**THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU**

**(An Autonomous Institute under VTU, Belagavi)**



In fulfilment of **Skill Development Program (DevOps**),

5th semester

Bachelor of Engineering

in

Computer Science and Engineering

# **“STUDENT RECORD MANAGEMENT SYSTEM”**

**Submitted by:**

Harshitha M

4NI22CS074

8088768988

2022cs\_harshitham\_b@nie.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# THE NATIONAL INSTITUTE OF ENGINEERING

Mysore-570 008

2024-2025

## DevOps

**DevOps** is a set of practices, principles, and cultural philosophies that aim to improve collaboration between software development (Dev) and IT operations (Ops) teams. The primary goal of DevOps is to shorten the development lifecycle, enhance the quality of software, and continuously deliver applications with greater speed and efficiency.

In traditional software development, developers and operations teams often work in silos, leading to delays, miscommunication, and inefficiencies in deploying and maintaining applications. DevOps seeks to break down these silos by encouraging cross-functional collaboration, automating repetitive tasks, and creating a continuous feedback loop that allows teams to rapidly adapt and improve.

**Key Concepts of DevOps:**

1. **Collaboration**: DevOps fosters close collaboration between development, operations, quality assurance, and other stakeholders to align goals and responsibilities throughout the software lifecycle.
2. **Automation**: Automating repetitive tasks such as testing, deployment, and infrastructure management helps to eliminate human error, speed up processes, and improve consistency. Common automation tools include Jenkins, Ansible, and Docker.
3. **Continuous Integration (CI)**: Developers frequently commit code to a shared repository, where automated tests are run to detect bugs early and ensure that changes integrate seamlessly with the rest of the system. This helps to avoid integration issues at later stages.
4. **Continuous Delivery (CD)**: In addition to CI, Continuous Delivery ensures that code changes are automatically deployed to production or staging environments after passing automated tests. This enables frequent, reliable releases with minimal downtime.
5. **Monitoring and Feedback**: DevOps emphasizes the use of monitoring tools (such as Prometheus, Grafana, or ELK Stack) to continuously track the performance and health of applications in production. Feedback loops allow teams to address issues proactively and iterate quickly based on user feedback or system metrics.
6. **Infrastructure as Code (IaC)**: Infrastructure provisioning and management are handled through code, rather than manual configuration, which ensures consistency, scalability, and easier recovery.

Popular tools include Terraform, AWS CloudFormation, and Kubernetes.

**Benefits of DevOps:**

* **Faster Time to Market**: By automating manual tasks and promoting continuous integration and delivery, DevOps allows teams to release new features and fixes faster.
* **Improved Collaboration and Communication**: Breaking down silos between development and operations fosters a culture of collaboration and shared responsibility.
* **Higher Quality**: Automated testing and continuous monitoring help detect issues early, ensuring higher software quality and fewer defects in production.
* **Scalability and Flexibility**: DevOps practices like Infrastructure as Code allow teams to quickly scale and adapt infrastructure to changing needs, without the need for manual intervention.
* **Cost Efficiency**: Automation reduces operational overhead and manual errors, while continuous feedback and iterative improvements reduce the costs of fixing defects in later stages.

**DevOps Tools:**

A variety of tools are used to implement DevOps practices, including:

* **Version Control**: Git, GitHub, Bitbucket
* **Continuous Integration/Continuous Deployment (CI/CD)**: Jenkins, GitLab CI, CircleCI  **Configuration Management**: Ansible, Chef, Puppet
* **Containerization**: Docker, Kubernetes
* **Monitoring**: Prometheus, Nagios, ELK Stack, New Relic  **Collaboration**: Slack, Microsoft Teams, Jira, Confluence

## Jenkins

Jenkins is an open-source automation server used to build, test, and deploy software continuously. It enables developers to automate parts of the software development process, like integration and deployment, through customizable pipelines. Jenkins supports numerous plugins, allowing integration with various tools for version control, testing, and deployment. It is widely used in CI/CD (Continuous Integration and Continuous Deployment) to improve development efficiency and ensure code quality.

## Docker

Docker is an open-source platform that enables developers to automate the deployment of applications in lightweight, portable containers. These containers package an application with its dependencies and configuration, ensuring it runs consistently across different environments, whether on a developer’s machine, in testing, or in production. By isolating applications from the underlying infrastructure, Docker simplifies the process of building, shipping, and running software and helps avoid issues like "it works on my machine." Docker containers are fast, resource-efficient, and ideal for modern microservices and CI/CD workflows.

## Integrating CI-CD

Integrating CI/CD (Continuous Integration and Continuous Deployment) involves setting up automated processes to build, test, and deploy your code every time there’s a change. Here’s a general guide to setting up a CI/CD pipeline using tools like Jenkins or GitHub Actions with Docker and Git:

1. Setup Version Control System
   * Use Git as your version control system, hosting your repository on GitHub, GitLab, Bitbucket, or another platform.
   * CI/CD tools will monitor this repository for code changes (like new commits or pull requests).
2. Choose a CI/CD Tool
   * Select a CI/CD tool that fits your project needs. Popular options include:
   * Jenkins: Highly customizable, suitable for complex workflows.
   * GitHub Actions: Integrated with GitHub, great for open-source and GitHub-hosted projects.
   * GitLab CI/CD: Integrated with GitLab repositories, provides CI/CD out of the box.
   * CircleCI, Travis CI, and Bitbucket Pipelines are other popular choices.
3. Define Pipeline Stages
   * A typical CI/CD pipeline has the following stages:
   * Build: Compiles the application and creates any necessary packages or Docker images.
   * Test: Runs automated tests to verify code quality and prevent bugs.
   * Deploy: Deploys the application to staging or production environments.

1. Create a Configuration File for CI/CD
   * Each CI/CD tool uses a specific configuration file to define pipeline stages:
   * Jenkins: Jenkinsfile (written in Groovy)
   * GitHub Actions: .github/workflows/main.yml (YAML format)
   * GitLab CI/CD: .gitlab-ci.yml (YAML format)
2. Automate Tests
   * Write automated tests for your application (e.g., unit tests, integration tests).
   * In the Test stage of your pipeline, configure the CI/CD tool to run these tests every time new code is pushed.
3. Build and Deploy Docker Containers (Optional)
   * If your application is containerized, build a Docker image and push it to a container registry (like Docker Hub or Amazon ECR).
   * Pull this image during the deployment stage and run it in a staging or production environment. 7. Set Up Notifications
   * Configure notifications (e.g., Slack, email) to alert the team if a build or deployment fails.

8. Monitor and Refine

 Monitor your CI/CD pipeline performance, refine it over time, and ensure it meets your team’s needs.

## Project Overview

The Student Management System is a web-based application designed to manage student data. It enables the user (such as an administrator or teacher) to perform essential operations like adding, viewing, updating, and

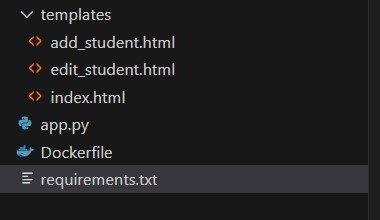
deleting student records. Built with the Flask web framework and SQLAlchemy ORM for database interactions, this application demonstrates the core concepts of CRUD operations and provides a simple interface for data management.

**Objectives:**

* To implement a web application that can manage student records.
* To allow administrators to easily add, view, edit, and delete student information.
* To provide a user-friendly interface that simplifies interaction with the database.
* To create a project that demonstrates the use of Flask, SQLAlchemy, HTML templates, and CRUD operations.

**Tools and Technologies:**

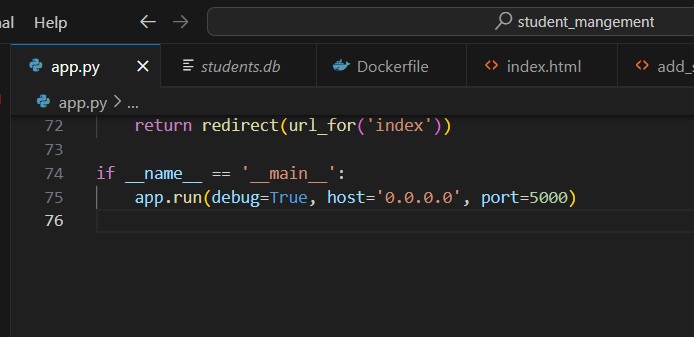
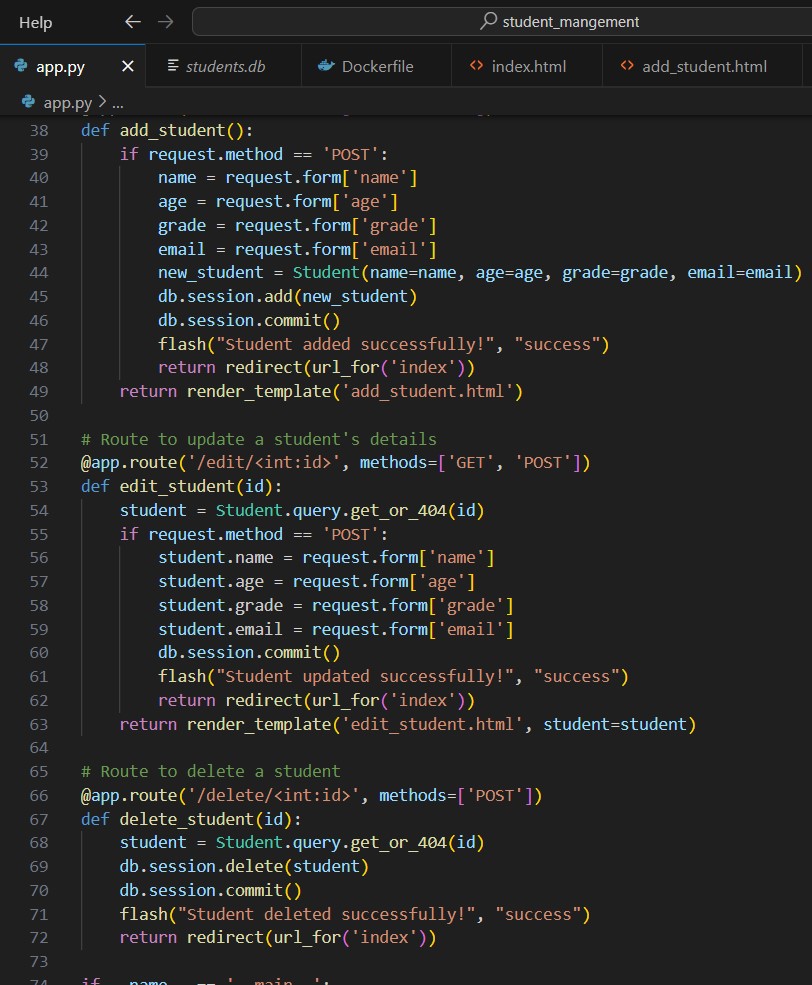
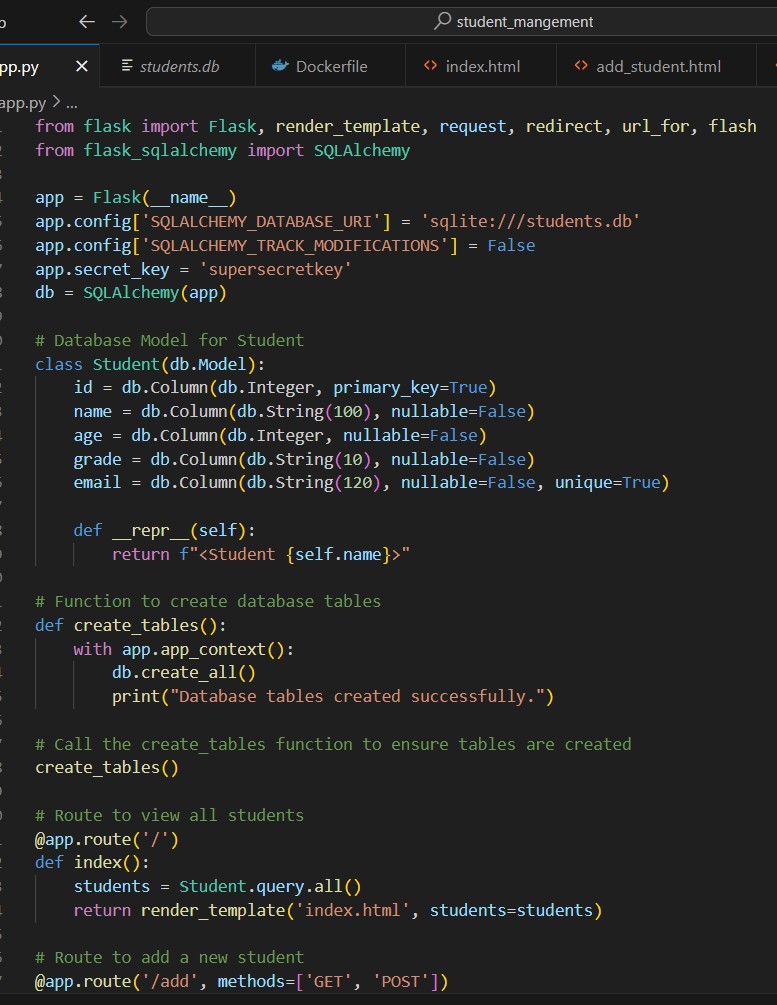
* Programming Language: Python
* Framework: Flask (a lightweight Python web framework)
* Database: SQLite (or PostgreSQL/MySQL for production environments)
* ORM (Object Relational Mapper): Flask-SQLAlchemy (for managing database interactions)
* Frontend: HTML, CSS, and Bootstrap for styling the web pages
* Development Environment: Visual Studio Code (VS Code), Docker  **File structure**



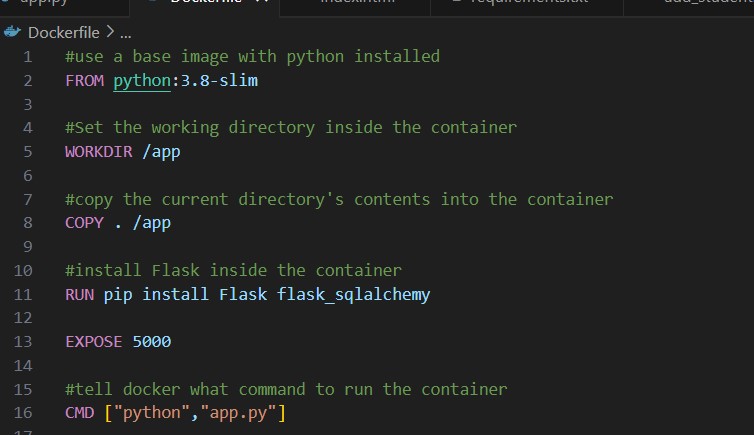
**Integrating CI-CD**

**Step 1**: Create the above files in vs code.

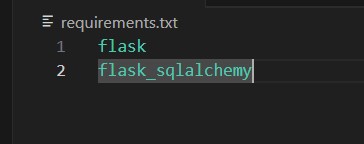
* app.py



* Dockerfile

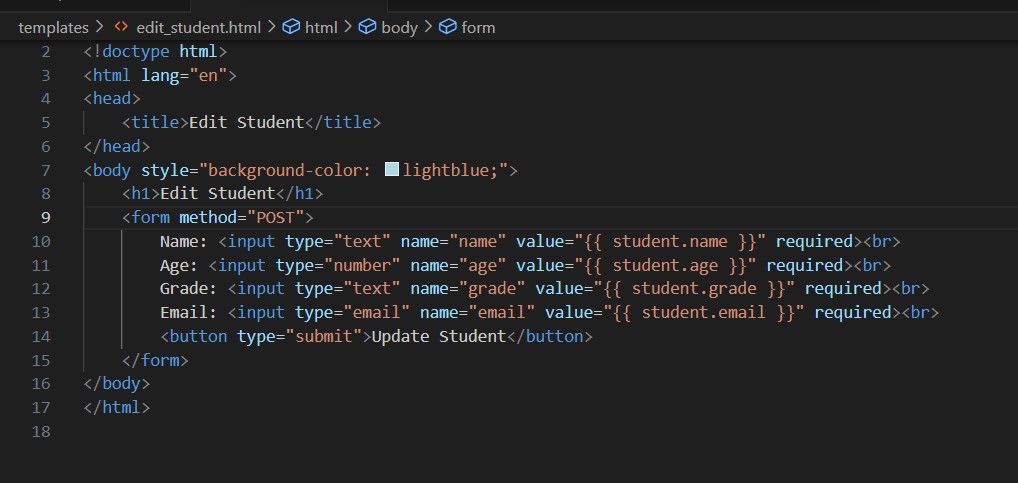


* requirements.txt

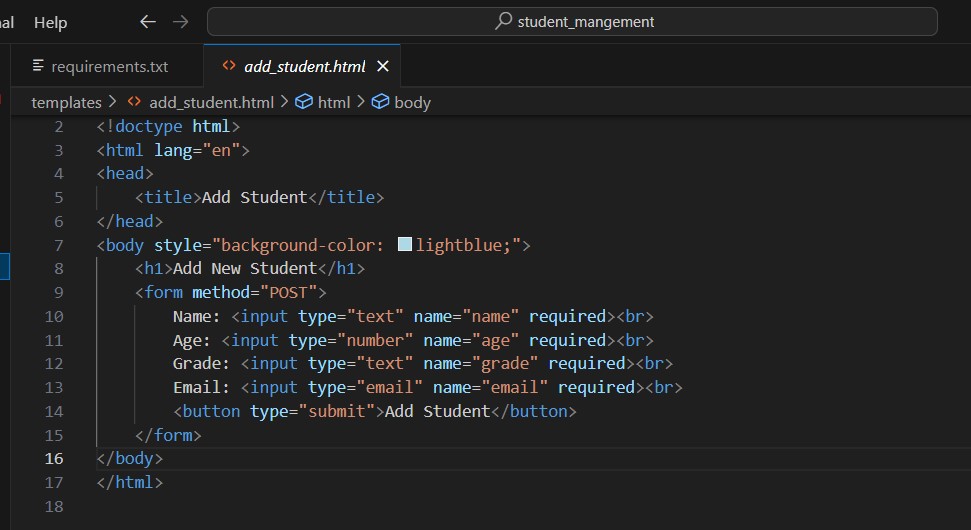


* Inside the folder templates:

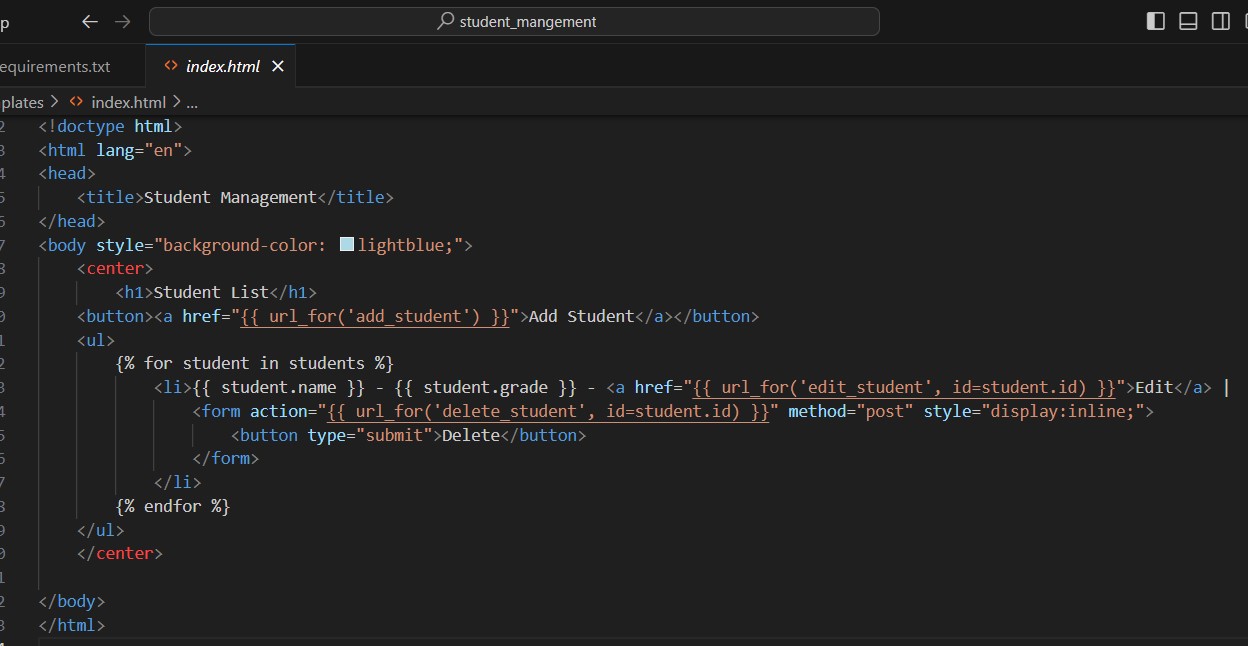
* 1. edit\_students.html



* 1. add\_students.html



* 1. index.html



**Step 2:** Create virtual environment using these commands.

* Python -m venv.venv
* .venv\Scripts\activate

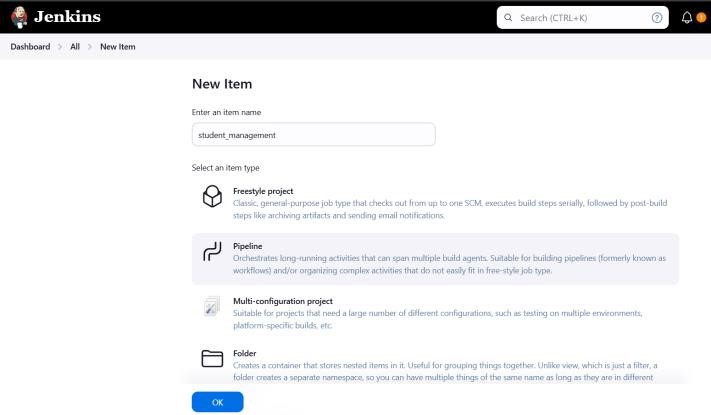
**Step 3:** Install flask by using *pip install flask* command .

**Step 4:** Save the files and run the code using *python app.py* command.

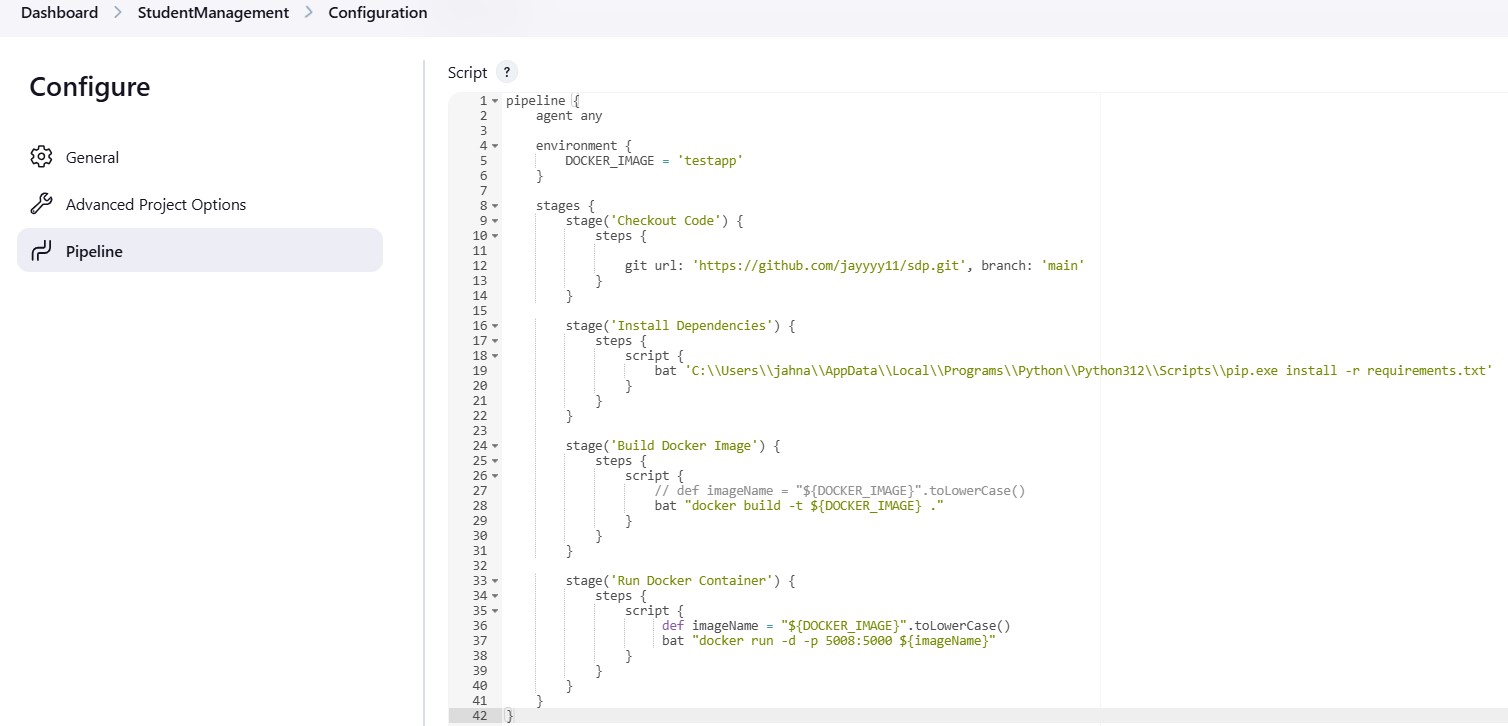
**Step 5:** Build docker image by using *docker build -t testapp .* command.

**Step 6:** Open Jenkins and login to your account

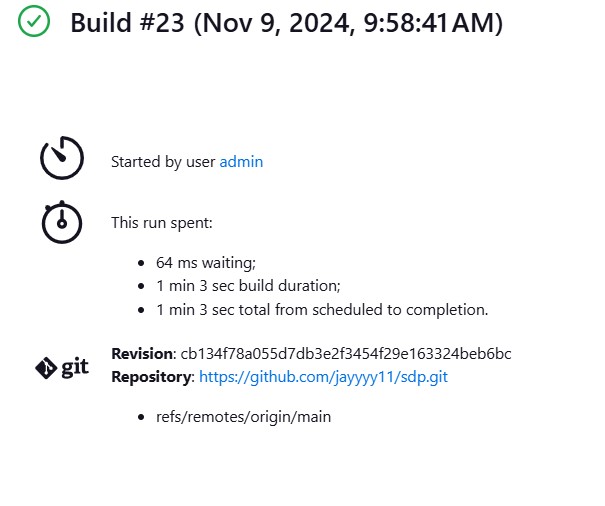
* Create a new job and select pipeline



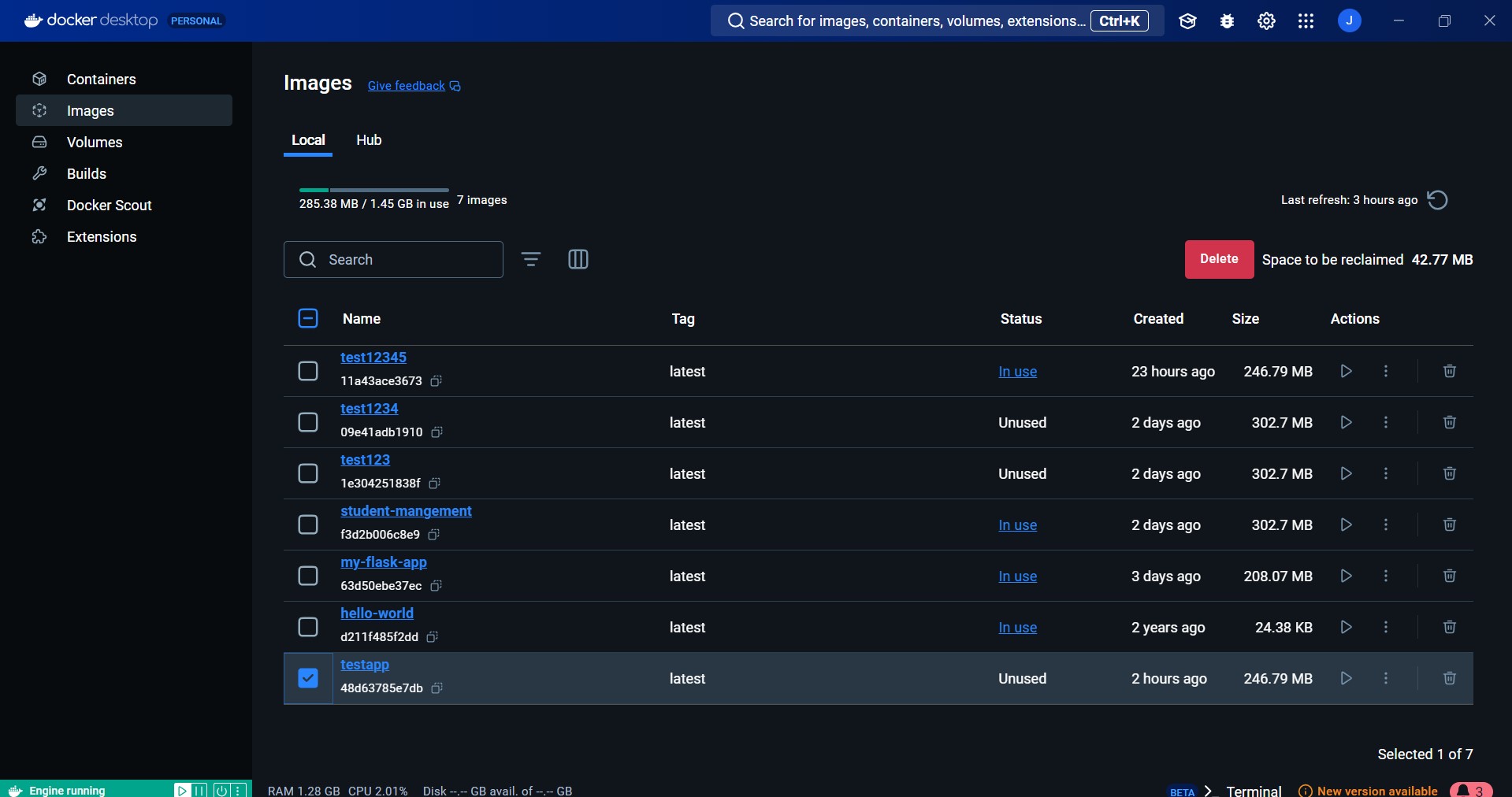
* Go to script and type this code and click on save



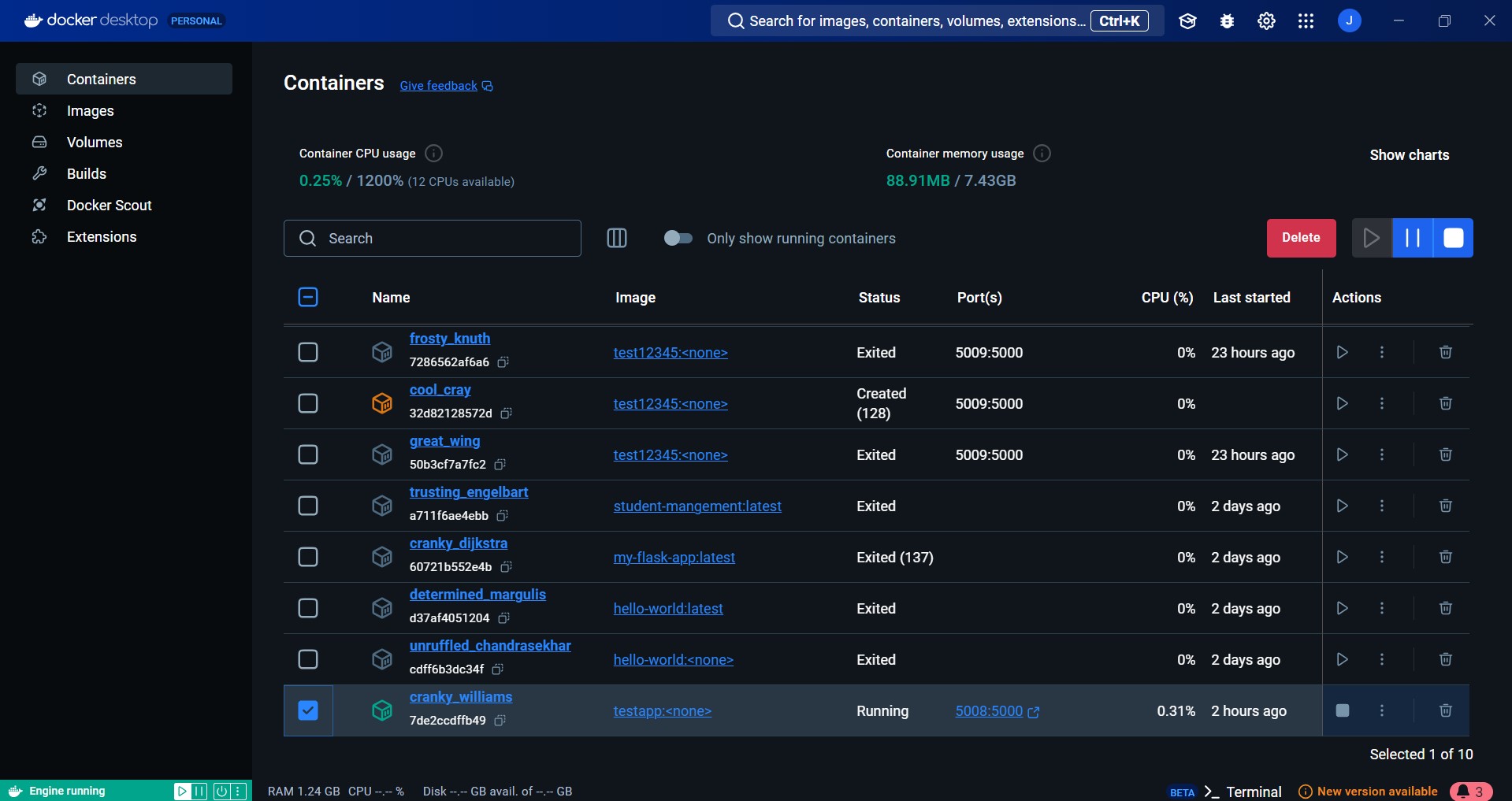
* Click on build now



**Step 7:** Open docker and check if container of image of testapp is created.



**Step 8**: Run the container.



**Step 9:** Click on the link to open website

