

Building kubernetes project

Day - 5

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 "Definition" and contains a dropdown menu set to "Pipeline script from SCM". Below this, there is a section for "SCM" (Source Control Management) 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 set to "ENG IN", the date and time as "09:39 AM 22-03-2025", and a network icon.

Configuring git in new Jenkins pipeline project: Step 2

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `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 system clock as 09:39 AM on 22-03-2025, with the language set to ENG IN.

Configuring git in new Jenkins pipeline project: Step 3

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, under the "Configure" section, the "Pipeline" option is selected. The main content area is divided into several sections:

- Credentials**: A dropdown menu shows "SysSyncer/*****". Below it are buttons for "+ Add" and "Advanced".
- Add Repository**: A button to add a new repository.
- Branches to build**: A section with a "Branch Specifier (blank for 'any')" label. A text input field contains "main". Below it is an "Add Branch" button.
- Repository browser**: A dropdown menu showing "(Auto)".
- Additional Behaviours**: A section with an "Add" button.

At the bottom of the configuration area are two buttons: "Save" (in blue) 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 set to "ENG IN", the date and time as "09:39 AM 22-03-2025", and a network icon.

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. 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  student@mcacc1-60: ~
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

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

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, File Explorer, Edge, Firefox, Chrome, Teams, Docker Desktop, VS Code, Task View, System Tray (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, Docker Desktop, Jenkins, Visual Studio Code, 09:51 AM 22-03-2025

Configuring Jenkins file: Step 4

```
student@mcacc1-60: ~/terraform  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, Teams, Docker Desktop, Jenkins, Visual Studio Code, Windows Defender, ENG IN, 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, Windows Defender, ENG IN, 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** (selected), **Installed plugins**, and **Advanced settings**. The main content area displays a search for 'kubernetes' with a table of available plugins. The 'Kubernetes' plugin is selected with a blue checkmark.

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

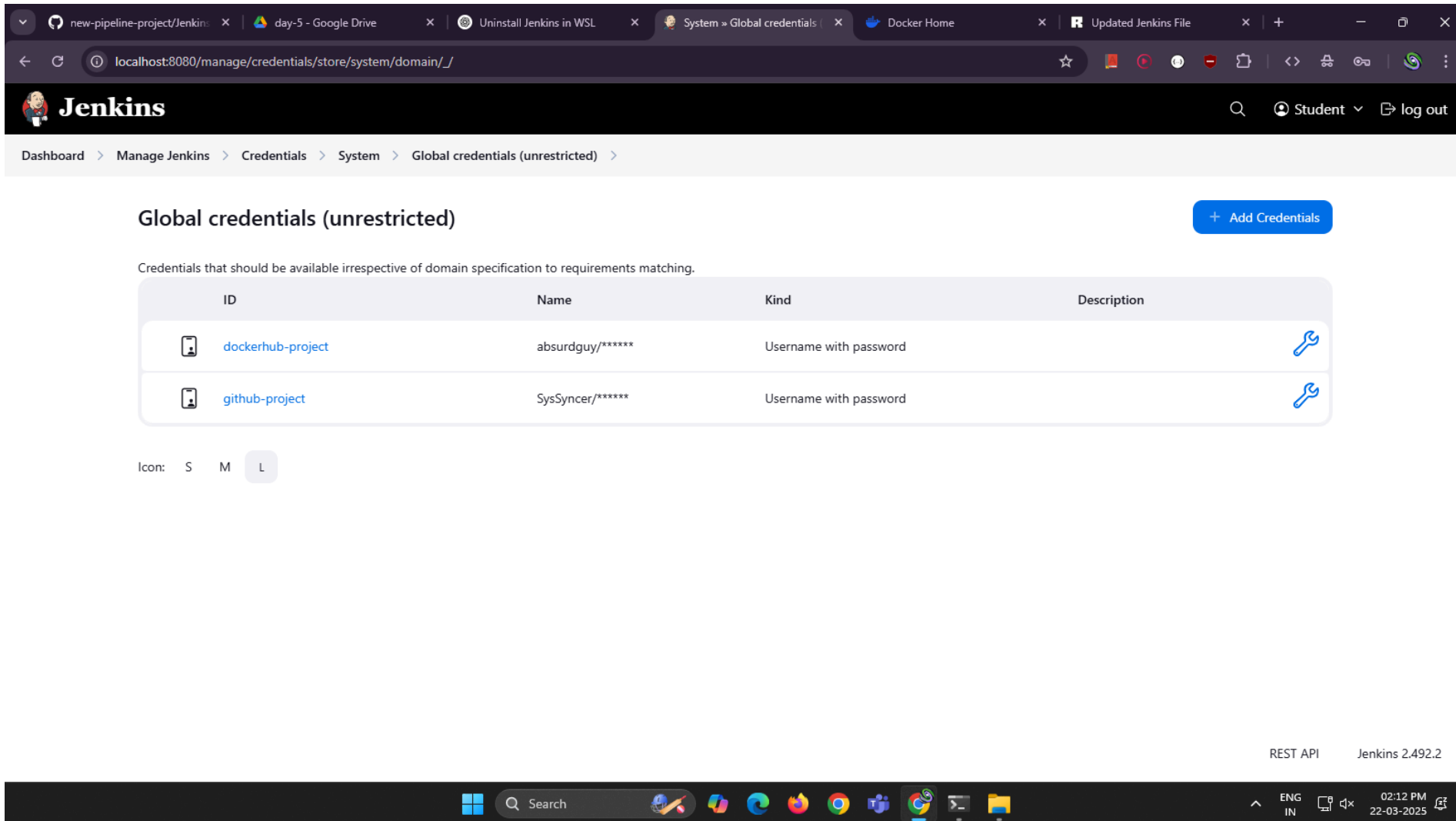
Loading plugin extensions ✓ Success

→ [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



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/manage/credentials/store/system/domain/_/`. The page title is "Global credentials (unrestricted)". A blue button labeled "+ Add Credentials" is in the top right. Below the title, a text line states: "Credentials that should be available irrespective of domain specification to requirements matching." A table lists two credentials:

ID	Name	Kind	Description
 dockerhub-project	absurdguy/*****	Username with password	
 github-project	SysSyncer/*****	Username with password	

Below the table, there is a filter section labeled "Icon:" with buttons for "S", "M", and "L", where "L" is selected. At the bottom right of the page, it says "REST API" and "Jenkins 2.492.2". The Windows taskbar at the bottom shows the time as 02:12 PM on 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 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 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 terminal window on the right shows the content of the Jenkinsfile, which defines a pipeline with stages for 'Checkout Code', 'Set Minikube Docker Environment', 'Build Backend Docker Image', and 'Build Frontend 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 Content:

```
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'
    }
  }

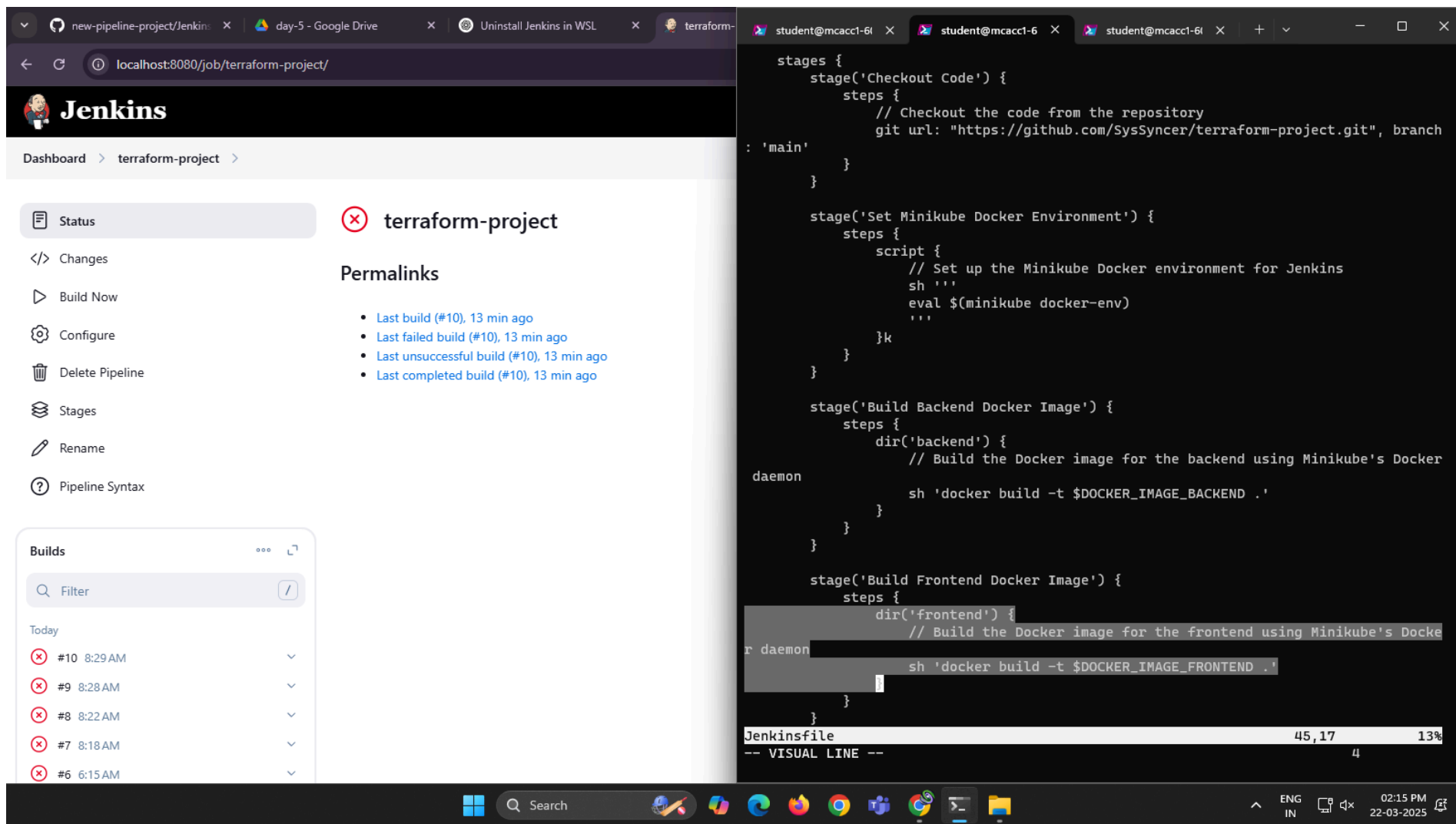
  stage('Set Minikube Docker Environment') {
    steps {
      script {
        // Set up the Minikube Docker environment for Jenkins
        sh '''
          eval $(minikube docker-env)
        '''
      }
    }
  }

  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
        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 .'
      }
    }
  }
}
```

Jenkinsfile 36,18 9%
-- VISUAL LINE -- 4

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 three stages: 'Checkout Code', 'Set Minikube Docker Environment', and 'Build Backend Docker Image'. The 'Build Backend Docker Image' stage is currently active, showing the directory change to 'backend' and the execution of the 'docker build' command. The terminal also shows the file size and line count of the Jenkinsfile (45,17 lines, 13% progress).

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 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)
        '''
      }
    }
  }

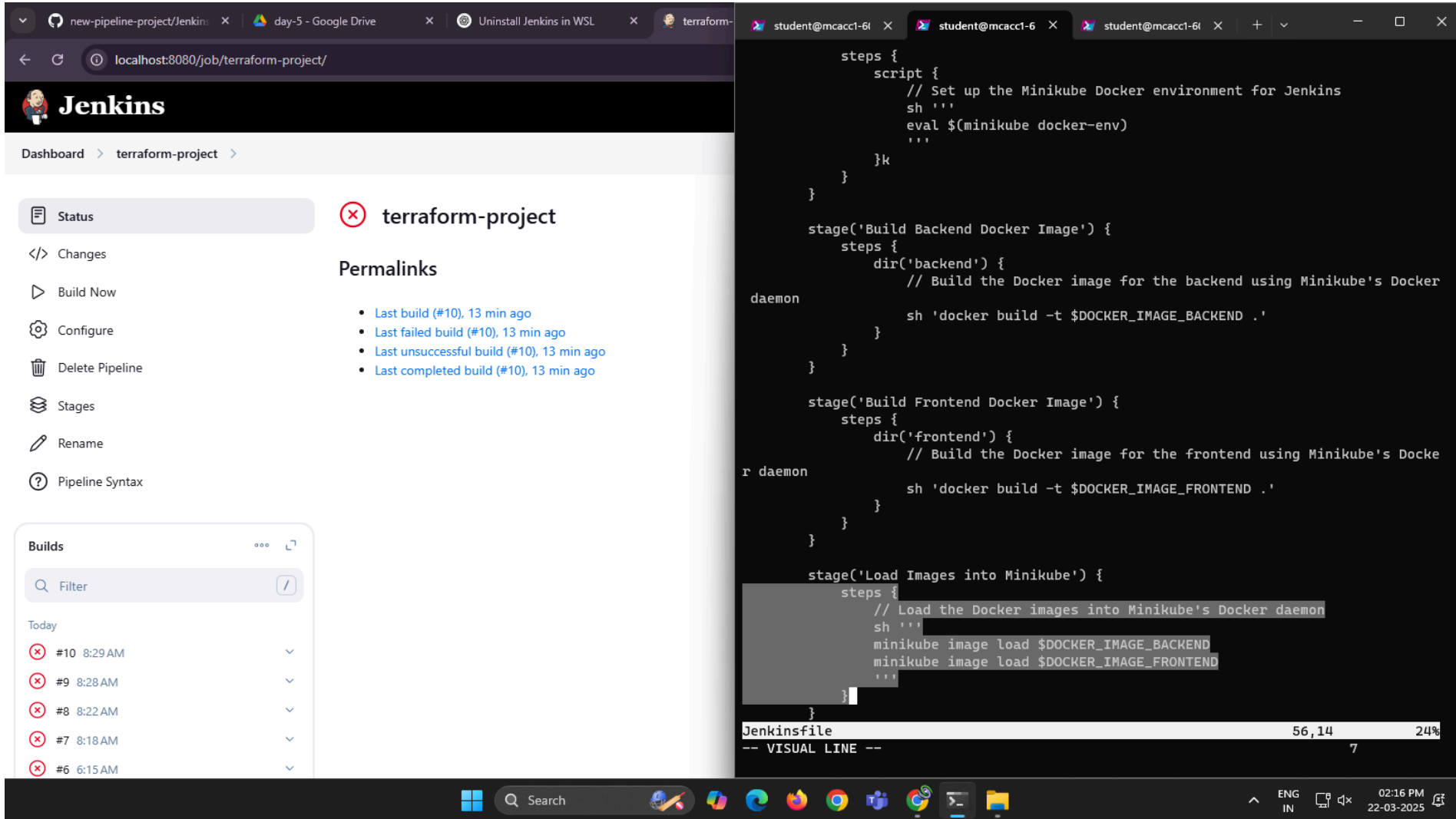
  stage('Build Backend Docker Image') {
    steps {
      dir('backend') {
        // Build the Docker image for the backend using Minikube's Docker
daemon
        sh 'docker build -t $DOCKER_IMAGE_BACKEND .'
      }
    }
  }

  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 'x' icon indicating a failure. The 'Builds' section lists several failed builds (#6 to #10). The code editor on the right shows the Jenkinsfile with the following content:

```
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 .'
    }
  }

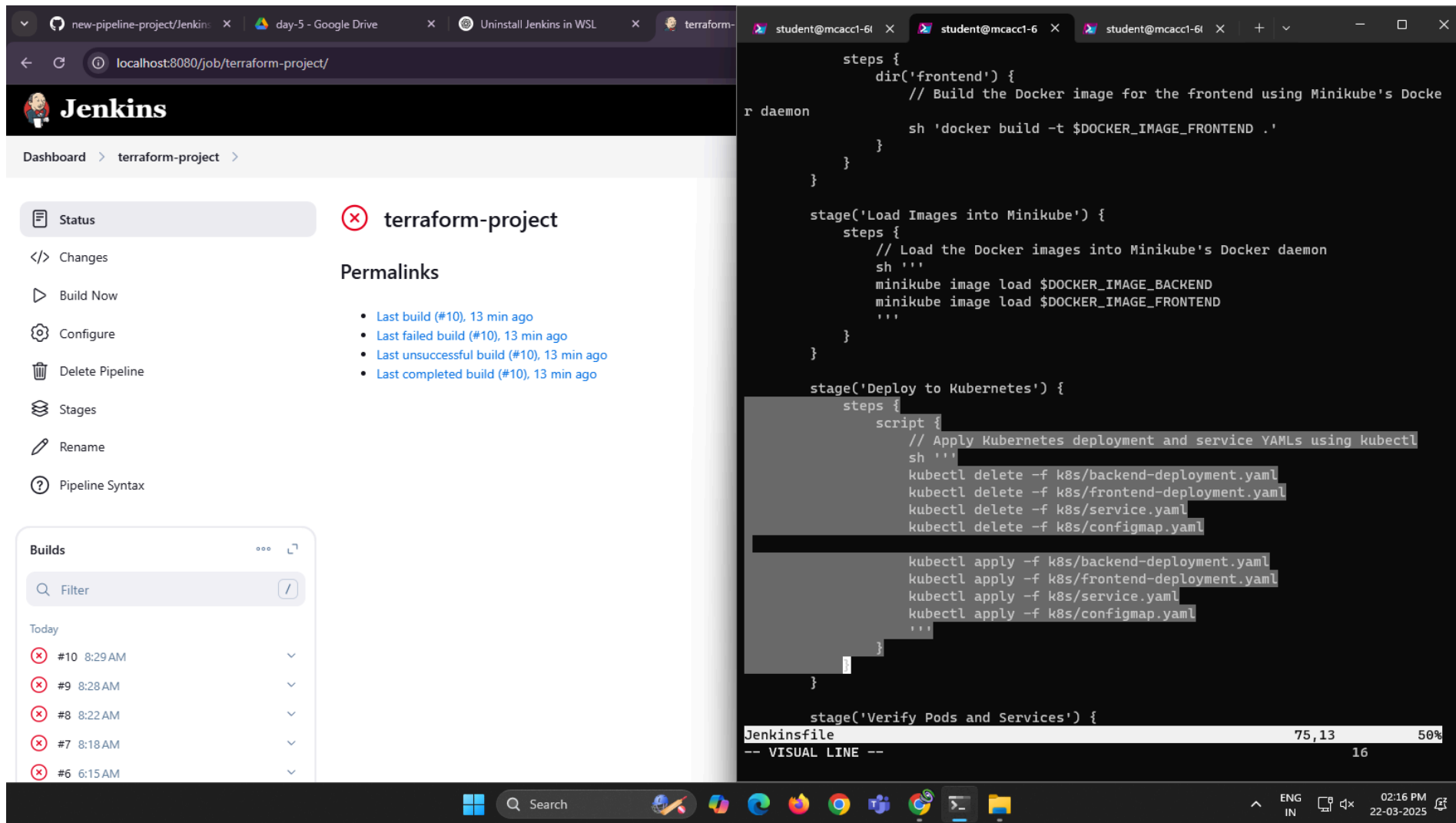
  stage('Build Frontend Docker Image') {
    steps {
      dir('frontend') {
        // Build the Docker image for the frontend using Minikube's Docker
        r daemon
        sh 'docker build -t $DOCKER_IMAGE_FRONTEND .'
      }
    }

    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
```

The code editor also shows a Windows taskbar at the bottom with the system clock at 02:16 PM on 22-03-2025.

Updating the Jenkinsfile: Step 7



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. The 'Builds' section lists several failed builds (#6 to #10). The terminal window shows a Jenkinsfile with stages for building Docker images, loading them into Minikube, and deploying to Kubernetes. The terminal also shows a progress bar for the Jenkinsfile update.

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

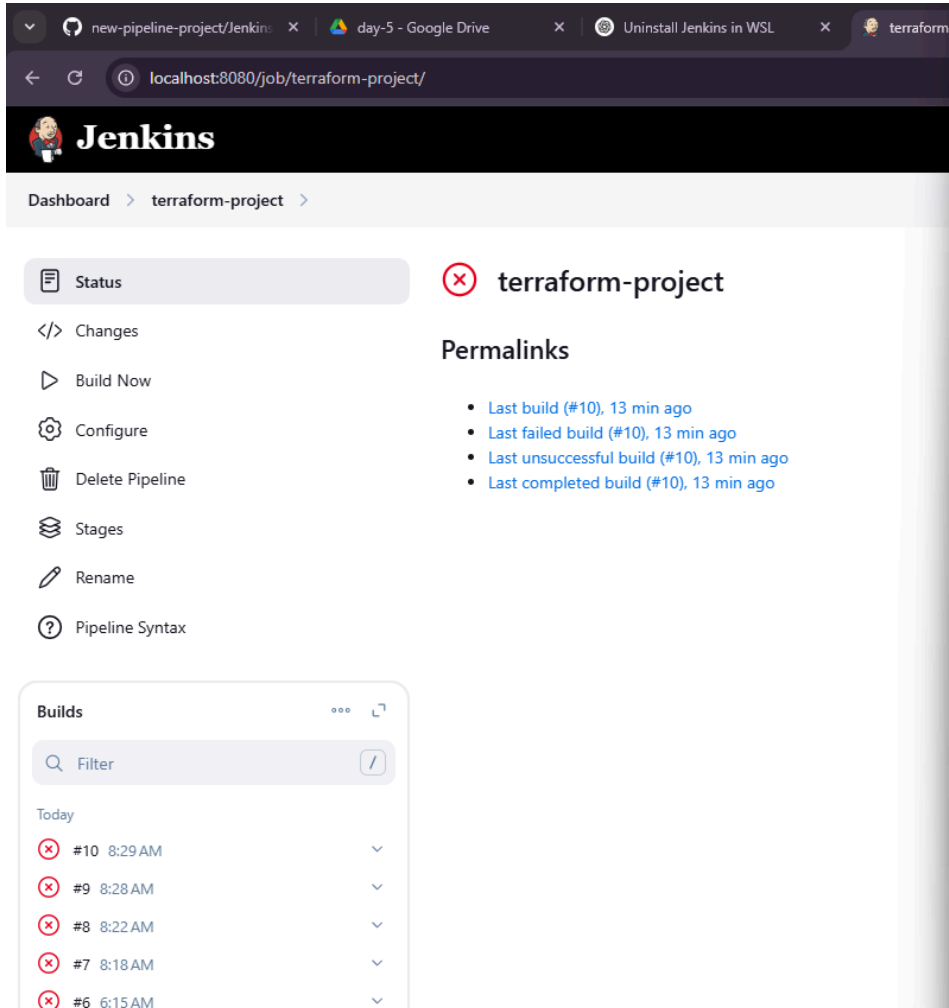
      kubectl apply -f k8s/backend-deployment.yaml
      kubectl apply -f k8s/frontend-deployment.yaml
      kubectl apply -f k8s/service.yaml
      kubectl apply -f k8s/configmap.yaml
      '''
    }
  }
}

stage('Verify Pods and Services') {
```

Progress Bar:

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

Updating the Jenkinsfile: Step 8



The screenshot shows the Jenkins web interface for a pipeline named 'terraform-project'. The interface includes a sidebar with navigation options: Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main area displays the pipeline status as 'Failed' (indicated by a red 'x' icon) and provides a list of 'Permalinks' for the last build (#10), including links for 'Last build', 'Last failed build', 'Last unsuccessful build', and 'Last completed build'. A 'Builds' section at the bottom shows a list of recent builds, all of which failed (indicated by red 'x' icons).

new-pipeline-project/Jenkins x day-5 - Google Drive x Uninstall Jenkins in WSL x terraform-

localhost:8080/job/terraform-project/

Jenkins

Dashboard > terraform-project >

Status

Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

terraform-project

Permalinks

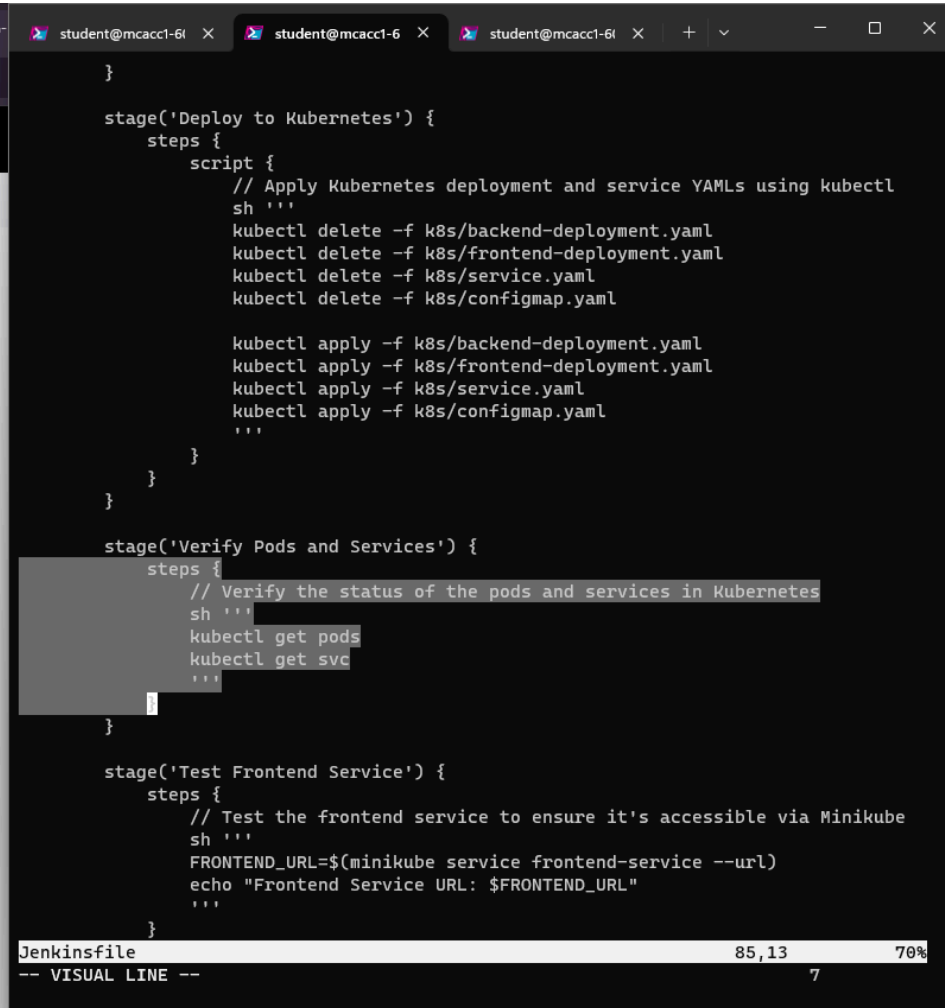
- Last build (#10), 13 min ago
- Last failed build (#10), 13 min ago
- Last unsuccessful build (#10), 13 min ago
- Last completed build (#10), 13 min ago

Builds

Filter

Today

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



The screenshot shows a code editor with the Jenkinsfile content. The file is named 'Jenkinsfile' and has a line count of 85,13 and a 70% completion status. The code defines two stages: 'Deploy to Kubernetes' and 'Verify Pods and Services'. The 'Deploy to Kubernetes' stage includes steps to delete and apply Kubernetes manifests. The 'Verify Pods and Services' stage includes a step to verify the status of the pods and services in Kubernetes. The 'Test Frontend Service' stage includes a step to test the frontend service to ensure it's accessible via Minikube.

```

}

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
    '''
  }
}

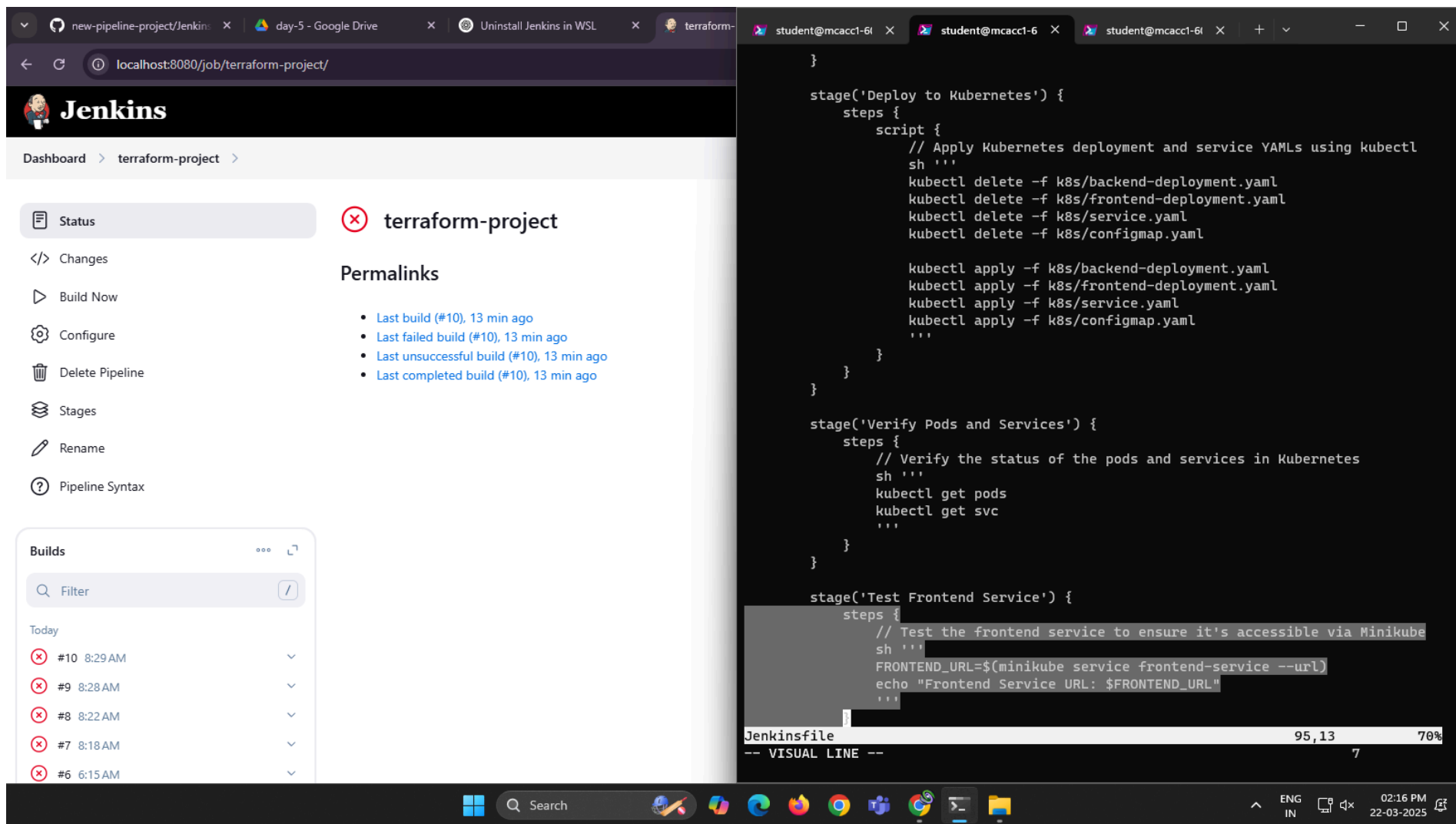
stage('Test Frontend Service') {
  steps {
    // Test the frontend service to ensure it's accessible via Minikube
    sh '''
      FRONTEND_URL=$(minikube service frontend-service --url)
      echo "Frontend Service URL: $FRONTEND_URL"
    '''
  }
}

```

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 interface 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 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

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
      '''
    }
  }
}

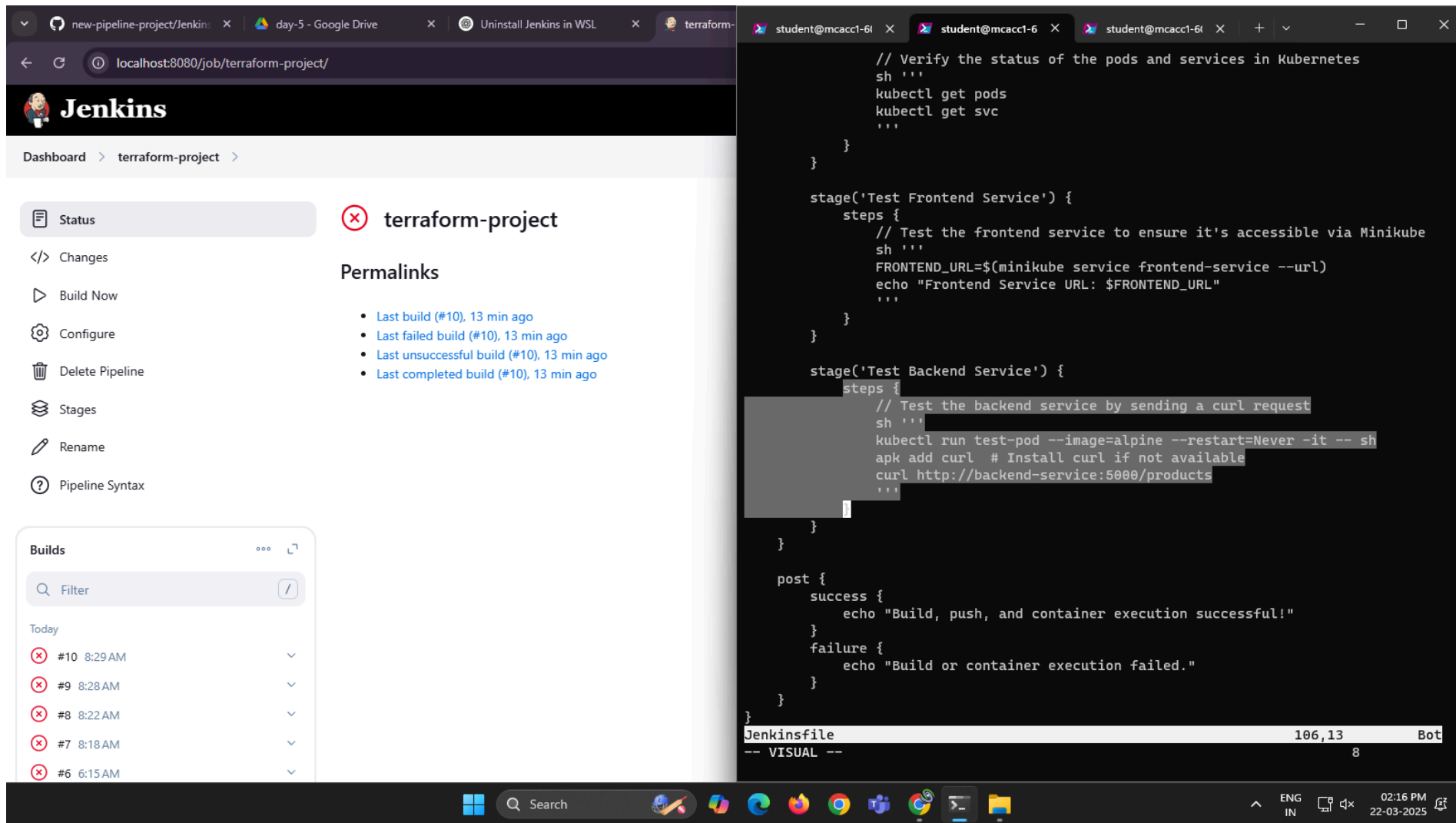
stage('Verify Pods and Services') {
  steps {
    // Verify the status of the pods and services in Kubernetes
    sh '''
      kubectl get pods
      kubectl get svc
    '''
  }
}

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 UI 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 shows the Jenkinsfile being updated with the new stage.

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 '''
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."
  }
}
```

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 build #10. The terminal window shows the Jenkinsfile being edited, with a new stage 'Test Frontend Service' and a 'post' block for success and failure messages.

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 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 right pane shows the details for the `Deploy to Kubernetes` step, which failed. It includes a timeline: `Started 1.1 sec ago`, `Queued 0 ms`, `Took 0.53 sec`, and `Failure`. A link `View as plain text` is provided. The console output shows a red error banner: `kubectl apply -f k8s/backend-deployment.yaml kubectl apply -f k8s/frontend-deployment.y...` with 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 output continues with `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 of the console area. The Windows taskbar is visible at the very bottom with the date `22-03-2025` and time `02:12 PM`.

new-pipeline-project/Jenkins x day-5 - Google Drive x Uninstall Jenkins in WSL x Build log [#10] [Jenkins] x Docker Home x Updated Jenkins File x

localhost:8080/job/terraform-project/10/pipeline-console/

Jenkins

Dashboard > terraform-project > #10 > Pipeline Console

< Build #10

Rebuild Overview Configure

In progress 8.5 sec ago in 8.5 sec and counting

- 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
- Post Actions

Started 1.1 sec ago

Queued 0 ms

Took 0.53 sec

Failure

Running on Jenkins

View as plain text

kubectl apply -f k8s/backend-deployment.yaml kubectl apply -f k8s/frontend-deployment.y... 0.3 sec

Shell Script

```
0 + kubectl apply -f k8s/backend-deployment.yaml
1 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;'>
  Authentication required
  <!--
```

Scroll to Bottom

Jenkins 2.492.2

ENG IN 02:12 PM 22-03-2025

End of day 5 assignment