St. Francis Institute of Technology, Mumbai-400 103

**Department Of Information Technology**

A.Y. 2024-2025

Class: TE-ITA/B, Semester: V

Subject: **DevOps Lab**

**Experiment – 7: a. To build pipeline of jobs in Jenkins, create a pipeline script to test and deploy an application.**

**b. To automatically build a job in Jenkins using webhooks (Topic Beyond Syllabus)**

1. **Aim:** To build pipeline of jobs in Jenkins, create a pipeline script to test and deploy an application
2. **Objectives:** Aim of this experiment is that, the students will be able

* To build pipeline of jobs in Jenkins, create a pipeline script to test and deploy an application

1. **Outcomes:** After study of this experiment, the students will be able

* To understand the importance of Jenkins to Build and deploy Software

Applications on server environment.

1. **Prerequisite:** Knowledge of software engineering concept of integration and deployment
2. **Requirements:** Jenkins,JDK, python,Personal Computer, Windows operating system, browser, Internet Connection, Microsoft Word.
3. **Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

1. **Laboratory Exercise**
   * + 1. **Procedure:**

**a. Answer the following:**

* What is Jenkins pipeline?
* What are the different ways to write a Jenkins pipeline?

**b**. **Execute following (Refer the shared material) and attach screenshots:**

* Create and build pipeline project with Git
* Create and build pipeline project with pipeline script
* Create and automatically build a pipeline project using webhooks

1. **Post-Experiments Exercise**
2. **Extended Theory:**

Nil

1. **Questions:**

* Explain the types of agents in a Jenkinsfile?
* What are webhooks?

1. **Conclusion:**

* Write what was performed in the experiment.
* Write the significance of the topic studied in the experiment.

1. **References:**

<https://jenkins.io/doc/>

<https://www.jenkins.io/doc/book/pipeline/syntax/>

<https://www.edureka.co/blog/jenkins-pipeline-tutorial-continuous-delivery>

<https://www.slideshare.net/abediaz/introduction-to-jenkins>

https://www.slideshare.net/jph98/jenkins-ci-presentation

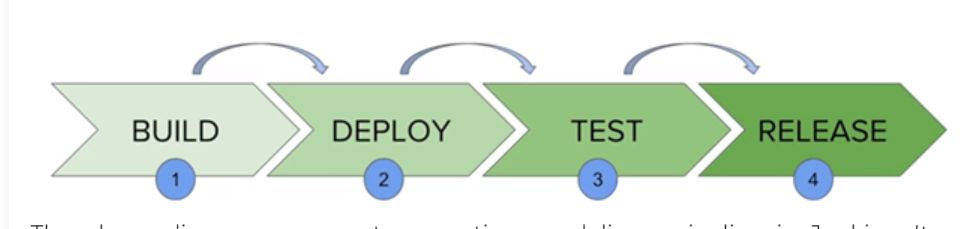
**9. Laboratory Exercise**

B. Answer the following:

1. **What is Jenkins pipeline?**

**Ans:**

1. In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.
2. It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.
3. In other words, a Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow.
4. A pipeline has an extensible automation server for creating simple or even complex delivery pipelines "as code", via DSL (Domain-specific language).



The above diagram represents a continuous delivery pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release. These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

A continuous delivery pipeline is an automated expression to show your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its way to being released. It also involves developing the software in a repeatable and reliable manner, and progression of the built software through multiple stages of testing and deployment.

**2. What are the different ways to write a Jenkins pipeline?**

**Ans:** Jenkins offers two types of syntax to create pipelines: declarative and scripted. Declarative syntax has recently been added to Jenkins to make pipeline code richer and more readable.

**DECLARATIVE:**

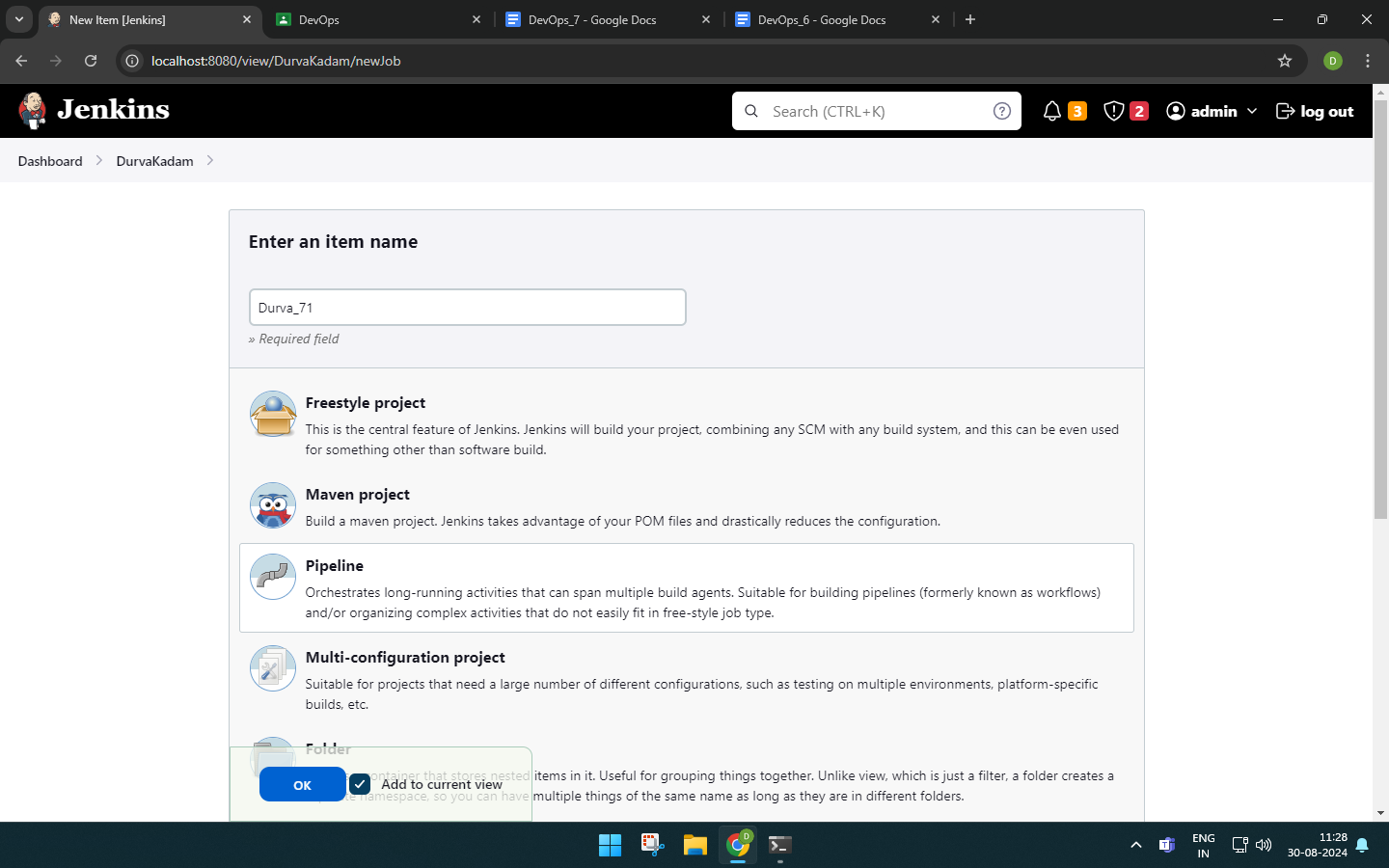
1. Structured Syntax: Uses a clear and organized format, divided into sections like pipeline, agent, stages, and steps.
2. Ease of Use: Simplifies the definition and management of CI/CD workflows, making it more readable and understandable.
3. Defined Stages: Breaks down the workflow into distinct phases (e.g., Build, Test, Deploy), enhancing visualization and organization.
4. Post Conditions: Supports actions to be taken after the pipeline execution, such as cleanup or notifications based on success or failure.
5. Encourages Best Practices: Promotes adherence to CI/CD best practices through its structured approach, making it ideal for standard tasks.

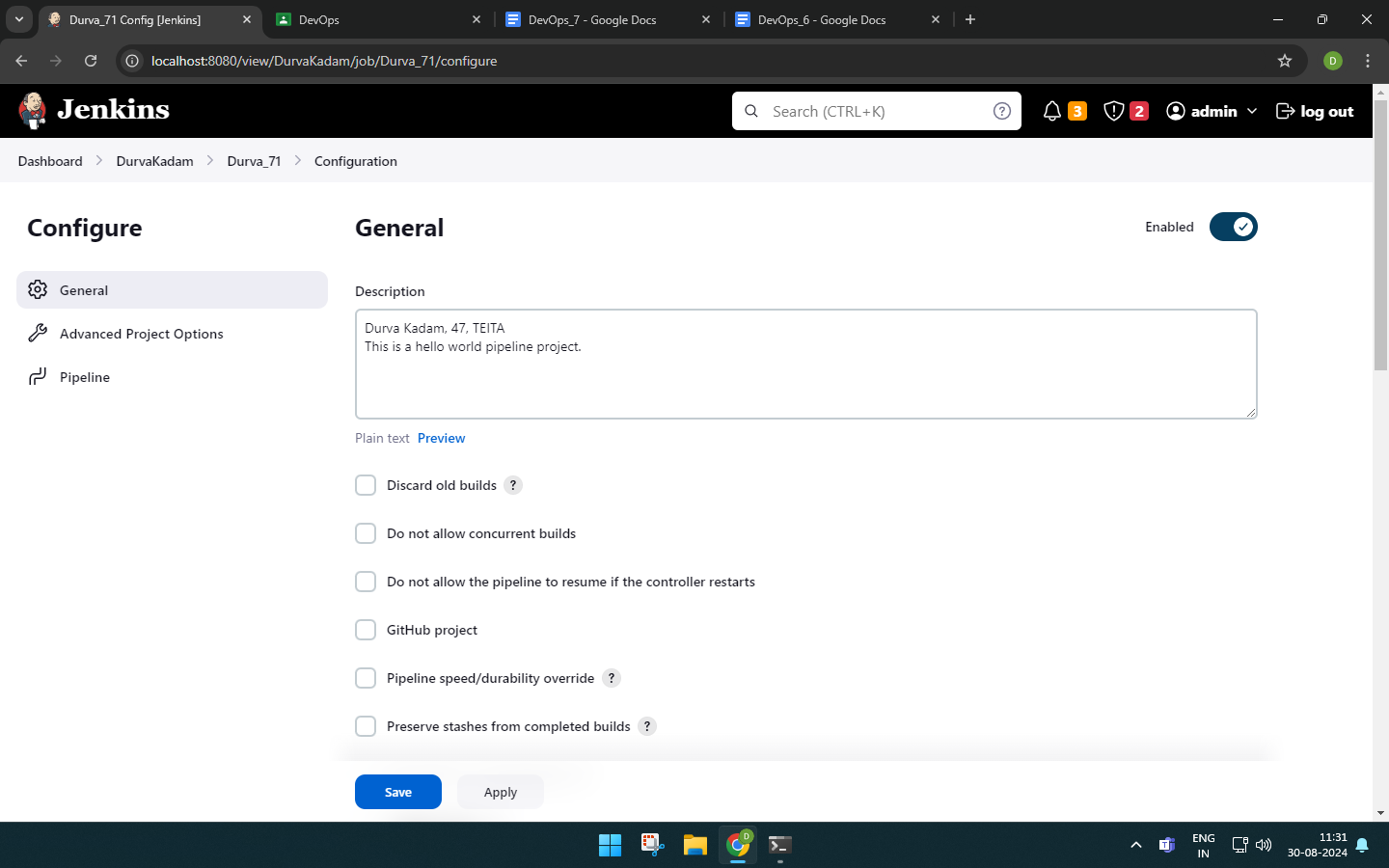
**SCRIPTED:**

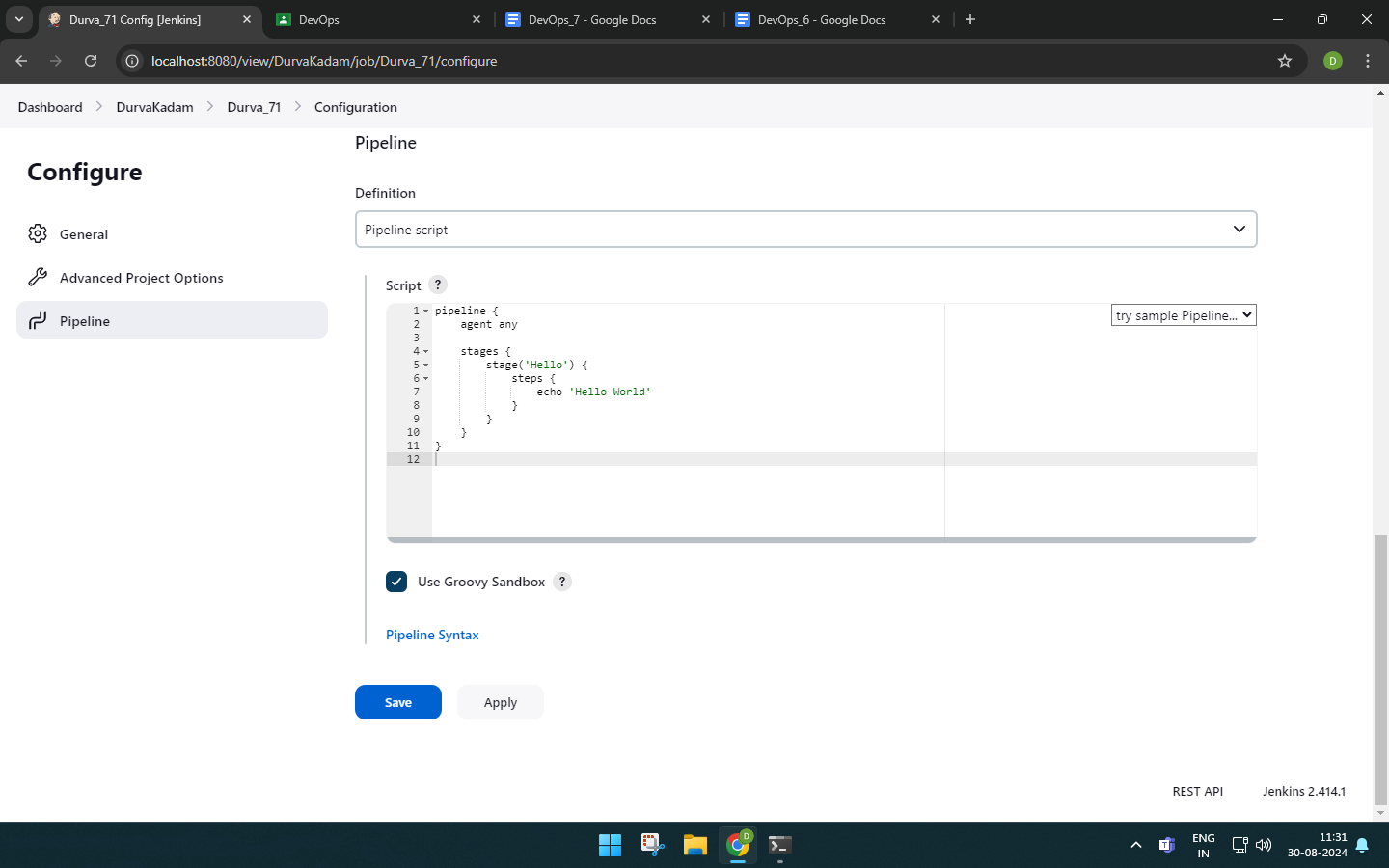
1. Flexible Syntax: Uses a free-form Groovy-based syntax, allowing for detailed programming constructs and complex logic.
2. Greater Control: Provides full control over the workflow, enabling intricate processes and customization.
3. Advanced Logic: Supports loops, conditionals, and other programming features, making it suitable for complex scenarios.
4. Less Structure: Offers flexibility in organization, but may lead to less readability if not managed properly.
5. Ideal for Experts: Best suited for experienced users who require advanced capabilities and custom solutions in their CI/CD workflows.

**1. Create and build pipeline project with pipeline script**

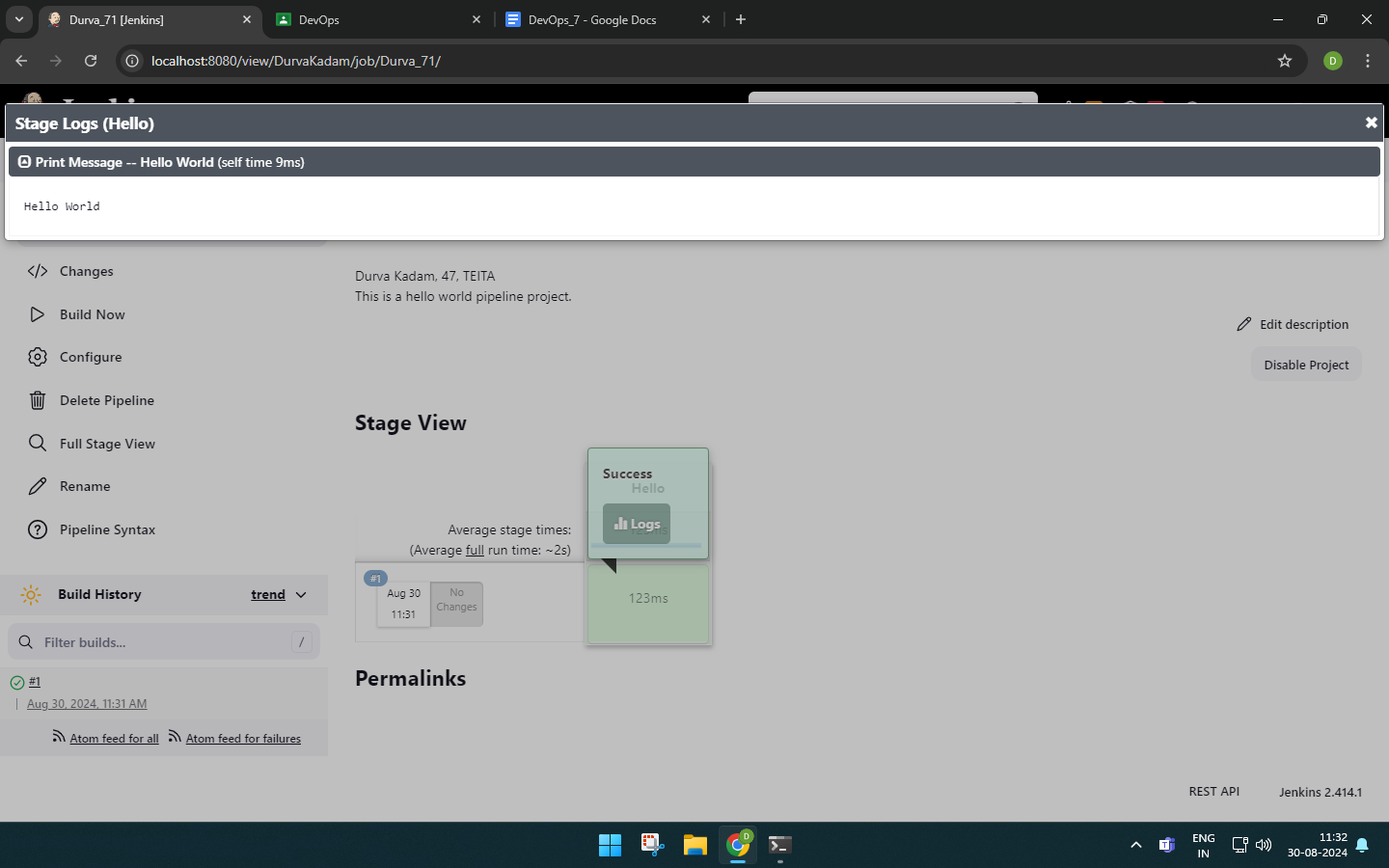
• Project with Hello World pipeline script



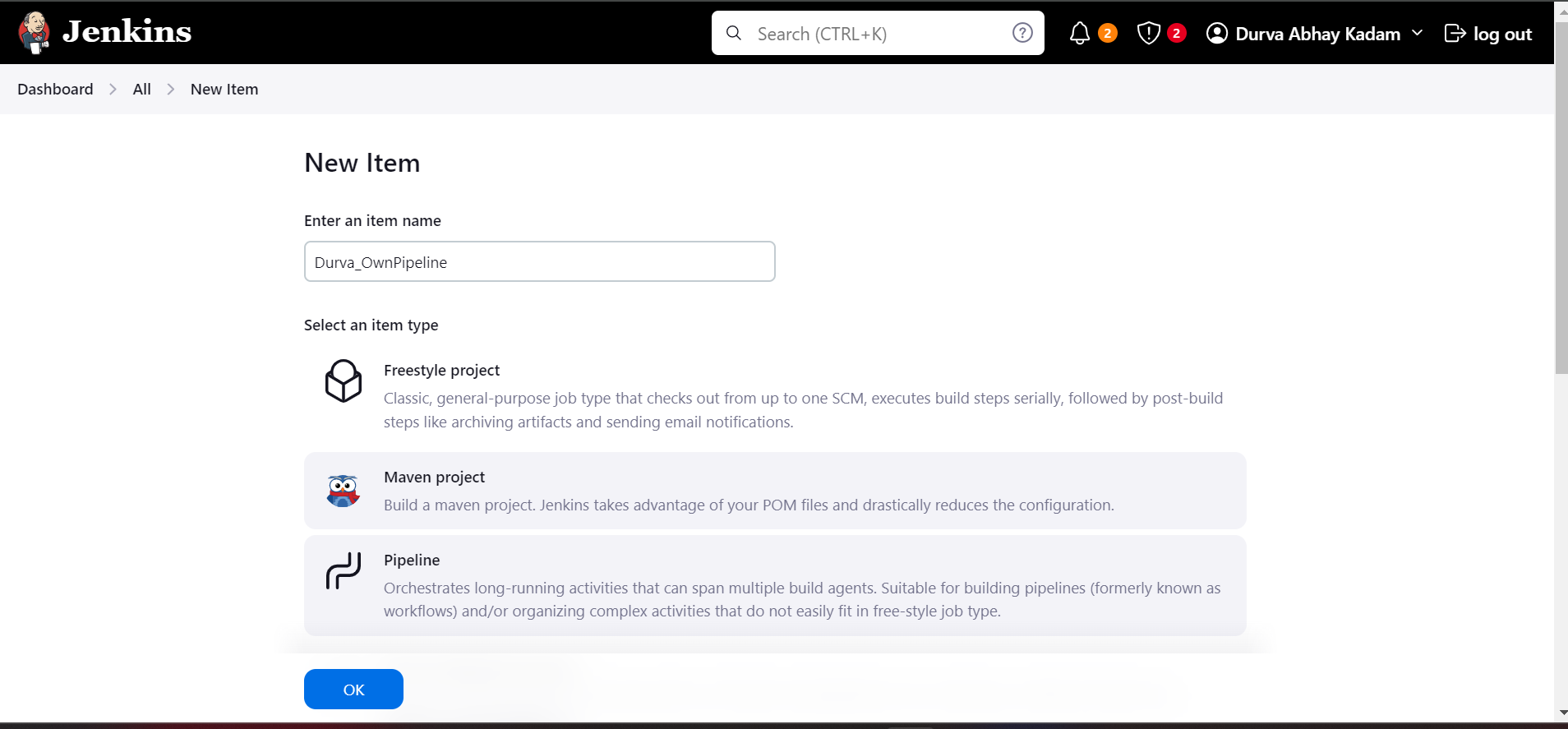




When we build it, we can see the script executed.



**• Project with your own pipeline script**

****

**My script:**

pipeline {

agent any

stages {

stage('Initialize') {

steps {

echo 'This is DURVA'

}

}

stage('Build') {

steps {

echo 'Building the application'

}

}

stage('Unit Tests') {

steps {

echo 'Running unit tests'

}

}

stage('Deploy to Dev') {

when {

branch 'develop'

}

steps {

echo 'Deploying to Development environment'

}

}

stage('Deploy to Production') {

when {

branch 'master'

}

steps {

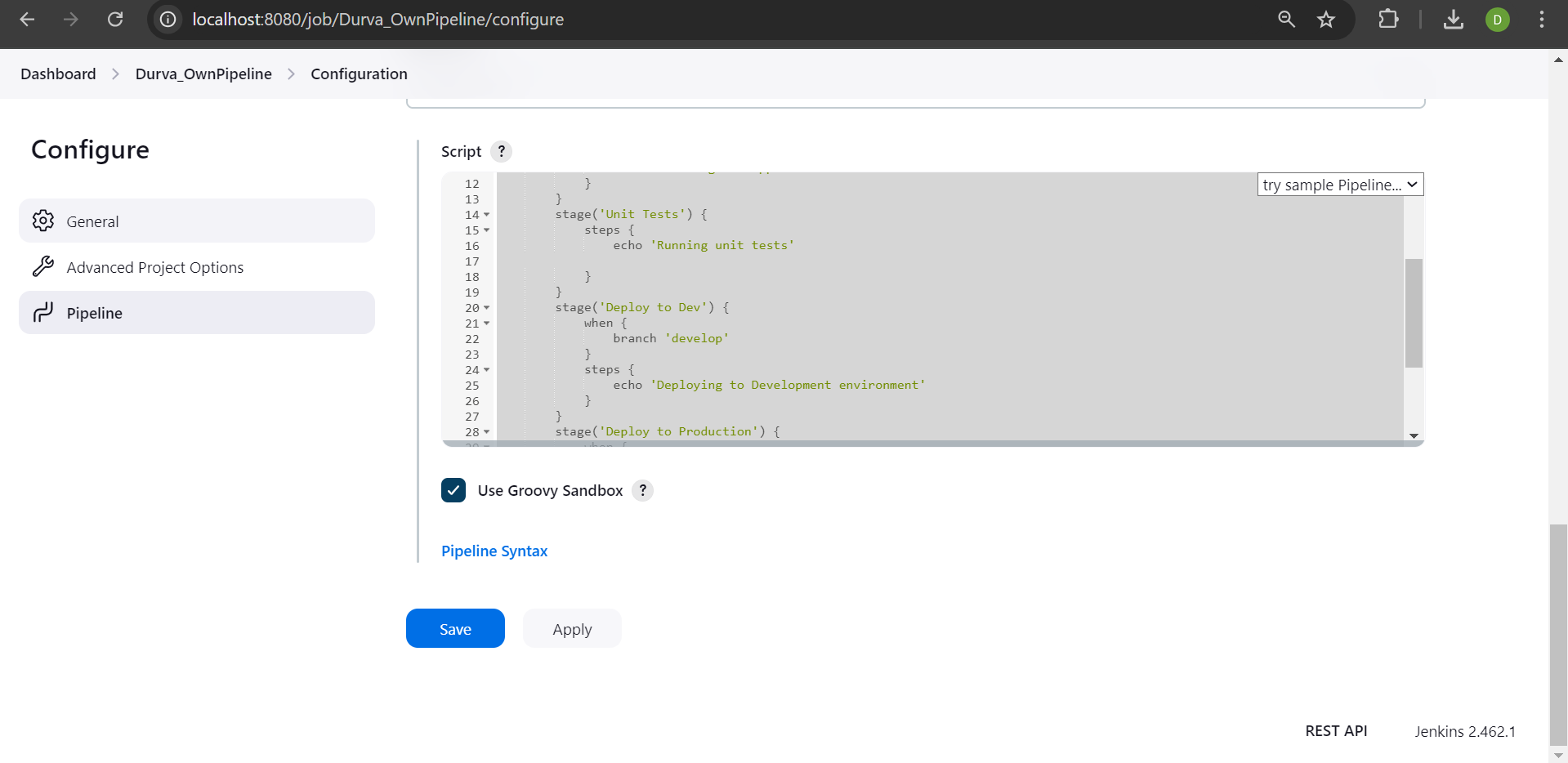
echo 'Deploying to Production environment'

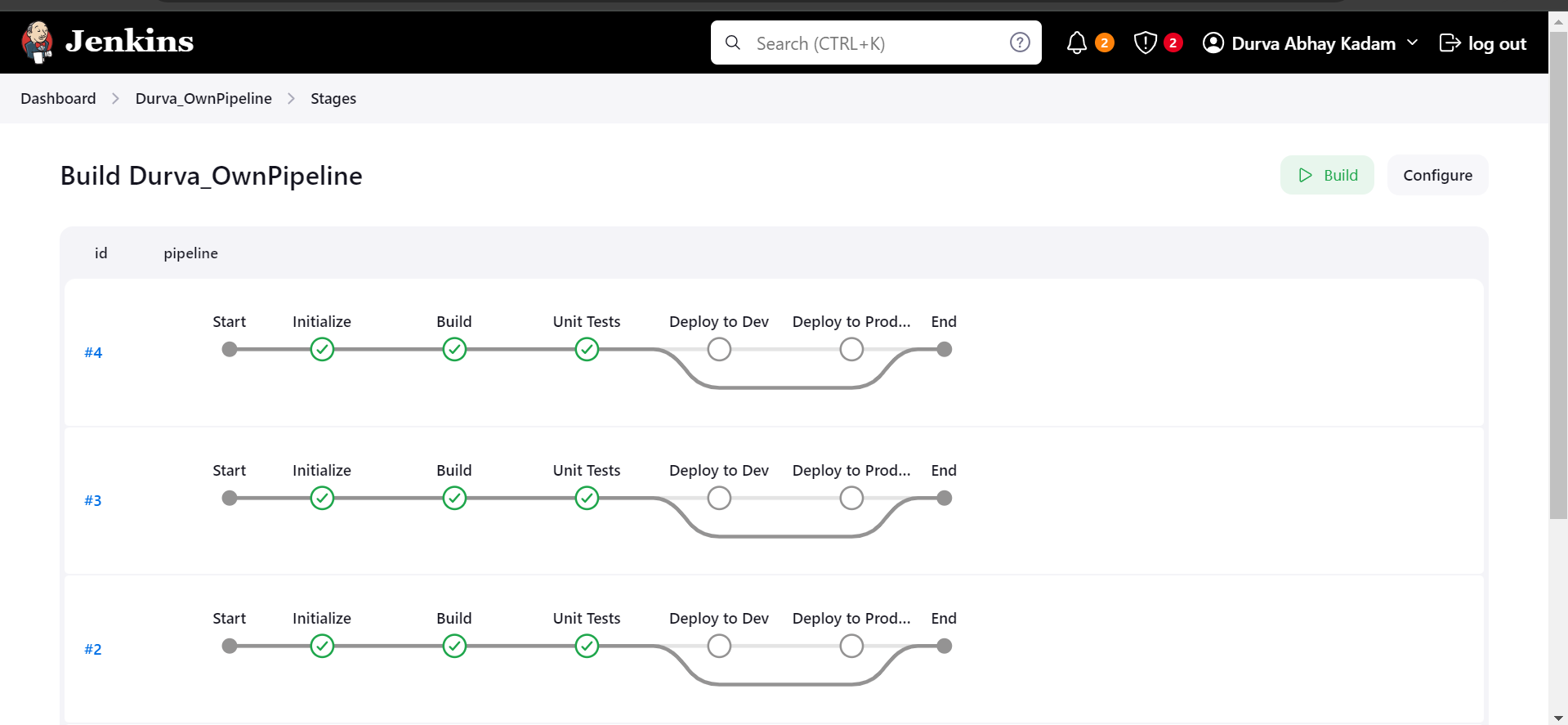
}

}

}

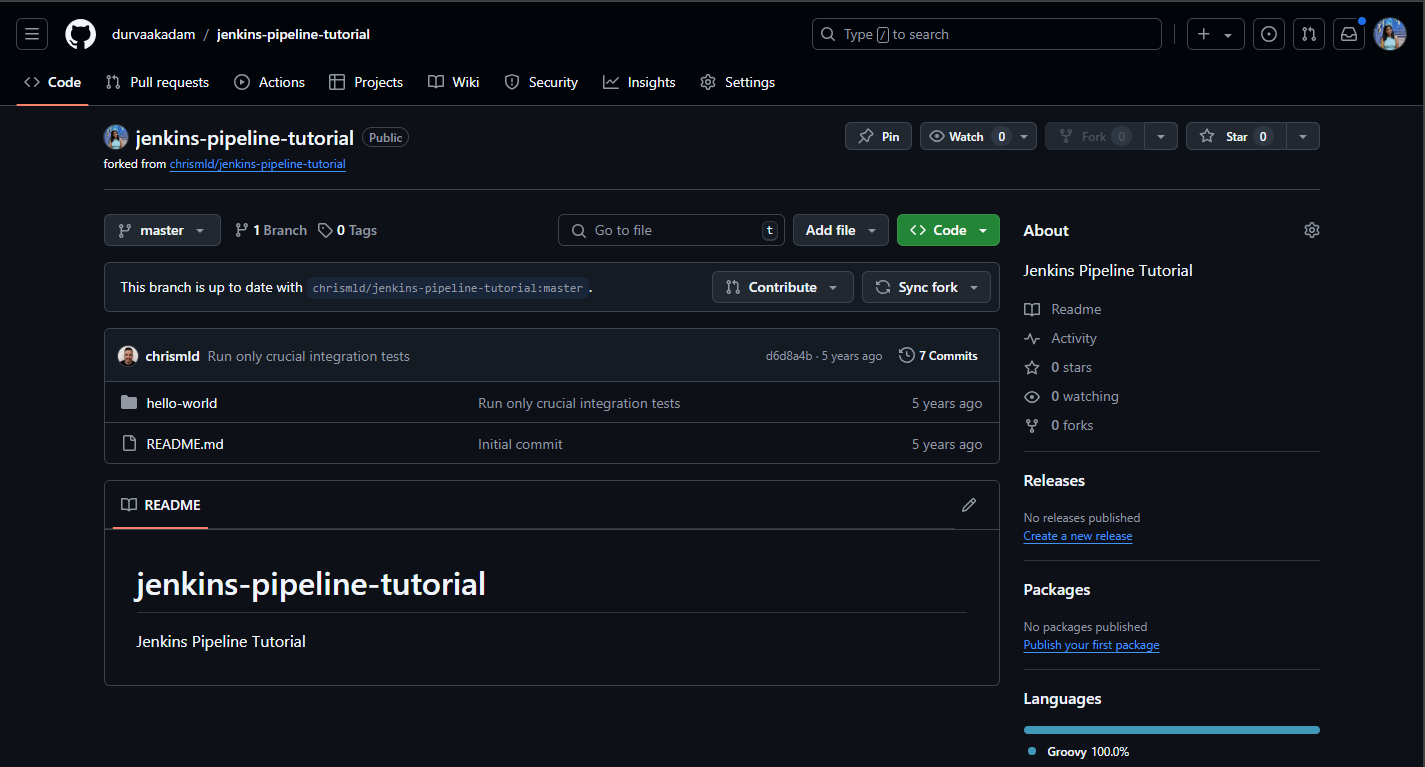
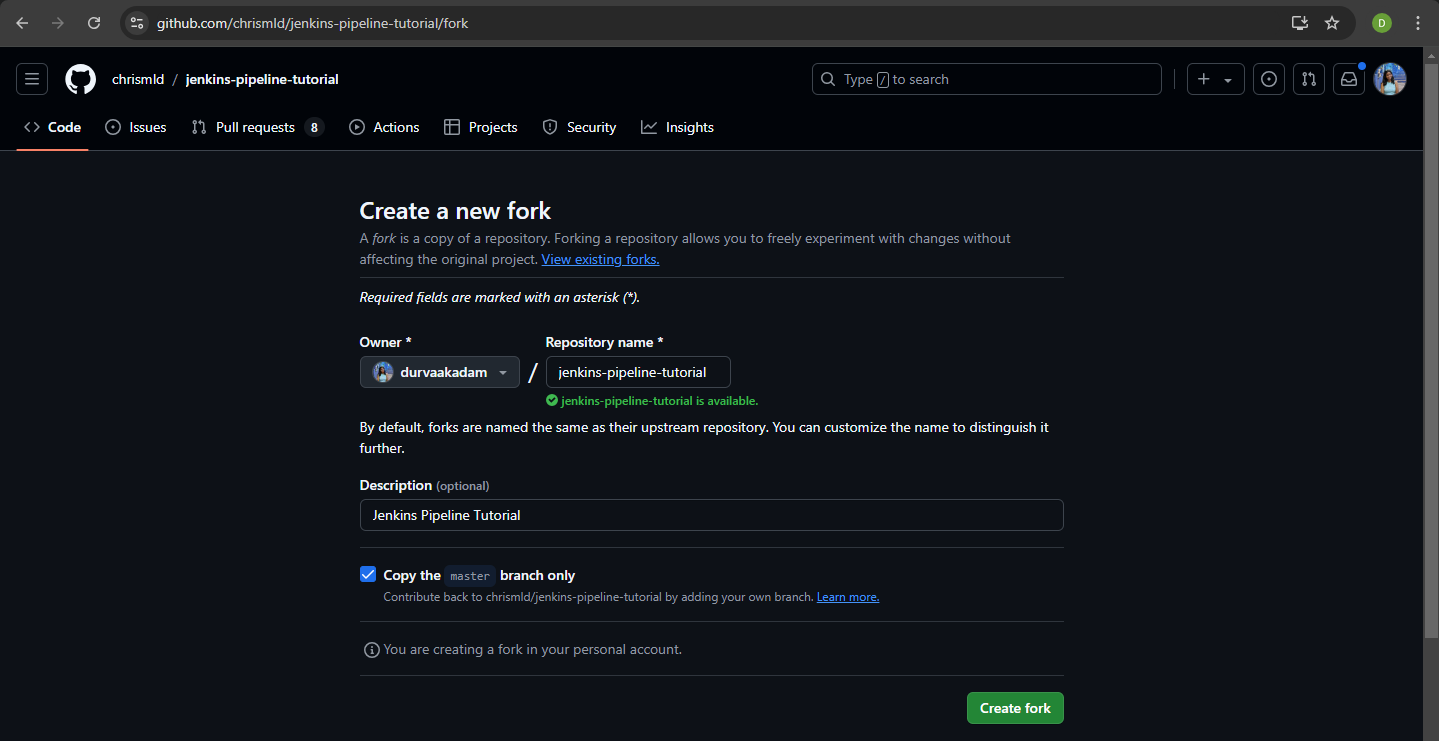
}



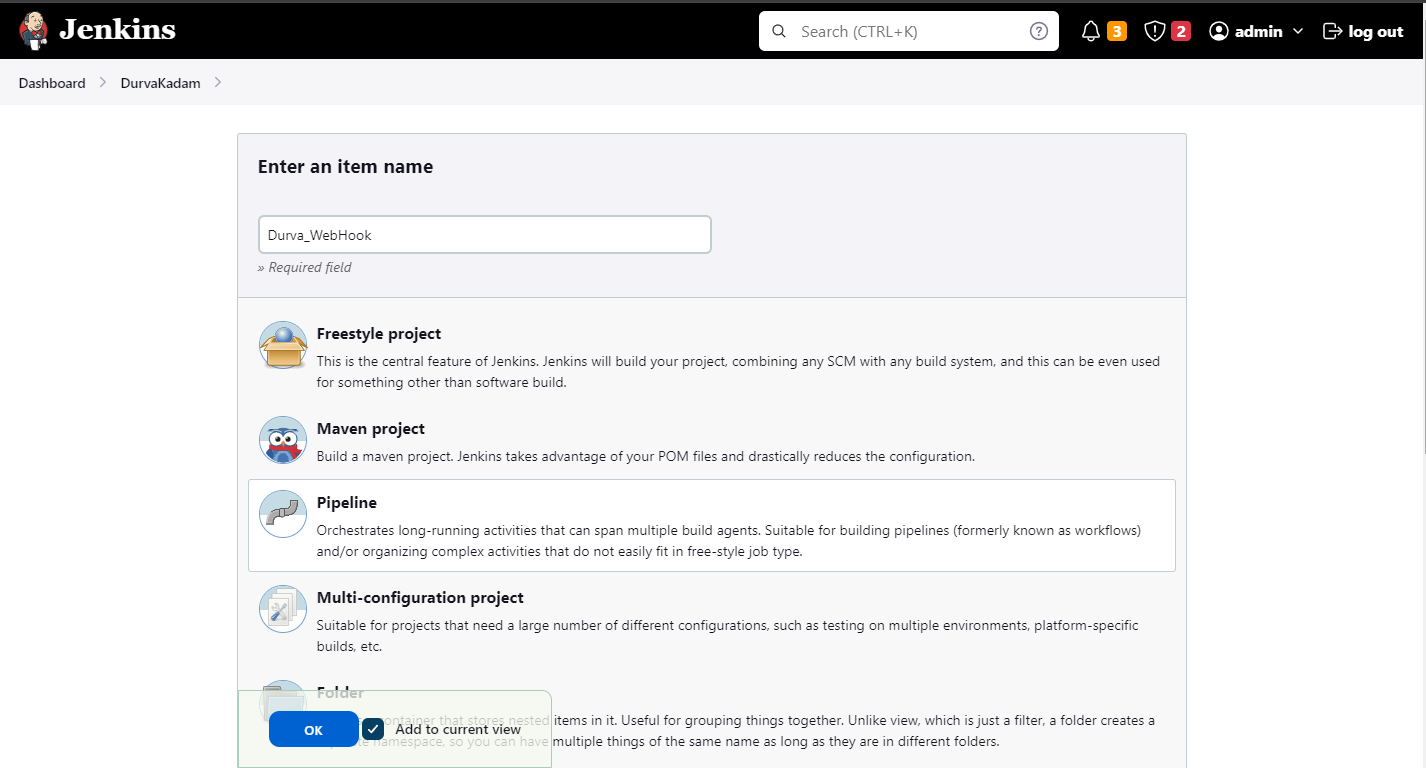


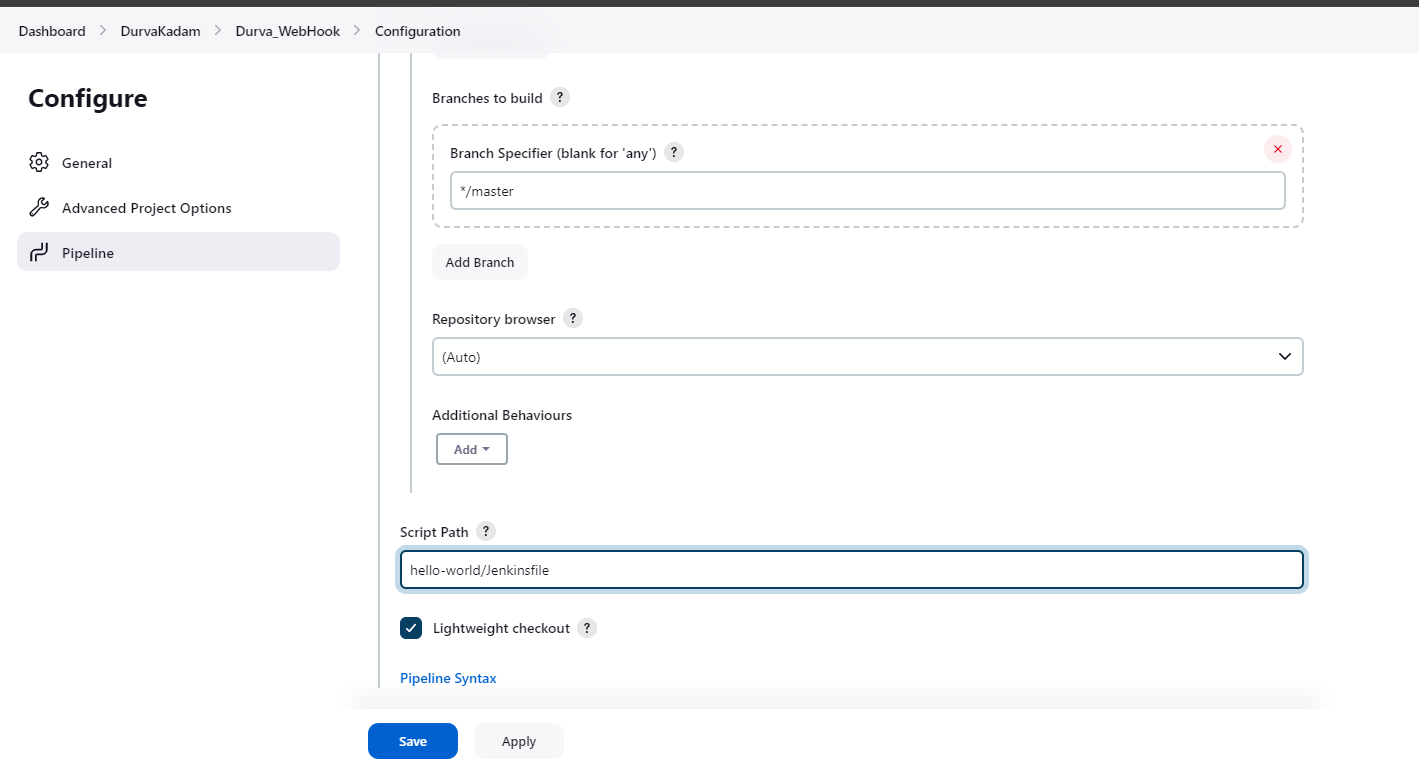
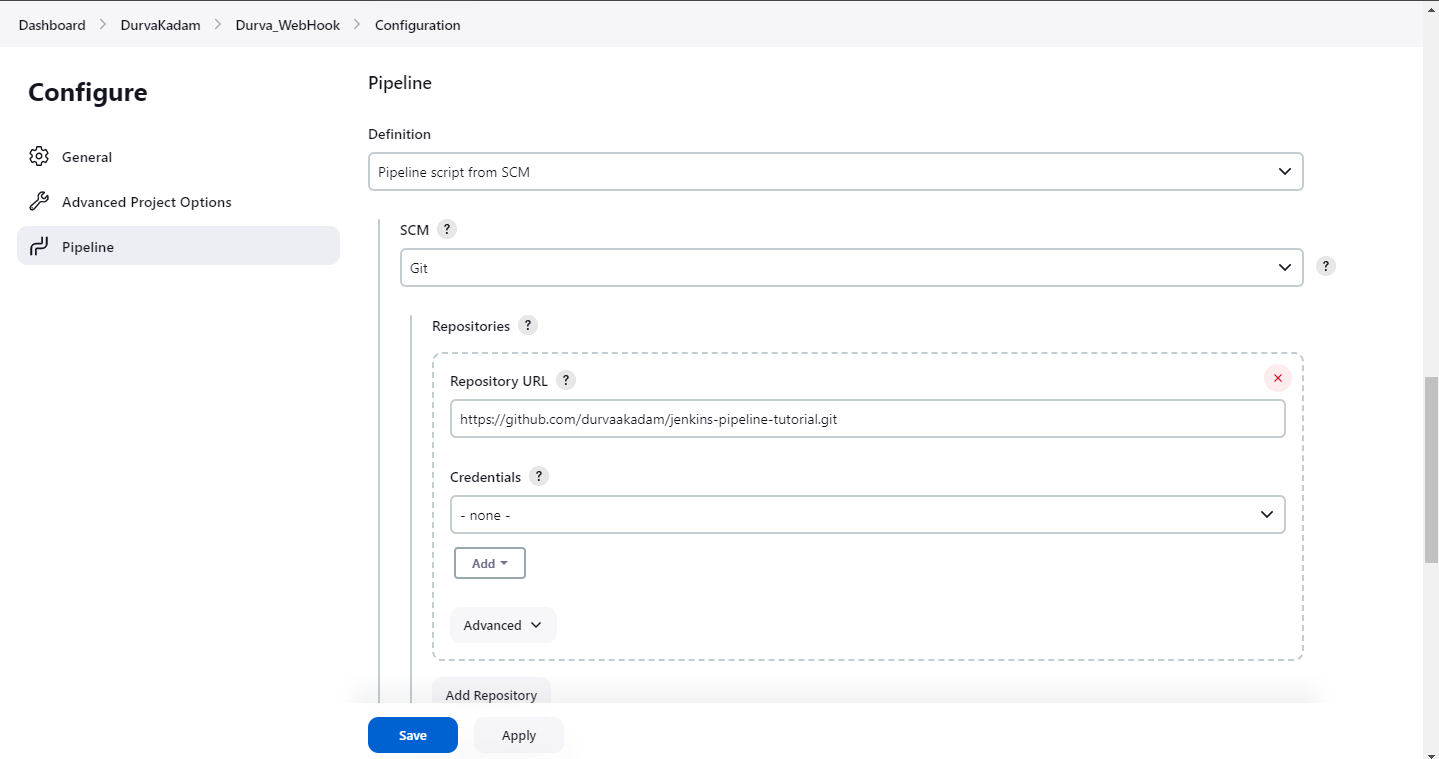
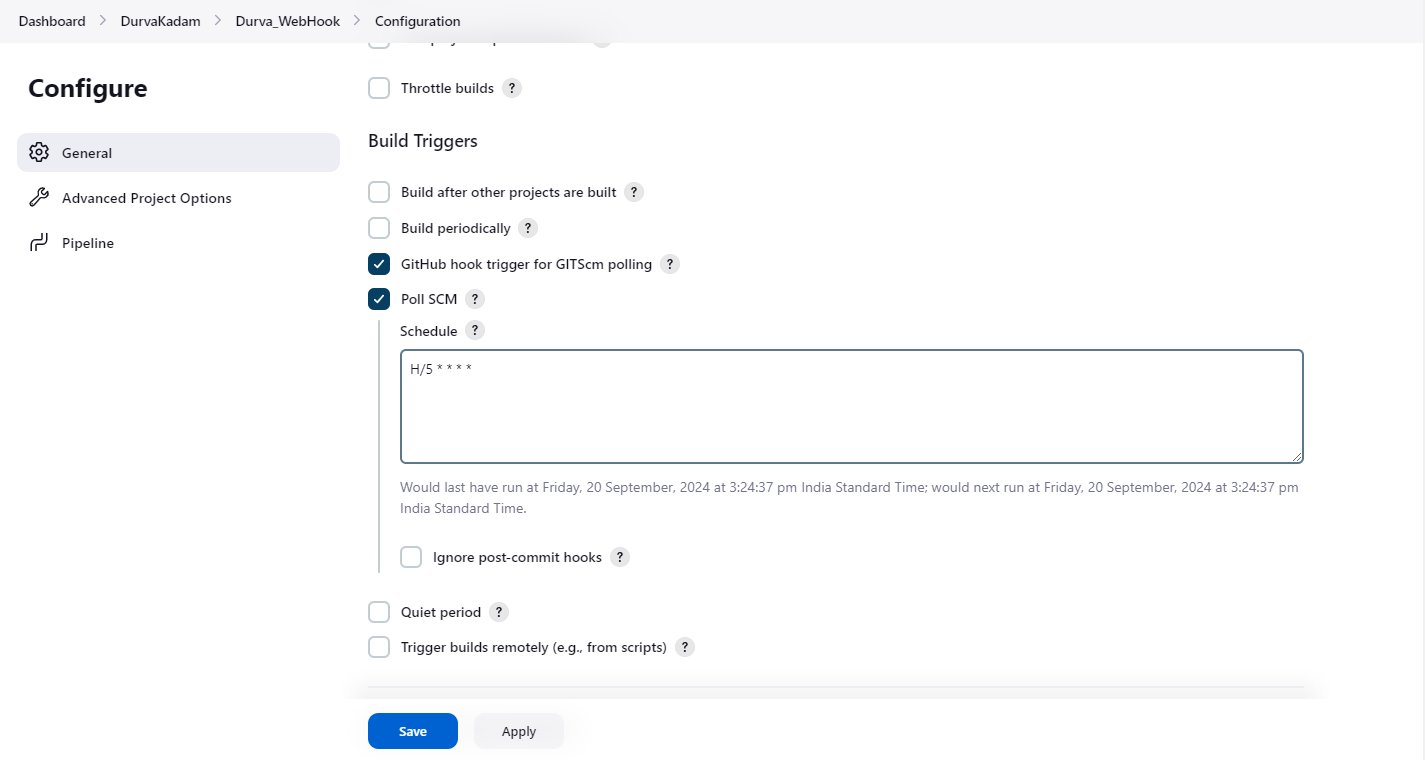
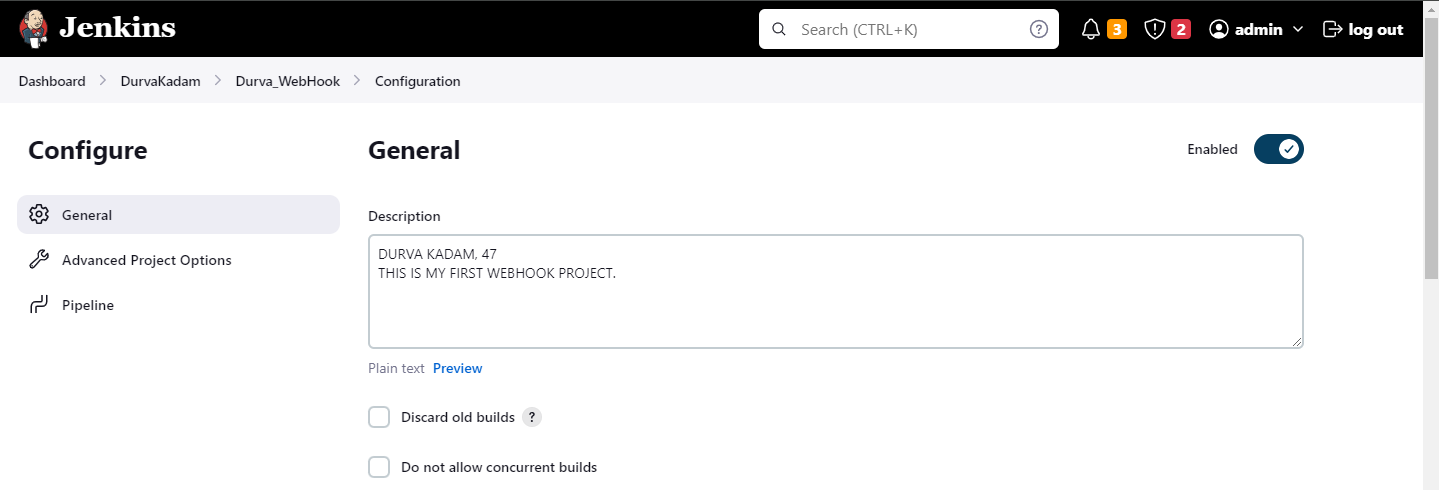
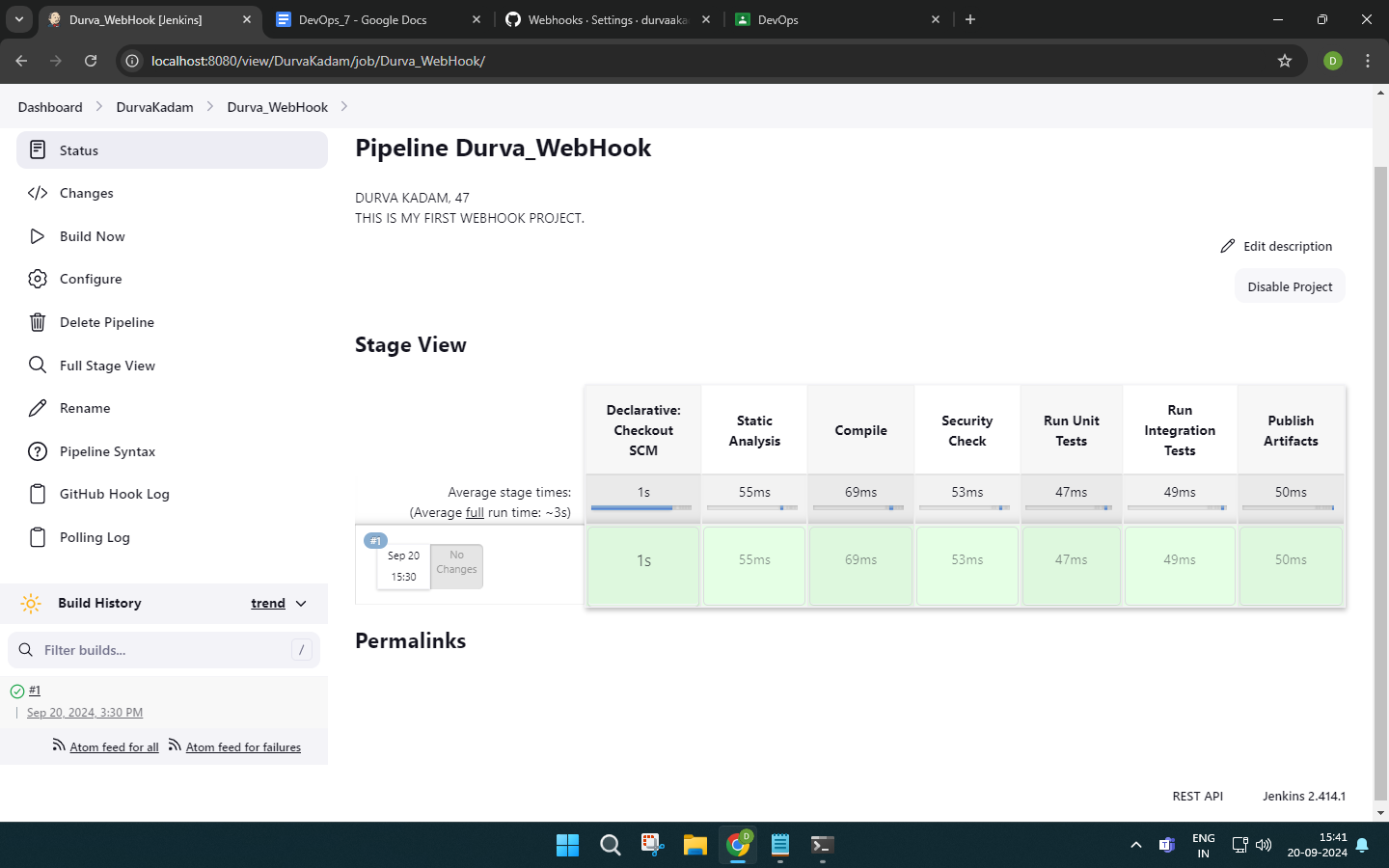
**2. Create and build pipeline project with Git**

• Fork repository on GitHub



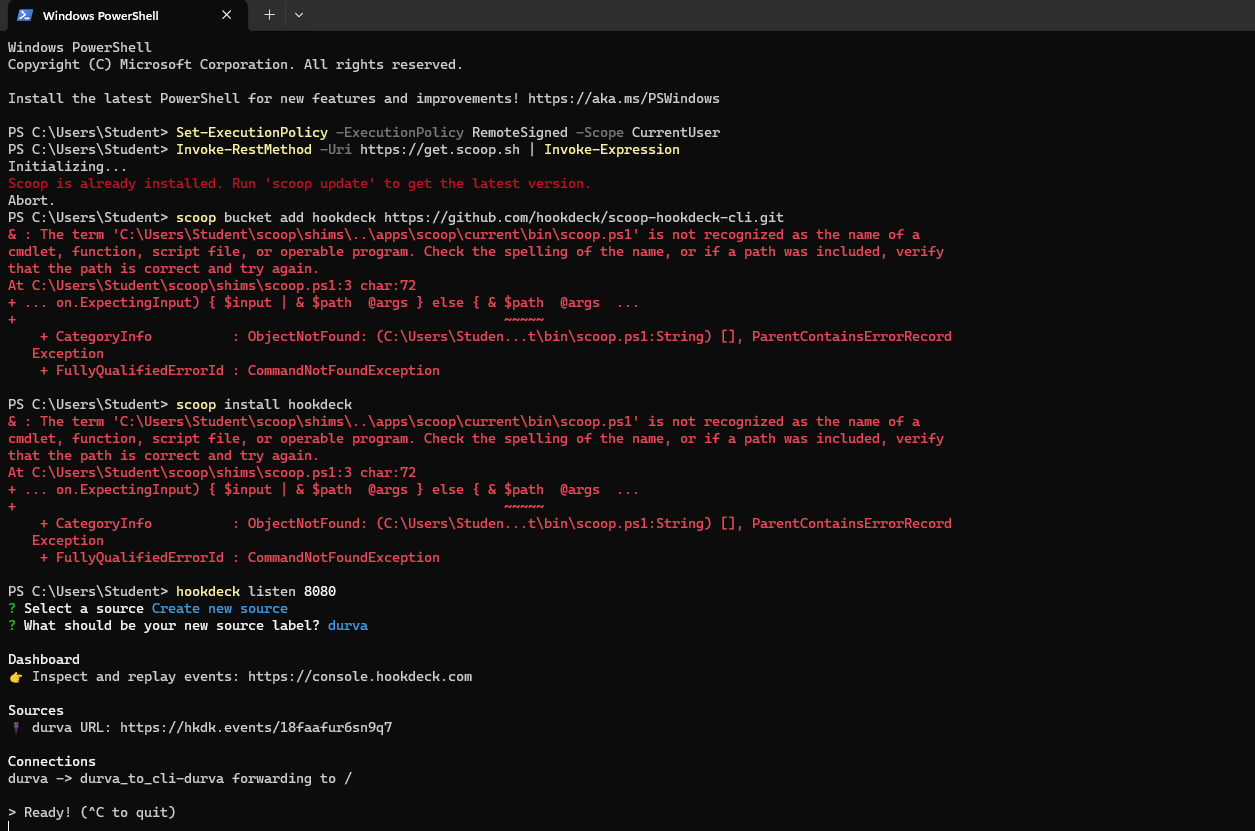
**• Create pipeline project with pipeline script from SCM**

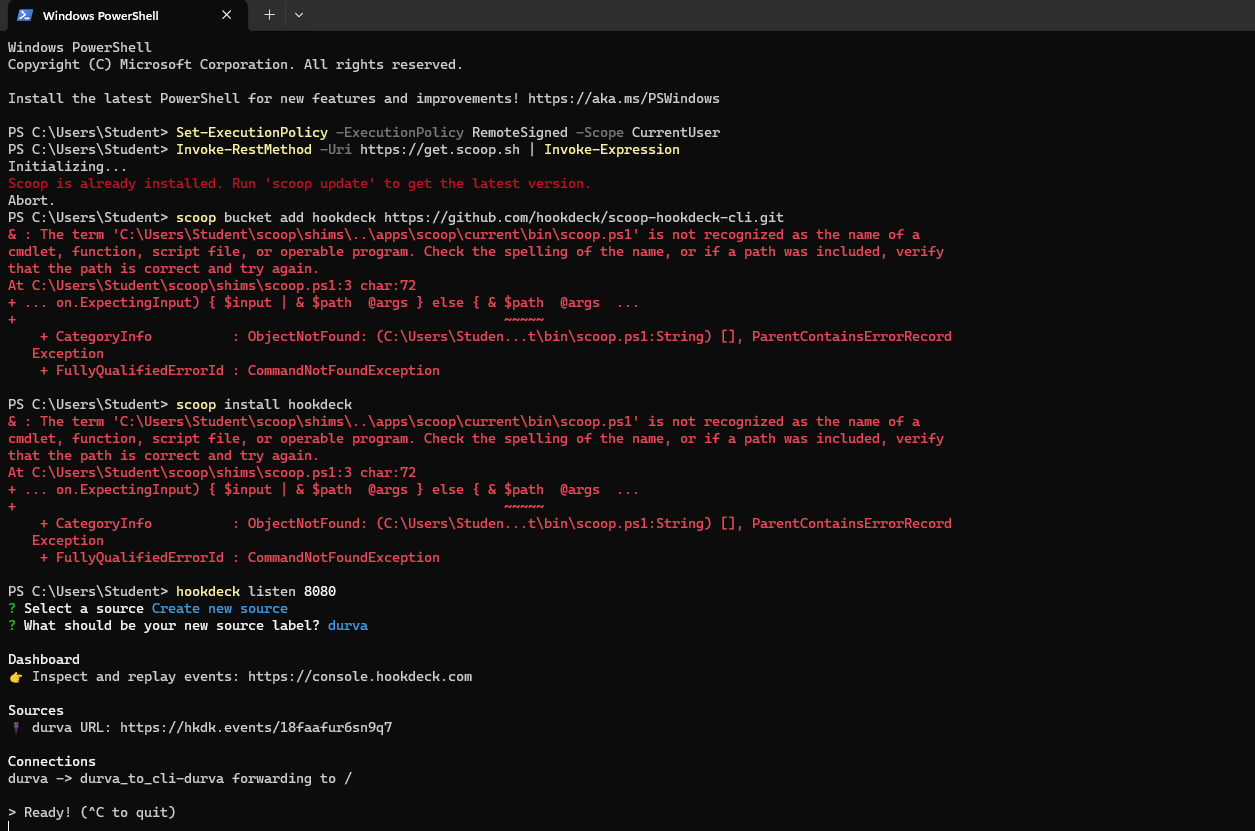


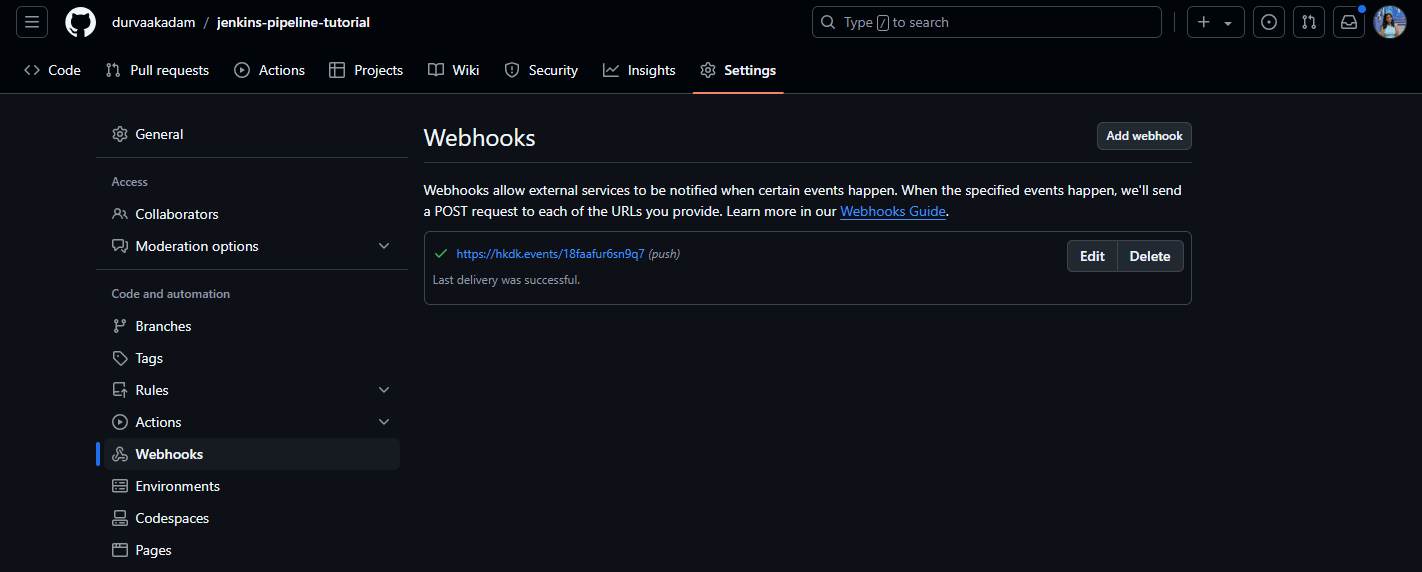
****

**• Add webhooks to the forked repository**

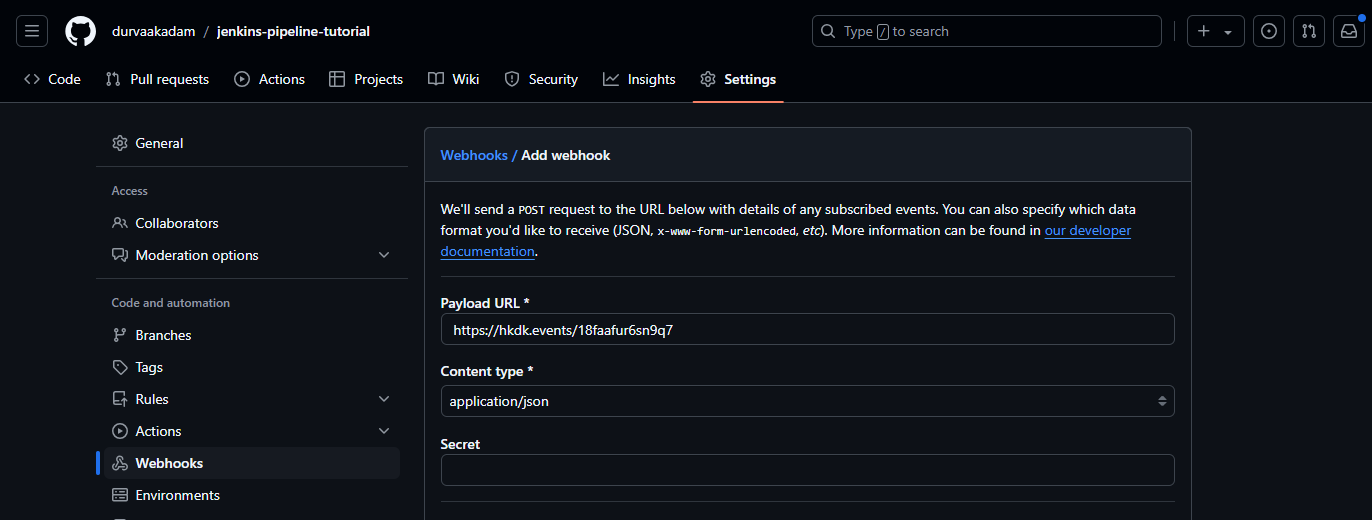
Install scoop using powershell commands.



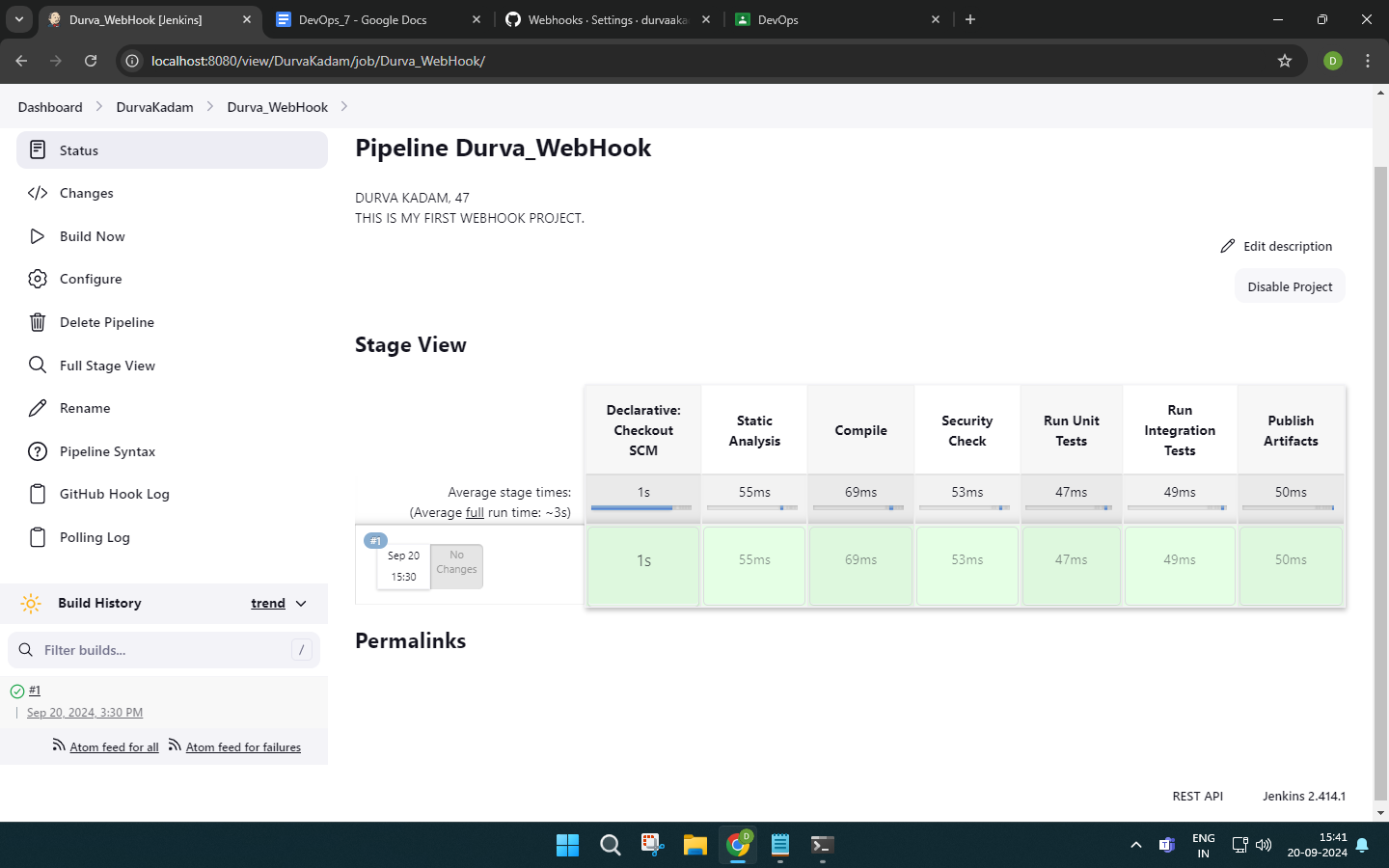


Goto Settings on GitHub, in WebHooks, add a WebHook.  


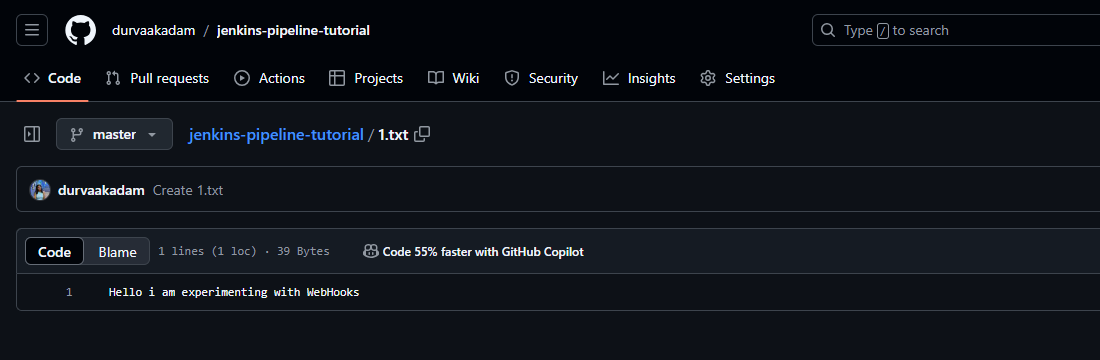
Paste the url from powershell and make the file type json and create the WebHook.

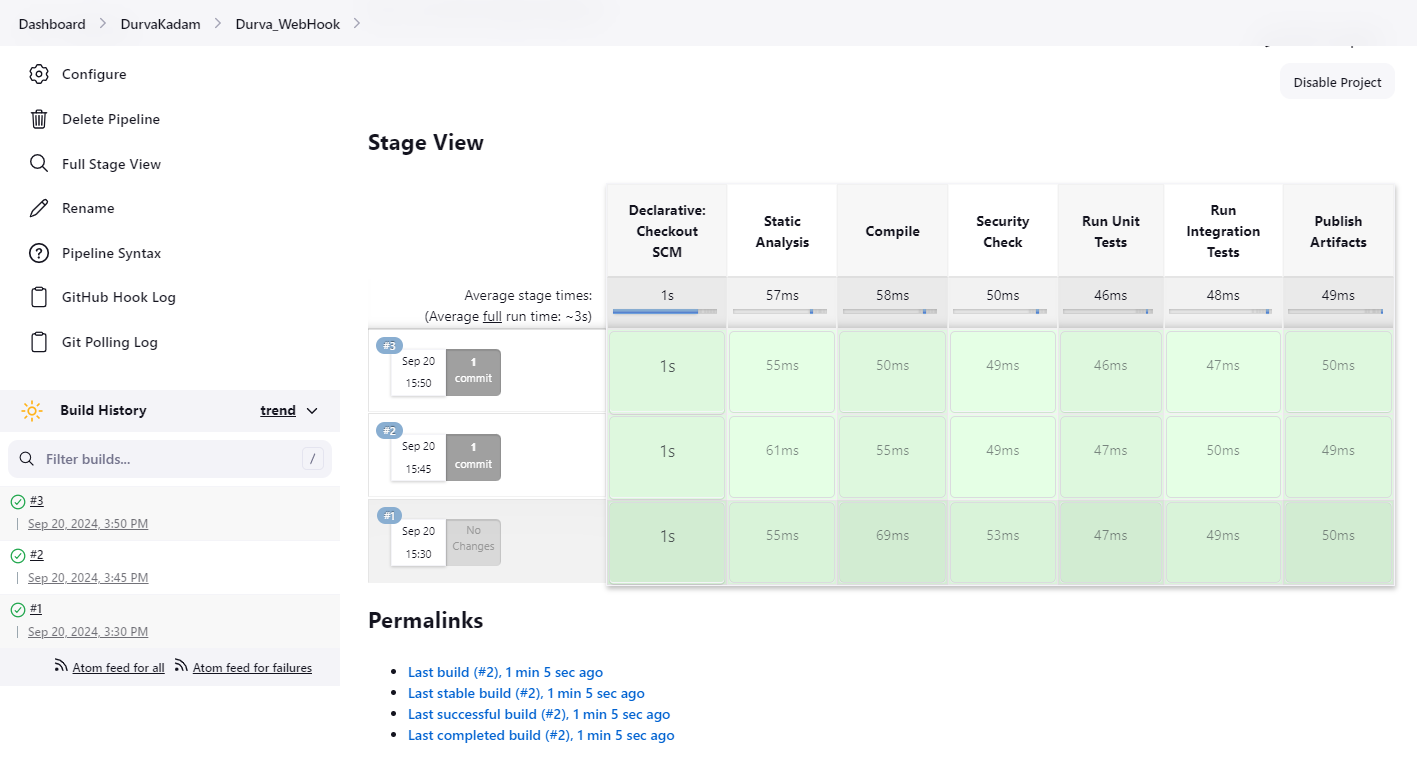


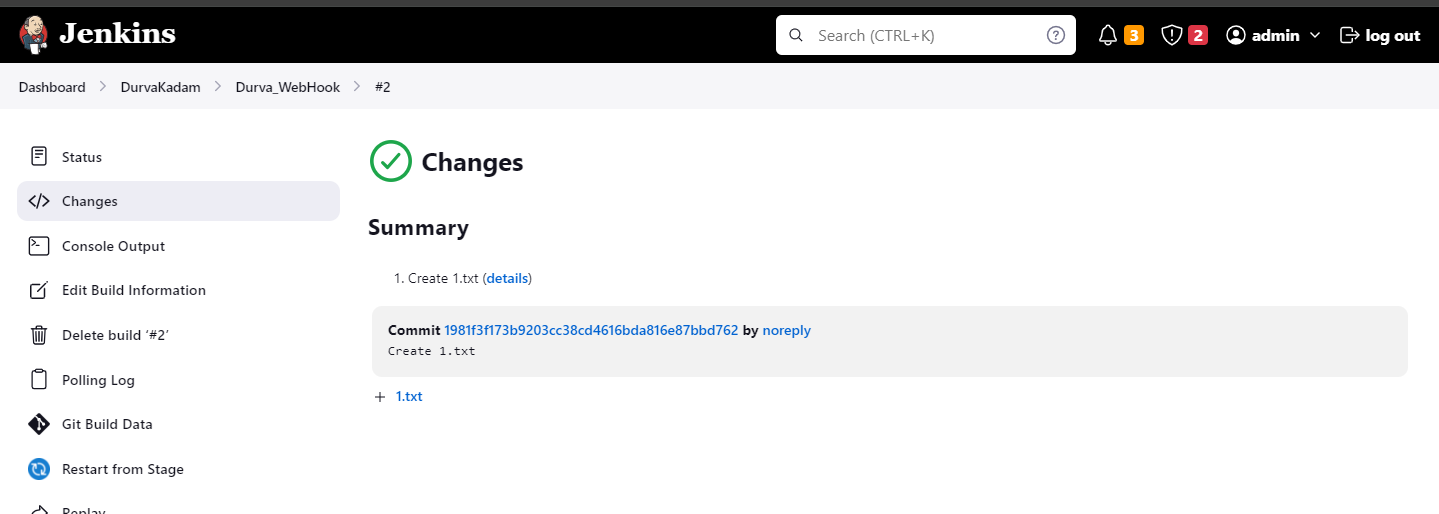
**• Build pipeline project**

****

**• Add file to forked repository and observe the automated build**







**• Make changes to Jenkins file on forked repository and observe the automated build**

Edit the existing file ‘1.txt’.

