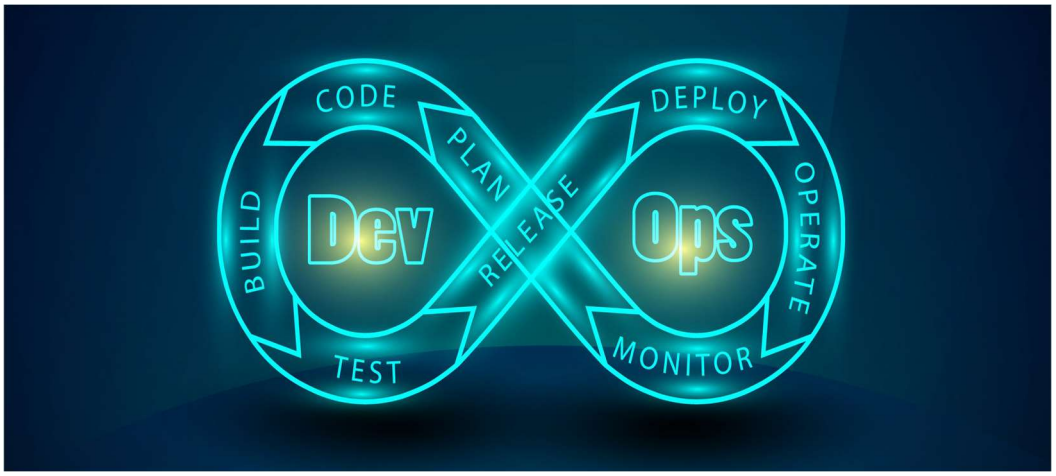


INTRODUCTION TO DEVOPS

S2-22_SESAPZG515

DONE BY

Krishna, Simran
2022SP93047

TABLE OF CONTENT

01 GIT

02 JENKINS

03 SELENIUM

04 SONARQUBE

05 KUBERNETES

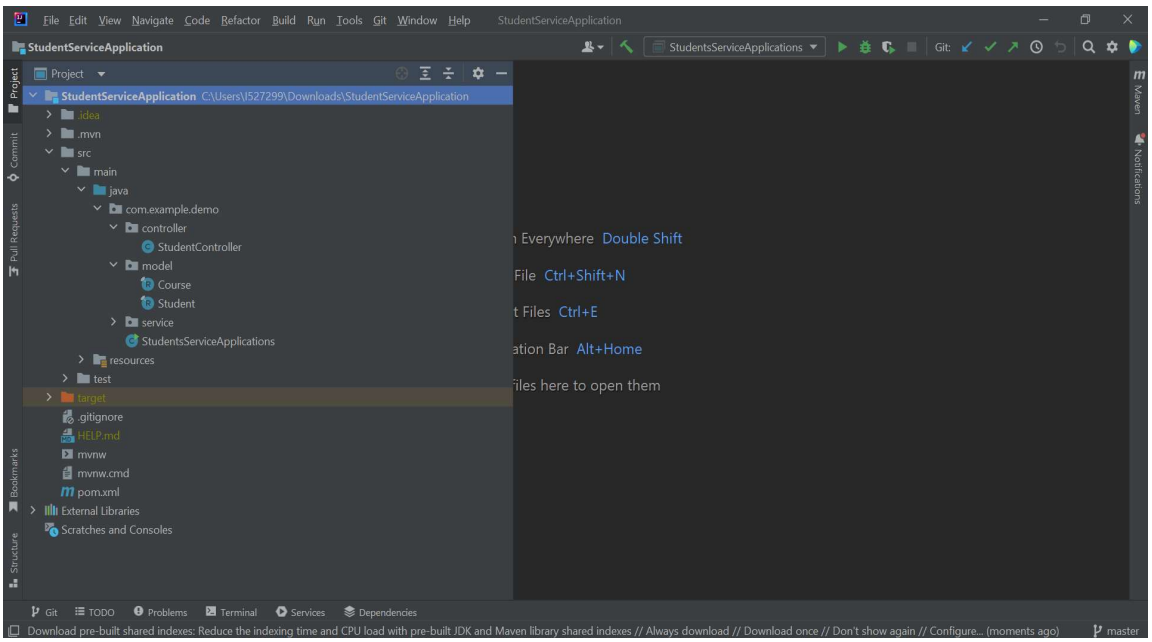
Devops Assignment

1. ## Git (3)
    1. ### Get your project up and running locally

Project – Student Service Application

In this project, there are three services using proper URIs and HTTP methods:

- `@GetMapping("/students/{studentId}/courses")`: You can ask the courses a specific student has registered for using request method Get and example uri /students/Student1/courses.
- `@GetMapping("/students/{studentId}/courses/{courseId}")`: You can ask a specific course for a specific student using request method Get and example uri /students/Student1/courses/Course1.
- `@PostMapping("/students/{studentId}/courses")` : You can register a student for a course by sending a POST request to URI /students/Student1/courses



A few details:

- StudentController.java - Rest controller exposing all the three service methods discussed above.
- Course.java, Student.java, StudentService.java - Business Logic for the application. StudentService exposes a couple of methods we would consume from our Rest Controller.
- StudentControllerIT.java - Integration Tests for the Rest Services.
- StudentControllerTest.java - Unit Tests for the Rest Services.
- StudentServicesApplication.java - Launcher for the Spring Boot Application. To run the application, just launch this file as Java Application.
- pom.xml - Contains all the dependencies needed to build this project. We will use Spring Boot Starter Web.

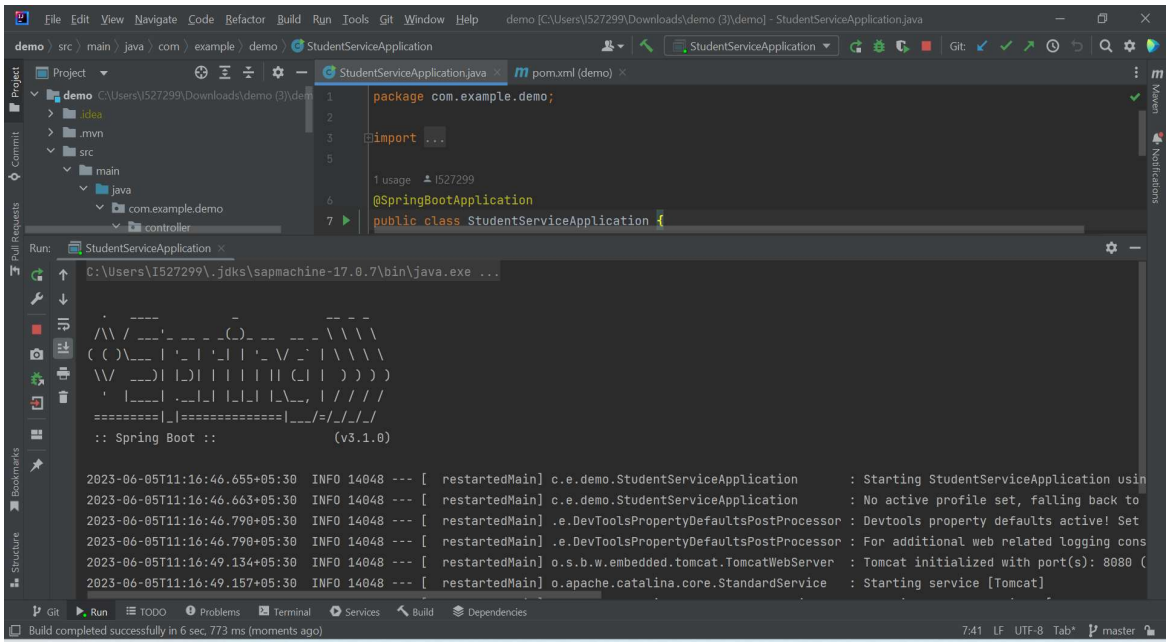
We have StudentService exposing methods to

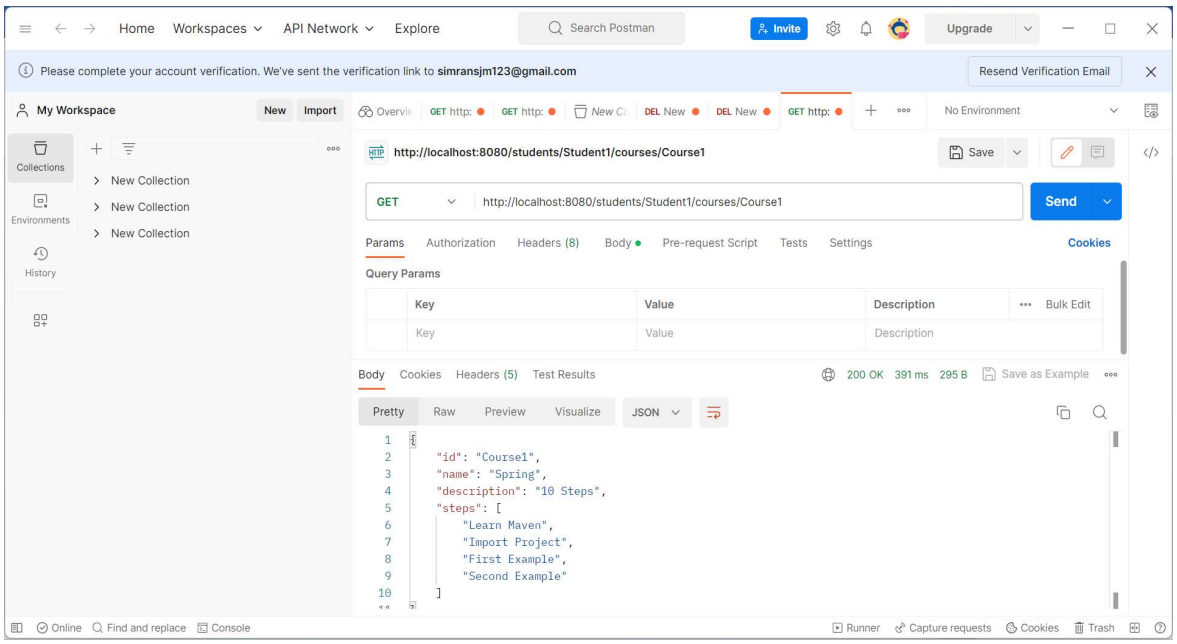
- public List<Student> retrieveAllStudents() - Retrieve details for all students
- public Student retrieveStudent(String studentId) - Retrieve a specific student details
- public List<Course> retrieveCourses(String studentId) - Retrieve all courses a student is registered for
- public Course retrieveCourse(String studentId, String courseId) - Retrieve details of a specific course a student is registered for
- public Course addCourse(String studentId, Course course) - Add a course to an existing student

Executing the Http Get Operation Using Postman

We will access a request to http://localhost:8080/students/Student1/courses/Course1 to test the service. And will receive the following response shown in the below.

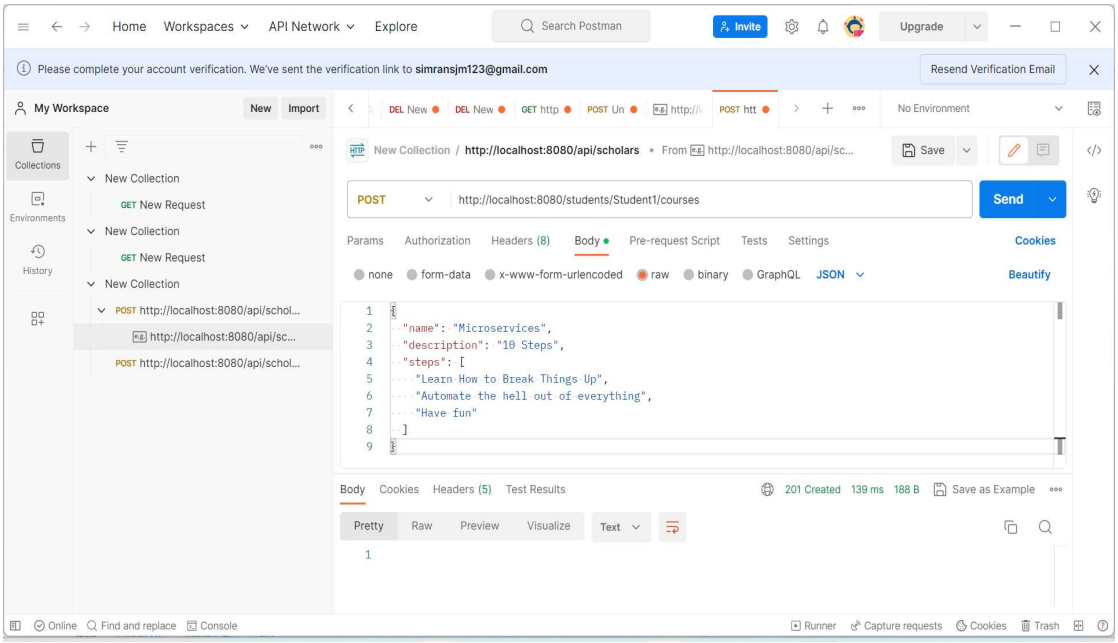
Below picture shows how we can execute this Get Service from Postman





Executing a Http POST Operation Rest Service

Example request is shown below. It contains all the details to register a course to a student.

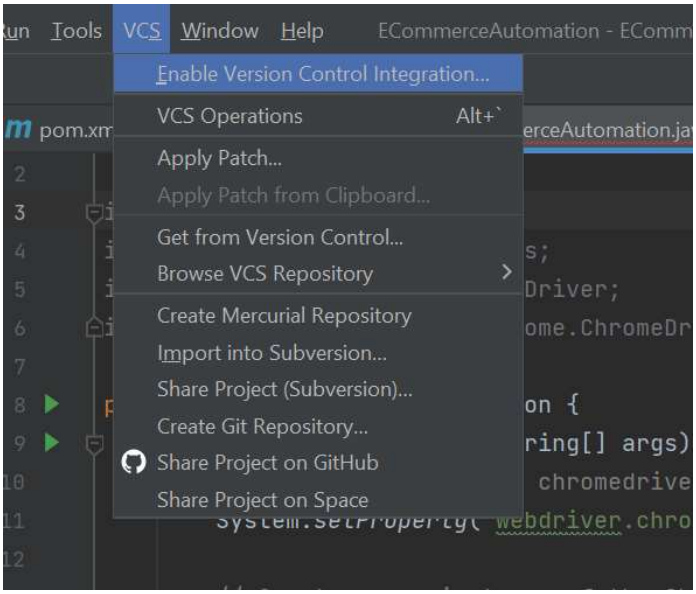


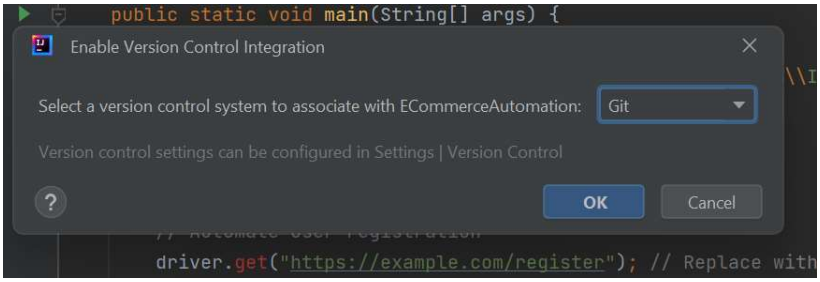
PUSHED THE PROJECT TO GITHUB THROUGH INTELIJ

STEPS

To push your project to GitHub using IntelliJ without using the terminal, you can follow these steps:

1. Make sure you have a Git repository initialized in your project. You can do this by going to "VCS" in the top menu, selecting "Enable Version Control Integration," and choosing Git.

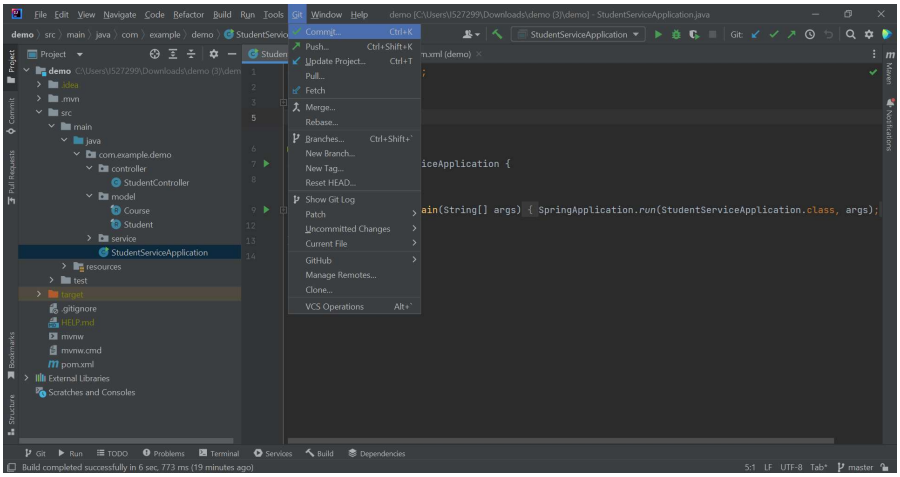




2. Once your project is under version control, go to the "Version Control" panel in IntelliJ. You can find it by going to "View" in the top menu, selecting "Tool Windows," and choosing "Version Control."

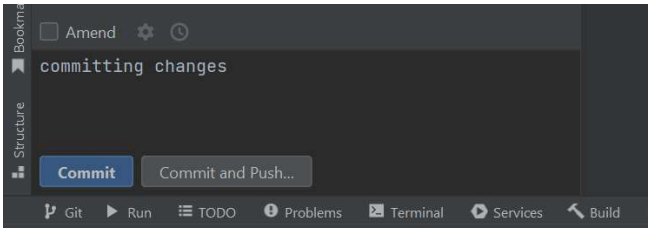
3. In the "Version Control" panel, you should see your changed files under the "Local Changes" tab. Select the files you want to commit and push to GitHub.

4. Right-click on the selected files and choose "Commit" from the context menu. Alternatively, you can click on the "Commit" button in the top toolbar.

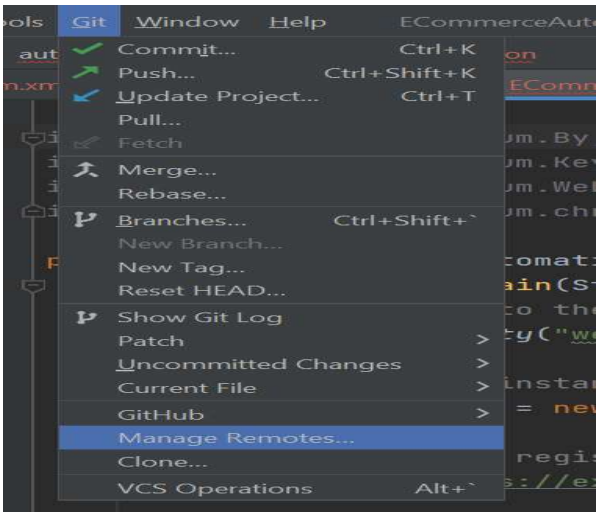


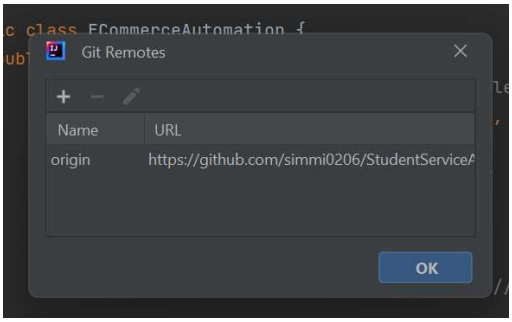
5. In the commit dialog, enter a commit message describing your changes. Make sure to provide a meaningful and concise message.

6. After entering the commit message, click on the "Commit and Push" button at the bottom of the dialog.

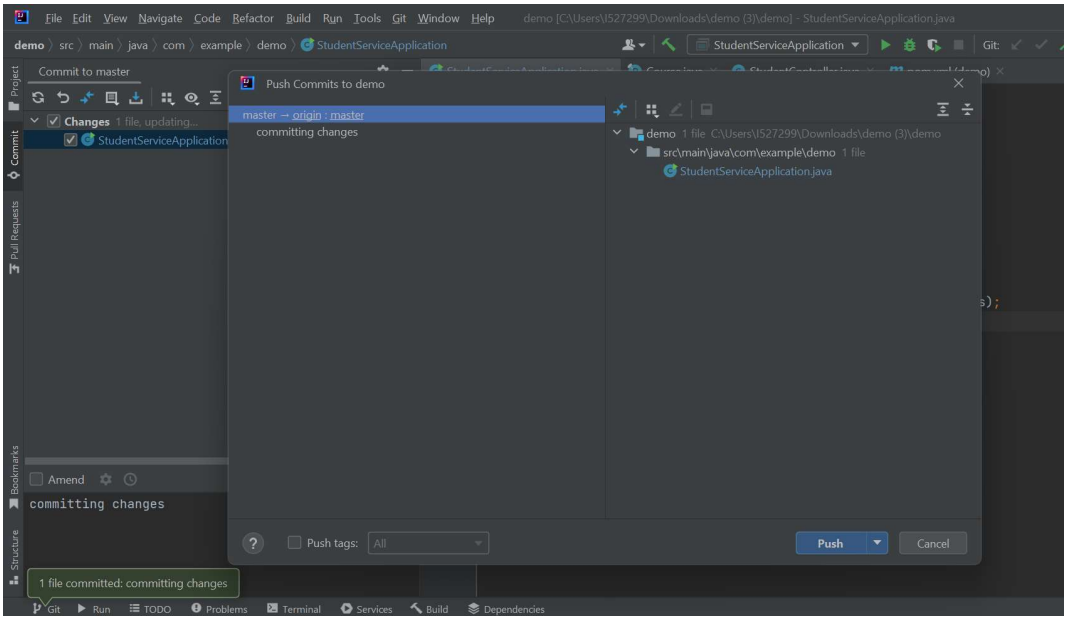


7. In the next dialog, select the remote repository where you want to push your changes. If you haven't added your GitHub account as a remote, click on the "+" button and enter the repository URL and your GitHub credentials.

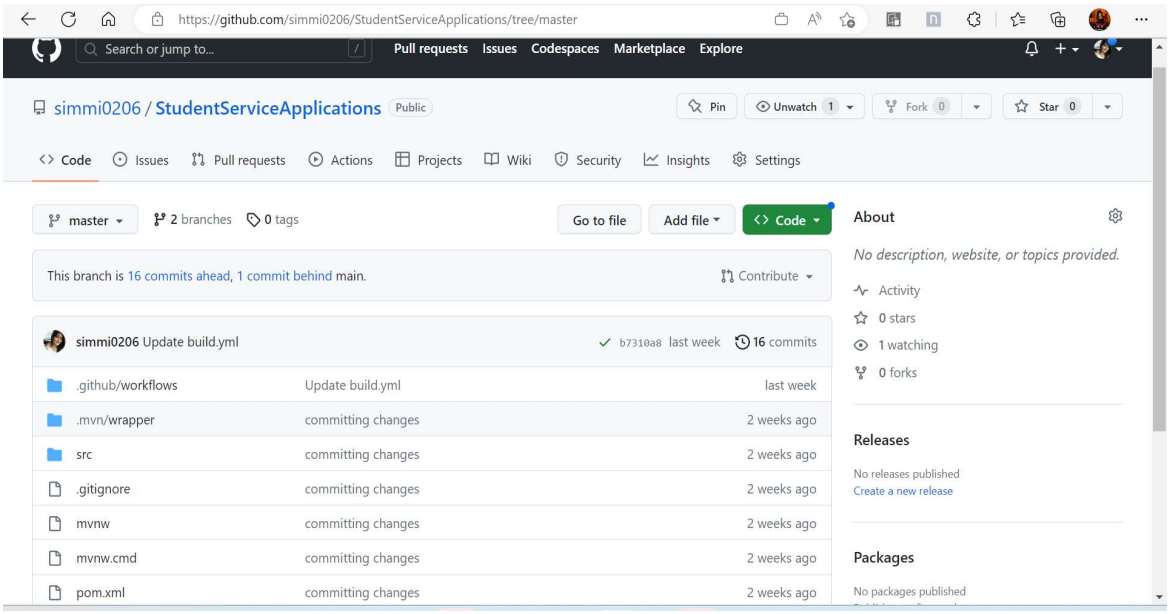




8. Once you have selected the remote repository, click on the "Push" button.



9. IntelliJ will now push your changes to the selected remote repository (in this case, GitHub).



Build an initial CI pipeline on Gitlab

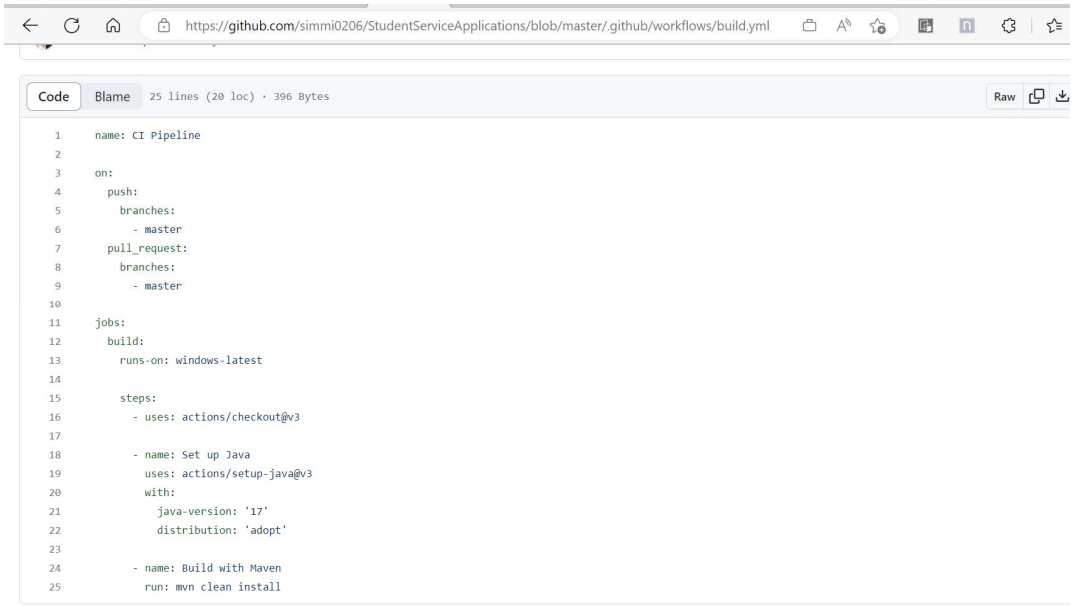
To build an initial CI (Continuous Integration) pipeline on GitHub, following are the steps:

1. Create a GitHub repository: Start by creating a new repository on GitHub, push code.

2. Choose GitHub Actions: GitHub Actions is an integrated CI/CD platform provided by GitHub. It supports Windows environments and can be used to build CI pipelines.

3. Define the CI workflow: In your repository, create a .github/workflows directory if it doesn't exist. Inside this directory, create a new YAML file (e.g., ci.yml) to define your CI workflow.

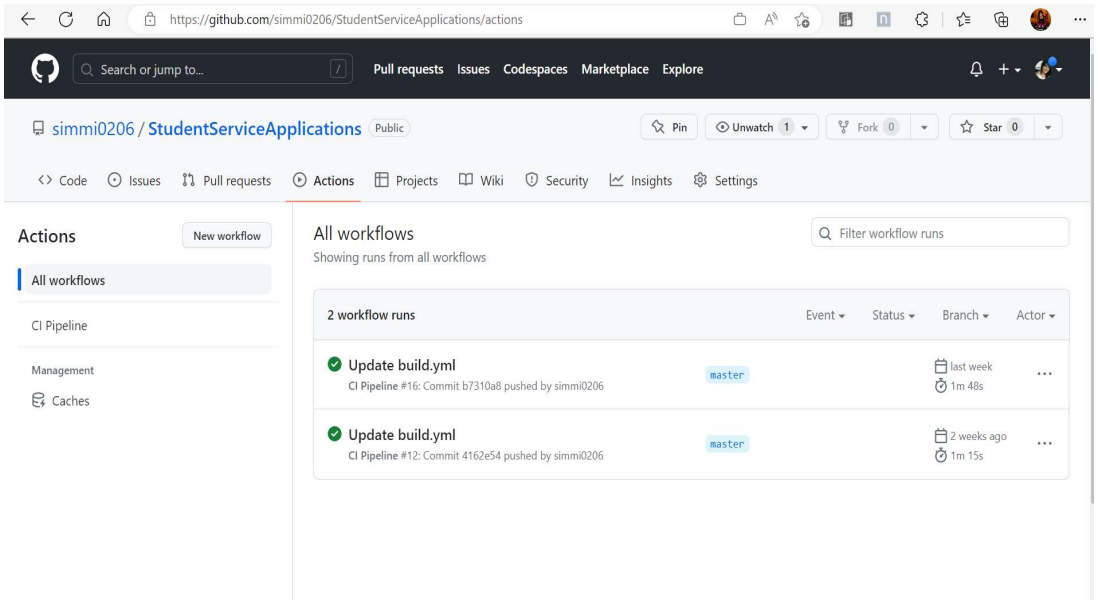
4. Configure workflow triggers: Within the YAML file, define the triggers that will activate the CI pipeline. For example, you can set it to trigger on every push to the repository or only on specific branches

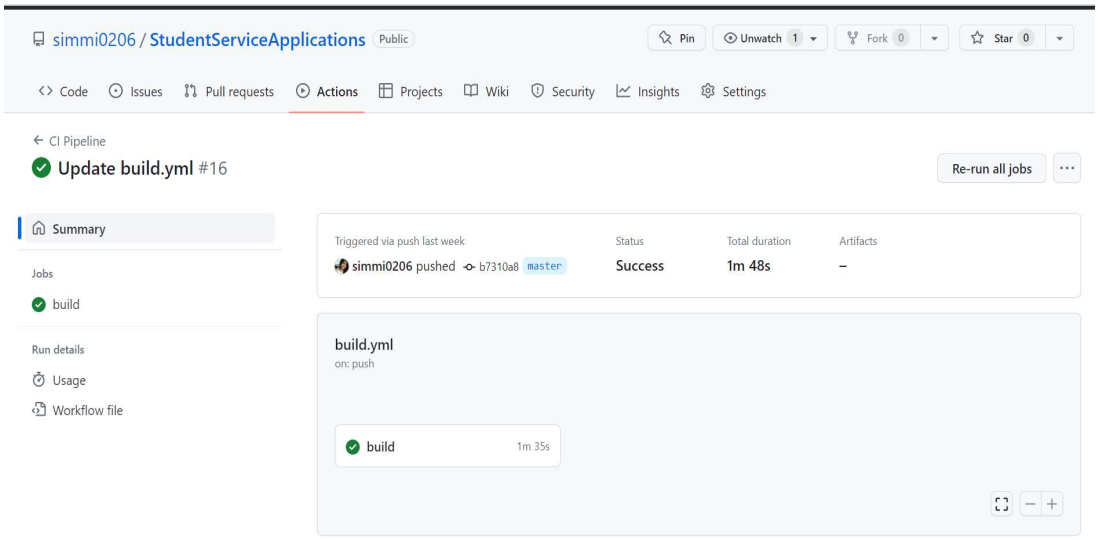


5. Specify the workflow steps: Define the steps required to build and test your Java Maven project. Here's an example workflow that builds the project using Maven. In this example, the workflow checks out the code, sets up Java 19 using the actions/setup-java action, and then builds and tests the project using Maven.

6. Commit and push the YAML file: Once you've defined the workflow, commit and push the YAML file to the .github/workflows directory in your repository. GitHub Actions will automatically detect the new workflow and start executing it based on the defined triggers.

7. Monitor the workflow: You can monitor the progress of your workflow by visiting the "Actions" tab in your GitHub repository. It will show you the status of each workflow run, including any errors or failures.


https://github.com/simmi0206/StudentServiceApplications/actions
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
simmi0206 / StudentServiceApplications
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Actions
New workflow
All workflows
CI Pipeline
Management
Caches
All workflows
Showing runs from all workflows
Filter workflow runs
2 workflow runs
Event
Status
Branch
Actor
Update build.yml
CI Pipeline #16: Commit b7310a8 pushed by simmi0206
master
last week
1m 48s
Update build.yml
CI Pipeline #12: Commit 4162e54 pushed by simmi0206
master
2 weeks ago
1m 15s


simmi0206 / StudentServiceApplications
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
CI Pipeline
Update build.yml #16
Re-run all jobs
Summary
Jobs
build
Run details
Usage
Workflow file
Triggered via push last week
simmi0206 pushed b7310a8 master
Status
Success
Total duration
1m 48s
Artifacts
–
build.yml
on: push
build
1m 35s

2. Maven and Jenkins

a. Setup a Jenkins Job with Apache Maven

To set up a Jenkins job with Apache Maven, steps are Following:

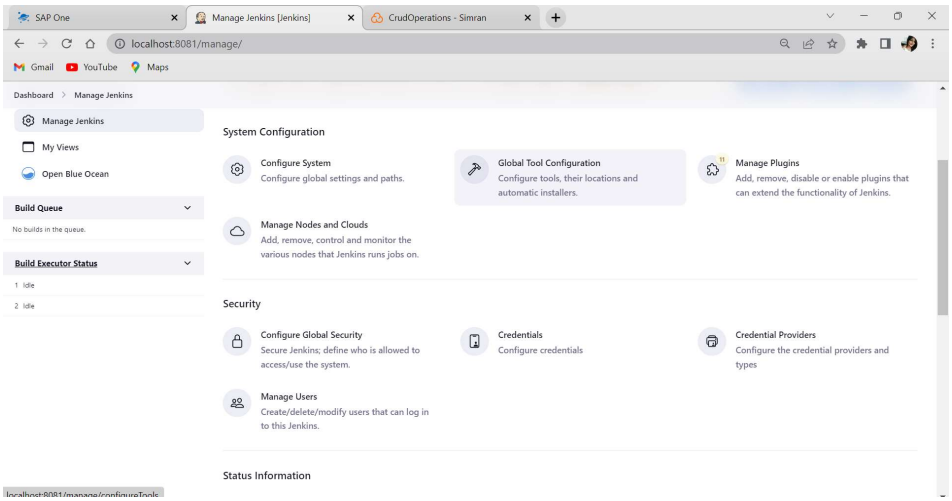
1. Install Jenkins: Download and install Jenkins on your server or local machine. Visit the official Jenkins website for installation instructions: https://www.jenkins.io/download/

2. Configure Jenkins:

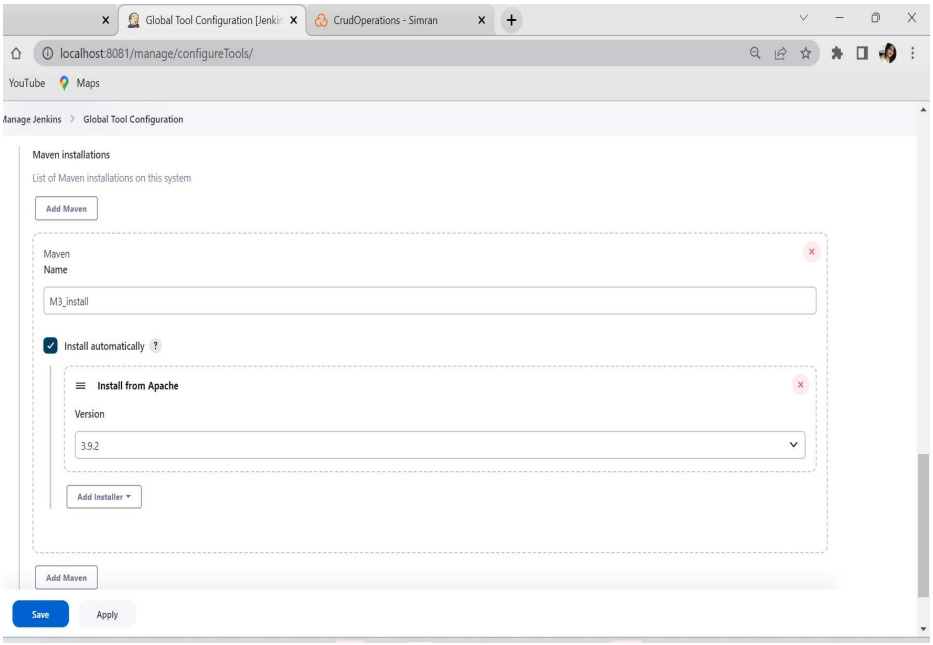
- • Access Jenkins through your web browser using the URL: http://localhost:8080 (if running locally) or the appropriate server address.
- Complete the initial setup by following the on-screen instructions, including creating an admin user and installing recommended plugins.

3. Install Maven on Jenkins: Open Jenkins and navigate to "Manage Jenkins" from the main dashboard.

- Select "Global Tool Configuration" to manage Jenkins tools.

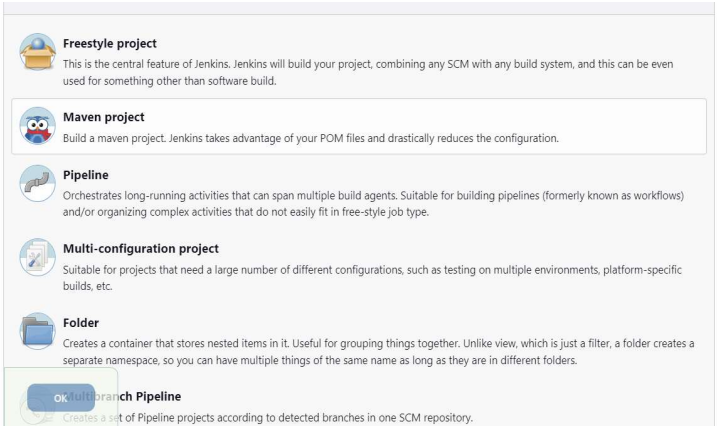


- Scroll down to the "Maven" section and click on "Add Maven."
- Provide a name for the Maven installation and select the Maven version to install.

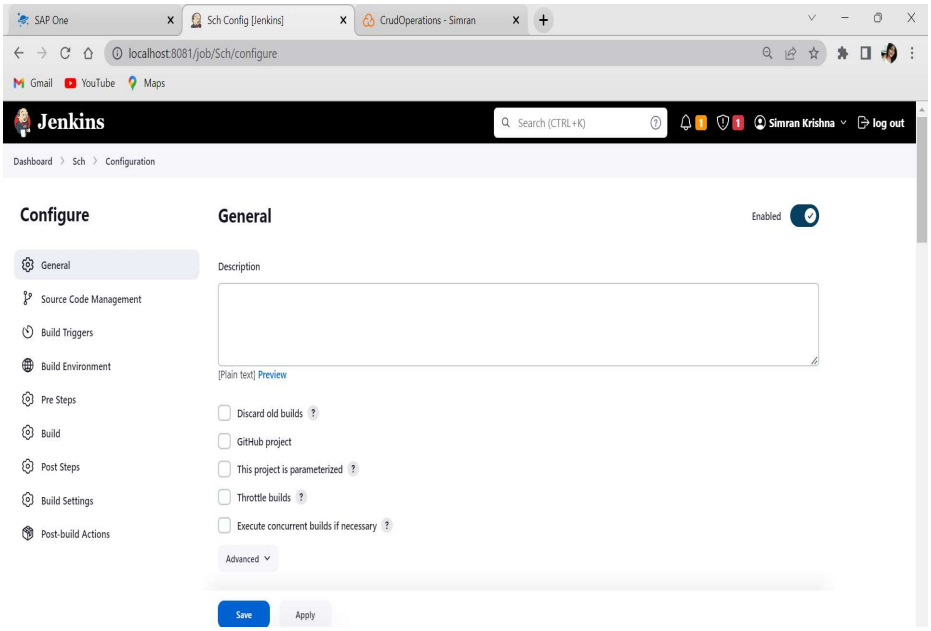


- Save the configuration.

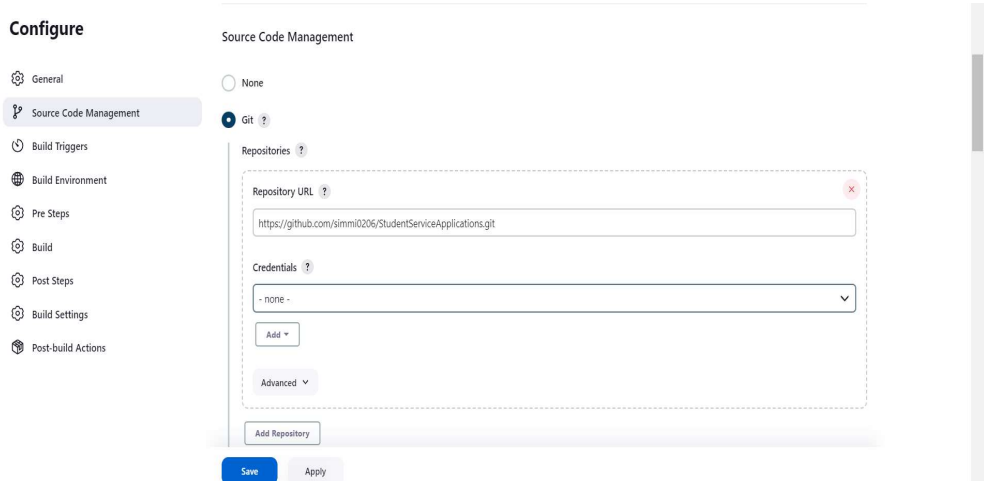
4. Create a new Jenkins job:
   - From the Jenkins main dashboard, click on "New Item" to create a new job.
   - Enter a name for your job and select the "Maven Project" option.



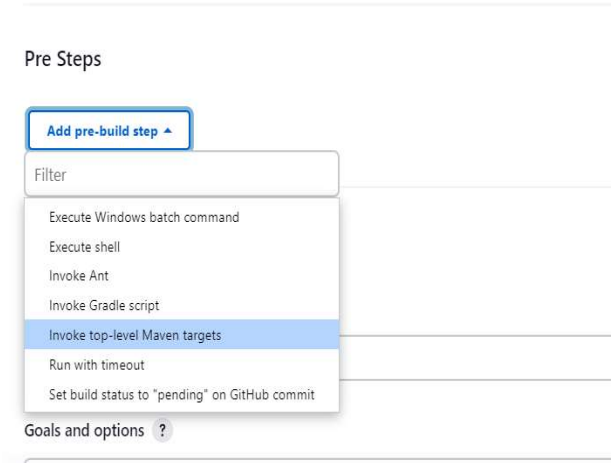
   - From the Jenkins main dashboard, click on "New Item" to create a new job.
   - Enter a name for your job and select the "Maven Project" option.
   - Click on "OK" to proceed.
   - Configure the Jenkins job:



5. Job configuration page → Source Code Management → Add git repository https or SSH link.



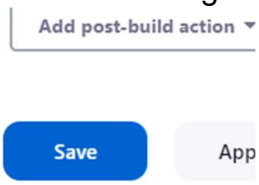
6. Under the "Build" section, click on "Add build step" and select "Invoke top-level Maven targets."



7. In the "Goals" field, specify the Maven goals you want to execute (e.g., clean install, test, package, etc.).

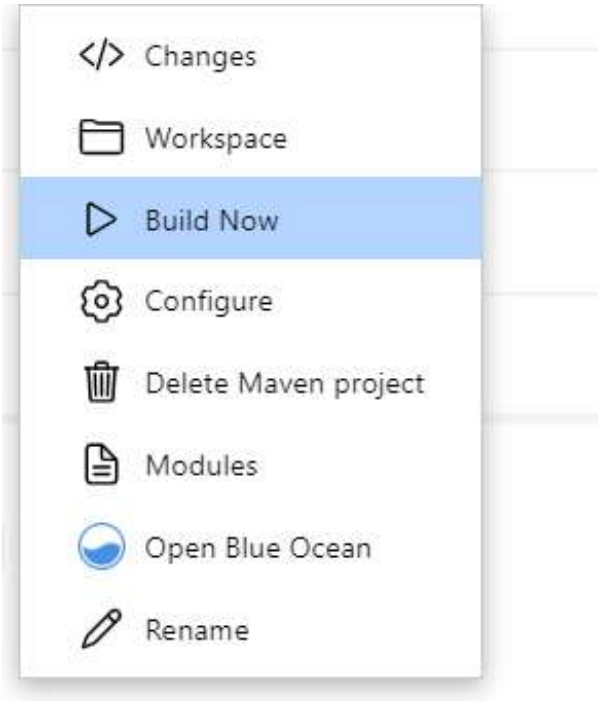
8. Set additional Maven options or specify the path to the Maven installation if different from the default.

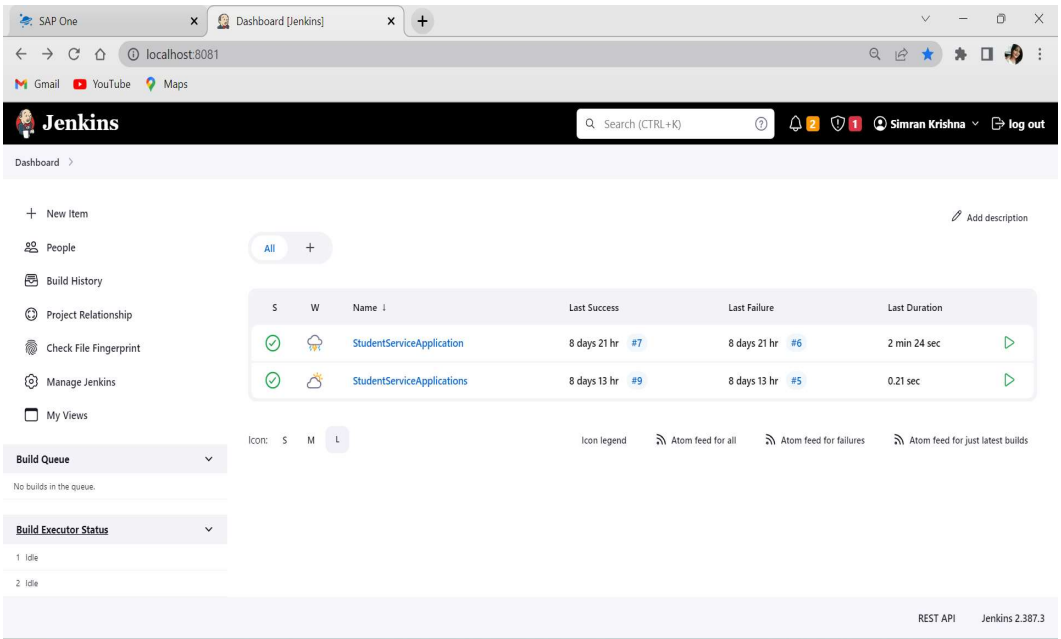
9. Save the configuration.



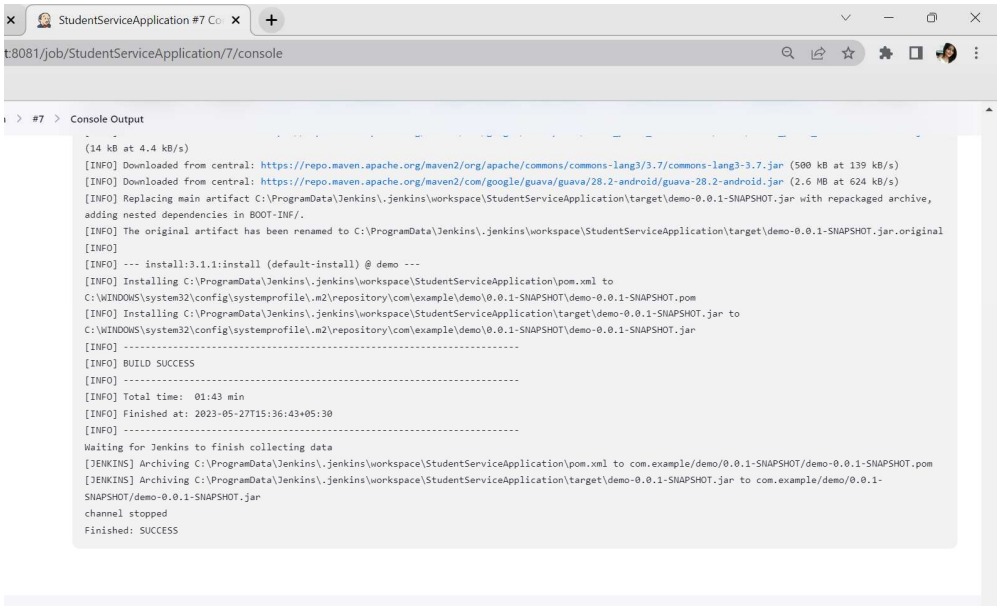
10. Build the Jenkins job:

11. On the main job dashboard, click on "Build Now" to initiate a build. (here Application is Build)





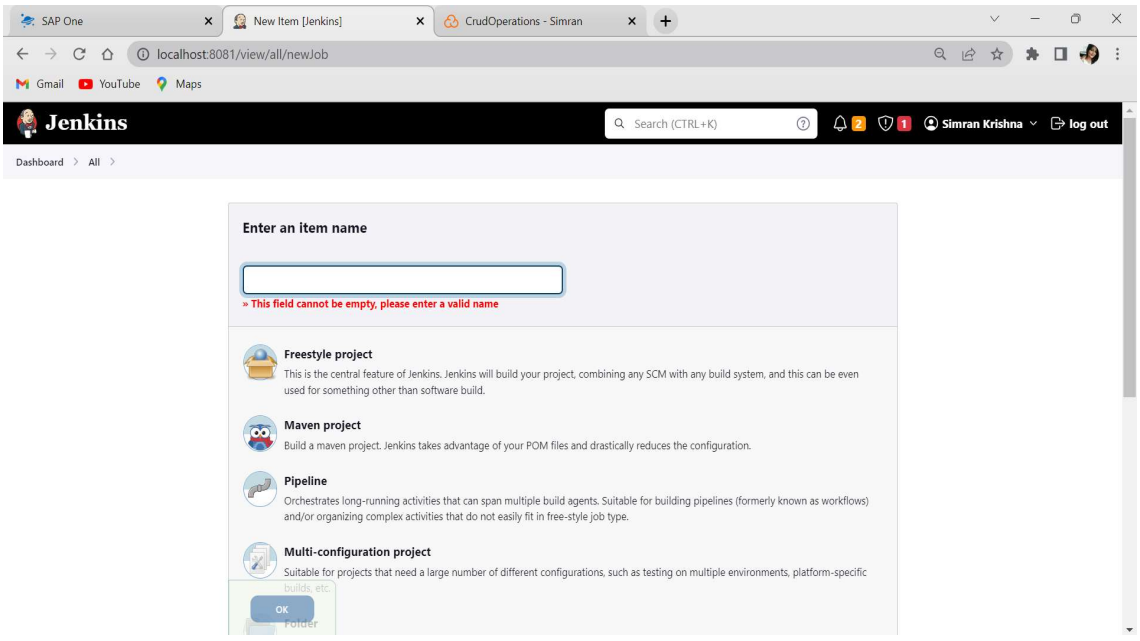
12. The console output will display the build progress, including any errors or warnings encountered



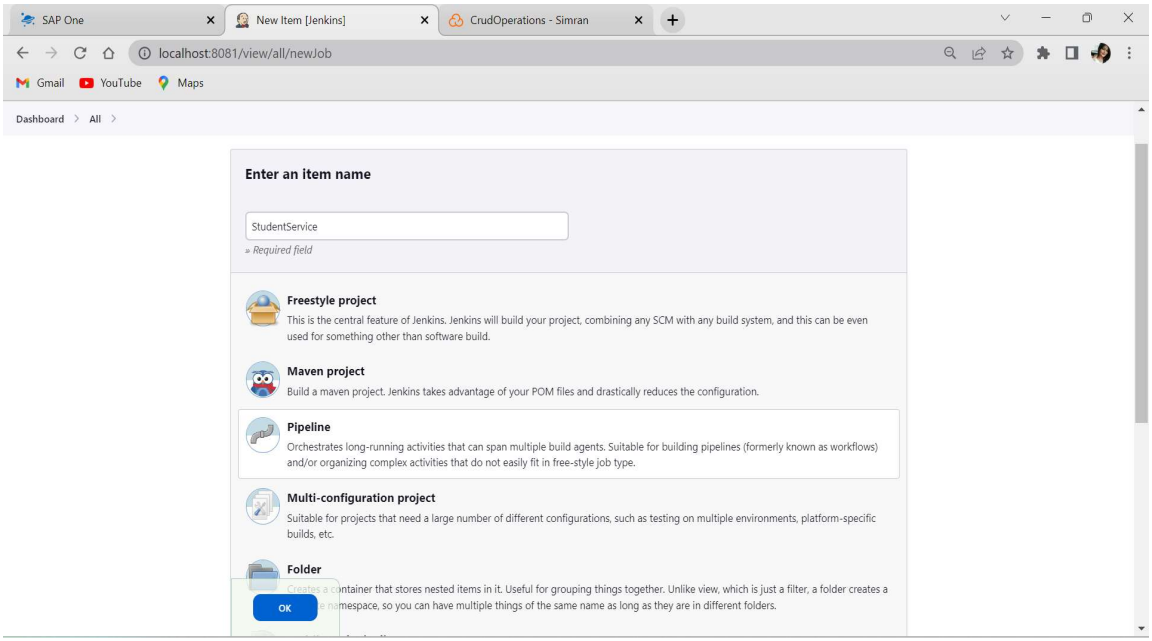
b. Setup a Jenkins build Pipeline with Build, Test, Package, Deploy

Step 1: Create a new Jenkins Pipeline job:

- From the Jenkins main dashboard, click on "New Item" to create a new job.

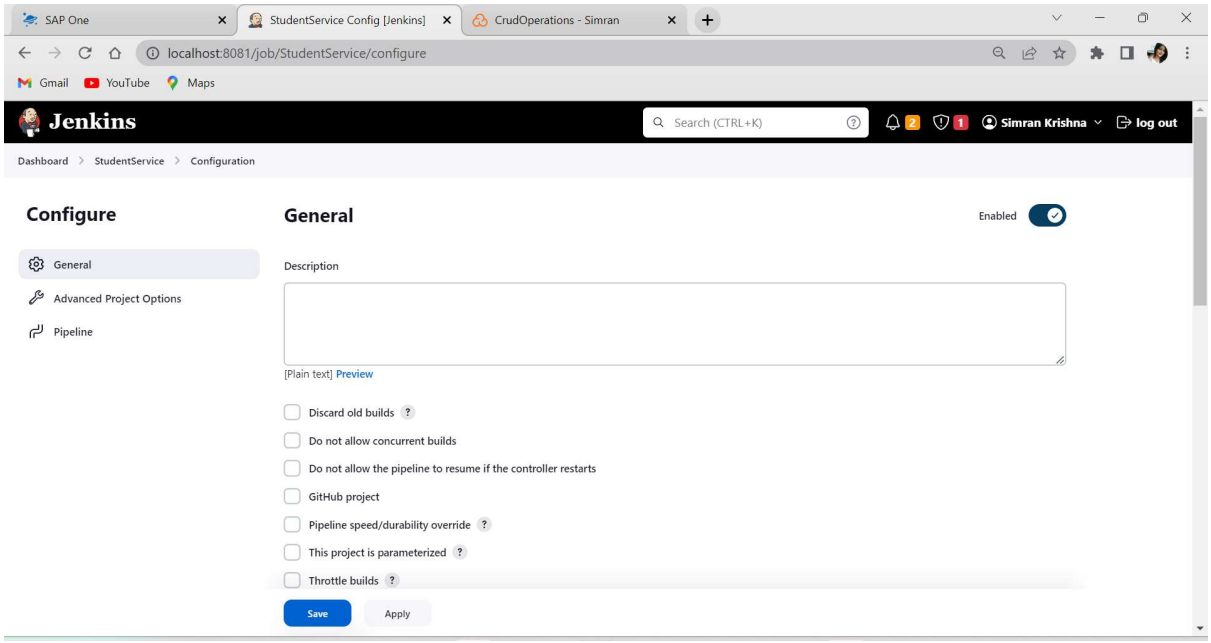


- Enter a name for your pipeline job and select the "Pipeline" option.



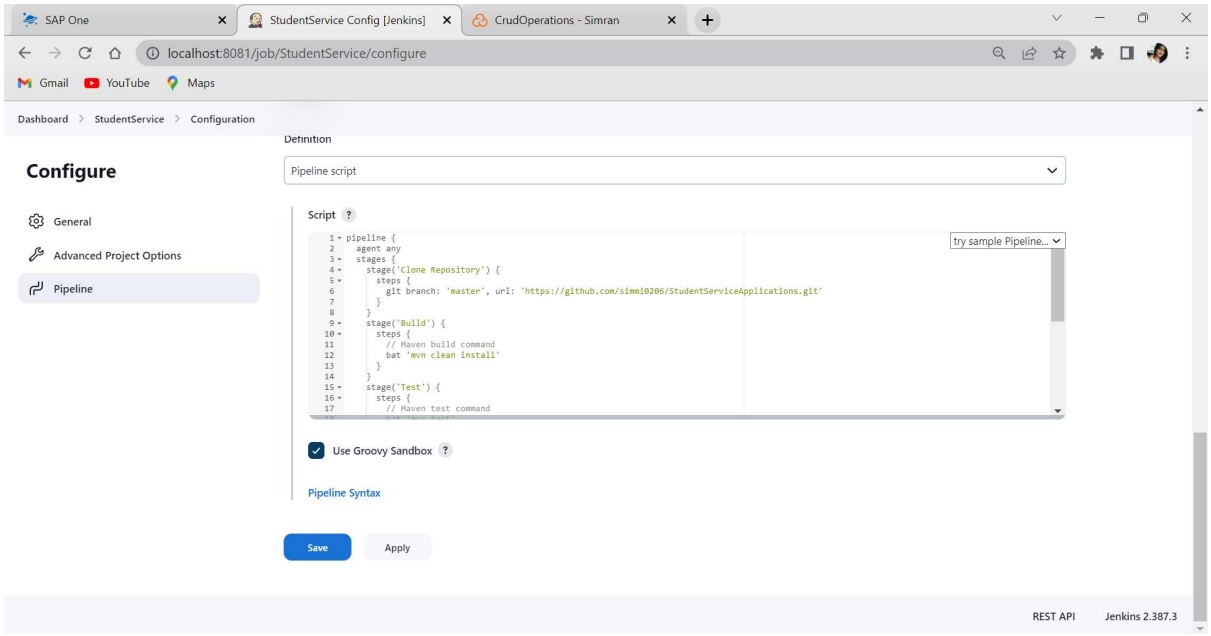
- Click on "OK" to proceed.

Step 2: Configure the Jenkins Pipeline job:



• In the pipeline configuration page, scroll down to the "Pipeline" section.

• In the "Definition" dropdown, select "Pipeline script"

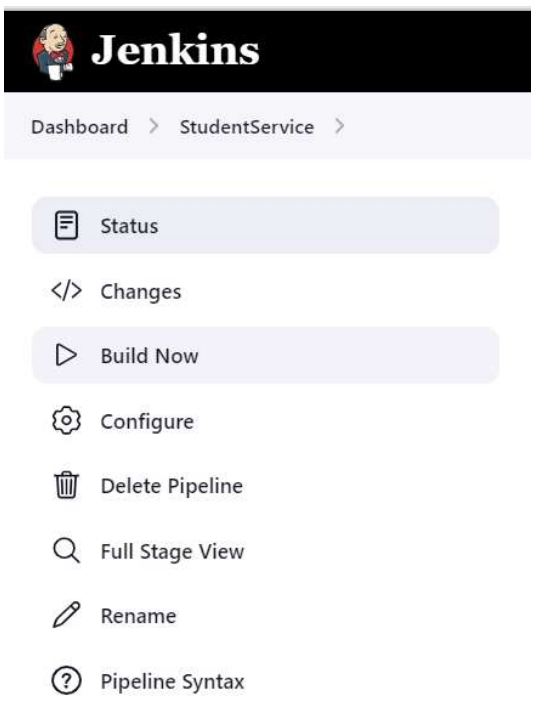


• In the script area, you can define your pipeline stages and steps. Use the following example script as a starting point

Step 3: Save the configuration.

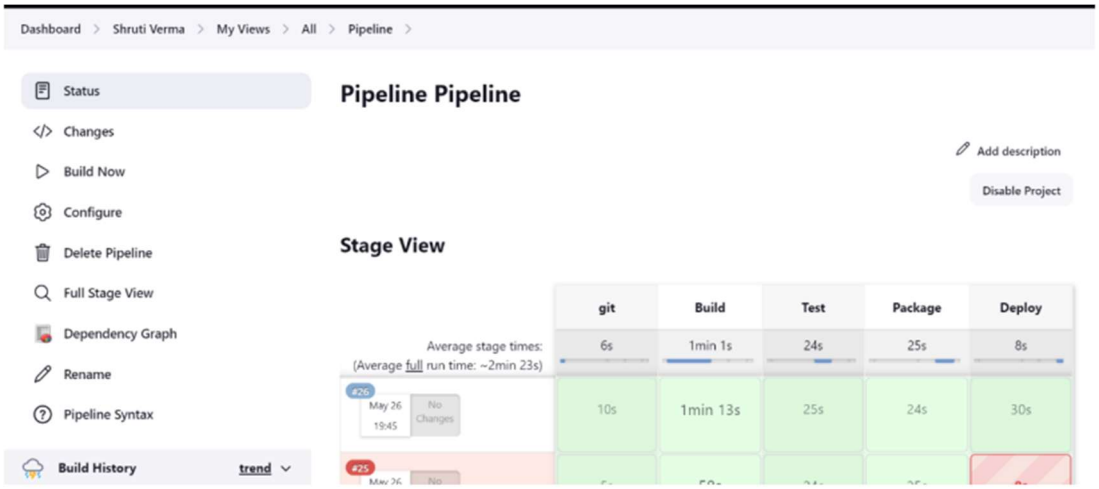
Step 4: Run the Jenkins Pipeline job:

• On the main job dashboard, click on "Build Now" to start the pipeline.



• Jenkins will execute each stage sequentially, running the defined steps within each stage.

• The console output will display the progress of each stage and any errors or warnings encountered



3.Selenium, SonarQube (3)

1. **Automate user registration & search product feature of e-commerce website with selenium**

   Automating user registration and search product features of an e-commerce website using Selenium can be done by following these steps:

   1. Install Selenium WebDriver: Download the latest version of Selenium WebDriver for your preferred programming language and install it.

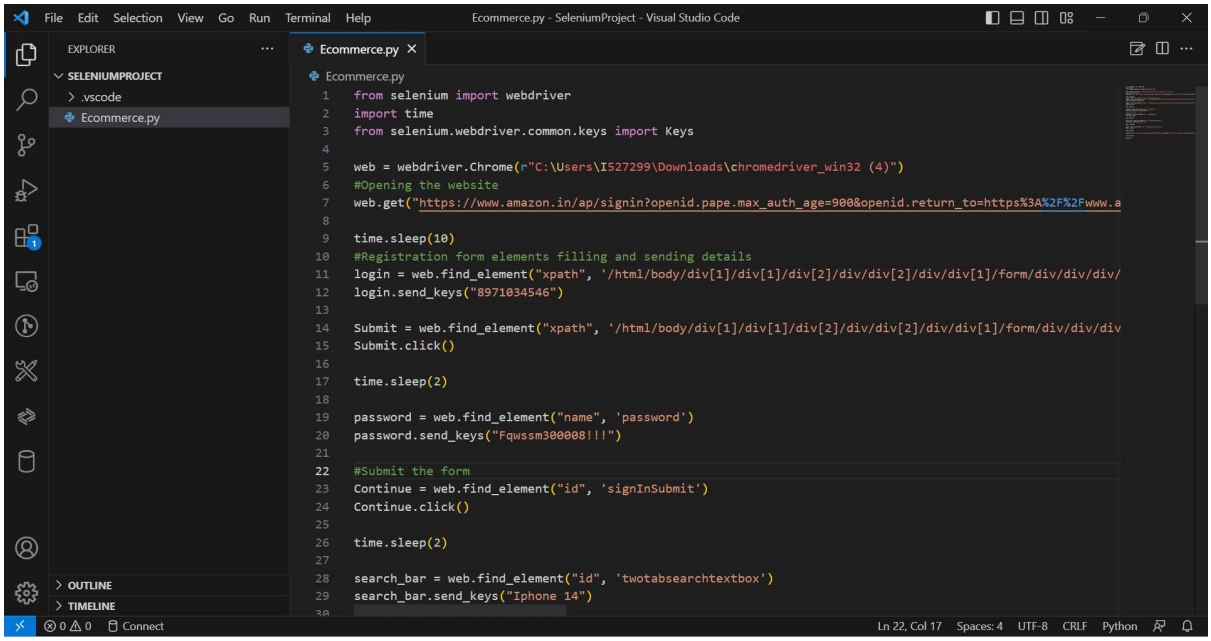
   2. Install selenium in CMD: pip install selenium

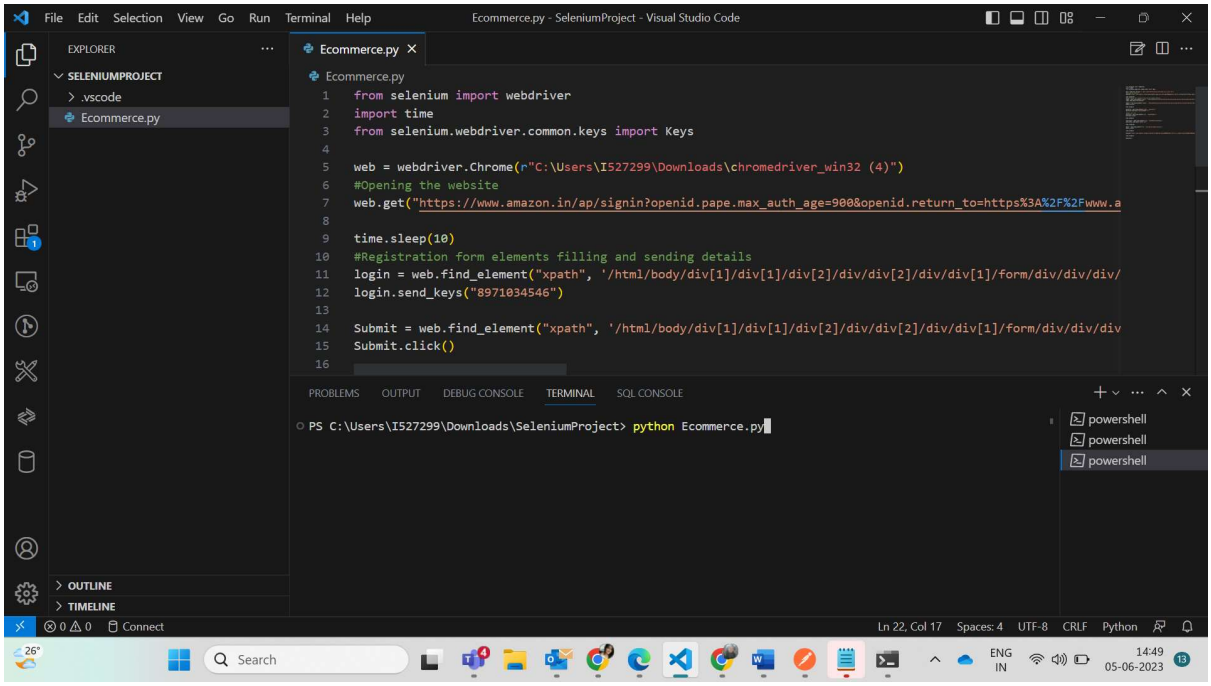
   3. Set up the development environment: Create a new project in your preferred integrated development environment (IDE) and import the Selenium WebDriver library.

   4. Open the website: Use the WebDriver to open the e-commerce website in a browser.

   5. Automate user registration: Find the registration form elements using the WebDriver and fill them out with user information. Then submit the form to complete the registration process.

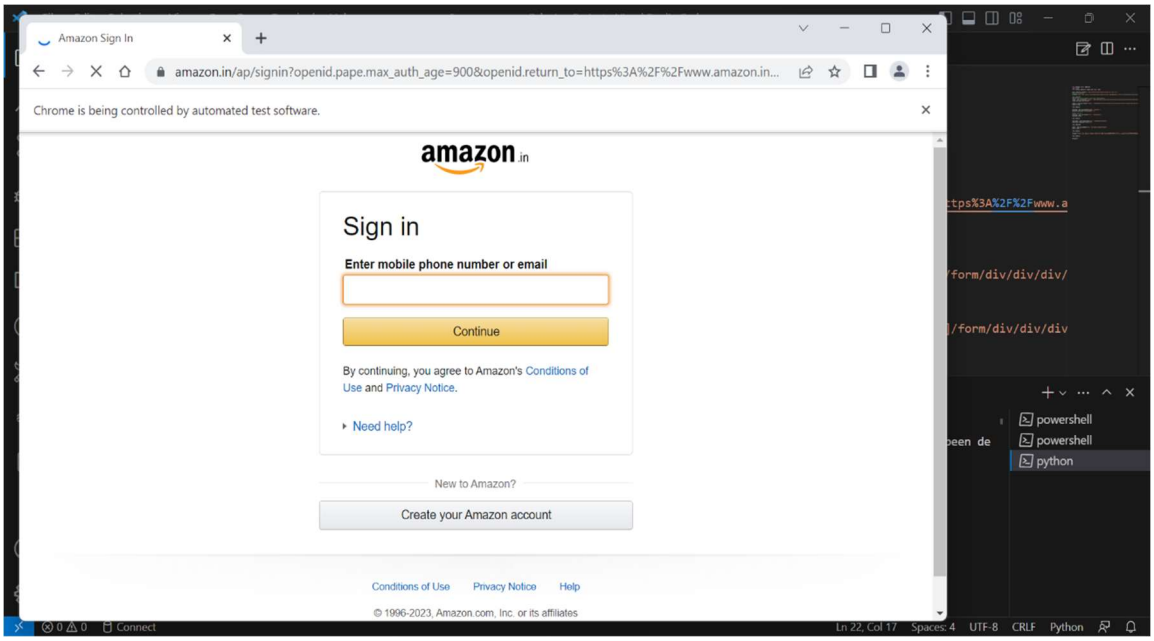
Here is an example code snippet for registering a new user:

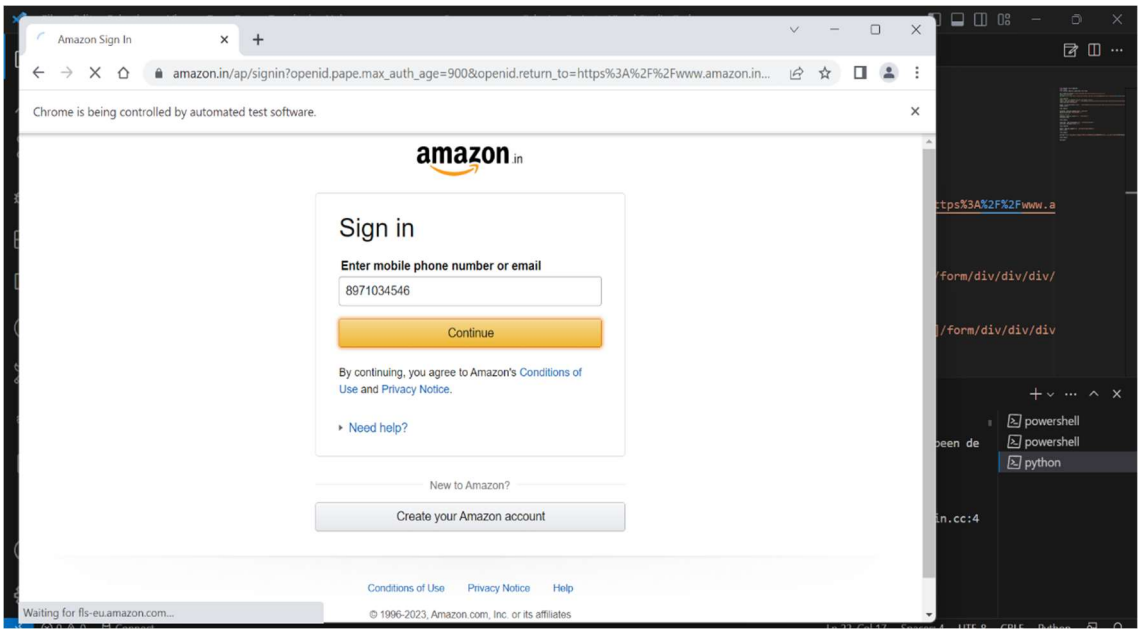


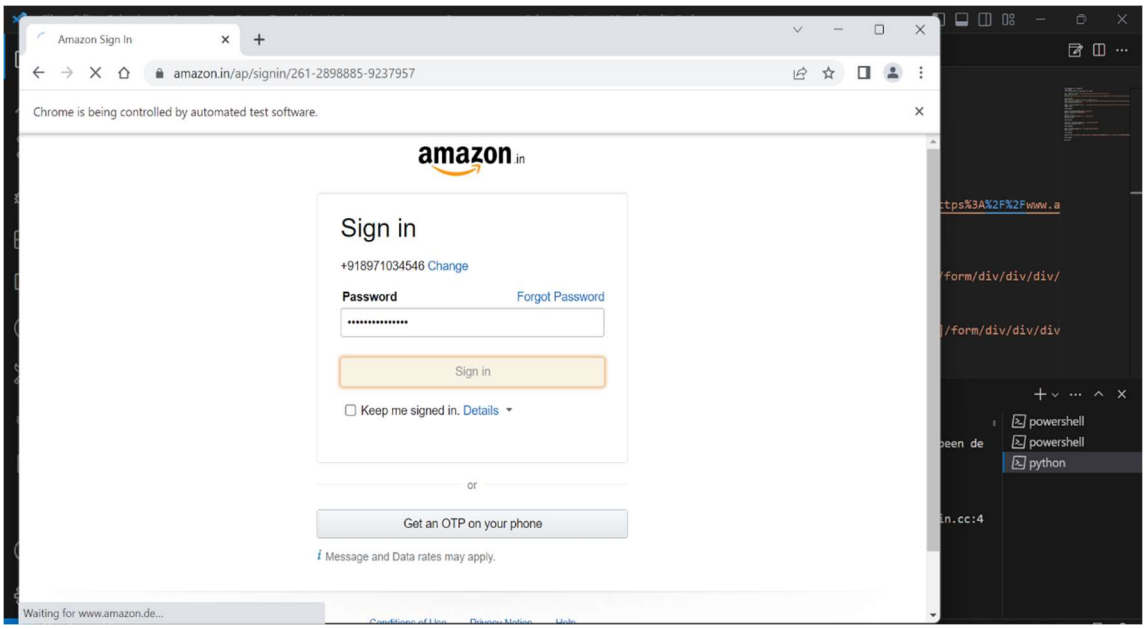


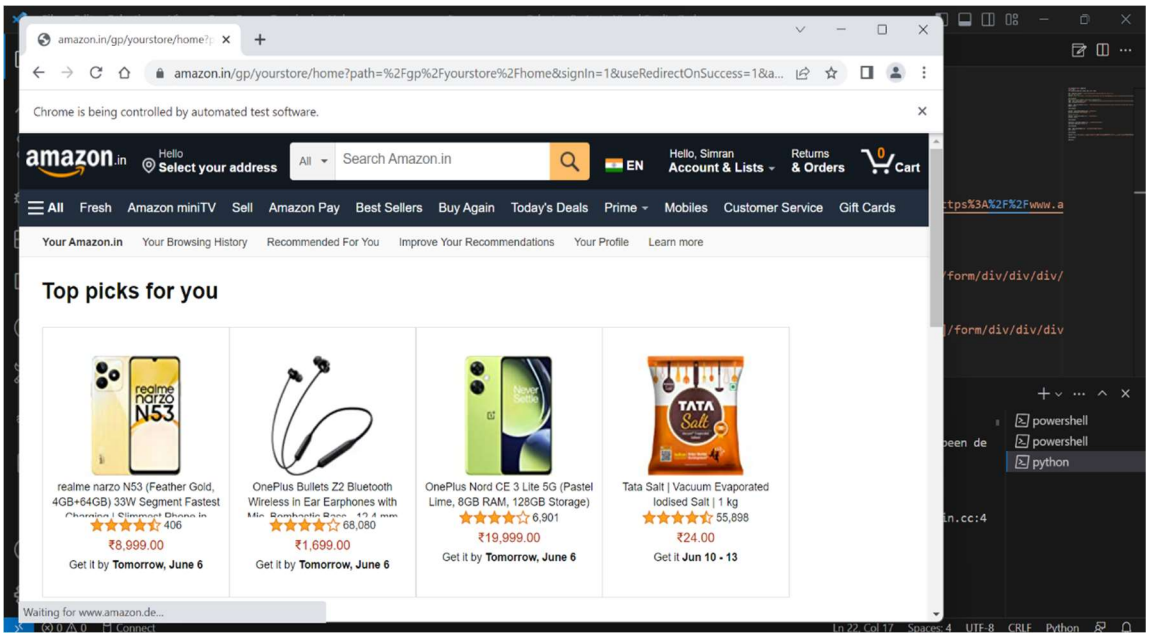
Output:

1. Registering on Amazon, automatically details are filled.




Amazon Sign In
amazon.in/ap/signin?openid.pape.max_auth_age=900&openid.return_to=https%3A%2F%2Fwww.amazon.in...
Chrome is being controlled by automated test software.
amazon.in
Sign in
Enter mobile phone number or email
8971034546
Continue
By continuing, you agree to Amazon's Conditions of Use and Privacy Notice.
Need help?
New to Amazon?
Create your Amazon account
Conditions of Use
Privacy Notice
Help
Waiting for fls-eu.amazon.com...
© 1996-2023, Amazon.com, Inc. or its affiliates
powershell
powershell
python


Amazon Sign In
amazon.in/ap/signin/261-2898885-9237957
Chrome is being controlled by automated test software.
amazon.in
Sign in
+918971034546 Change
Password
Forgot Password
Sign in
Keep me signed in. Details
or
Get an OTP on your phone
Message and Data rates may apply.
Waiting for www.amazon.de...
powershell
powershell
python

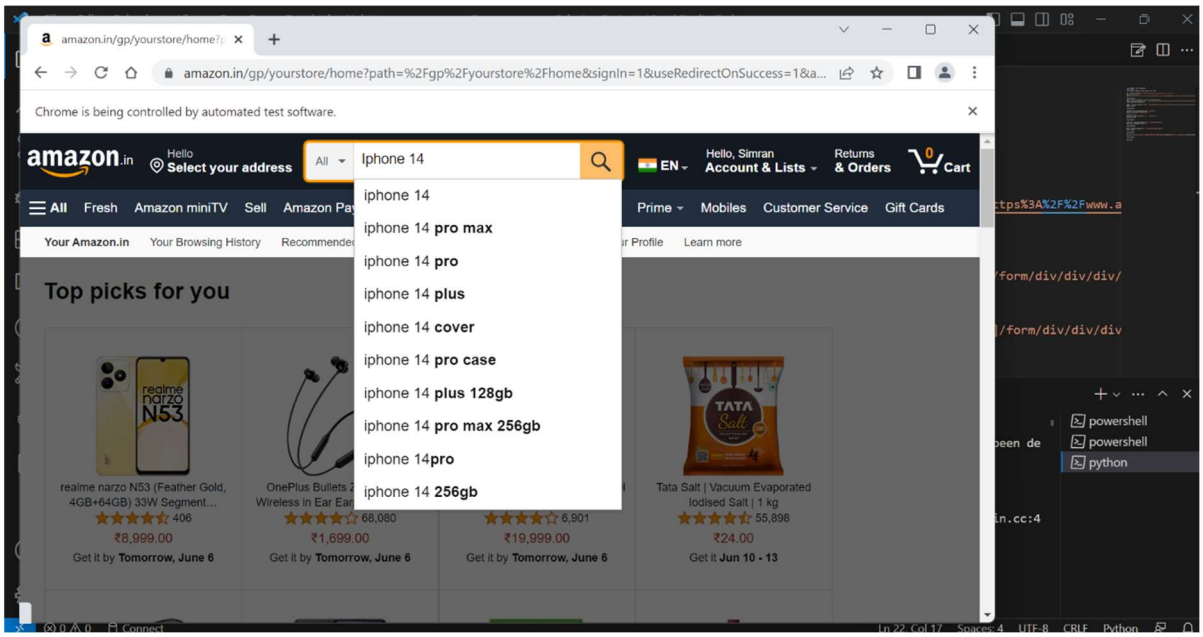


Automate product search: Find the search bar element using the WebDriver and enter the product name or keywords. Then, submit the search query to display the search results.

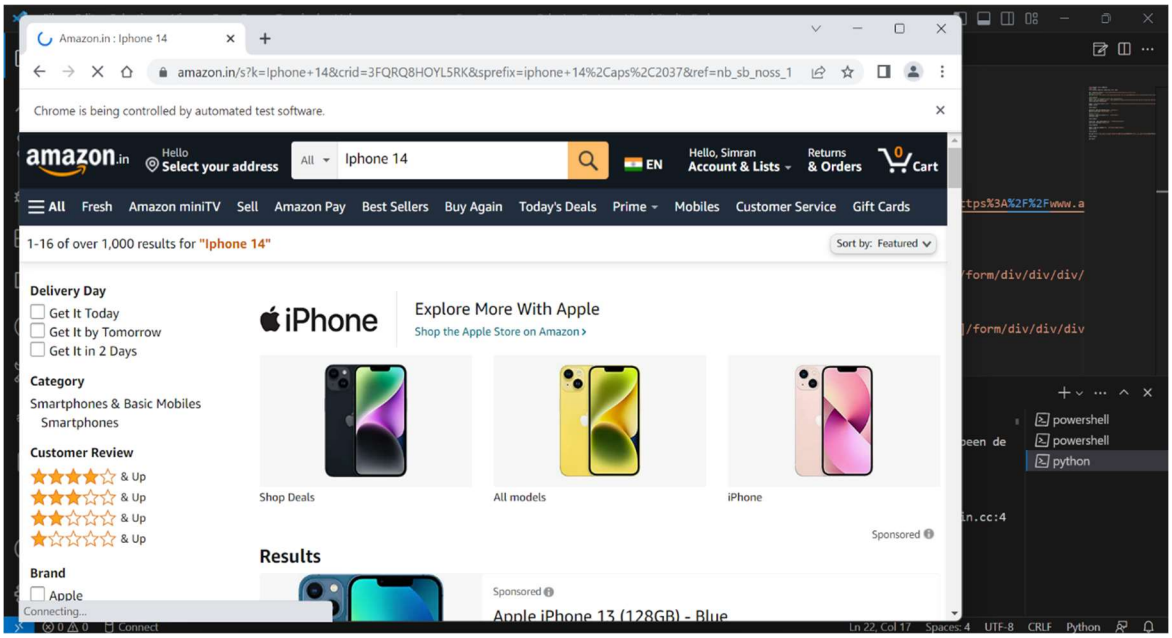
Here is a code snippet for searching a product

Output:

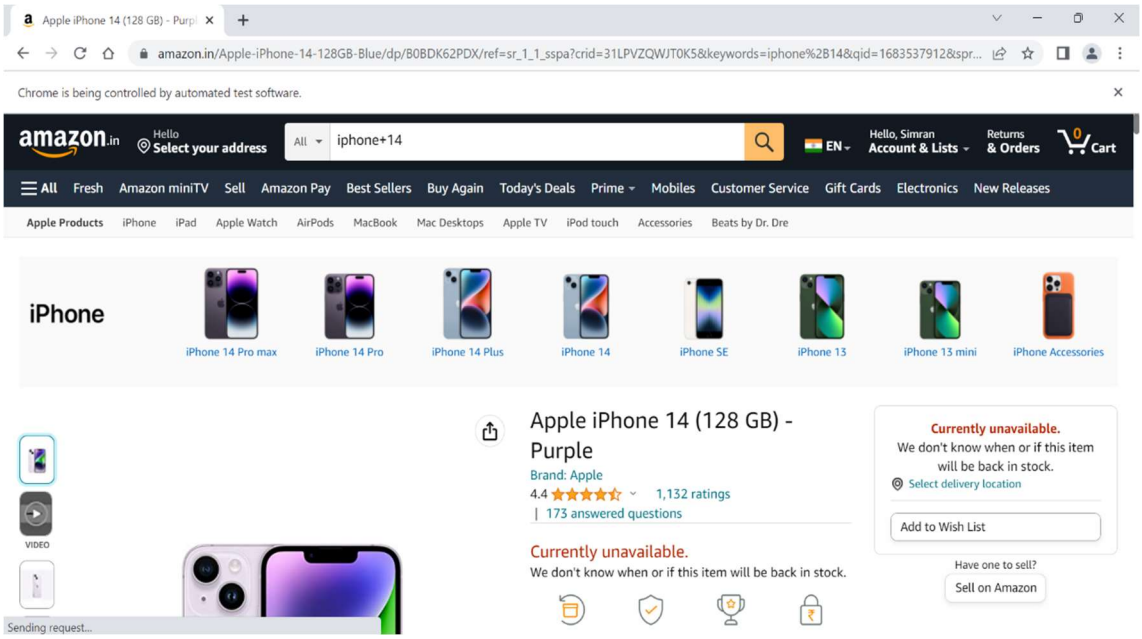
1. Automated Searching.



2. Automated Web-Results.



3. Product Features.



SonarQube(3)

Run a basic SonarQube scan and show dashboard with 3 types of issues ie Bug, Vulnerability and Code Smell

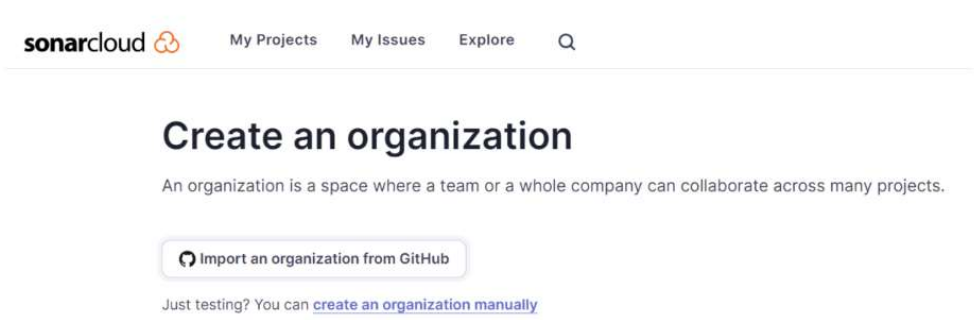
For this project, we use Sonarcloud and we connect it with our GitHub

organization and scan a existing repository. The following steps are

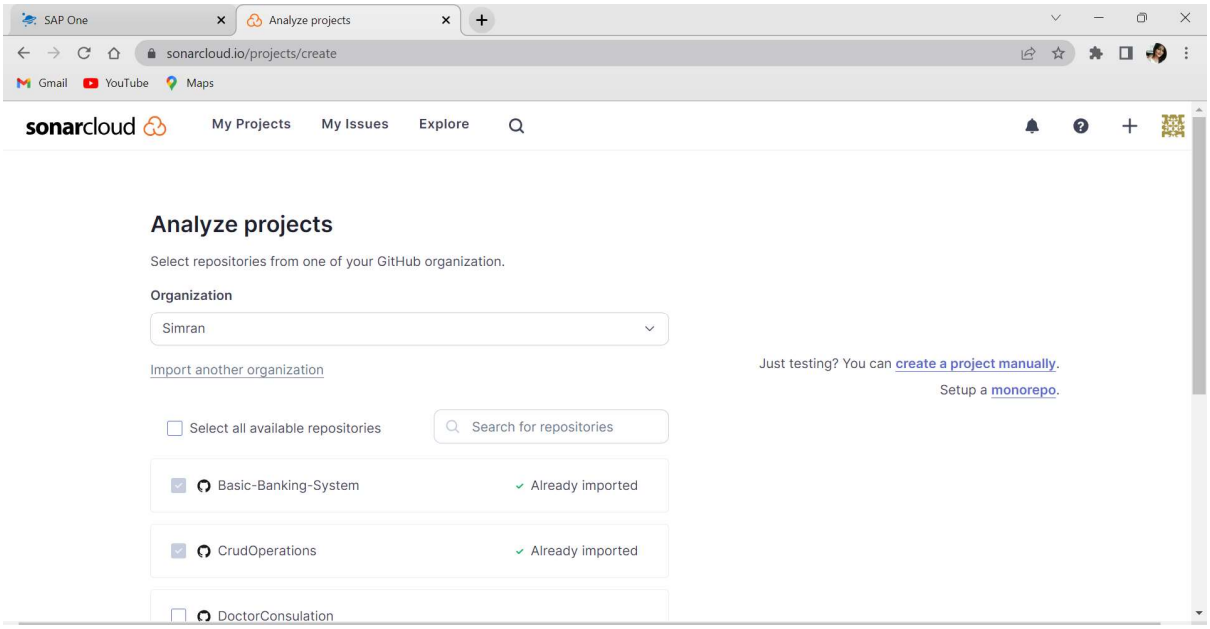
followed: -

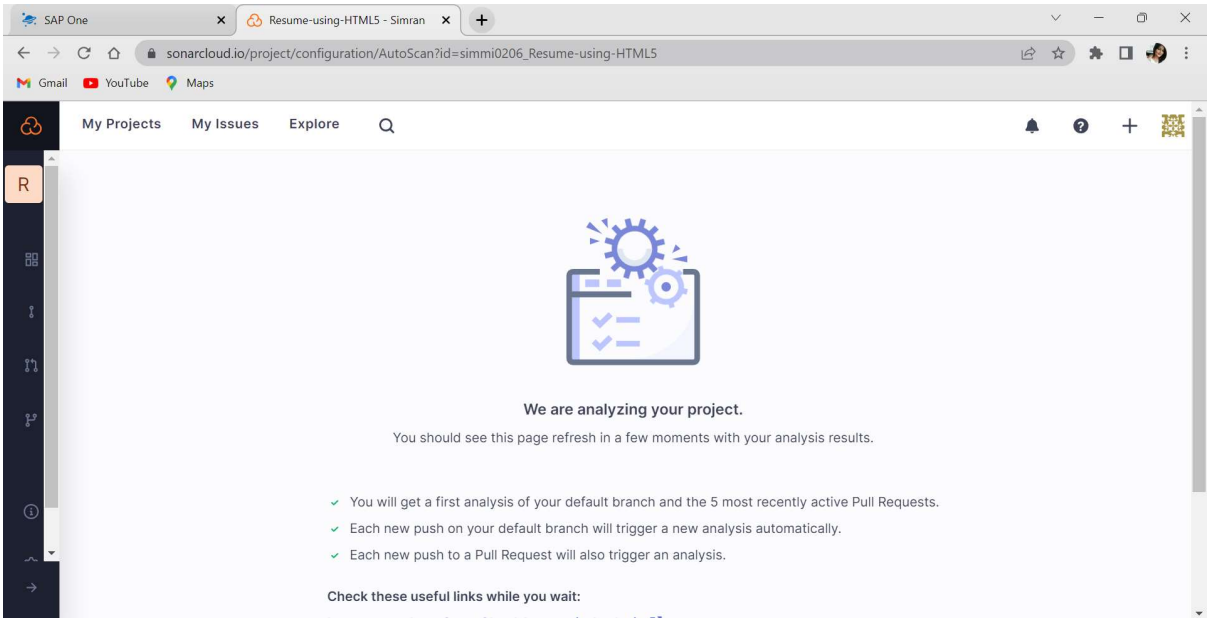
- Create a SonarCloud Account: We sign up for a SonarCloud account at https://sonarcloud.io using our GitHub credentials.

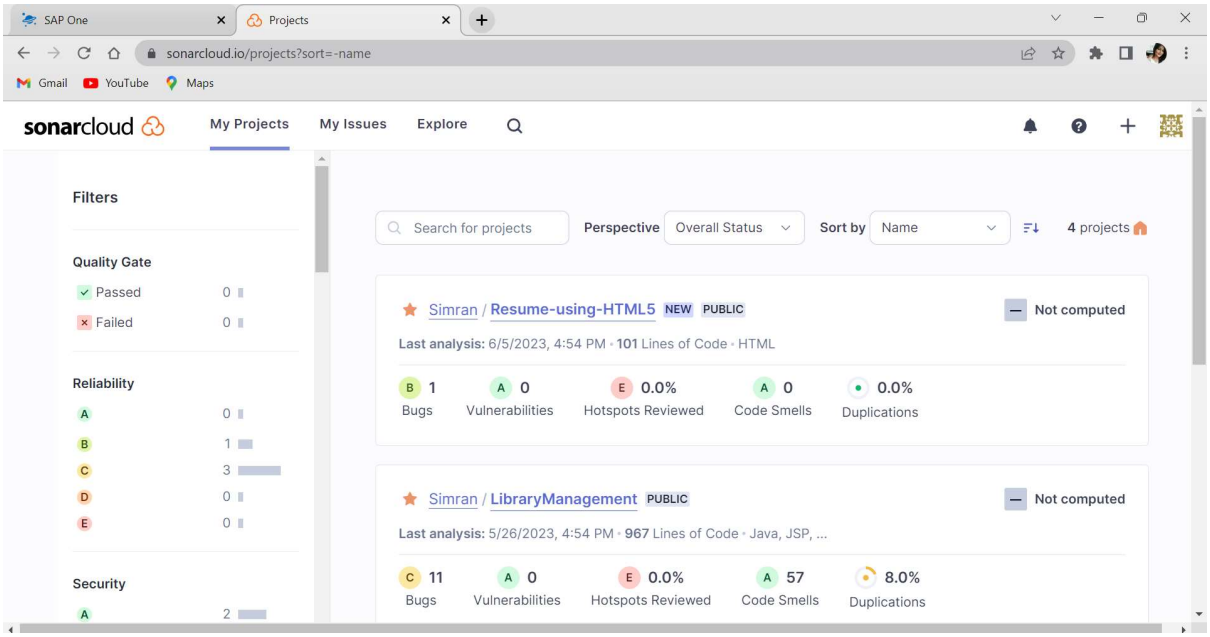
- Connect SonarCloud with GitHub Organization: In SonarCloud, we go to our account settings and select "Organizations" from the sidebar. We click on "Create Organization" and choose the option to connect with our GitHub organization. We follow the prompts to authorize SonarCloud to access our GitHub organization.

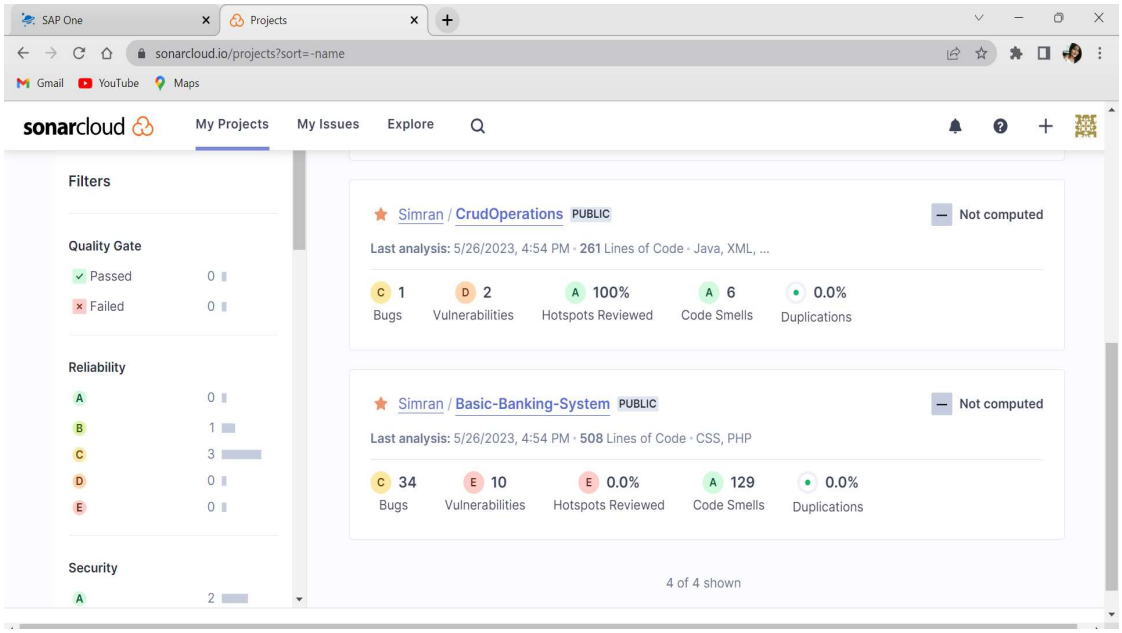


- Create a New Project in SonarCloud: After connecting with our GitHub organization, we create a new project in SonarCloud.
- Select the organization we connected with and we choose the repository that we want to analyse. Lasty, we configure any additional project settings as needed.

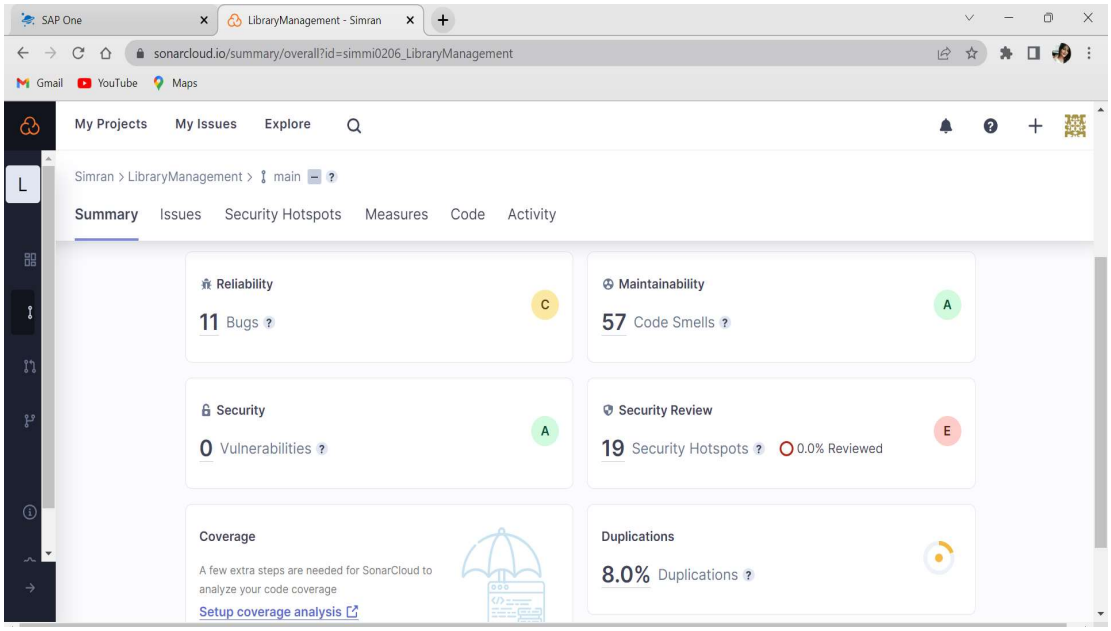

SAP One
Analyze projects
sonarcloud.io/projects/create
Gmail
YouTube
Maps
sonarcloud
My Projects
My Issues
Explore
Analyze projects
Select repositories from one of your GitHub organization.
Organization
Simran
Import another organization
Just testing? You can create a project manually.
Setup a monorepo.
Select all available repositories
Search for repositories
Basic-Banking-System
Already imported
CrudOperations
Already imported
DoctorConsulation

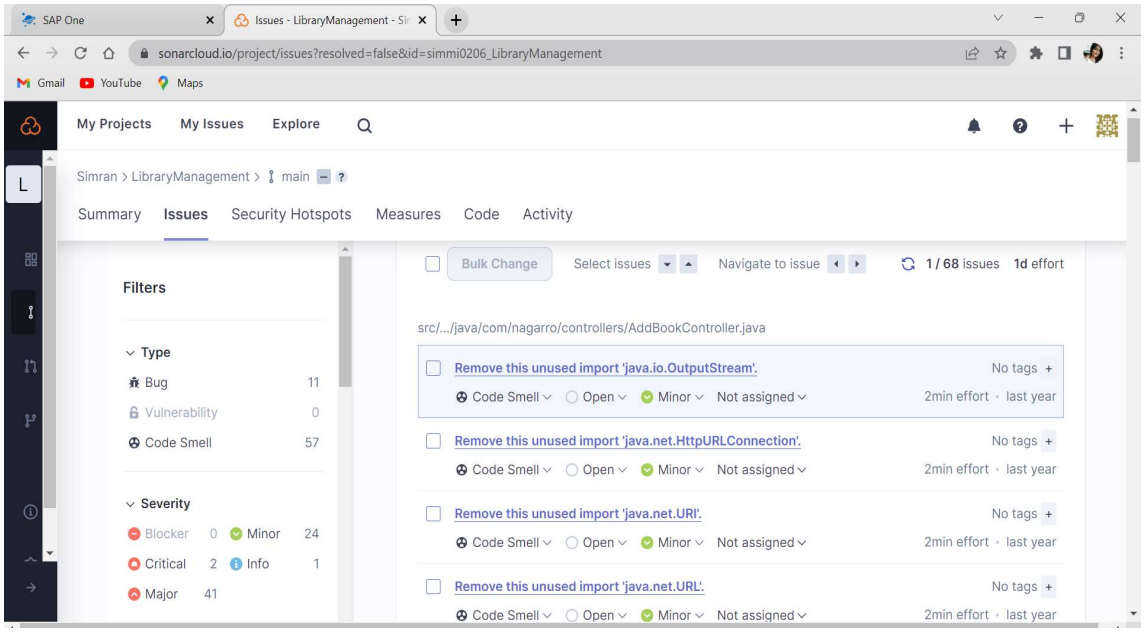

SAP One
Resume-using-HTML5 - Simran
sonarcloud.io/project/configuration/AutoScan?id=simmi0206_Resume-using-HTML5
Gmail
YouTube
Maps
My Projects
My Issues
Explore
R
We are analyzing your project.
You should see this page refresh in a few moments with your analysis results.
You will get a first analysis of your default branch and the 5 most recently active Pull Requests.
Each new push on your default branch will trigger a new analysis automatically.
Each new push to a Pull Request will also trigger an analysis.
Check these useful links while you wait:





We can view the detailed analysis of our projects by clicking on them.





- It shows the part/line where vulnerabilities/bugs are present.
- This helps us to maintain high code quality, enhance security, and improve maintainability, ultimately leading to a more reliable and efficient software development process.

Kubernetes (3)

1. **Take a Kubernetes image and deploy with high availability, fault tolerance, scalability provided by Kubernetes.**

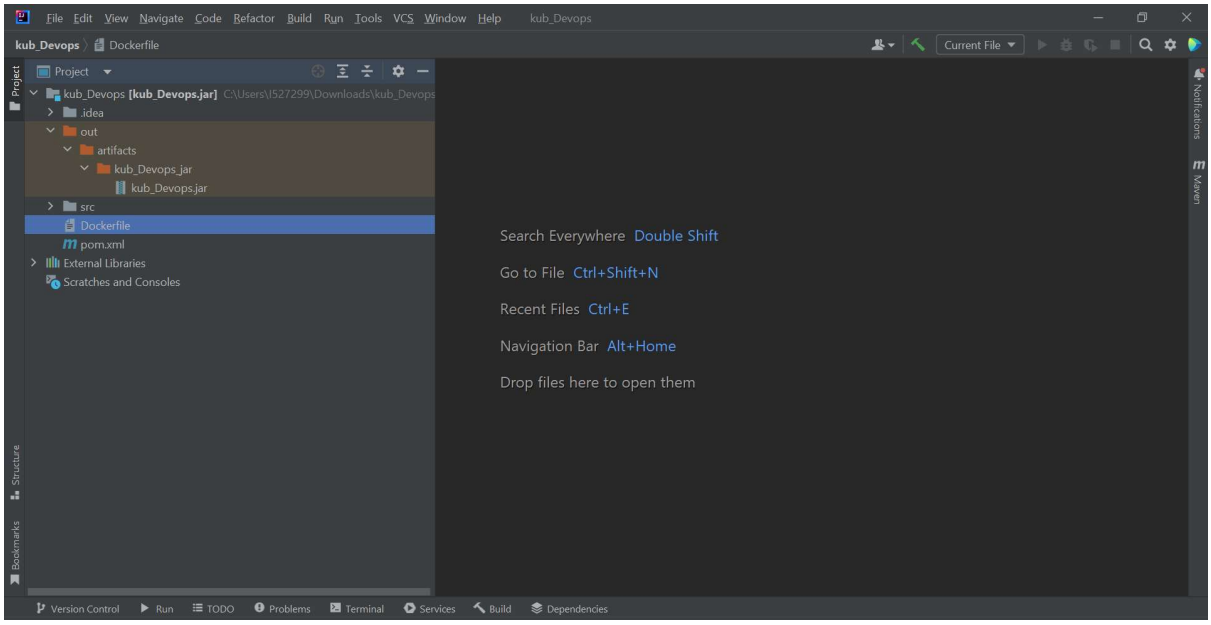
   Pre-requisites

   1. Docker should be installed on the system and should be in running state

   2. Docker Hub free account is created

   3. Git Bash is installed to run the docker commands To create a Kubernetes image, you'll first need to create a Docker image since Kubernetes works with container images.

   Step 1: Create Dockerfile in application root directory.



   Step 2: Create docker image (make sure your docker desktop is running and in terminal you are in application directory)

   In cmd : docker build -t dockerimage .

   Step 3: Login to docker hub (make sure you have docker hub account)

   In cmd : docker login

   Step 4: Give tag to your image as username/repository name

   In cmd : Docker tag dockerimage simran111111/repodocker

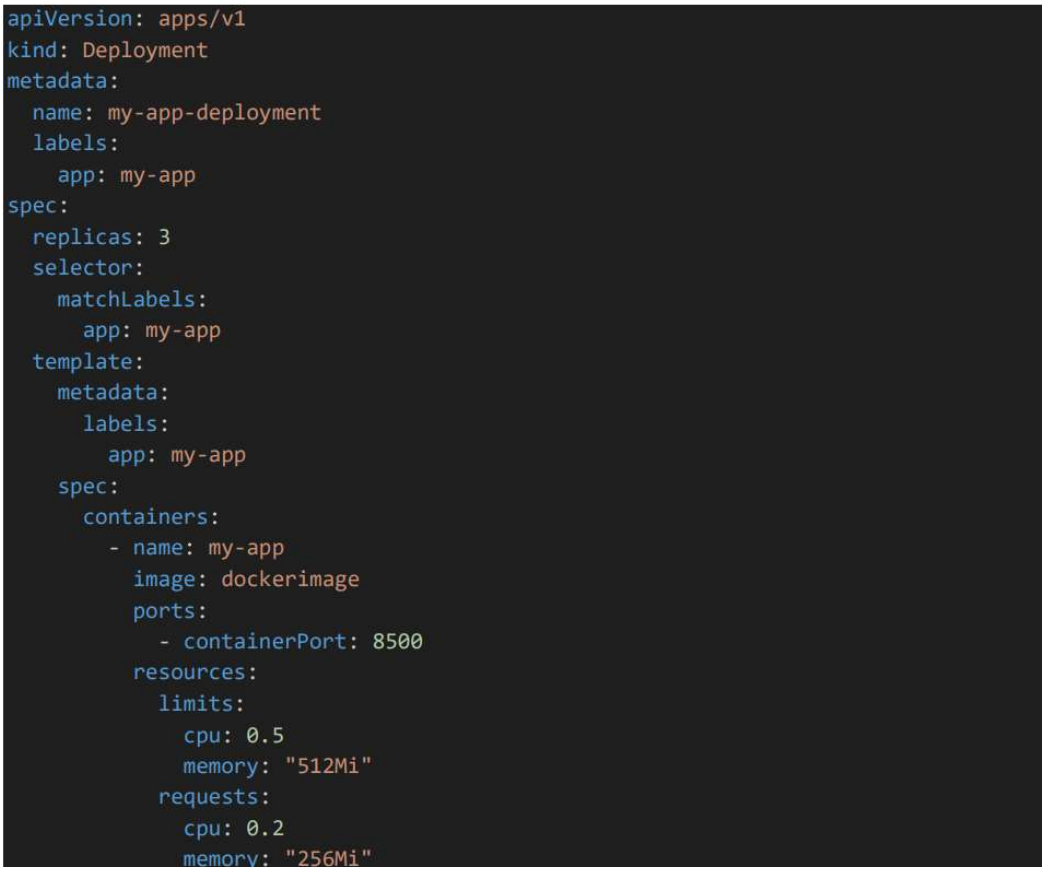
Step 5: Push docker image to docker hub

In cmd: Docker push simran111111/repodocke

Step 6: Install kubcetl with link: https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/

Step 7: Add path to environment variable.

Step 8: Create a Kubernetes deployment manifest file, that describe application's state. configure the file with image details and options to enable high availability, fault tolerance, scalability



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-app
          image: dockerimage
          ports:
            - containerPort: 8500
          resources:
            limits:
              cpu: 0.5
              memory: "512Mi"
            requests:
              cpu: 0.2
              memory: "256Mi"
```

Step 9: Create Kubernetes cluster: Enable Kubernetes in docker desktop

Step 10: Deploy the image to created cluster

In cmd: kubectl apply -f demo.yaml

Step 11: Get deployments:

cmd: kubectl get deployments

Step 12. Get pods:

cmd: kubectl get pods

Step 13. Get info :

cmd: kubectl cluster-inf