**Description**

**1. Jenkins CI CD pipeline for flask application**

**Objective:**

Set up a Jenkins pipeline that automates the testing and deployment of a simple Python web application.

**Requirements:**

1. Setup:

   - Install Jenkins on a virtual machine or use a cloud-based Jenkins service.

   - Configure Jenkins with Python and any necessary libraries.

2. Source Code:

  - Fork the provided Python web application repository on GitHub (provide a link to a sample Python web application repository).

  - Clone the forked repository into your Jenkins server.

3. Jenkins Pipeline:

   - Create a Jenkinsfile in the root of your Python application repository.

   - Define a pipeline with the following stages:

    - Build: Install dependencies using pip.

    - Test: Run unit tests using a testing framework like pytest.

    - Deploy: If tests pass, deploy the application to a staging environment.

4. Triggers:

   - Configure the pipeline to trigger a new build whenever changes are pushed to the main branch of the repository.

5. Notifications:

   - Set up a notification system to alert via email when the build process fails or succeeds.

6. Documentation:

   - Document the pipeline process and any prerequisites needed for the setup in a README.md file in the repository.

7. Submission:

   - Provide the URL to the GitHub repository with the Jenkinsfile and updated README.md.

   - Include screenshots of the Jenkins pipeline showing the build, test, and deployment stages.

**Deliverables:**

- Forked GitHub repository with Jenkinsfile.

- Documentation in README.md.

- Screenshots of the Jenkins pipeline execution.

**2. GitHub Actions CI/CD Pipeline Flask App**

**Objective:**

Implement a CI/CD workflow using GitHub Actions for a Python application.

**Requirements:**

1. Setup:

   - Use a provided Python application repository on GitHub (provide a link to a sample Python application repository).

   - Ensure the repository has a main branch and a staging branch.

2. GitHub Actions Workflow:

   - Create a .github/workflows directory in your repository.

   - Inside the directory, create a YAML file to define the workflow.

3. Workflow Steps:

     - Define a workflow that performs the following jobs:

     - Install Dependencies: Install all necessary dependencies for the Python application using pip.

     - Run Tests: Execute the test suite using a framework like pytest.

     - Build: If tests pass, prepare the application for deployment.

     - Deploy to Staging: Deploy the application to a staging environment when changes are pushed to the staging branch.

     - Deploy to Production: Deploy the application to production when a release is tagged.

4. Environment Secrets:

   - Use GitHub Secrets to store sensitive information required for deployments (e.g., deployment keys, API tokens).

5. Documentation:

   - Update the README.md file with instructions on how the GitHub Actions workflow works and how to configure the necessary secrets.

6. Submission:

   - Provide the URL to the GitHub repository with the workflow file and updated README.md.

   - Include screenshots of the GitHub Actions workflow runs showing successful execution of all steps.

**Deliverables:**

- GitHub repository with the workflow file.

- Documentation in README.md.

- Screenshots of the GitHub Actions workflow runs.

**Submission Instructions**

To submit your assignment, please follow these guidelines:

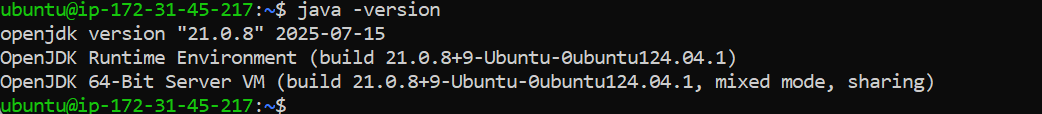
- Ensure that your assignment is fully completed.

- Push your code to a GitHub repository.

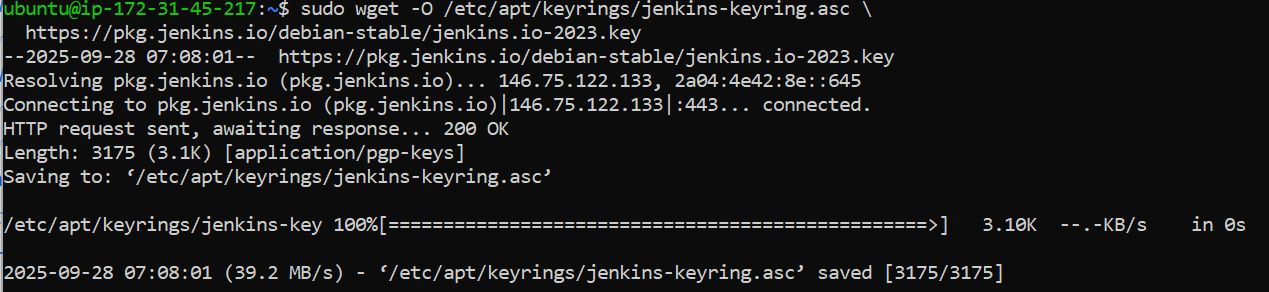
- Share the repository link by including it in a text, Word, or PDF file format.

Submit the file/text containing the repository link via Vlearn.

1. Create an EC2
   1. Use “instance type” as “t2.micro”.
   2. Enable inbound rule for port 8080.
   3. Allocate Elastic IP.
2. Connect to EC2 via SSH.
   1. Use Command “ssh -I ‘Path-to-keyFile.pem’ ubuntu@publicIP”
3. Run below command to update and upgrade EC2.
   1. sudo apt update
   2. sudo apt upgrade
4. Install “Open jdk Java Jre”
   1. use command “sudo apt install fontconfig openjdk-21-jre”
   2. Run “java -version” to verify if its installed



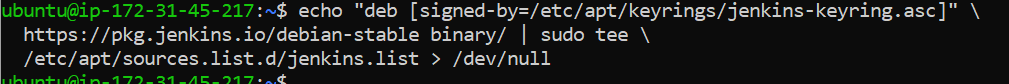
1. Follow below steps to install Jenkins.
   1. Run “sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \ <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key>”



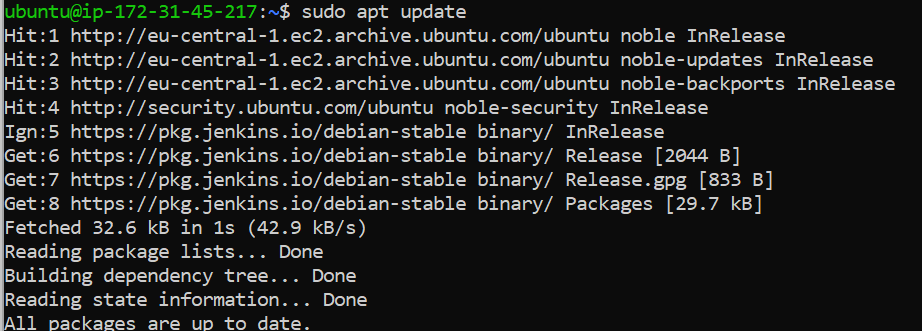
* 1. Run “echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

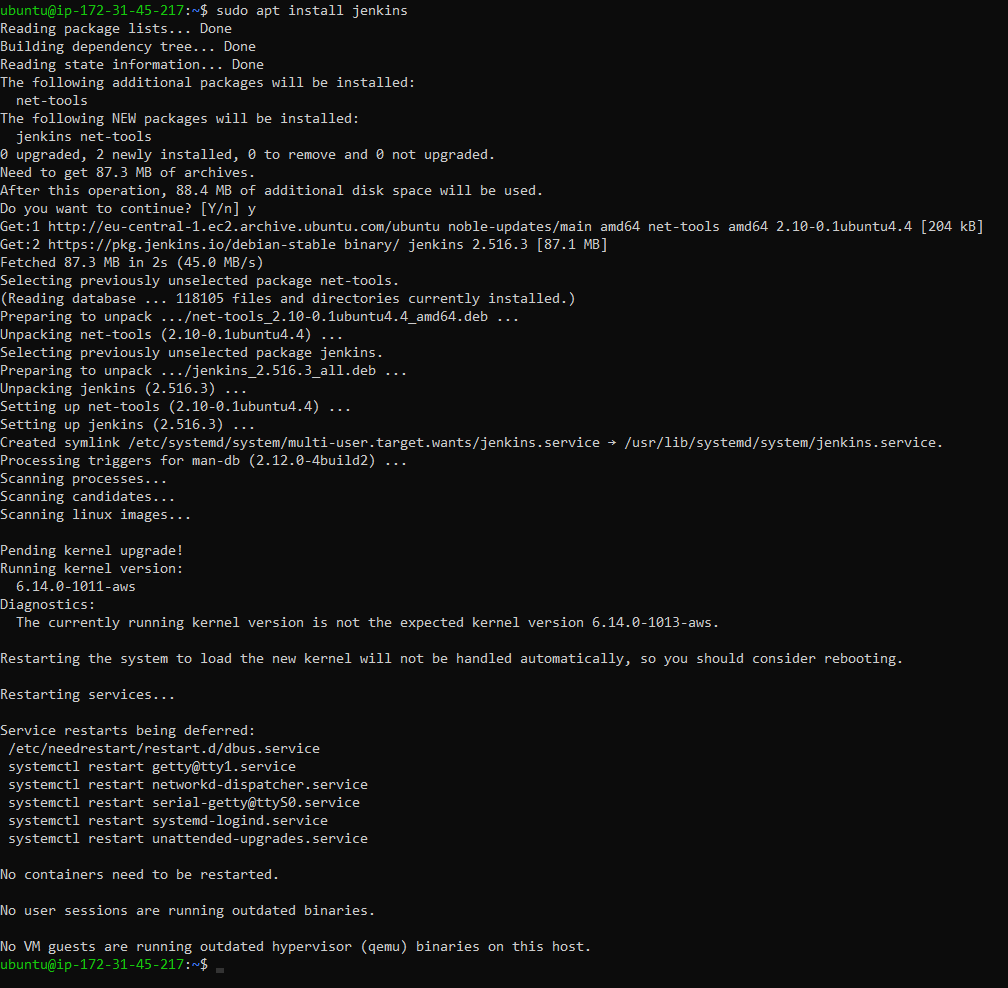
/etc/apt/sources.list.d/jenkins.list **>** /dev/null”



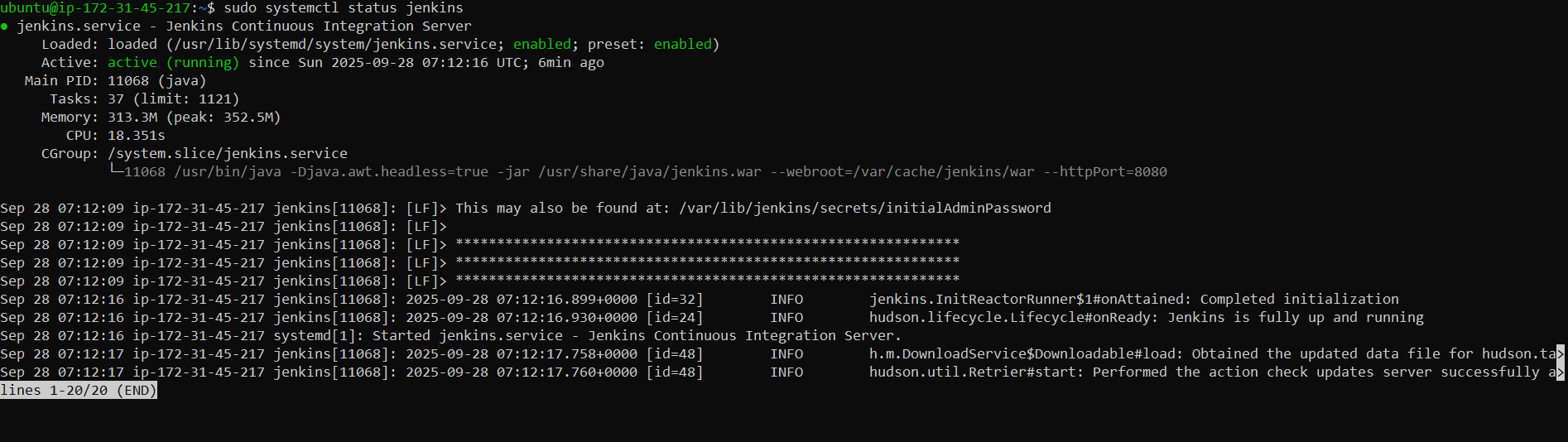
* 1. Run “sudo apt update”



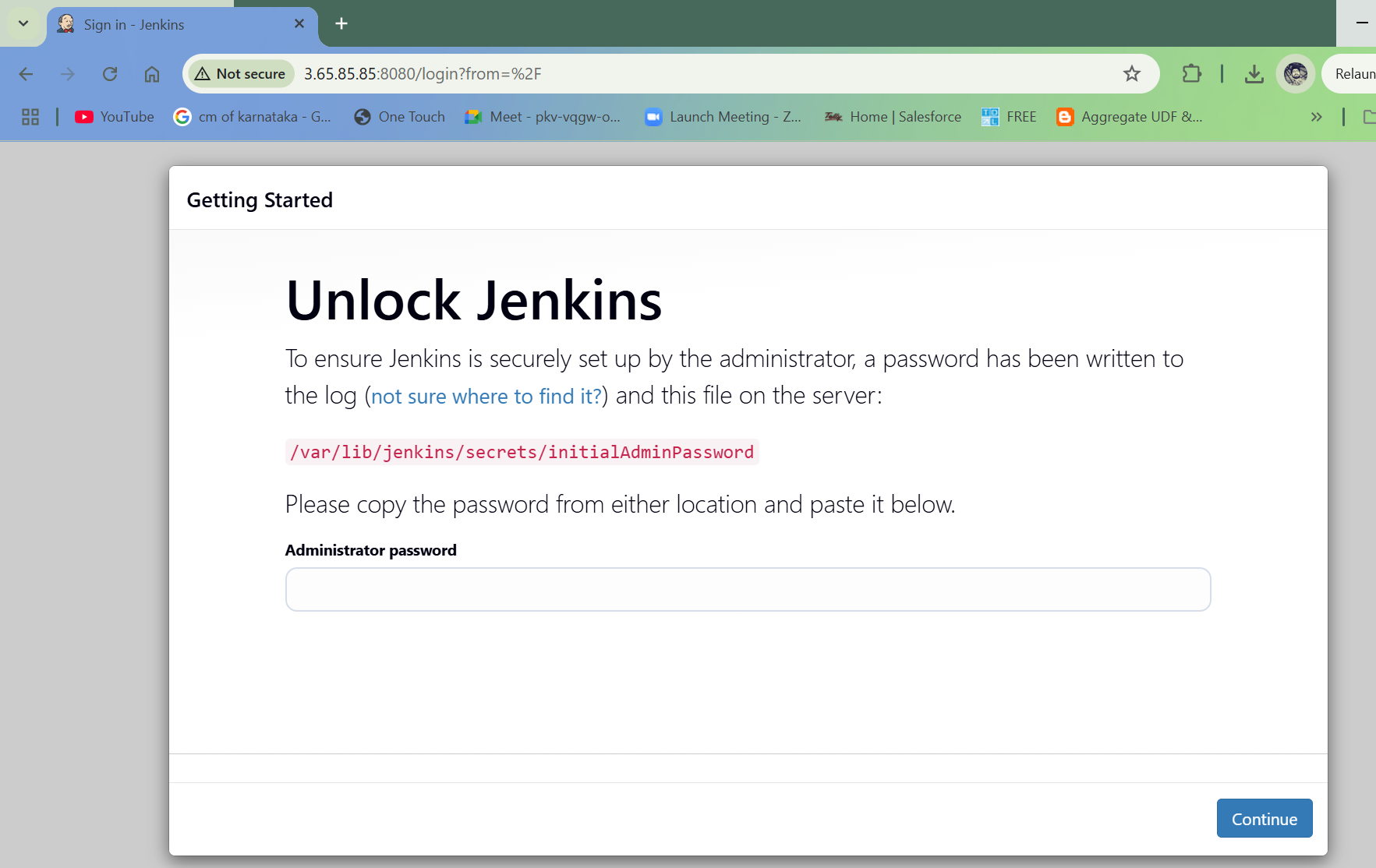
* 1. Run “sudo apt install jenkins”



1. Check status of “Jenkins”
   1. Run “sudo systemctl status jenkins”



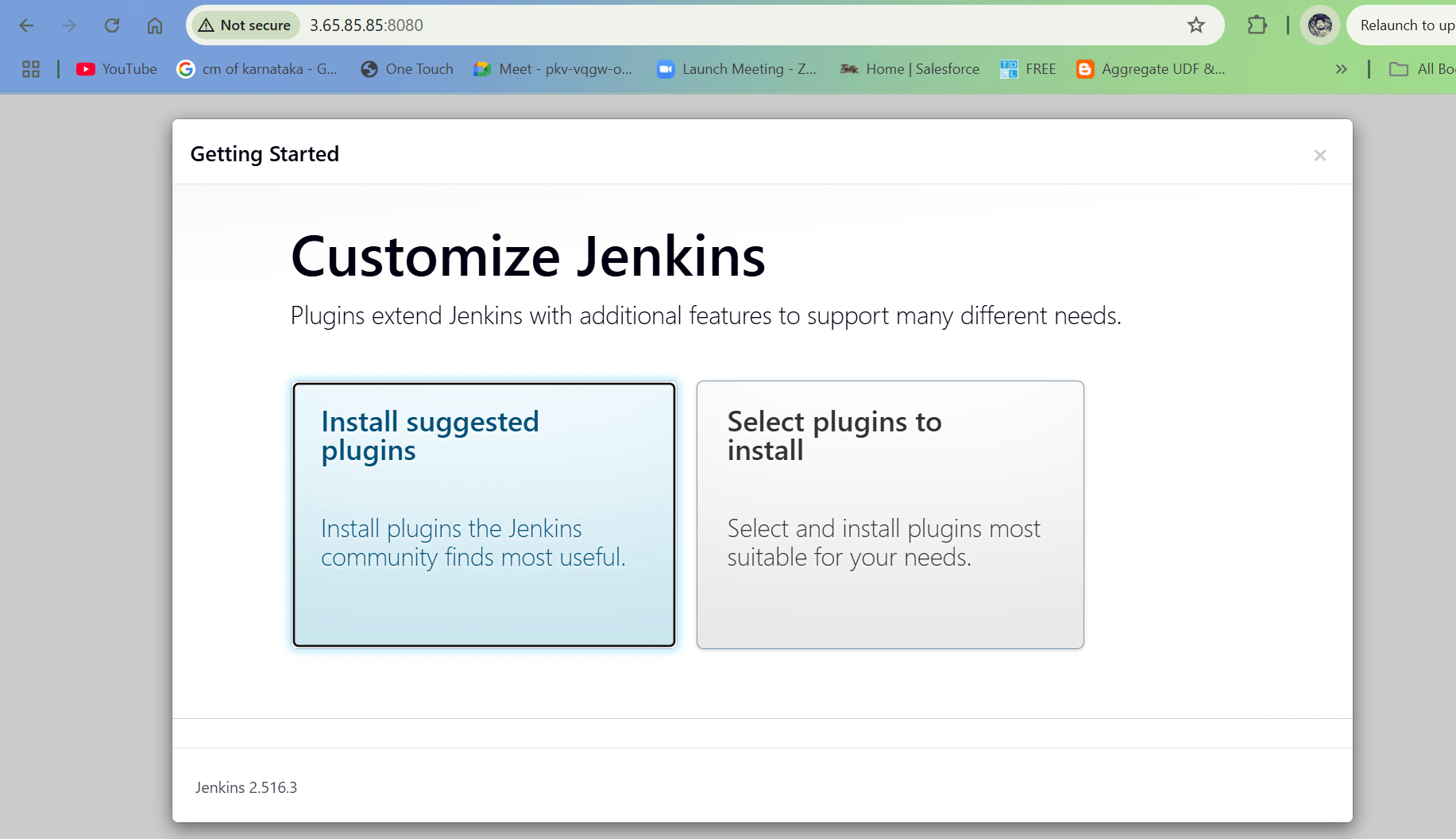
Since it is already running, we don’t have to start Jenkins, if it was not running, we need to start “jenkins” by running “sudo systemctl start jenkins”

1. To setup and configure Jenkins, follow below steps.
   1. Navigate to “PublicIPAddress:8080”

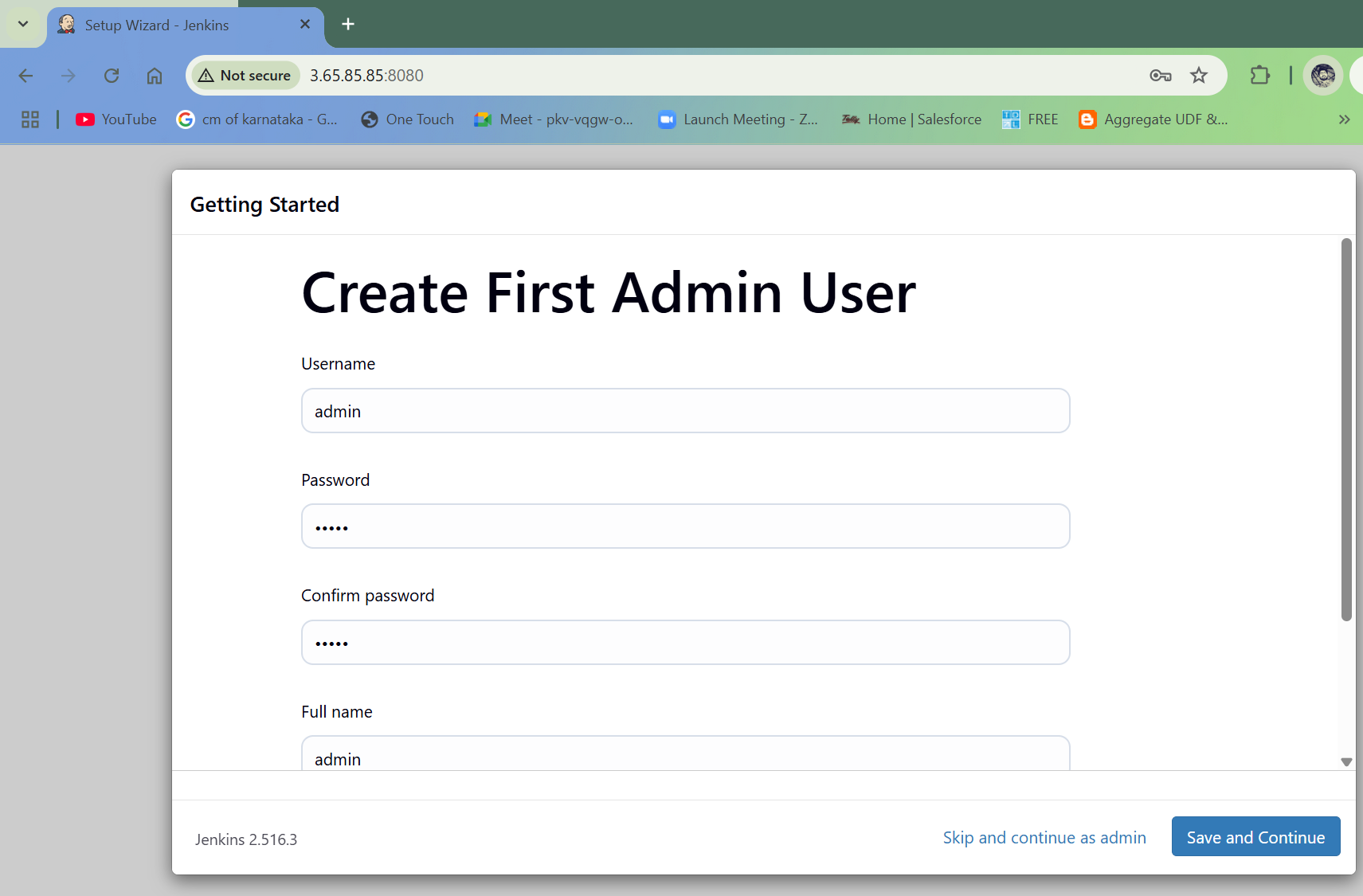
Enter “administrator Password” by reading the file located at “/var/lib/Jenkins/secrets/initialAdminPassword”.

Run “sudo cat /var/lib/Jenkins/secrets/initialAdminPassword” to read the file

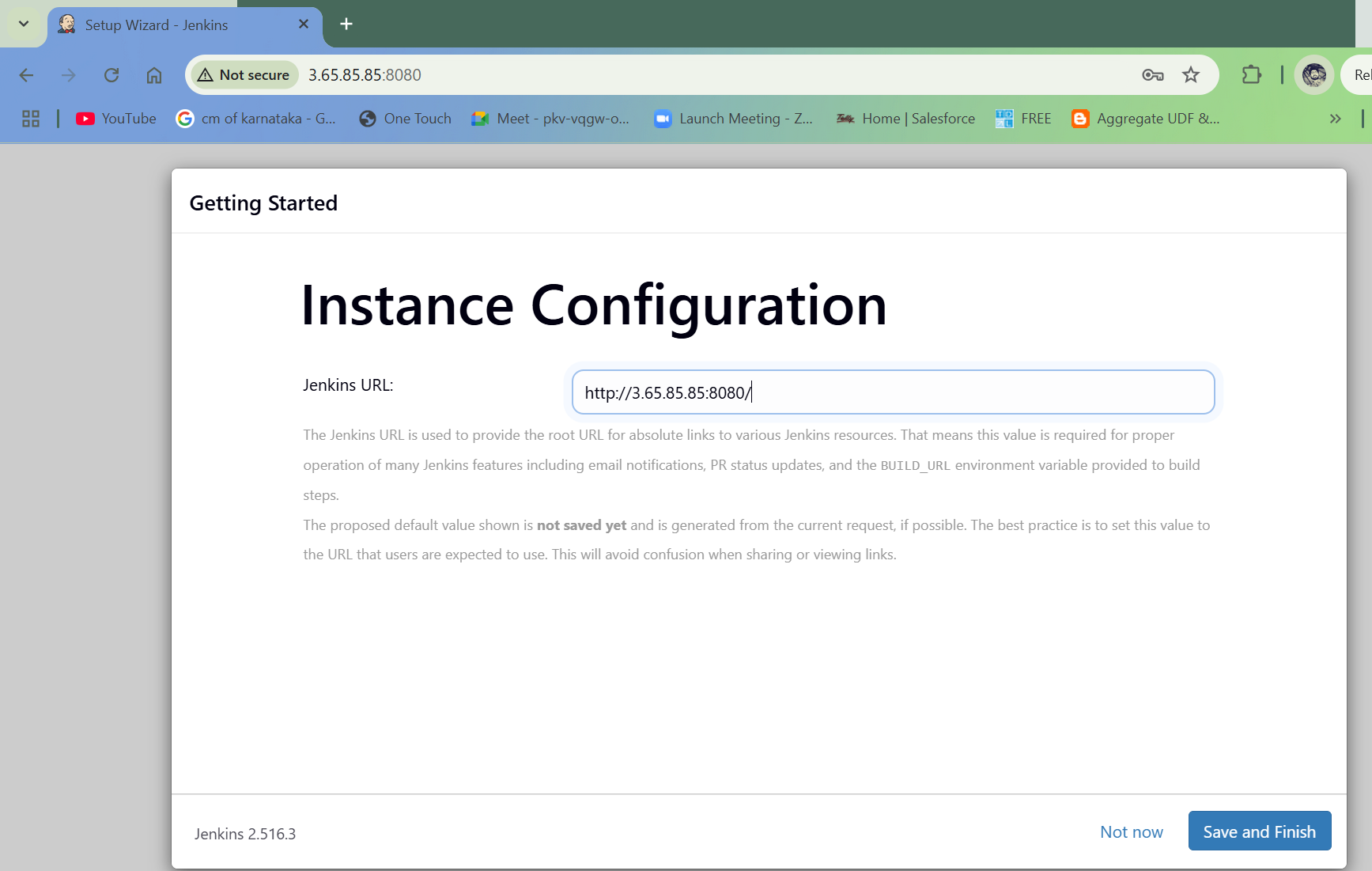
* 1. Click on “Install Suggested Plugins”



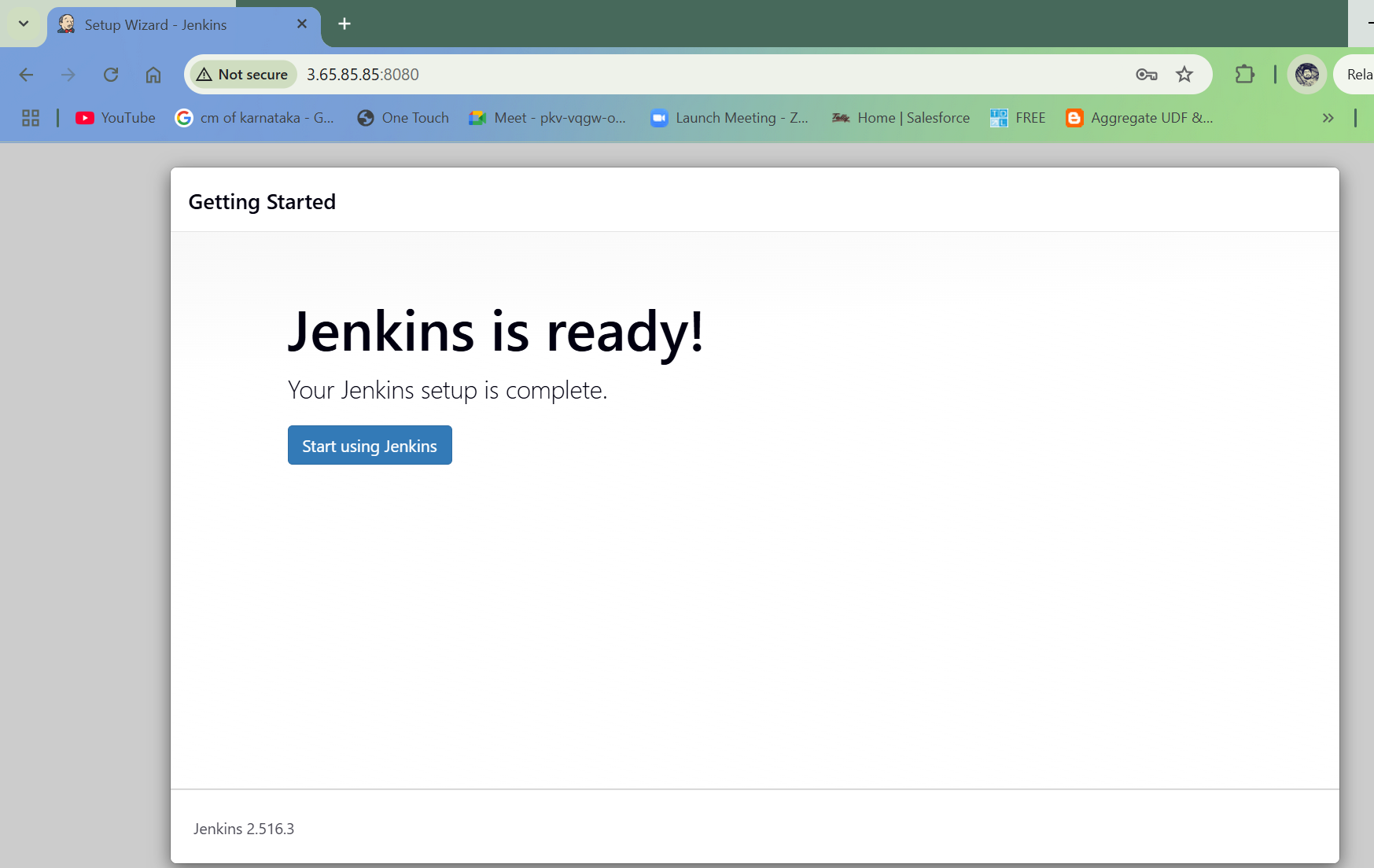
* 1. Enter details and click on “Save and Continue”



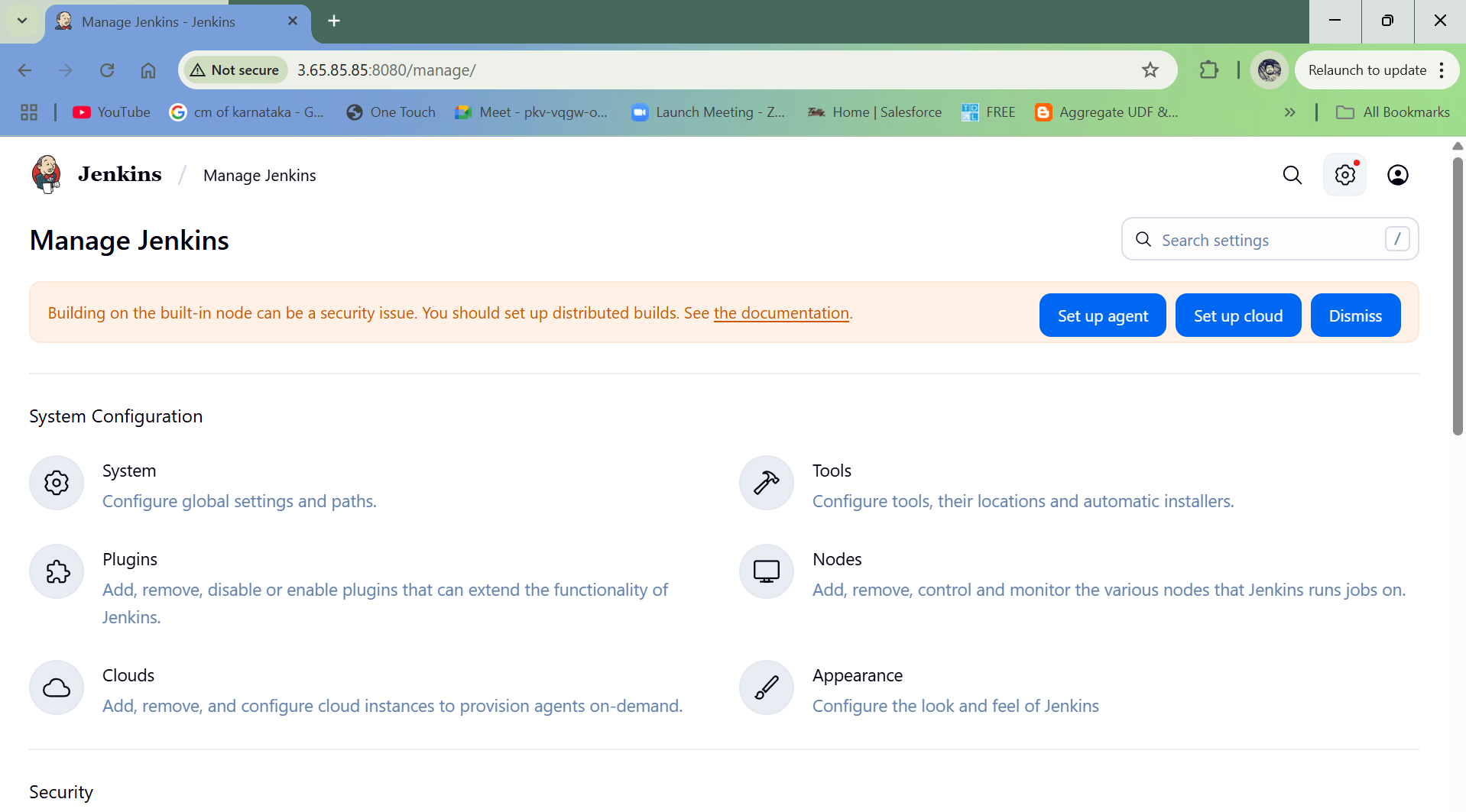
* 1. Click on “Save and finish”



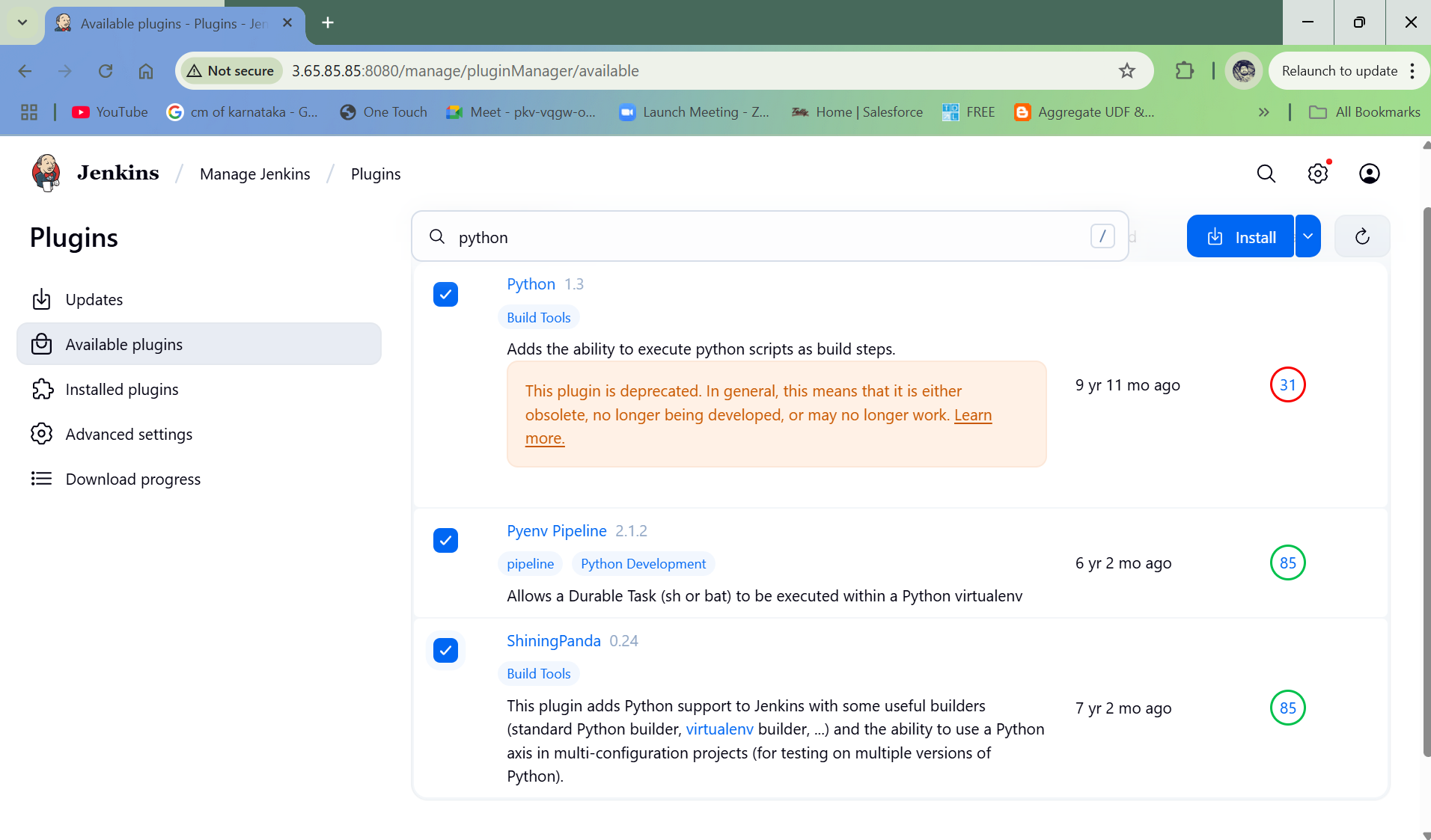
* 1. Jenkins set up is completed. Click on “Start using Jenkins”



1. Configure Jenkins with Python and necessary plugins
   1. Go to “Manage Jenkins” and click on “Plugins”

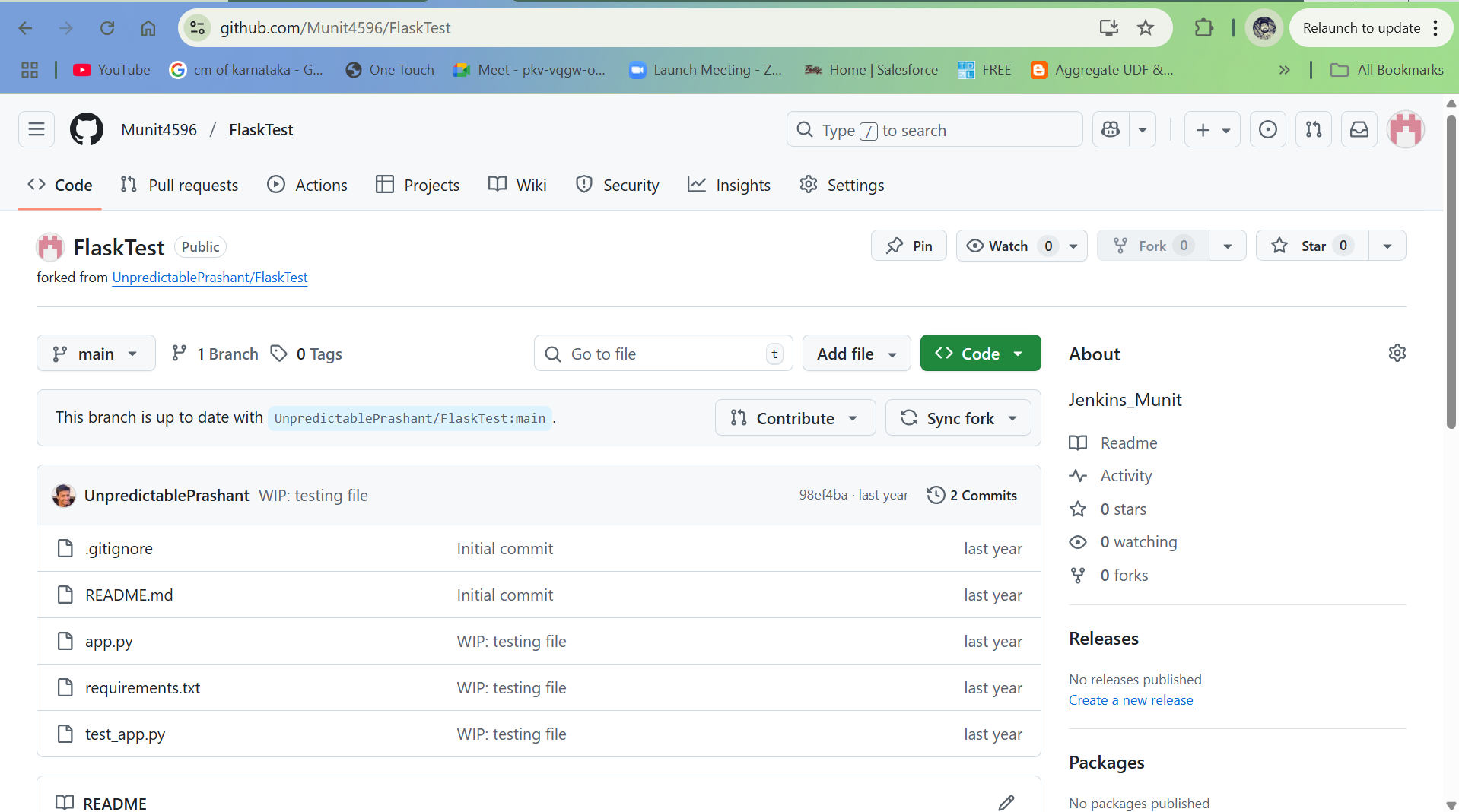


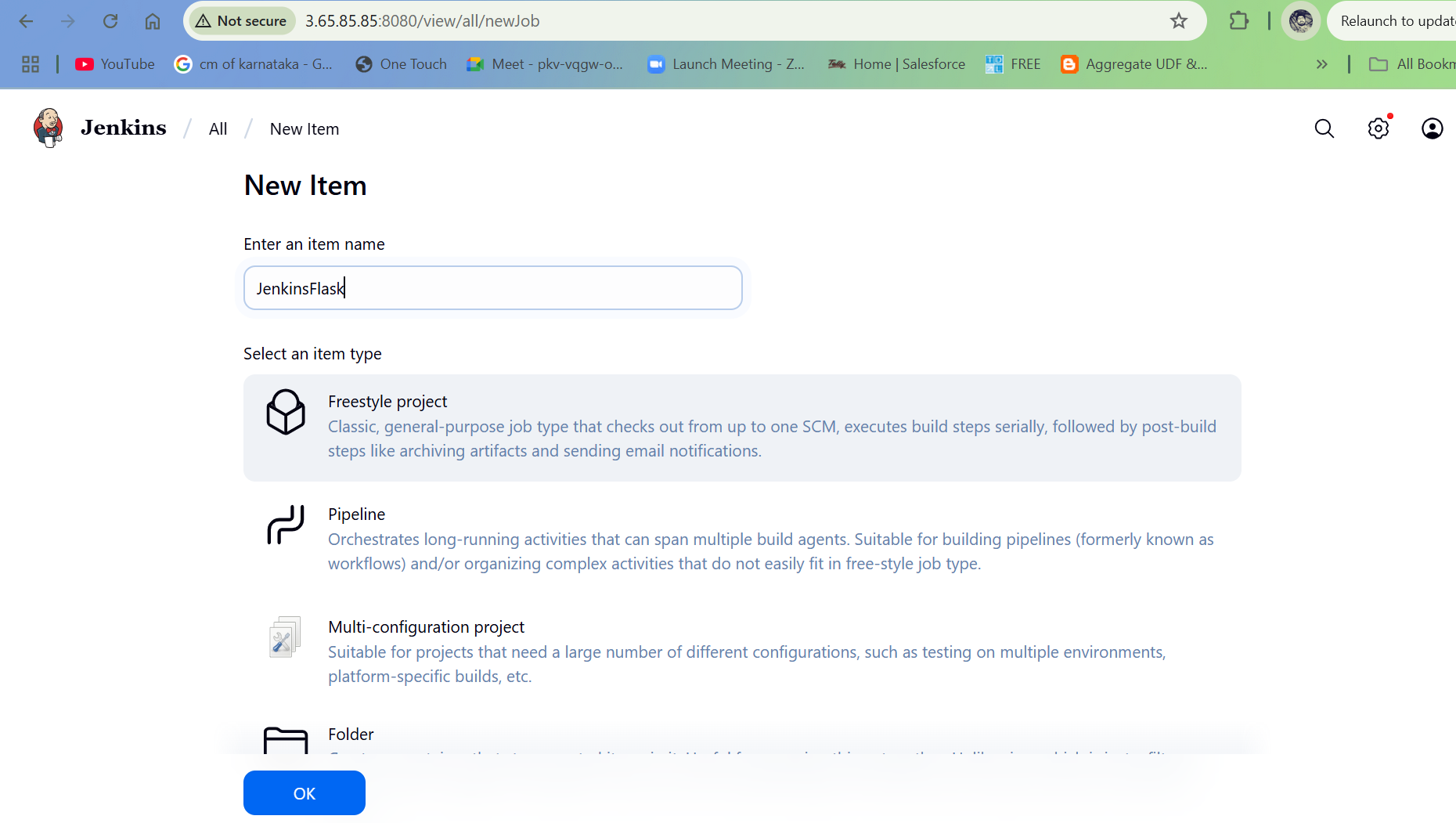
* 1. Search for “Python” and install the required plugins.

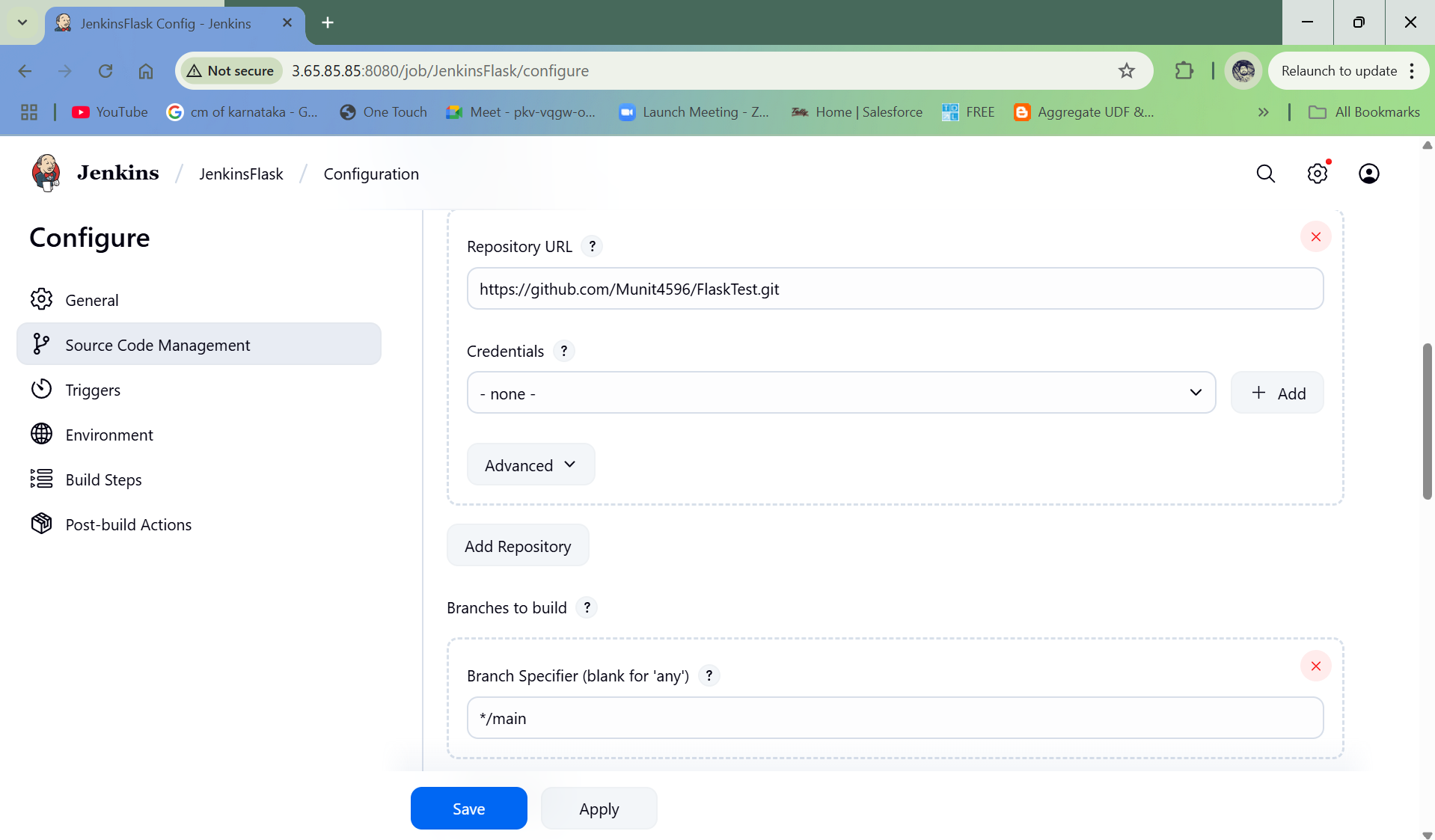


* 1. Once all the plugins are installed, return to the Jenkins Dashboard.

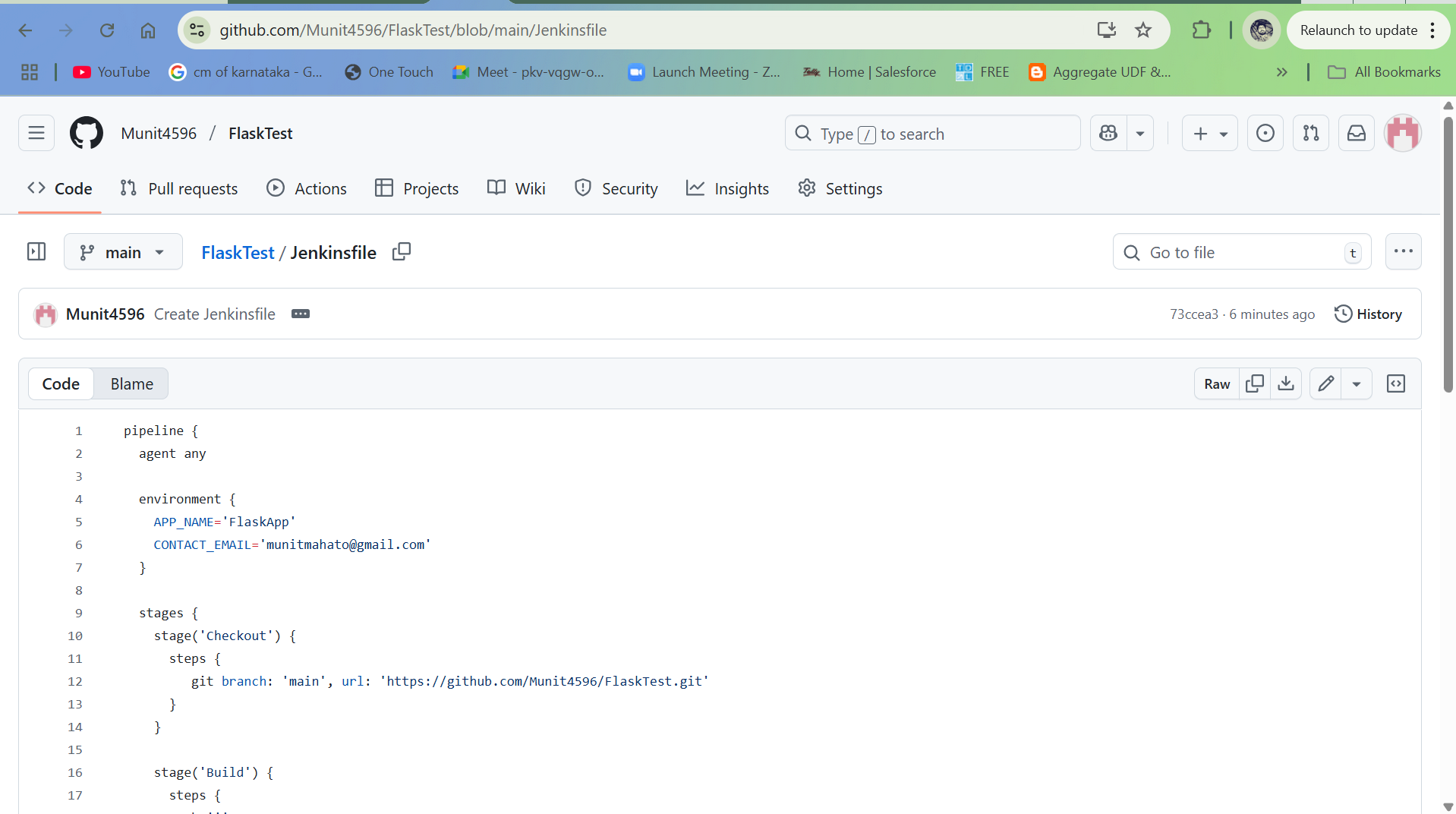
1. Fork the provided Python web application repository on GitHub (provide a link to a sample Python web application repository).
   1. Navigate to <https://github.com/UnpredictablePrashant/FlaskTest.git> and fork the repository.



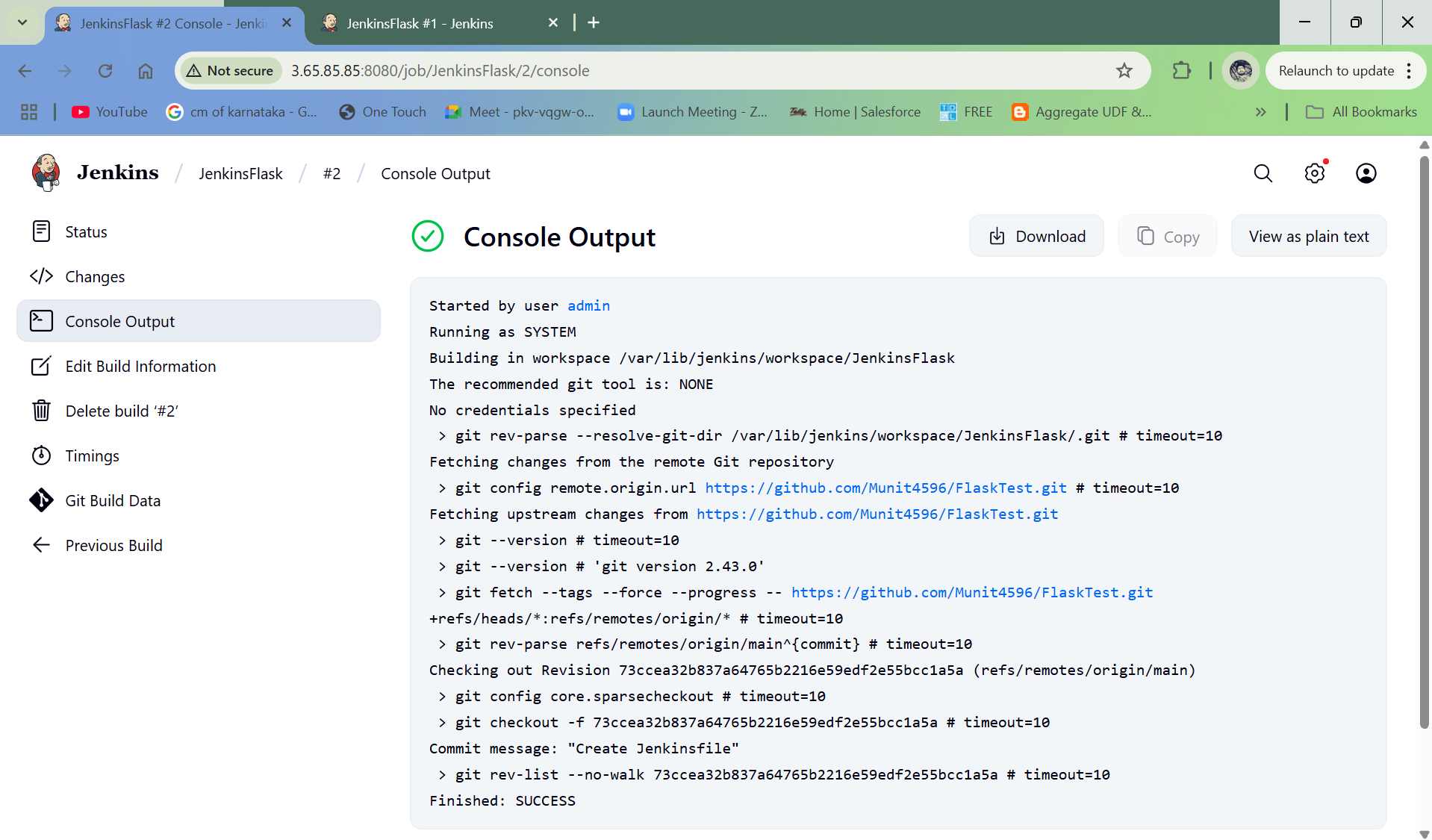
1. Clone the forked repository into your Jenkins server.
   1. From Jenkins Dashboard, click on “New Item” > Freestyle Project. 
   2. Provide the Repository URL and Branch. Click on Apply and then Save.



1. Jenkins Pipeline
   1. Create a Jenkinsfile in the root of Python application repository and Commit Changes.



* 1. Go to Jenkins Dashboard and then navigate to the item which has been forked. Click on “Build Now”

****

* 1. Finished:SUCCESS is observed in the output