAINER_NAME_BACKEND = "docker-k8s-backend-app" // Backend container name
}

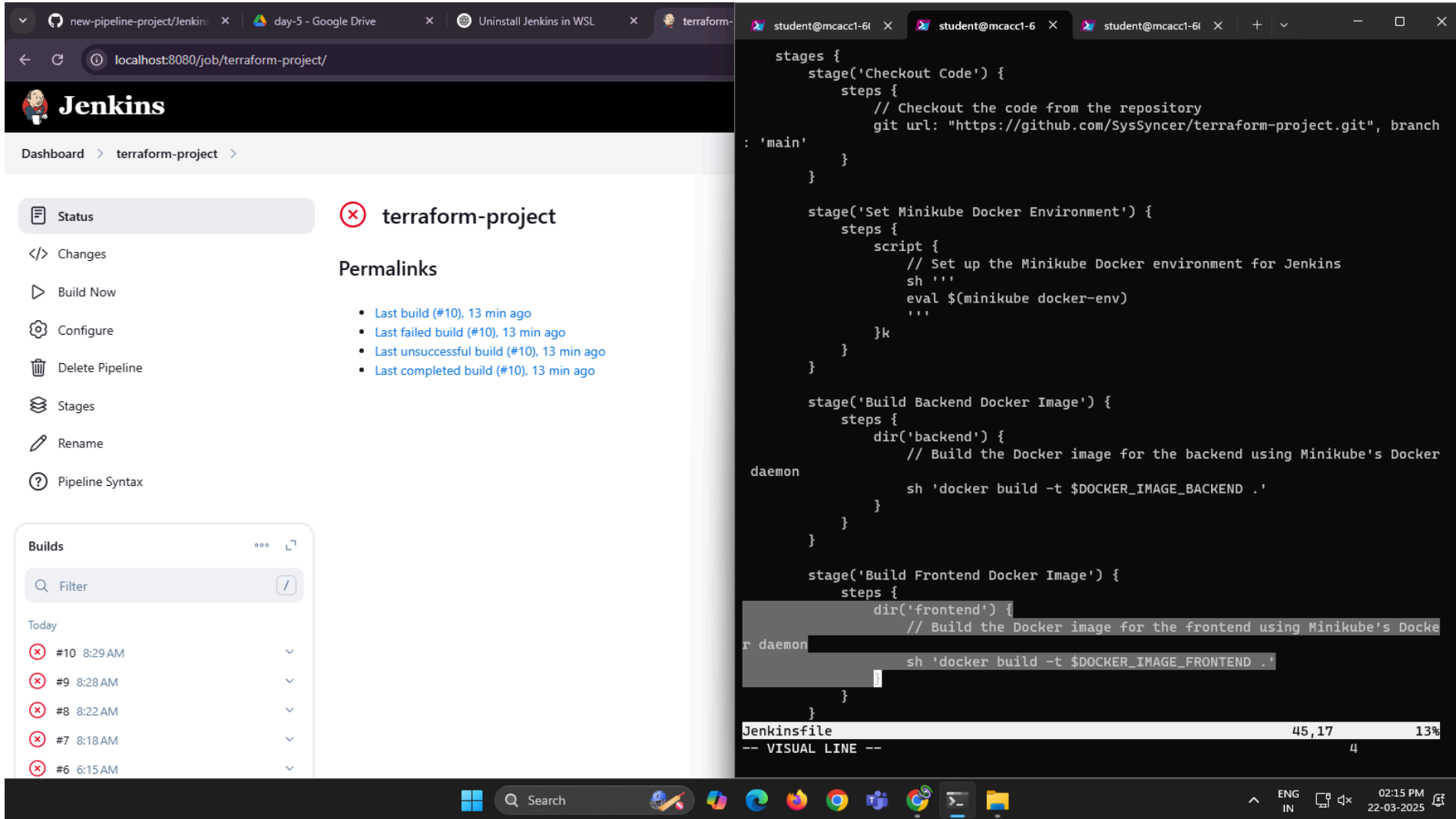
stages {
  stage('Checkout Code') {
    steps {
      // Checkout the code from the repository
      git url: "https://github.com/SysSyncer/terraform-project.git", branch
: 'main'
    }
  }

  stage('Set Minikube Docker Environment') {
    steps {
      script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
          eval $(minikube docker-env)
        '''
      }k
    }
  }

  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
daemon
        sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
      }
    }
  }

  stage('Build Frontend Docker Image') {
    steps {
      dir('frontend') {
        // Build the Docker image for the frontend using Minikube's Docke
r daemon
        sh 'docker build -t $DOCKER_IMAGE_FRONTEND .'
      }
    }
  }
}
```

# Updating the Jenkinsfile: Step 5



The screenshot displays the Jenkins web interface on the left and a terminal window on the right. The Jenkins interface shows the 'terraform-project' pipeline with a failed status (red X). The 'Builds' section lists recent builds, all marked as failed. The terminal window shows the Jenkinsfile code, which defines a pipeline with three stages: 'Checkout Code', 'Set Minikube Docker Environment', and 'Build Backend Docker Image'. The 'Build Backend Docker Image' stage is currently highlighted in the terminal.

**Jenkins Interface:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds:
  - Today
  - #10 8:29 AM (Failed)
  - #9 8:28 AM (Failed)
  - #8 8:22 AM (Failed)
  - #7 8:18 AM (Failed)
  - #6 6:15 AM (Failed)

**Jenkinsfile Code:**

```
stages {
  stage('Checkout Code') {
    steps {
      // Checkout the code from the repository
      git url: "https://github.com/SysSyncer/terraform-project.git", branch
: 'main'
    }
  }

  stage('Set Minikube Docker Environment') {
    steps {
      script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
          eval $(minikube docker-env)
        '''
      }
    }
  }

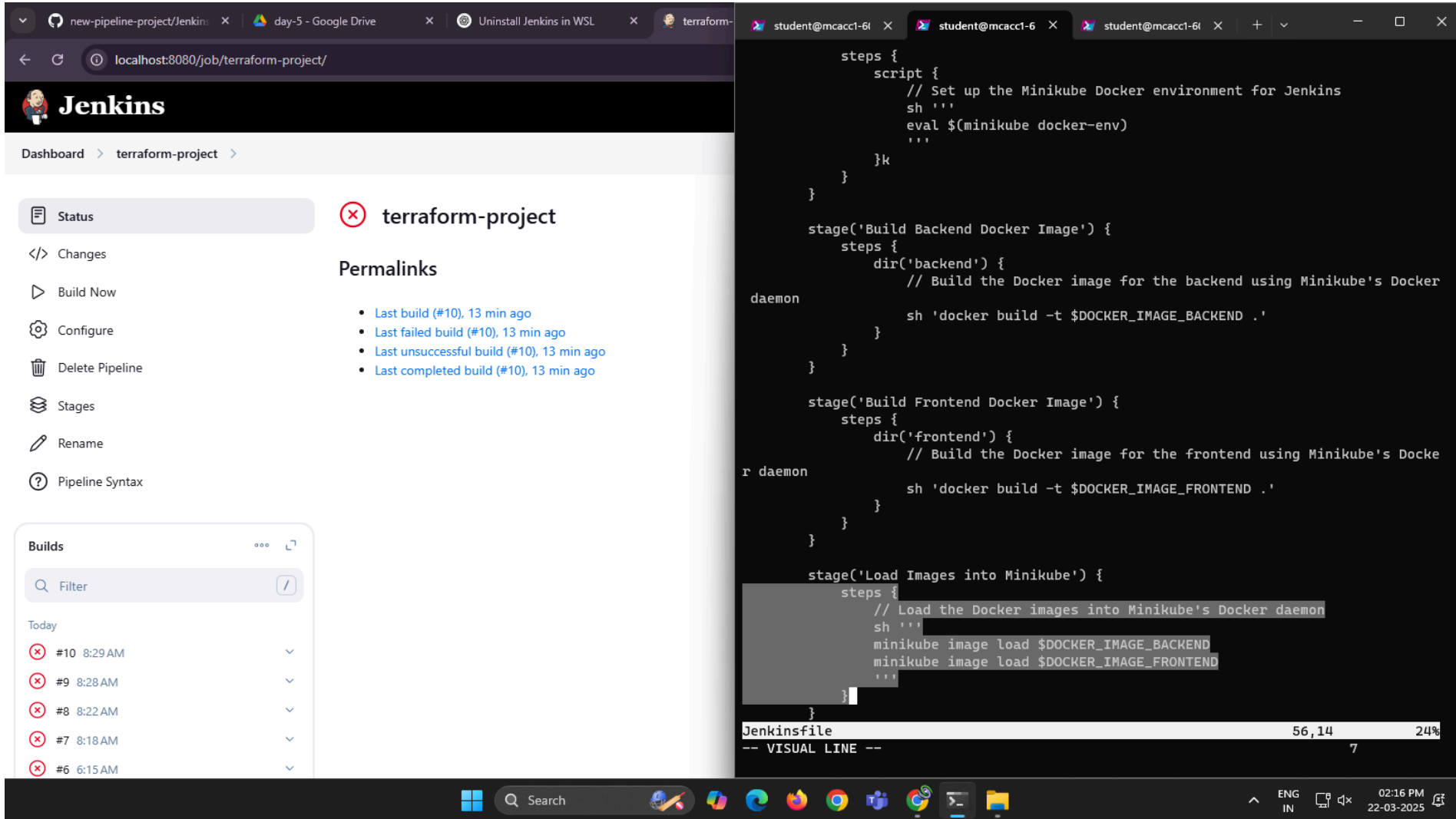
  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
daemon
        sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
      }
    }
  }

  stage('Build Frontend Docker Image') {
    steps {
      dir('frontend') {
        // Build the Docker image for the frontend using Minikube's Docker
r daemon
        sh 'docker build -t $DOCKER_IMAGE_FRONTEND .'
      }
    }
  }
}
```

**Terminal Output:**

```
Jenkinsfile 45,17 13%
-- VISUAL LINE -- 4
```

# Updating the Jenkinsfile: Step 6



The screenshot displays the Jenkins web interface on the left and a code editor on the right. The Jenkins UI shows the 'terraform-project' pipeline with a red status icon and a list of builds, all marked as failed. The code editor on the right shows the Jenkinsfile with several updates highlighted in grey, including the addition of a 'daemon' step in the 'Build Backend Docker Image' stage and the 'Load Images into Minikube' stage.

**Jenkins UI (Left Panel):**

- Dashboard > terraform-project
- Status: terraform-project (with a red status icon)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds List:
  - #10 8:29 AM (Failed)
  - #9 8:28 AM (Failed)
  - #8 8:22 AM (Failed)
  - #7 8:18 AM (Failed)
  - #6 6:15 AM (Failed)

**Jenkinsfile (Right Panel):**

```
steps {
    script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
            eval $(minikube docker-env)
        '''
    }k
}

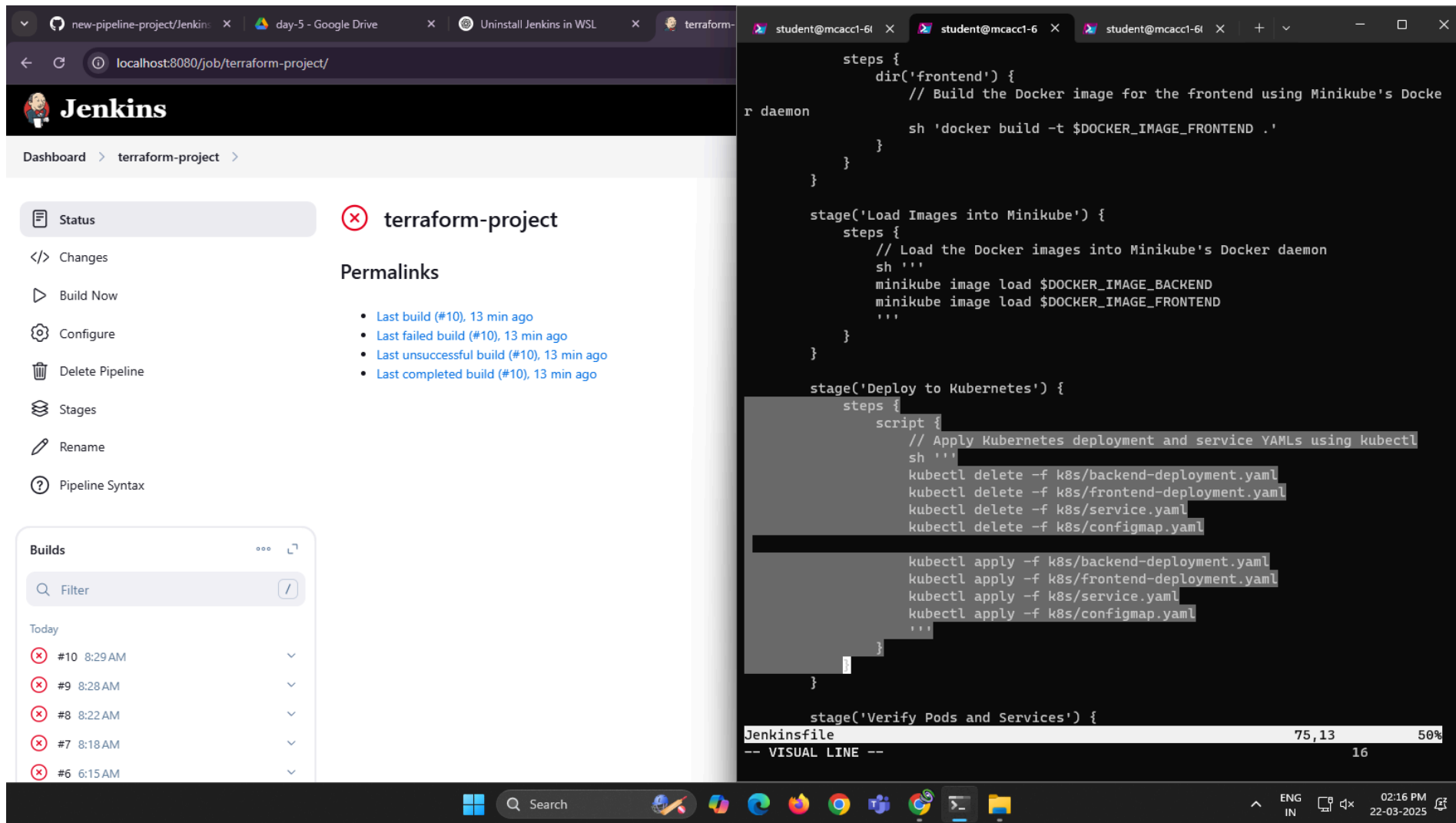
stage('Build Backend Docker Image') {
    steps {
        dir('backend') {
            // Build the Docker image for the backend using Minikube's Docker
            daemon
            sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
        }
    }
}

stage('Build Frontend Docker Image') {
    steps {
        dir('frontend') {
            // Build the Docker image for the frontend using Minikube's Docker
            r daemon
            sh 'docker build -t $DOCKER_IMAGE_FRONTEND .'
        }
    }
}

stage('Load Images into Minikube') {
    steps {
        // Load the Docker images into Minikube's Docker daemon
        sh '''
            minikube image load $DOCKER_IMAGE_BACKEND
            minikube image load $DOCKER_IMAGE_FRONTEND
        '''
    }
}
```

Jenkinsfile 56,14 24%  
-- VISUAL LINE -- 7

# Updating the Jenkinsfile: Step 7



The screenshot displays the Jenkins web interface on the left and a terminal window on the right. The Jenkins UI shows the 'terraform-project' pipeline with a red status icon. The 'Builds' section lists recent builds, all marked with red 'X' icons, indicating failures. The terminal window shows the Jenkinsfile content, which includes stages for building Docker images, loading them into Minikube, deploying to Kubernetes, and verifying pods and services. The terminal also shows the command to run the Jenkinsfile.

**Jenkins UI:**

- Dashboard > terraform-project
- Status (selected)
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax

**terraform-project Permalinks:**

- Last build (#10), 13 min ago
- Last failed build (#10), 13 min ago
- Last unsuccessful build (#10), 13 min ago
- Last completed build (#10), 13 min ago

**Builds:**

Build Number	Time	Status
#10	8:29 AM	Failed
#9	8:28 AM	Failed
#8	8:22 AM	Failed
#7	8:18 AM	Failed
#6	6:15 AM	Failed

**Jenkinsfile:**

```
steps {
  dir('frontend') {
    // Build the Docker image for the frontend using Minikube's Docker daemon
    sh 'docker build -t $DOCKER_IMAGE_FRONTEND .'
  }
}

stage('Load Images into Minikube') {
  steps {
    // Load the Docker images into Minikube's Docker daemon
    sh '''
minikube image load $DOCKER_IMAGE_BACKEND
minikube image load $DOCKER_IMAGE_FRONTEND
'''
  }
}

stage('Deploy to Kubernetes') {
  steps {
    script {
      // Apply Kubernetes deployment and service YAMLs using kubectl
      sh '''
kubectl delete -f k8s/backend-deployment.yaml
kubectl delete -f k8s/frontend-deployment.yaml
kubectl delete -f k8s/service.yaml
kubectl delete -f k8s/configmap.yaml

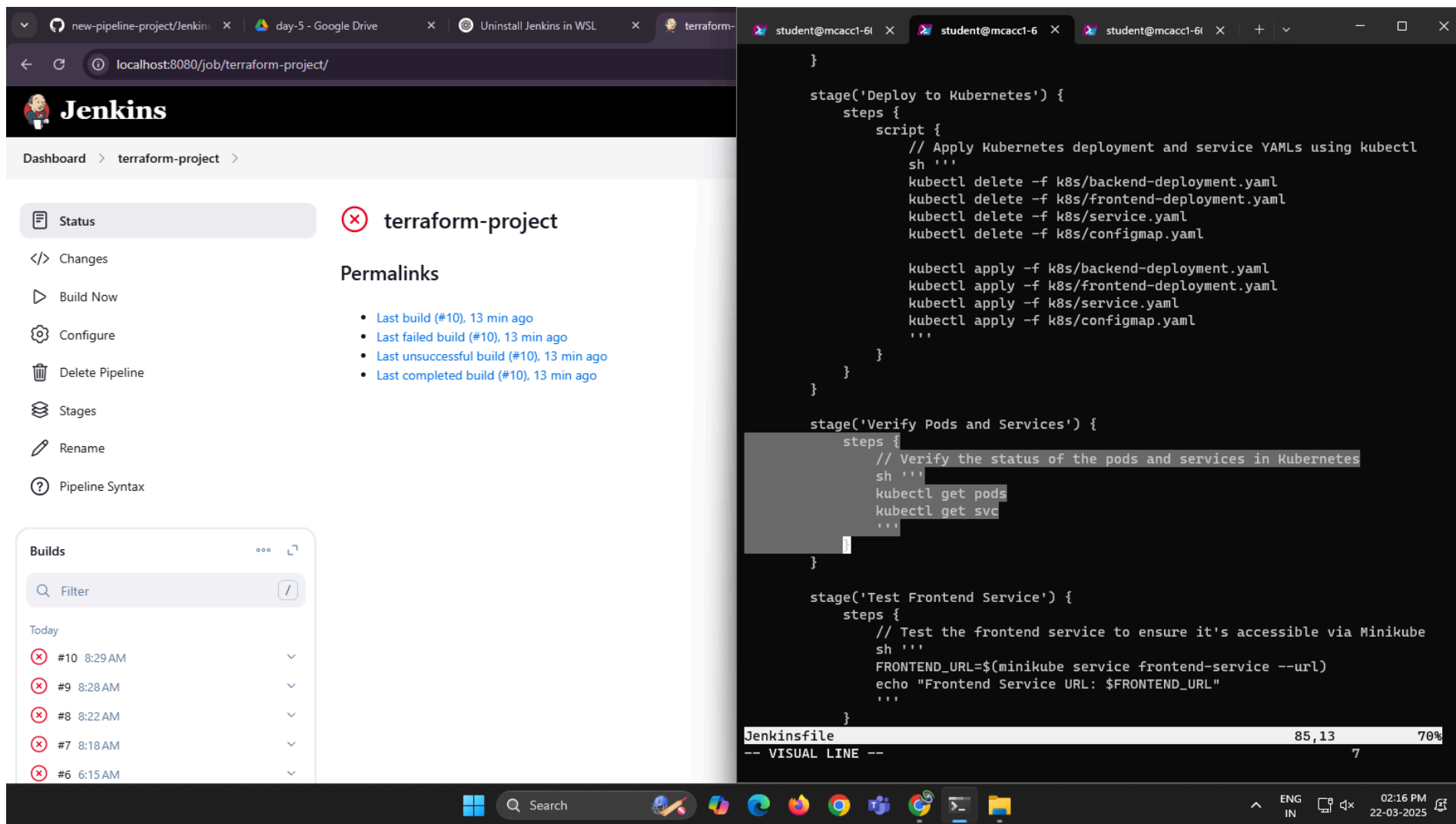
kubectl apply -f k8s/backend-deployment.yaml
kubectl apply -f k8s/frontend-deployment.yaml
kubectl apply -f k8s/service.yaml
kubectl apply -f k8s/configmap.yaml
'''
    }
  }
}

stage('Verify Pods and Services') {
  // ...
}
```

**Terminal Output:**

```
Jenkinsfile 75,13 50%
-- VISUAL LINE -- 16
```

# Updating the Jenkinsfile: Step 8



The screenshot displays the Jenkins web interface on the left and a code editor on the right. The Jenkins UI shows the 'terraform-project' pipeline with a status of 'Failed' (indicated by a red 'x'). The 'Builds' section lists several failed builds, with the most recent being build #10 at 8:29 AM. The code editor on the right shows the Jenkinsfile for this pipeline, which is being updated. The file contains two stages: 'Deploy to Kubernetes' and 'Verify Pods and Services'. The 'Deploy to Kubernetes' stage includes steps to delete and then apply Kubernetes manifests. The 'Verify Pods and Services' stage includes a step to verify the status of pods and services. The 'Test Frontend Service' stage is also present, with a step to test the frontend service. The code editor shows the file is 85,13 lines long and 70% complete.

**Jenkins UI:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax
- Builds: #10 8:29 AM, #9 8:28 AM, #8 8:22 AM, #7 8:18 AM, #6 6:15 AM

**Jenkinsfile:**

```
stage('Deploy to Kubernetes') {
  steps {
    script {
      // Apply Kubernetes deployment and service YAMLs using kubectl
      sh '''
        kubectl delete -f k8s/backend-deployment.yaml
        kubectl delete -f k8s/frontend-deployment.yaml
        kubectl delete -f k8s/service.yaml
        kubectl delete -f k8s/configmap.yaml

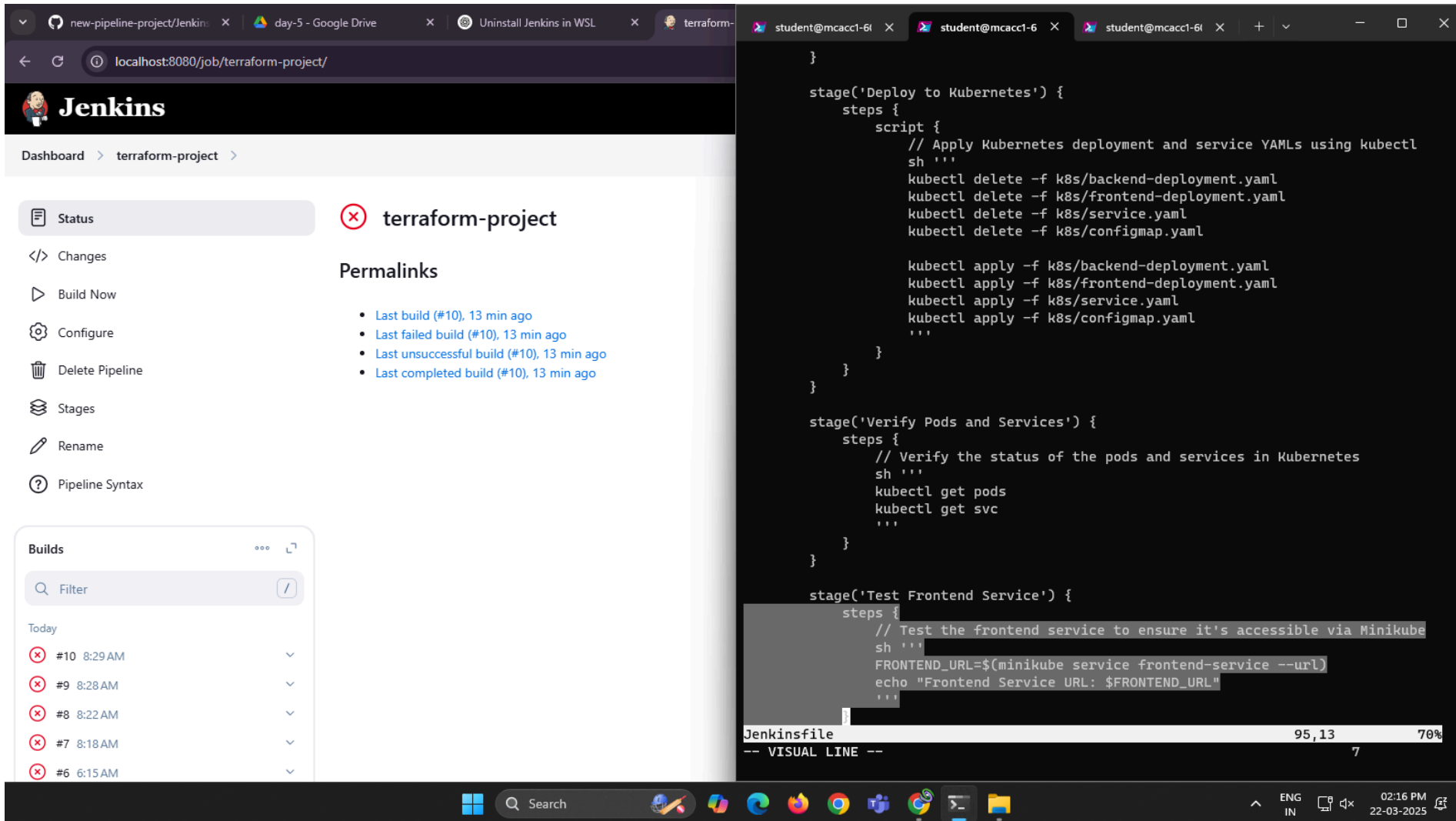
        kubectl apply -f k8s/backend-deployment.yaml
        kubectl apply -f k8s/frontend-deployment.yaml
        kubectl apply -f k8s/service.yaml
        kubectl apply -f k8s/configmap.yaml
      '''
    }
  }
}

stage('Verify Pods and Services') {
  steps {
    // Verify the status of the pods and services in Kubernetes
    sh '''
      kubectl get pods
      kubectl get svc
    '''
  }
}

stage('Test Frontend Service') {
  steps {
    // Test the frontend service to ensure it's accessible via Minikube
    sh '''
      FRONTEND_URL=$(minikube service frontend-service --url)
      echo "Frontend Service URL: $FRONTEND_URL"
    '''
  }
}
```

Jenkinsfile 85,13 70%  
-- VISUAL LINE -- 7

# Updating the Jenkinsfile: Step 9



The screenshot displays the Jenkins web interface on the left and a terminal window on the right. The Jenkins UI shows the 'terraform-project' pipeline with a red 'X' icon, indicating a failure. The 'Builds' section lists several failed builds, with the most recent being build #10 at 8:29 AM. The terminal window on the right shows the Jenkinsfile content, which is being edited. The file contains three stages: 'Deploy to Kubernetes', 'Verify Pods and Services', and 'Test Frontend Service'. The 'Test Frontend Service' stage is currently selected, and the terminal shows the command to run the Jenkinsfile.

**Jenkins UI:**

- Dashboard > terraform-project
- Status (selected)
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax

**terraform-project Permalinks:**

- Last build (#10), 13 min ago
- Last failed build (#10), 13 min ago
- Last unsuccessful build (#10), 13 min ago
- Last completed build (#10), 13 min ago

**Builds:**

Build Number	Time	Status
#10	8:29 AM	Failed
#9	8:28 AM	Failed
#8	8:22 AM	Failed
#7	8:18 AM	Failed
#6	6:15 AM	Failed

**Jenkinsfile Content:**

```
stage('Deploy to Kubernetes') {
  steps {
    script {
      // Apply Kubernetes deployment and service YAMLs using kubectl
      sh '''
        kubectl delete -f k8s/backend-deployment.yaml
        kubectl delete -f k8s/frontend-deployment.yaml
        kubectl delete -f k8s/service.yaml
        kubectl delete -f k8s/configmap.yaml

        kubectl apply -f k8s/backend-deployment.yaml
        kubectl apply -f k8s/frontend-deployment.yaml
        kubectl apply -f k8s/service.yaml
        kubectl apply -f k8s/configmap.yaml
      '''
    }
  }
}

stage('Verify Pods and Services') {
  steps {
    // Verify the status of the pods and services in Kubernetes
    sh '''
      kubectl get pods
      kubectl get svc
    '''
  }
}

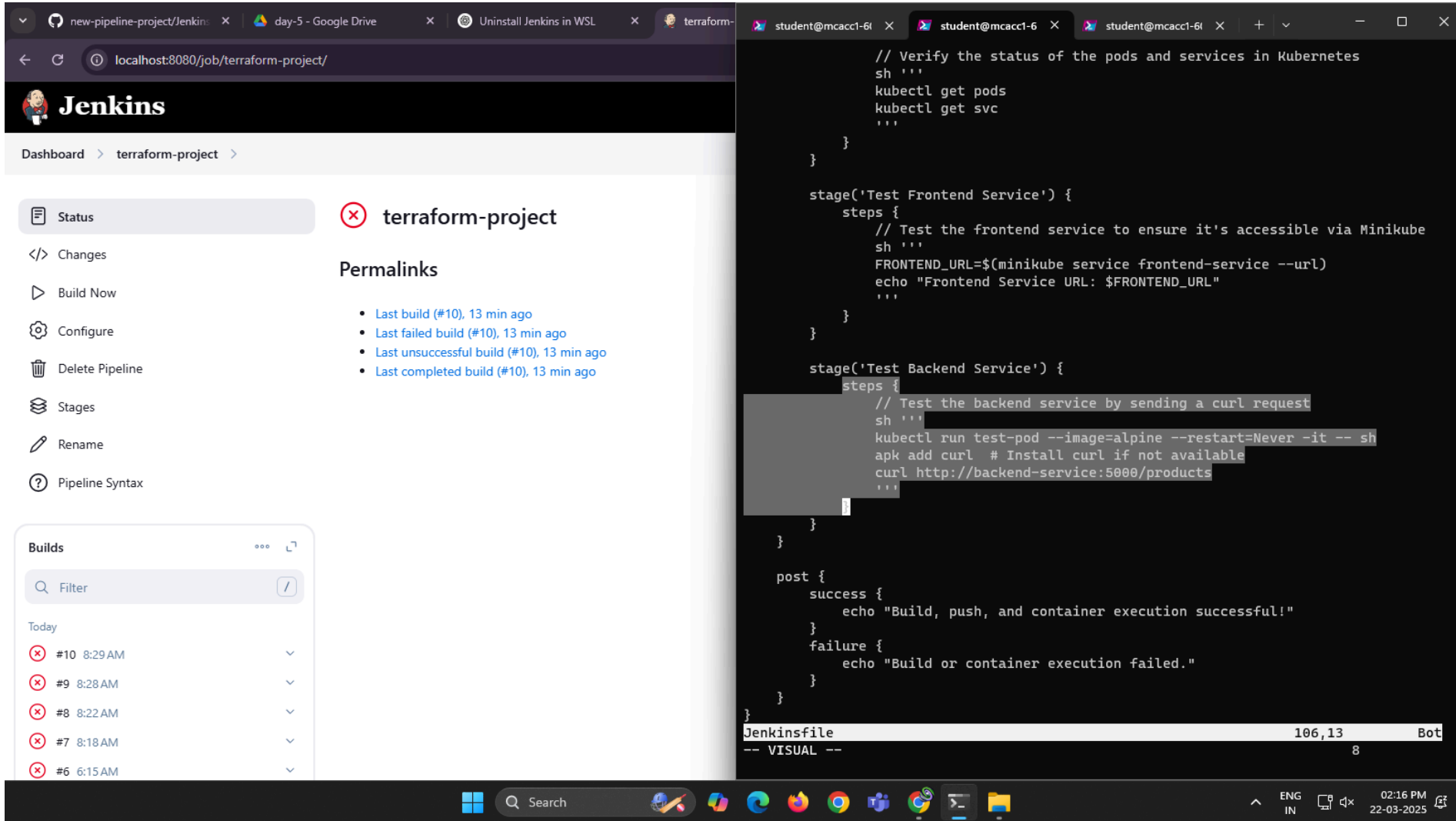
stage('Test Frontend Service') {
  steps {
    // Test the frontend service to ensure it's accessible via Minikube
    sh '''
      FRONTEND_URL=$(minikube service frontend-service --url)
      echo "Frontend Service URL: $FRONTEND_URL"
    '''
  }
}
```

**Terminal Output:**

```
Jenkinsfile 95,13 70%
-- VISUAL LINE -- 7
```



# Updating the Jenkinsfile: Step 10



The screenshot displays the Jenkins web interface on the left and a terminal window on the right. The Jenkins interface shows the 'terraform-project' pipeline with a red status icon and a list of recent builds, all marked as failed. The terminal window shows the Jenkinsfile being edited, with a new stage 'Test Backend Service' being added. The terminal output at the bottom indicates the file size is 106,13 bytes.

**Jenkins UI:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds:
  - #10 8:29 AM (Failed)
  - #9 8:28 AM (Failed)
  - #8 8:22 AM (Failed)
  - #7 8:18 AM (Failed)
  - #6 6:15 AM (Failed)

**Jenkinsfile:**

```

// Verify the status of the pods and services in Kubernetes
sh '''
kubectll get pods
kubectll get svc
'''

}

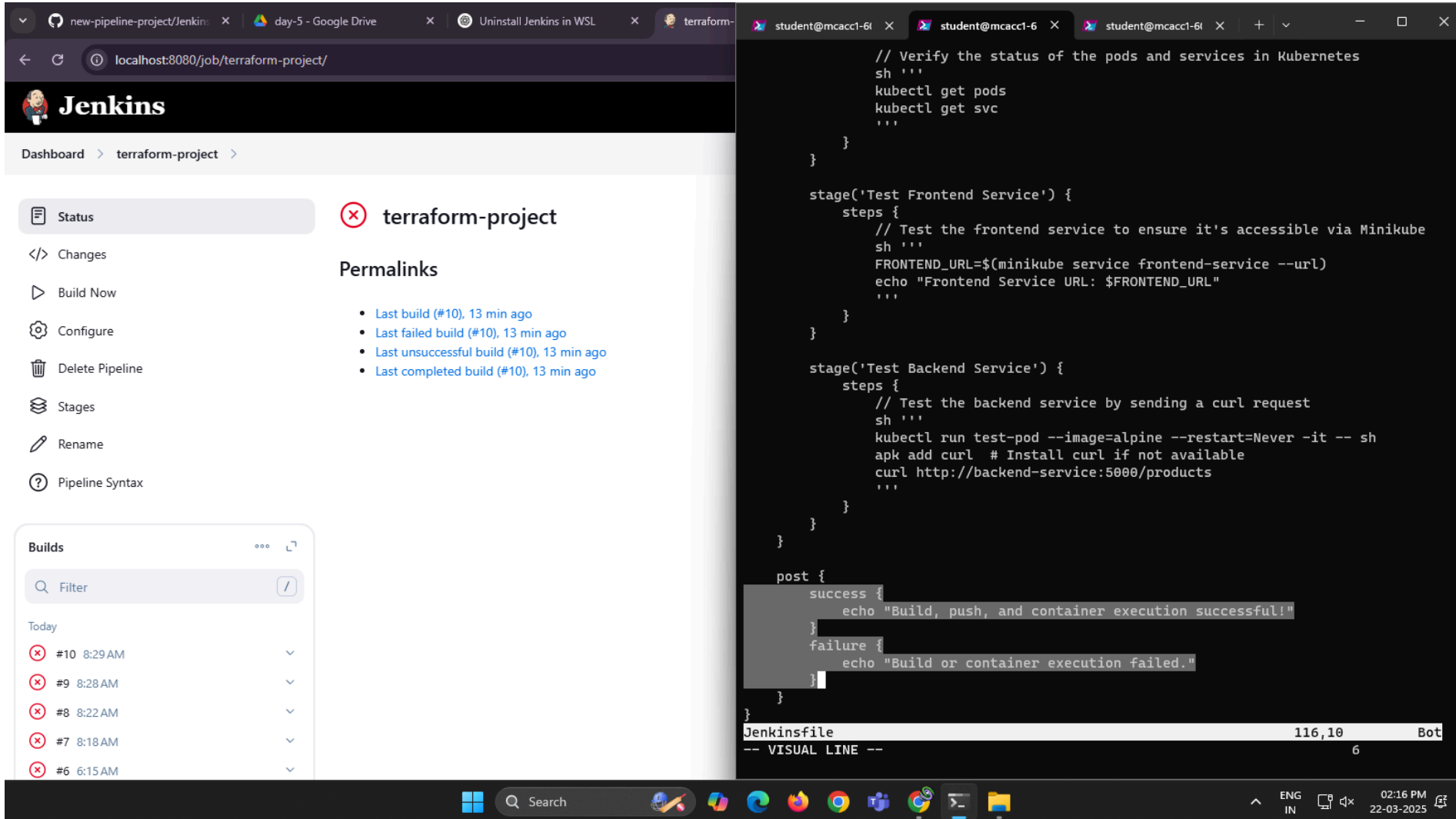
stage('Test Frontend Service') {
  steps {
    // Test the frontend service to ensure it's accessible via Minikube
    sh '''
    FRONTEND_URL=$(minikube service frontend-service --url)
    echo "Frontend Service URL: $FRONTEND_URL"
    '''
  }
}

stage('Test Backend Service') {
  steps {
    // Test the backend service by sending a curl request
    sh '''
    kubectll run test-pod --image=alpine --restart=Never -it -- sh
    apk add curl # Install curl if not available
    curl http://backend-service:5000/products
    '''
  }
}

post {
  success {
    echo "Build, push, and container execution successful!"
  }
  failure {
    echo "Build or container execution failed."
  }
}
}

Jenkinsfile 106,13 Bot
-- VISUAL -- 8
```

# Updating the Jenkinsfile: Step 11



The screenshot displays the Jenkins web interface on the left and a terminal window on the right. The Jenkins UI shows the 'terraform-project' pipeline with a failed status (red X). The 'Builds' section lists recent builds, with build #10 being the most recent and failed. The terminal window shows the Jenkinsfile content, which includes stages for testing frontend and backend services. The terminal also shows the output of the Jenkinsfile execution, indicating a successful build.

**Jenkins UI:**

- Dashboard > terraform-project
- Status: terraform-project (Failed)
- Permalinks:
  - Last build (#10), 13 min ago
  - Last failed build (#10), 13 min ago
  - Last unsuccessful build (#10), 13 min ago
  - Last completed build (#10), 13 min ago
- Builds:
  - Today
  - #10 8:29 AM (Failed)
  - #9 8:28 AM (Failed)
  - #8 8:22 AM (Failed)
  - #7 8:18 AM (Failed)
  - #6 6:15 AM (Failed)

**Jenkinsfile:**

```
// Verify the status of the pods and services in Kubernetes
sh '''
kubectl get pods
kubectl get svc
'''

}

stage('Test Frontend Service') {
  steps {
    // Test the frontend service to ensure it's accessible via Minikube
    sh '''
    FRONTEND_URL=$(minikube service frontend-service --url)
    echo "Frontend Service URL: $FRONTEND_URL"
    '''
  }
}

stage('Test Backend Service') {
  steps {
    // Test the backend service by sending a curl request
    sh '''
    kubectl run test-pod --image=alpine --restart=Never -it -- sh
    apk add curl # Install curl if not available
    curl http://backend-service:5000/products
    '''
  }
}

post {
  success {
    echo "Build, push, and container execution successful!"
  }
  failure {
    echo "Build or container execution failed."
  }
}
```

**Terminal Output:**

```
Jenkinsfile 116,10 Bot
-- VISUAL LINE -- 6
```

# Building the project pipeline

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/job/terraform-project/10/pipeline-console/`. The Jenkins logo and navigation links are at the top. The breadcrumb trail is `Dashboard > terraform-project > #10 > Pipeline Console`. The main heading is `< Build #10` with a red 'X' icon. To the right are buttons for `Rebuild`, `Overview`, `Configure`, and a menu icon. Below the heading, it says `In progress 8.5 sec ago in 8.5 sec and counting`. On the left is a list of pipeline steps: `Checkout SCM`, `Checkout Code`, `Build Backend Docker Image`, `Build Frontend Docker Image`, `Login to Docker Registry`, `Push Backend to Container Registry`, `Push Frontend to Container Registry`, `Stop & Remove Existing Containers`, `Run Backend Docker Container`, `Run Frontend Docker Container`, `Deploy to Kubernetes` (highlighted in orange with a red 'X'), and `Post Actions`. The main console area shows the build timeline: `Started 1.1 sec ago`, `Queued 0 ms`, `Took 0.53 sec`, and `Failure`. It also indicates `Running on Jenkins` and provides a `View as plain text` link. A red error banner for the `Deploy to Kubernetes` step shows the command `kubectl apply -f k8s/backend-deployment.yaml kubectl apply -f k8s/frontend-deployment.y...` and a duration of `0.3 sec`. The error message is: `error: error validating "k8s/backend-deployment.yaml": error validating data: failed to download openapi: <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fopenapi%2Fv2%3Ftimeout%3D32s'></script><body style='background-color:white; color:white;'>`. The console also shows `Authentication required` and `<!--`. A `Scroll to Bottom` link is at the bottom right of the console. The Jenkins version `Jenkins 2.492.2` is noted at the bottom right. The Windows taskbar at the bottom shows the Start button, search bar, and various application icons, with the system clock displaying `02:12 PM 22-03-2025`.

*End of day 5 assignment*