**Experiment No. 2**

**Aim:** To perform Continuous Integration & Continuous Delivery using Jenkins

**Objective:**

1. To understand CI/CD

2. To know SDLC

**Theory:**

Performing Continuous Integration and Continuous Delivery (CI/CD) using Jenkins is a common use case for DevOps teams. Jenkins is an open-source automation server that can be used to automate various stages of the software development lifecycle, including building, testing, and deploying software.

Here are the general steps for setting up a CI/CD pipeline using Jenkins:

Install Jenkins: First, you need to install Jenkins on your machine or server. Jenkins is available for Windows, Linux, and macOS operating systems. You can download the installation package from the official Jenkins website and follow the installation instructions.

Configure Jenkins: Once Jenkins is installed, you need to configure it to work with your environment and tools. This includes setting up plugins, configuring security settings, and defining build agents.

Create a Jenkins job: Jenkins jobs are the building blocks of your CI/CD pipeline. You can create a new Jenkins job for each stage of your pipeline, such as building, testing, and deploying. To create a job, you need to define the build steps, such as executing shell scripts, running tests, and deploying the software.

Integrate with version control: To trigger builds automatically when code changes are pushed to the repository, you need to integrate Jenkins with your version control system (VCS), such as Git. Jenkins has plugins that allow it to work with different VCS providers.

Add testing and deployment stages: After the build stage, you can add testing and deployment stages to your pipeline. For example, you can add unit tests, integration tests, and acceptance tests. You can also add deployment steps, such as deploying to a staging environment and then promoting to production.

Monitor and optimize: Once your CI/CD pipeline is up and running, you can monitor its performance and optimize it for efficiency and reliability. Jenkins provides various metrics and reports to help you identify areas for improvement.

CI/CD stands for Continuous Integration and Continuous Delivery (or Deployment). CI/CD is a DevOps practice that involves automating the software build, test, and deployment process to improve the speed, quality, and reliability of software releases. Continuous Integration refers to the practice of automatically building and testing code changes whenever new code is pushed to a shared repository. Continuous Delivery (or Deployment) refers to the practice of automatically deploying code changes to a production environment once they have been successfully built and tested.

Jenkins is an open-source automation server that can be used to implement CI/CD pipelines. Jenkins allows you to define workflows that automate the build, test, and deployment process. Jenkins is highly extensible, with a large number of plugins available to integrate with various tools and technologies.

The benefits of CI/CD include:

Improved software quality: By automating the build and testing process, CI/CD helps ensure that code changes are thoroughly tested before they are deployed to production, reducing the likelihood of bugs and errors.

Faster time-to-market: By automating the build, test, and deployment process, CI/CD helps reduce the time it takes to release software updates, allowing organizations to respond to customer needs and market changes more quickly.

Greater collaboration: By providing a shared pipeline for developers, testers, and operations personnel, CI/CD encourages greater collaboration and communication across teams.

Increased efficiency: By automating manual tasks and reducing the need for manual intervention, CI/CD helps increase efficiency and reduce costs.

Jenkins Workflow is a plugin that allows you to define complex build and deployment workflows using a domain-specific language (DSL). Jenkins Workflow allows you to define pipelines that can include multiple stages and steps, allowing you to automate complex build, test, and deployment processes.

A Pipeline in Jenkins is a way to define a build and deployment workflow as code. There are two types of pipelines in Jenkins: Scripted Pipeline and Declarative Pipeline. Scripted Pipeline uses a Groovy-based DSL to define the build and deployment workflow, while Declarative Pipeline uses a YAML-based syntax. Declarative Pipeline is more structured and easier to read and maintain, while Scripted Pipeline is more flexible and allows for more complex logic.

By default, your Jenkins runs at https://localhost:8080/. This can be changed by editing jenkins.xml , which is located in your installation directory. This file is also the place to change other boot configuration parameters, such as JVM options, HTTPS setup, etc.

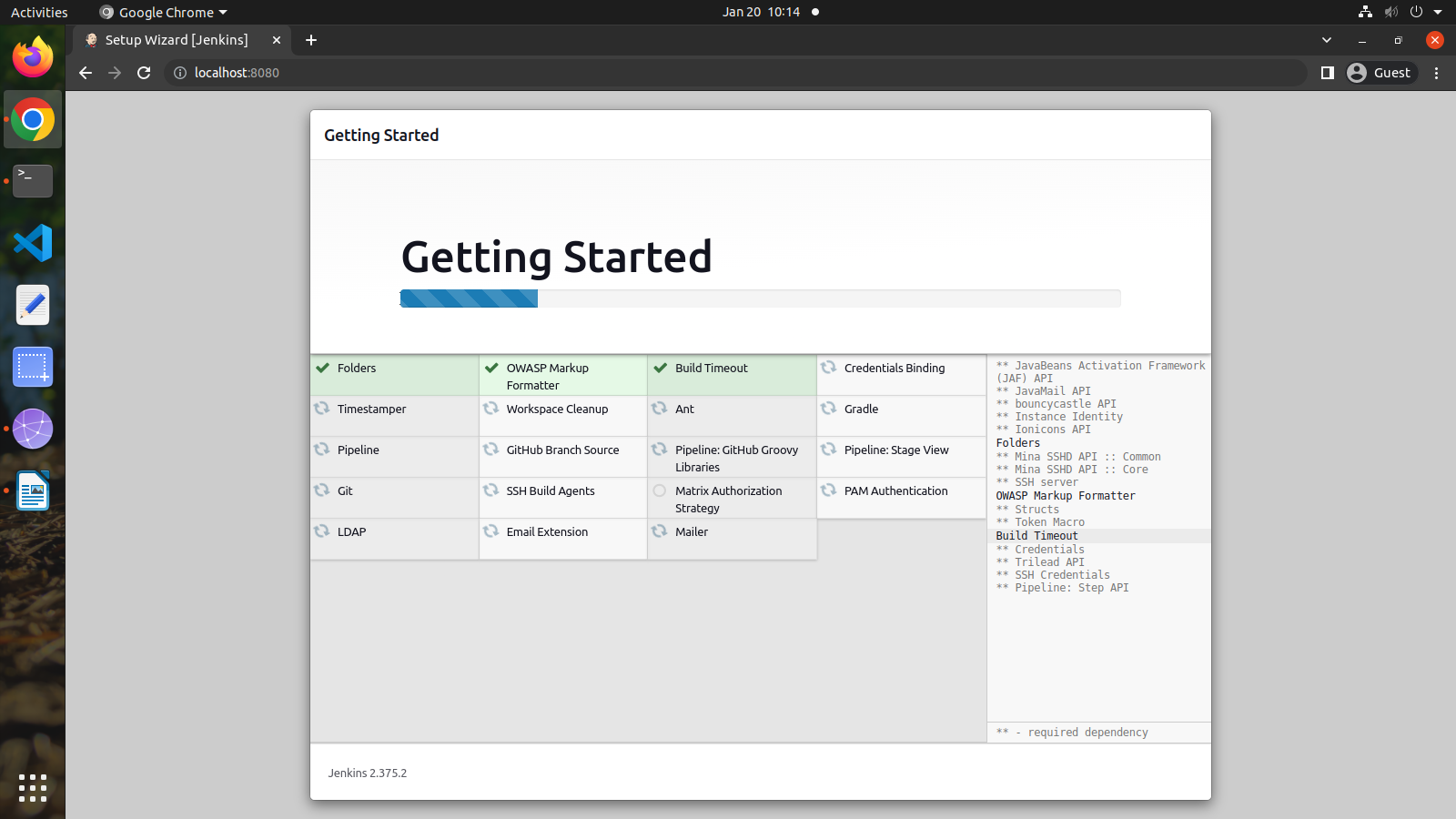
**Conclusion:**

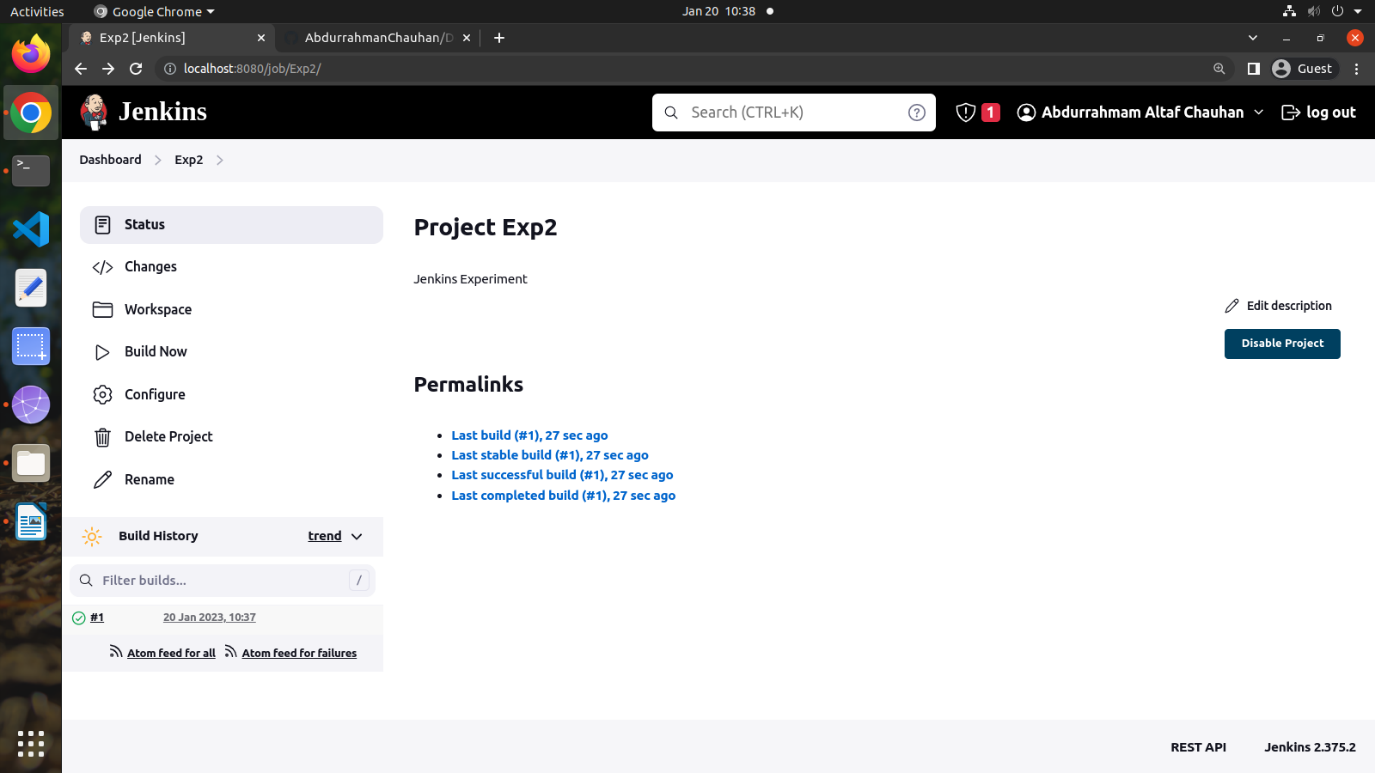
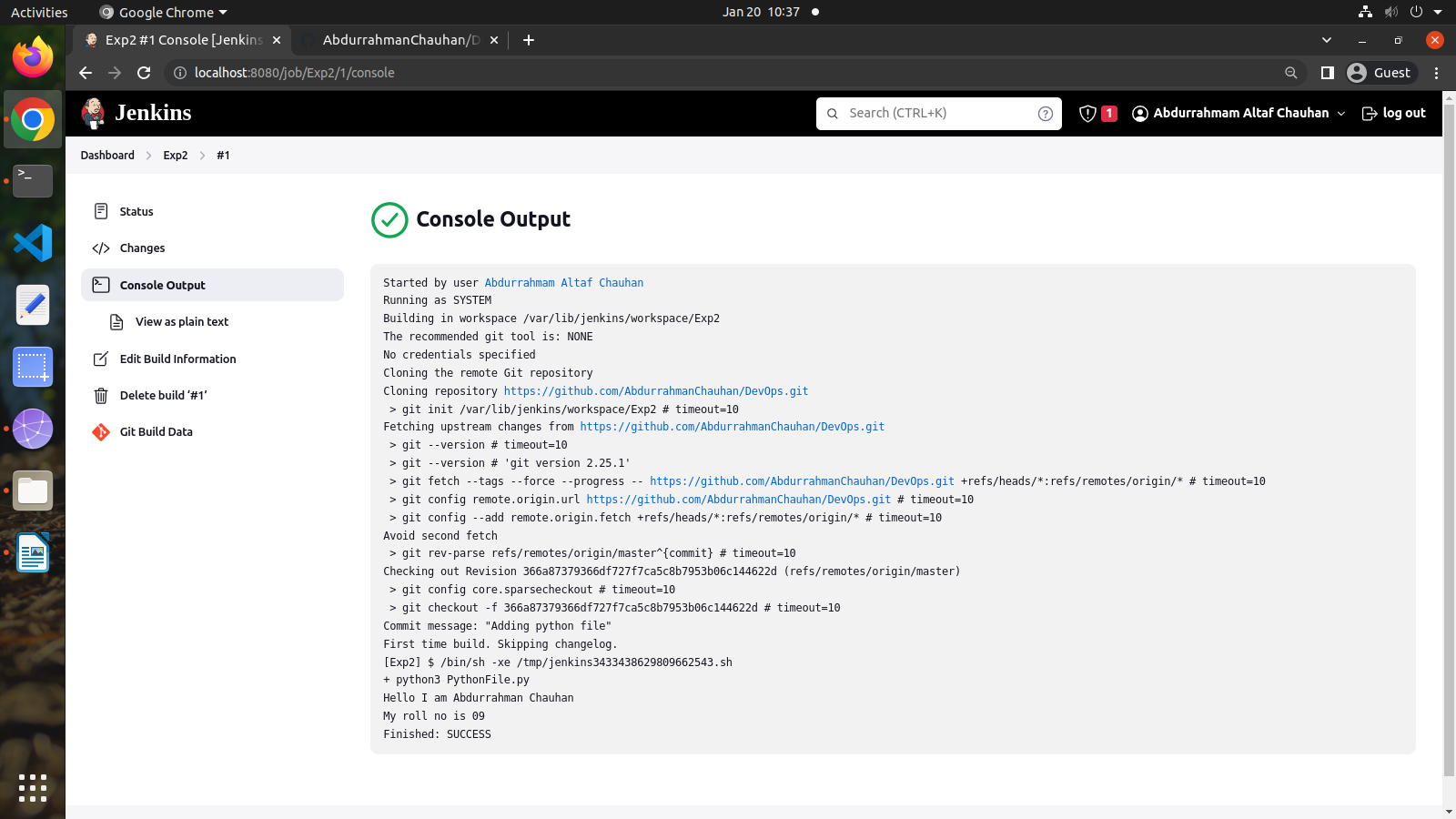
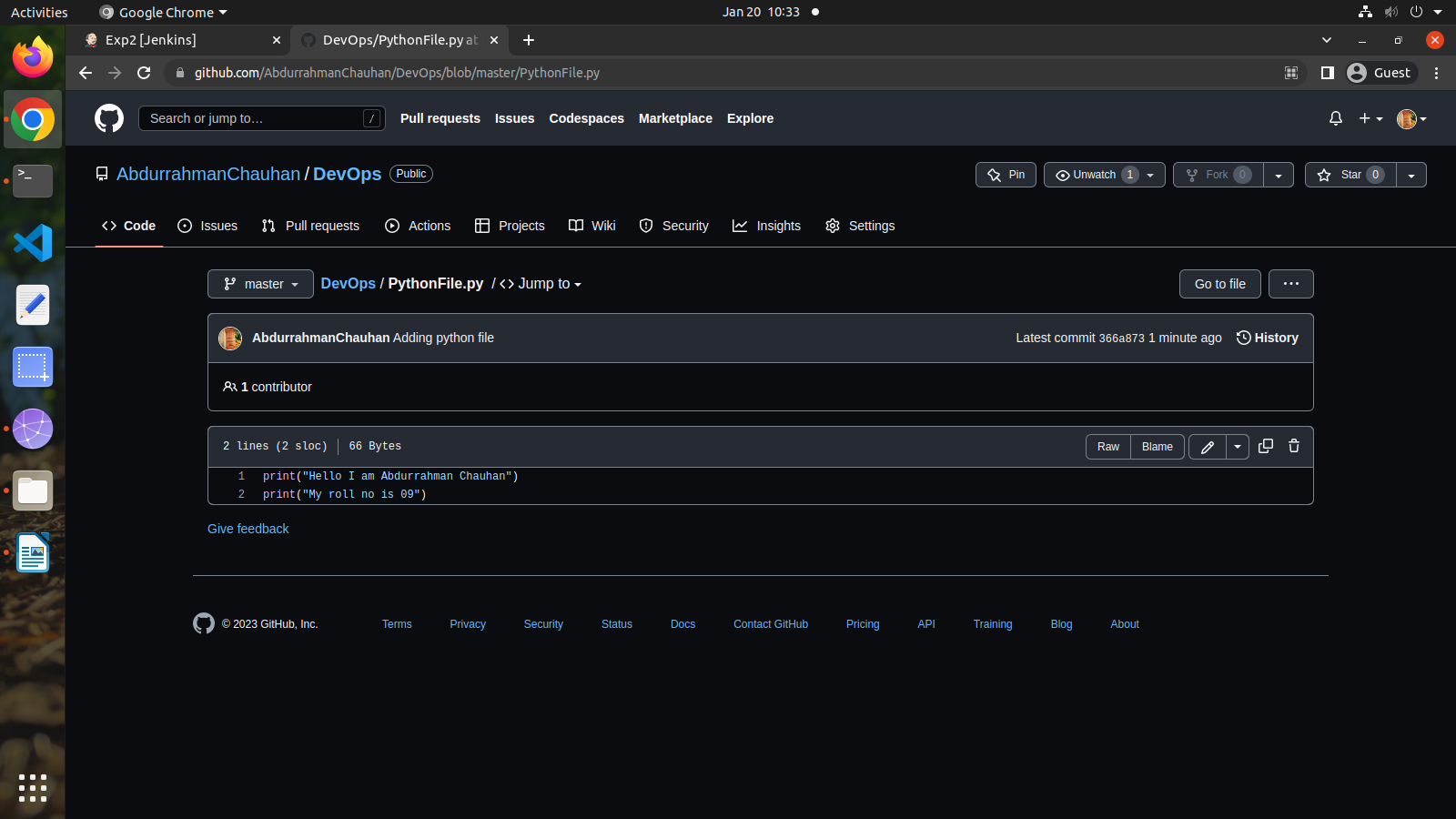
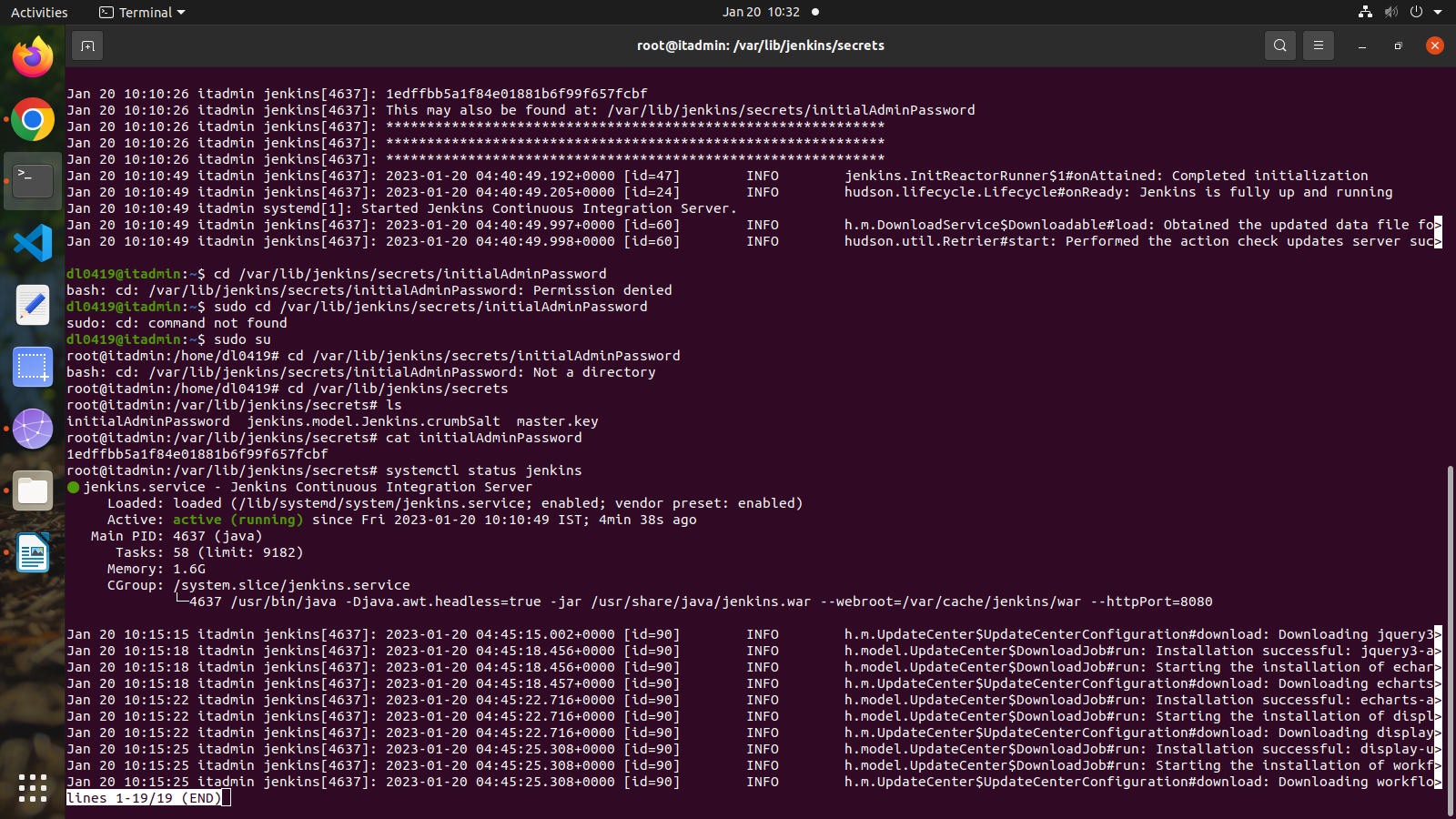
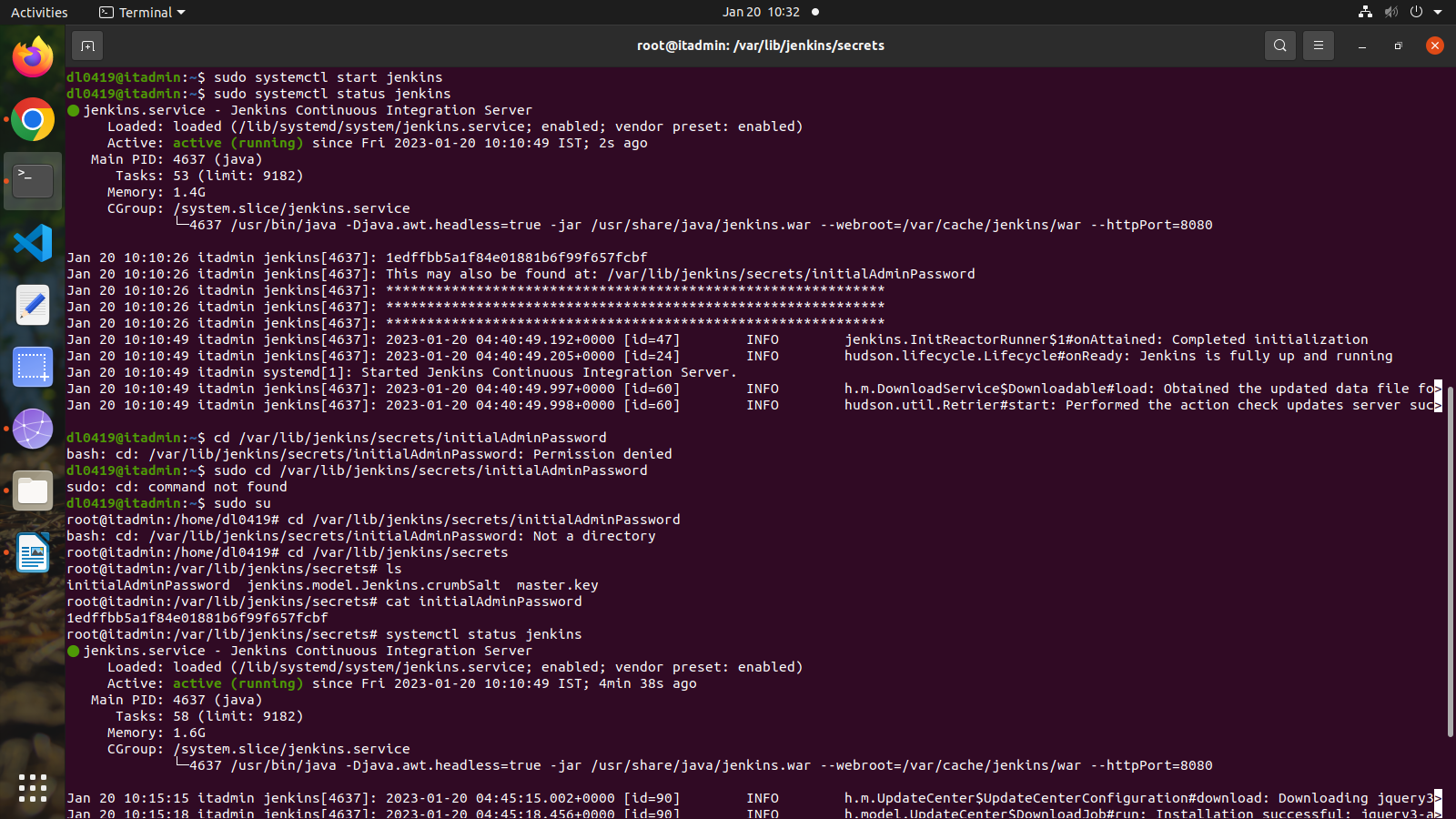
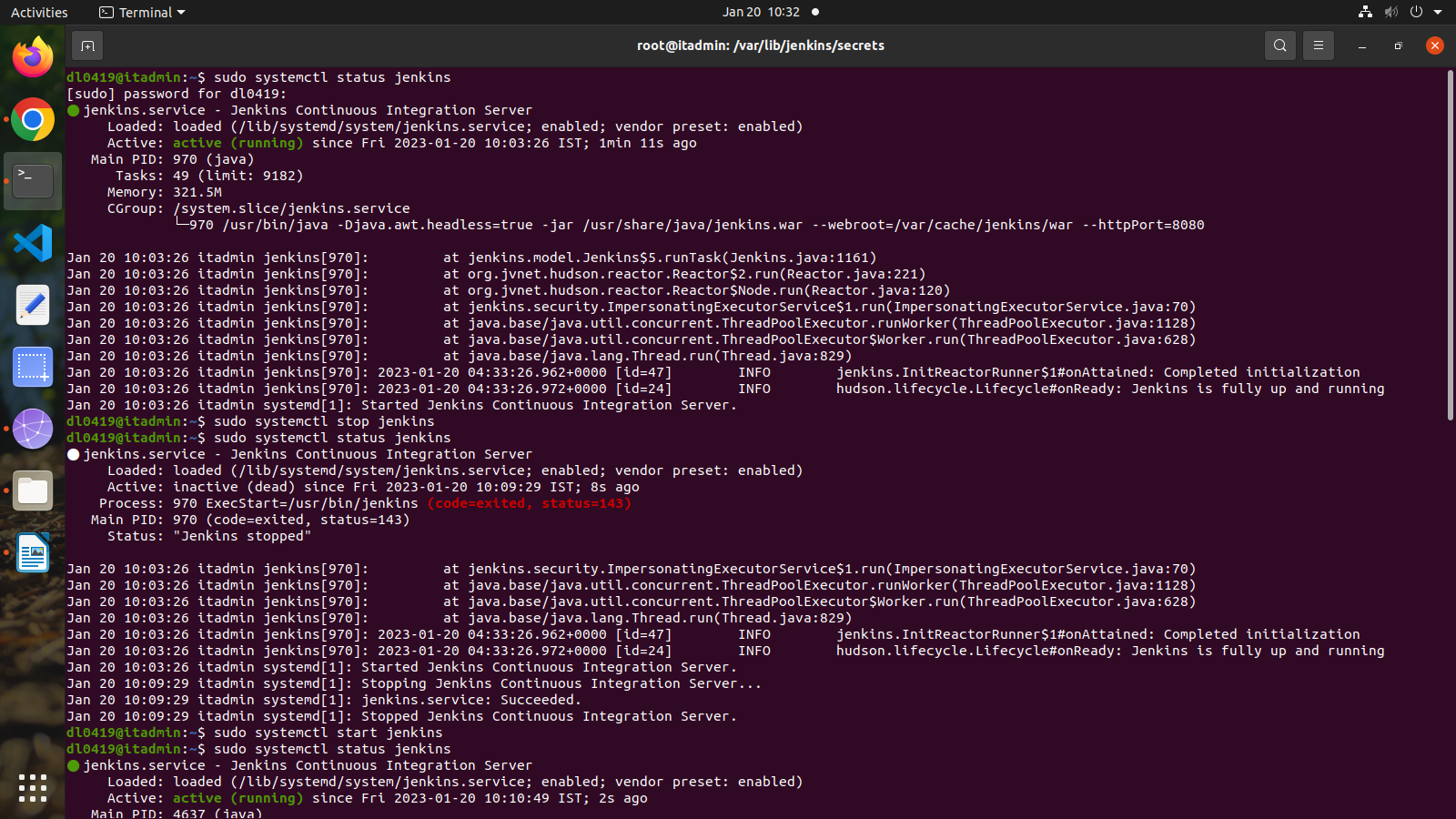
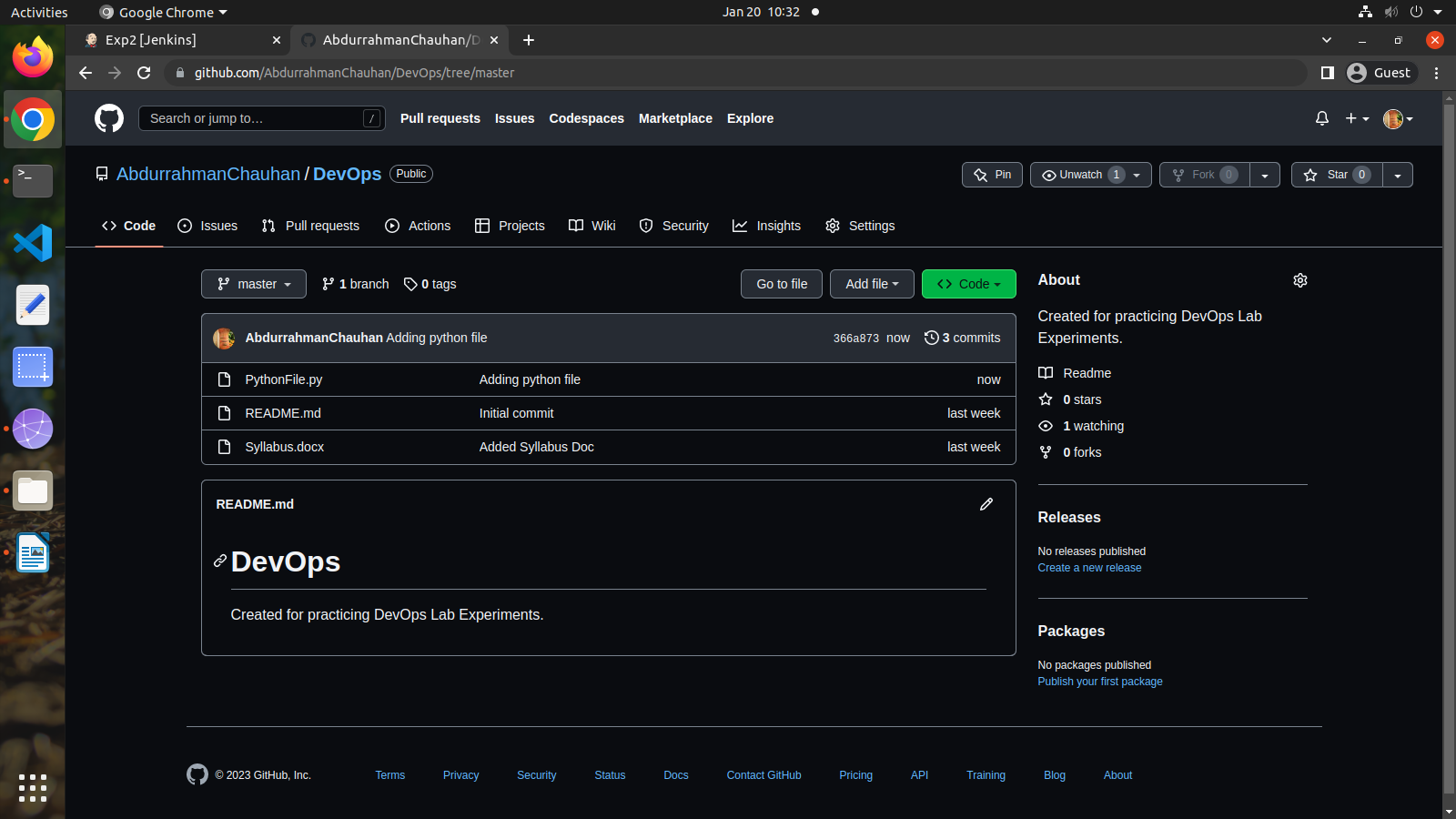
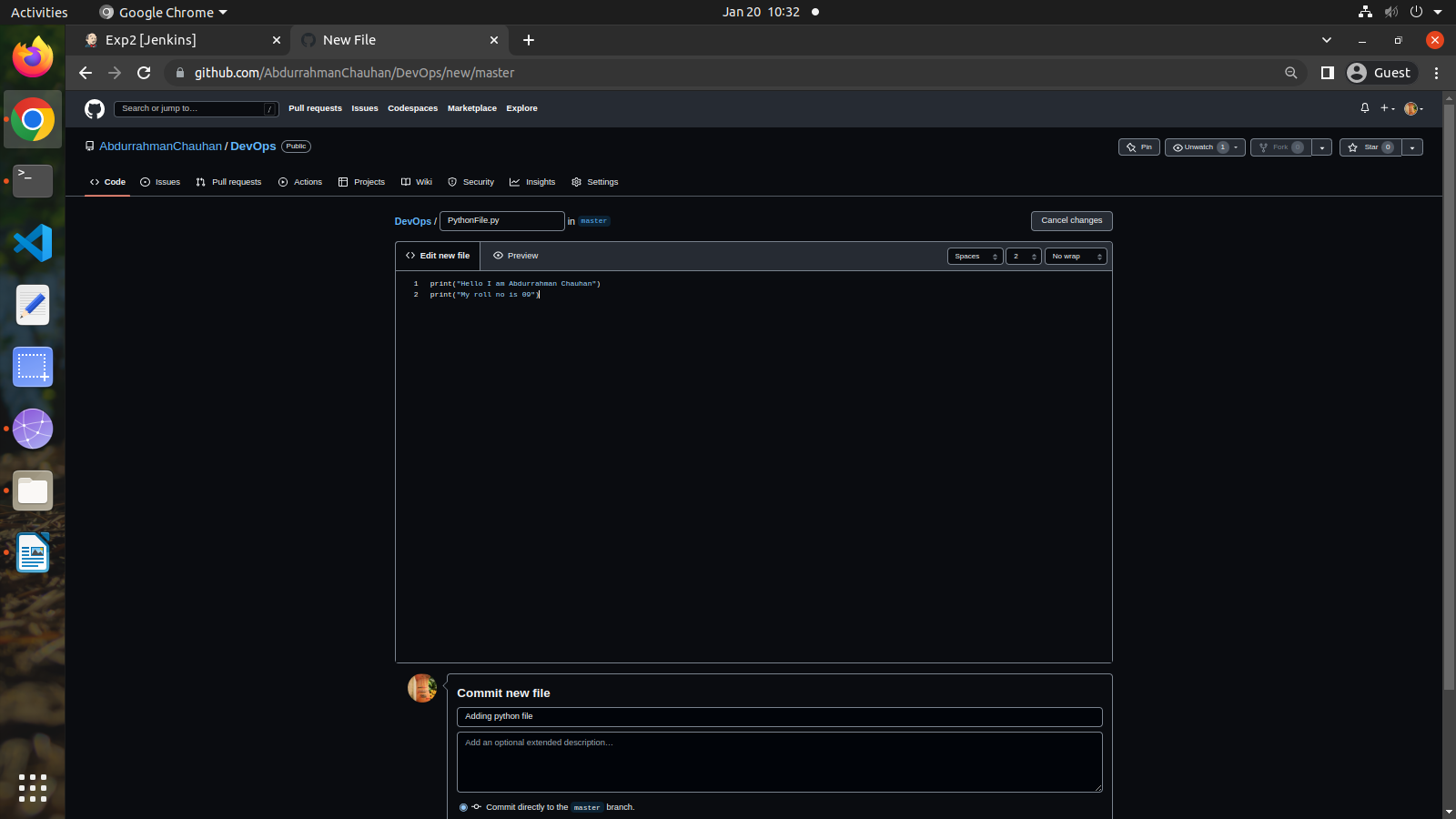
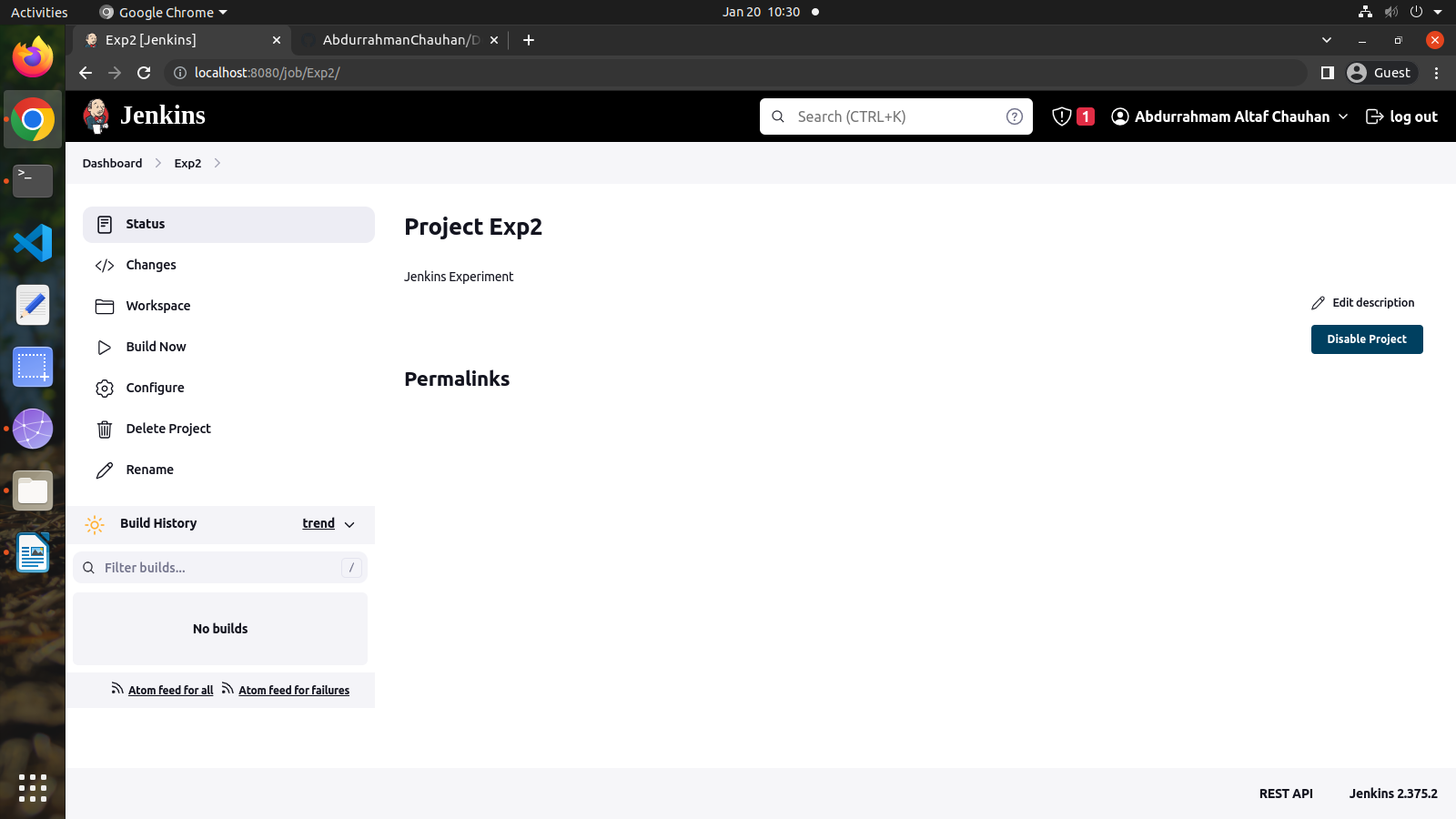
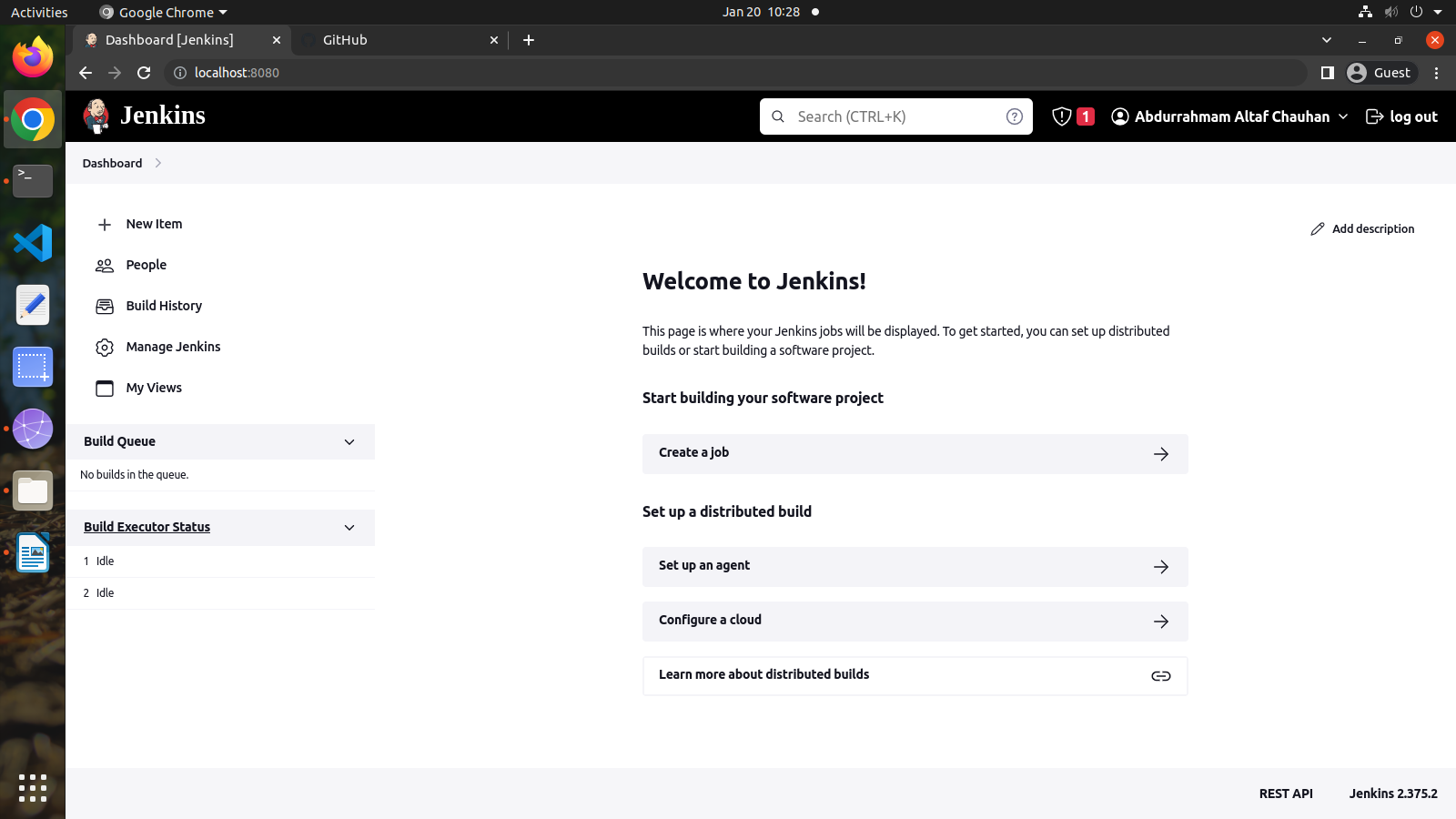
By setting up a CI/CD pipeline using Jenkins, you can automate many of the manual tasks involved in software development and deployment, which can help improve the speed and quality of your software releases.

**Outcome:**

1. Installed Jenkins

2. Used Jenkins Server for Continuous Integration.

**Output: **

****