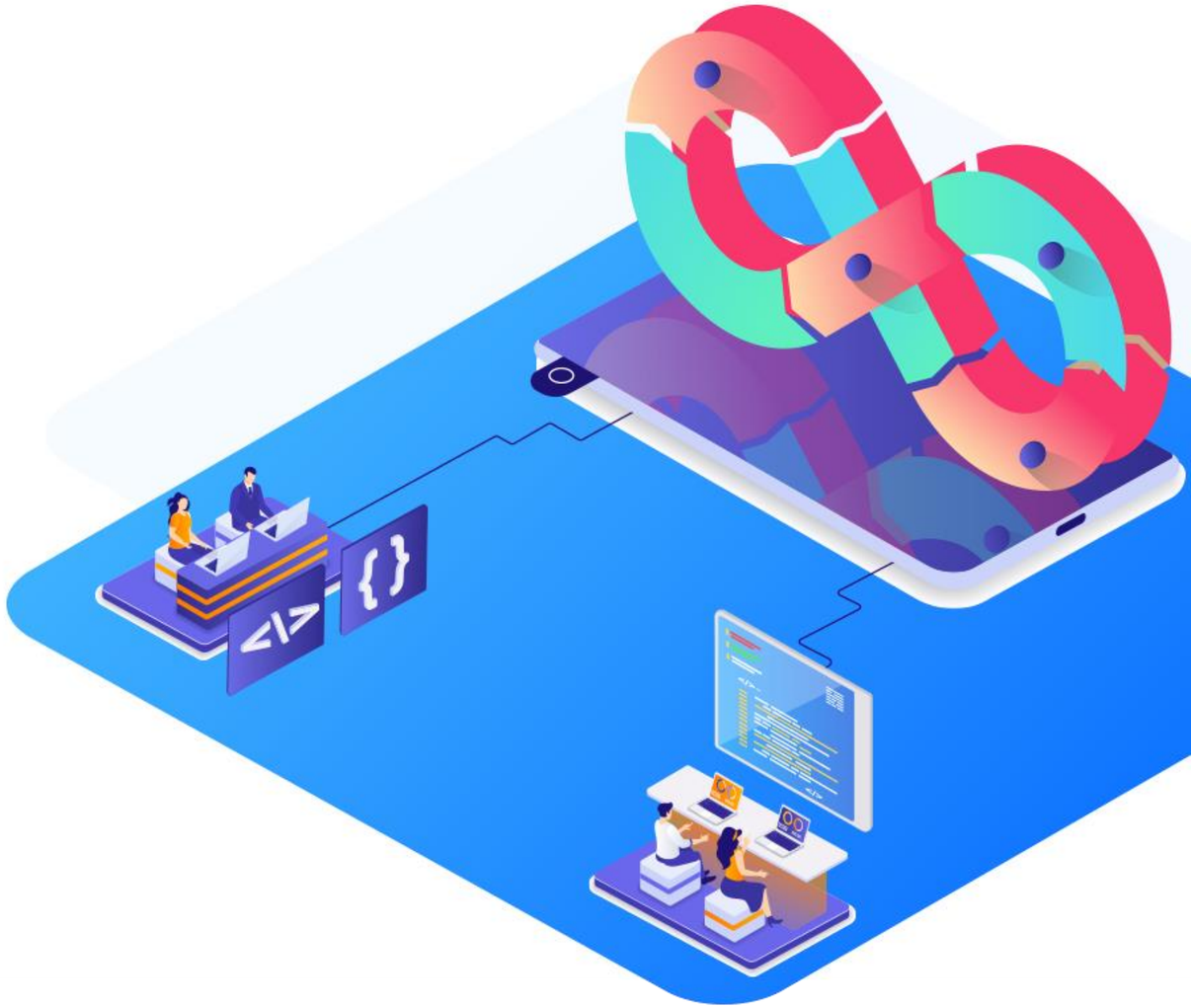
DevOps Foundations: Version Control and CI/CD with Jenkins



Course-End Project

Insured Assurance

Objective

To create a GitHub Actions CI/CD pipeline workflow for invoking the deployment of a Java application as a Jenkins job using Tomcat Apache

COURSE-END PROJECT

Problem Statement and Motivation

Real-time scenario:

Insured Assurance, a leading global insurance provider based in the US, offers a range of products including home, health, car, and life insurance.

The company is transitioning to a DevOps architecture and aims to automate code builds and deployments across various environments. To meet this need, it has adopted GitHub Actions for code checkout, building, and testing automation and Jenkins for continuous deployment.

As a DevOps engineer at Insured Assurance, you are tasked with implementing a CI/CD pipeline using GitHub Actions and Jenkins.

Industry Relevance

The following tools used in this project serve specific purposes within the industry:

1. **Jenkins:** It is an open-source automation server commonly used in the software development industry for building, testing, and deploying projects. It supports continuous integration and continuous delivery (CI/CD) practices, enabling teams to increase efficiency, detect issues early, and accelerate software releases.

2. **GitHub Actions:** It is a CI/CD platform that streamlines the code integration and deployment processes by automating the software development process directly within GitHub. This makes it invaluable for developers to manage and deploy code updates efficiently.

3. **Tomcat Apache:** It is an open-source web server and servlet container that supports Java servlets and JSPs, making it a staple in web application development and deployment for organizations relying on Java technologies.

Tasks

The following tasks outline the process of implementing CI/CD using GitHub Actions and Jenkins:

1. Create a code repository on GitHub
2. Create a GitHub Actions pipeline to perform continuous integration
3. Configure Tomcat Apache for automated code deployment
4. Integrate the GitHub Actions pipeline to invoke the Jenkins pipeline
5. Invoke pipeline to validate automated deployment

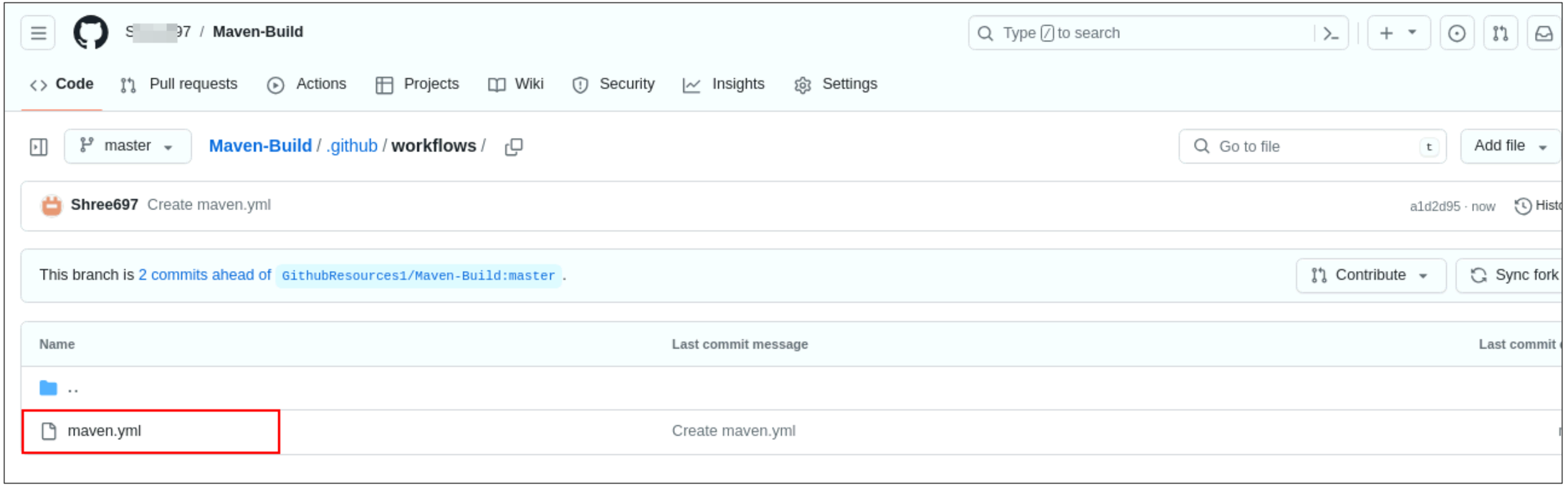
Project References

- **Task 1:** Lesson 02
- **Task 2:** Lesson 10
- **Task 3:** Lesson 07
- **Task 4:** Lesson 10
- **Task 5:** Lessons 08 and 10

COURSE-END PROJECT

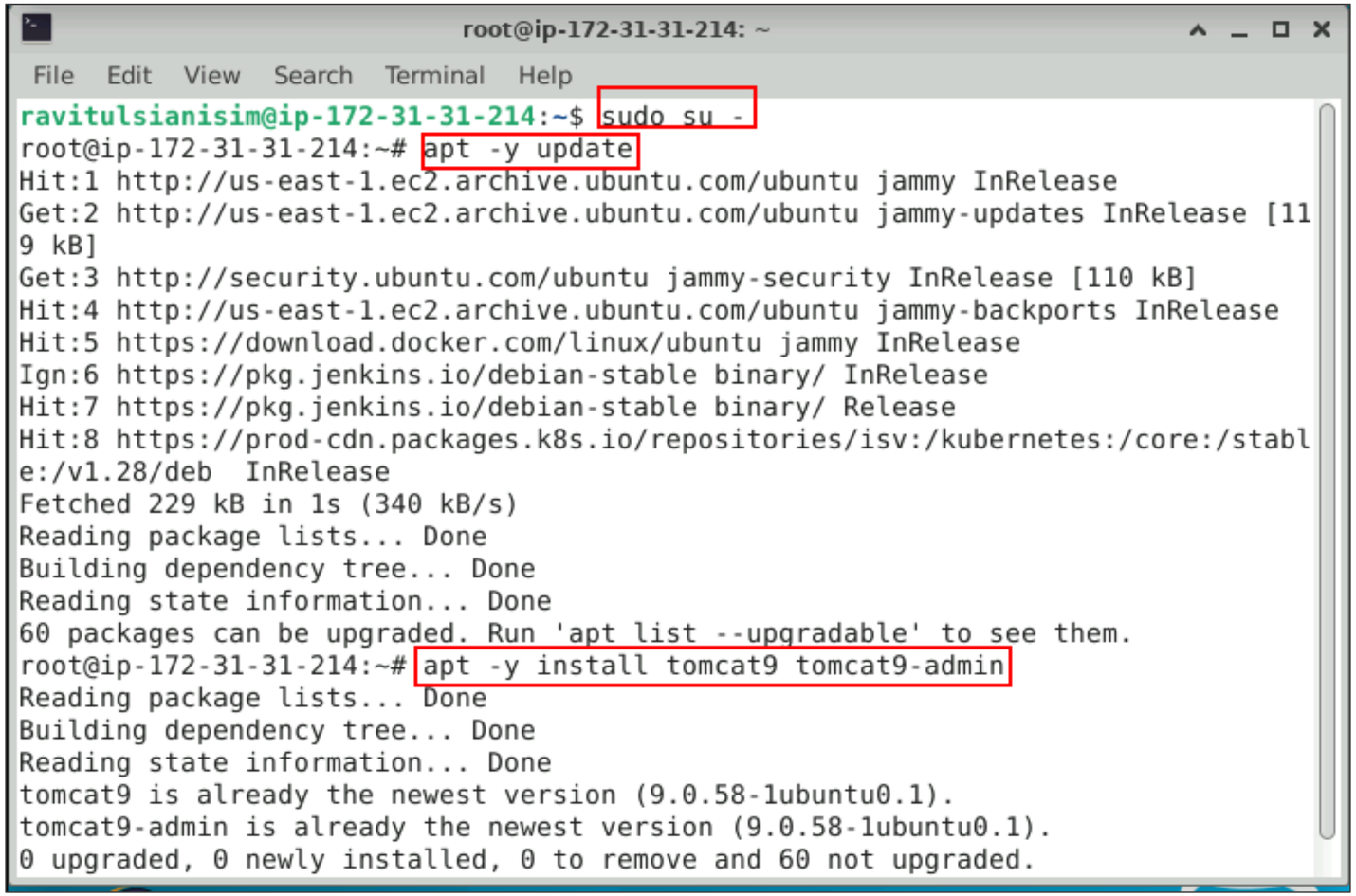
Output Screenshots

Maven workflow file configured in GitHub Actions:



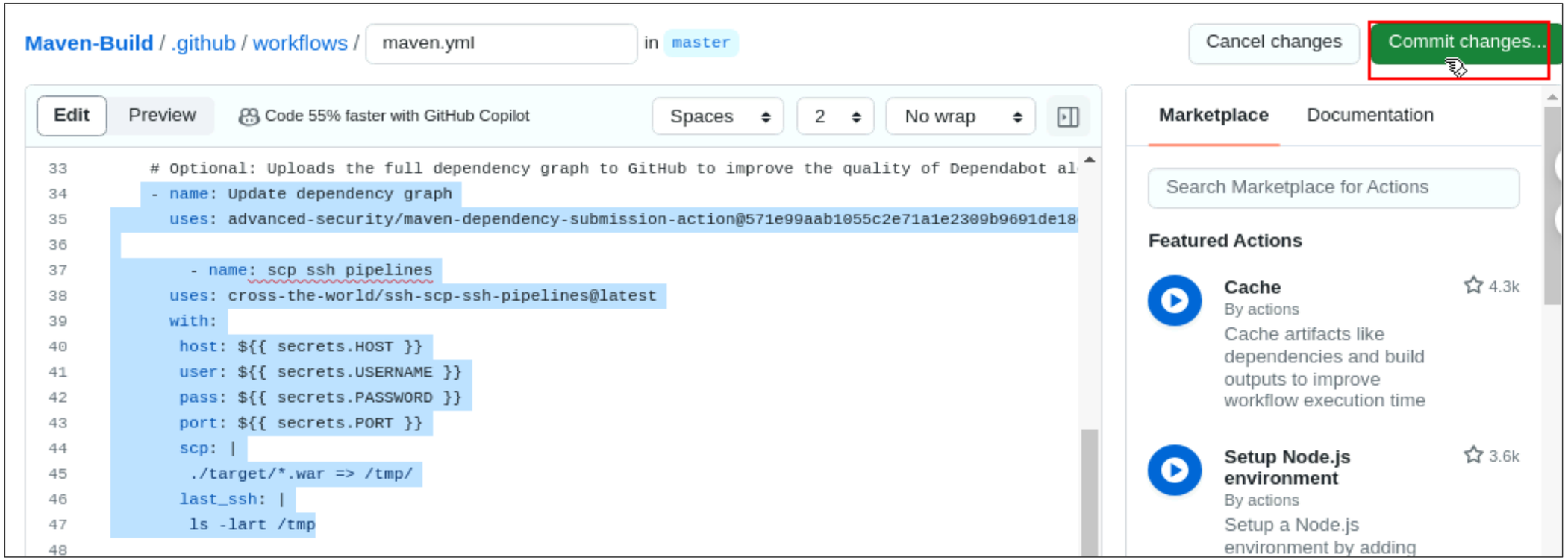
Output Screenshots

Tomcat Apache setup:



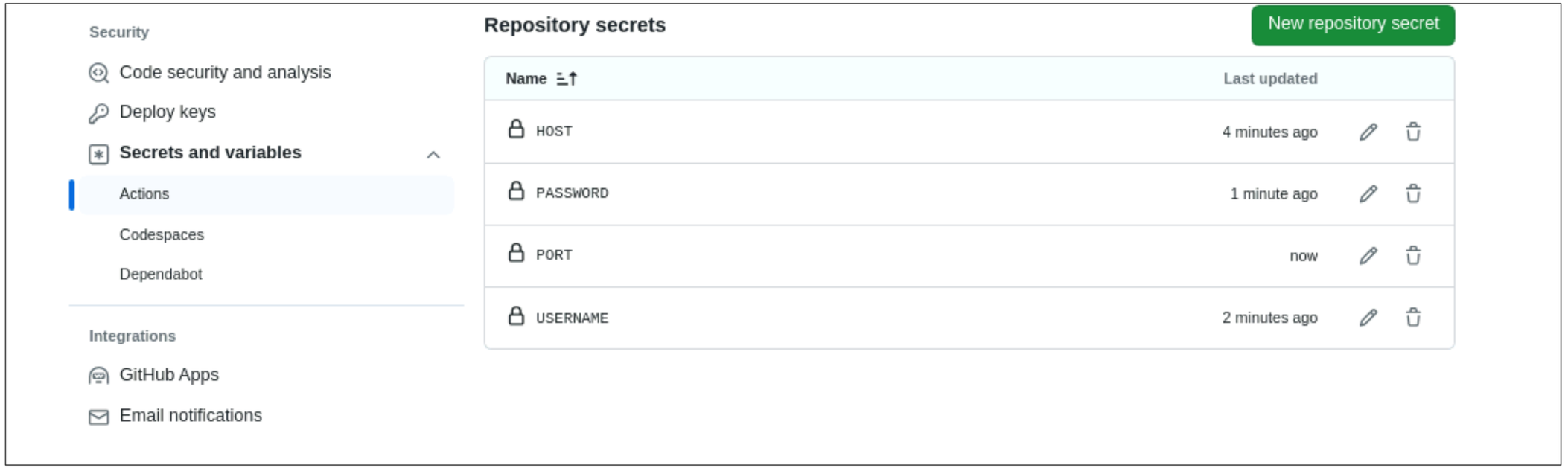
Output Screenshots

SSH pipeline configuration in **maven.yml** workflow file:



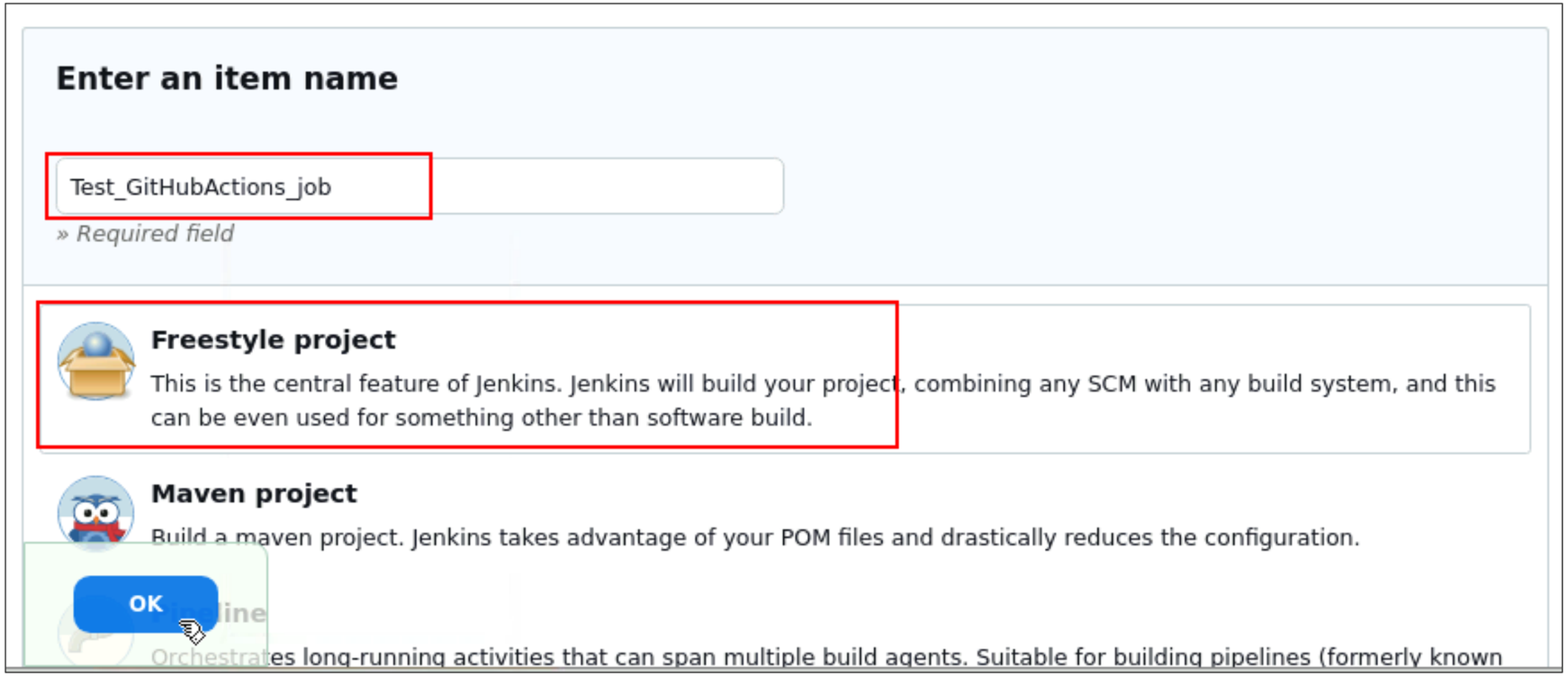
Output Screenshots

GitHub repository secrets:



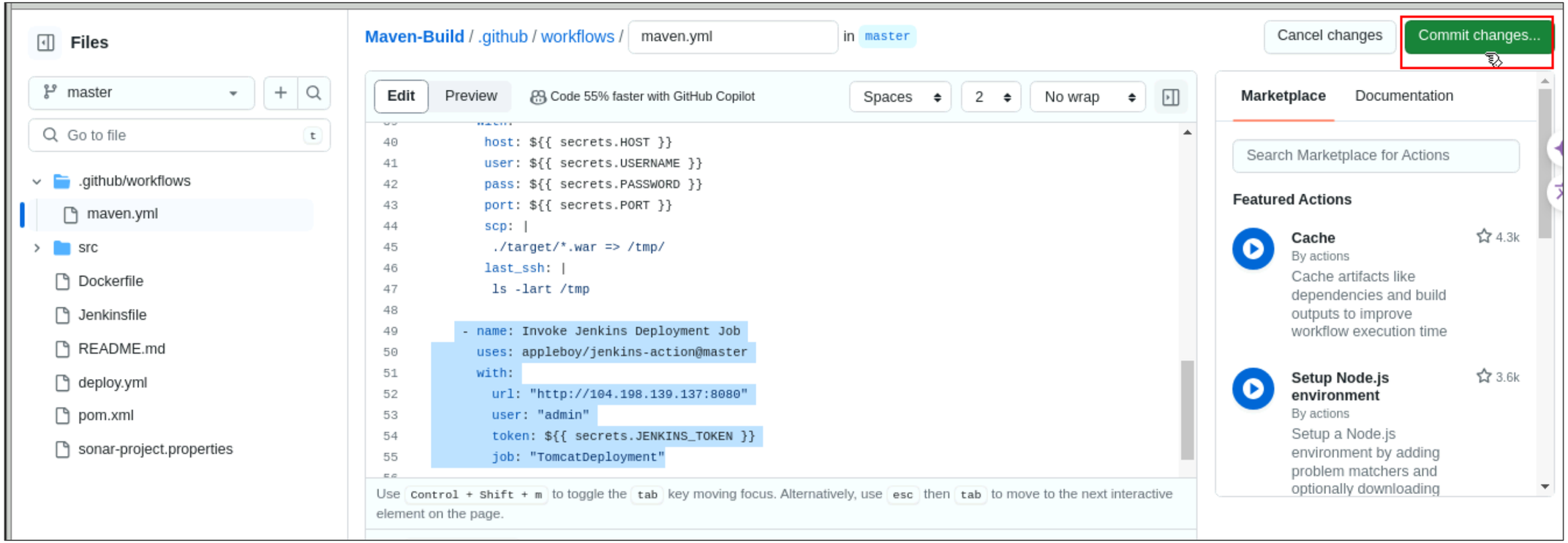
Output Screenshots

Freestyle job creation in Jenkins for continuous deployment:



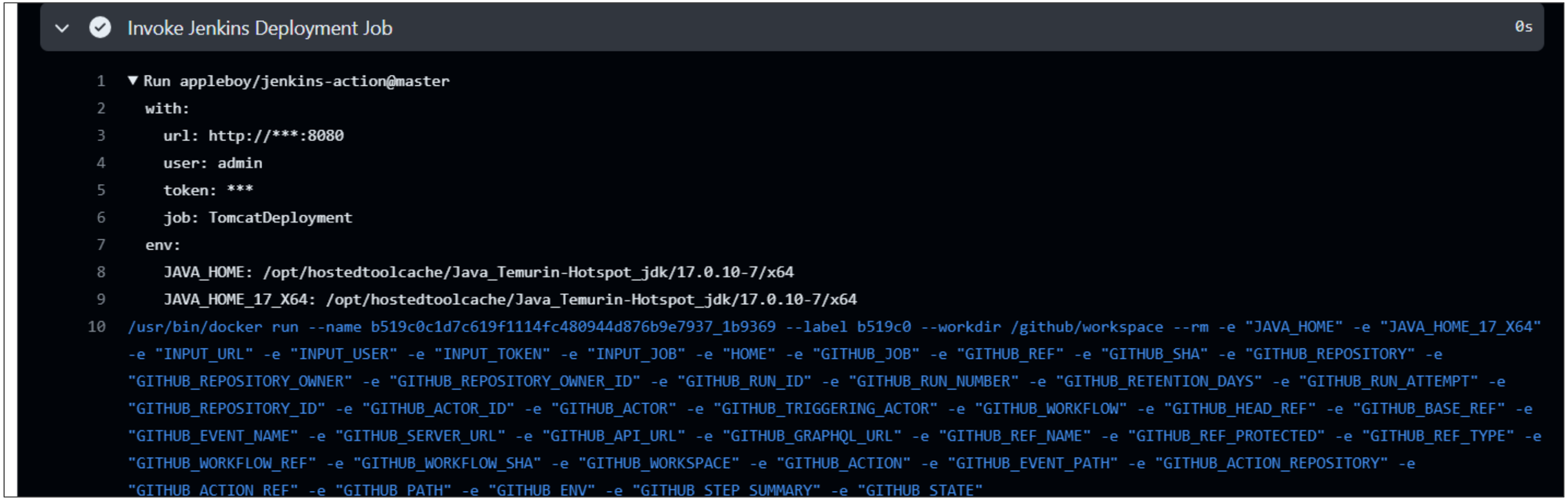
Output Screenshots

Workflow configuration to integrate Jenkins and GitHub Actions:



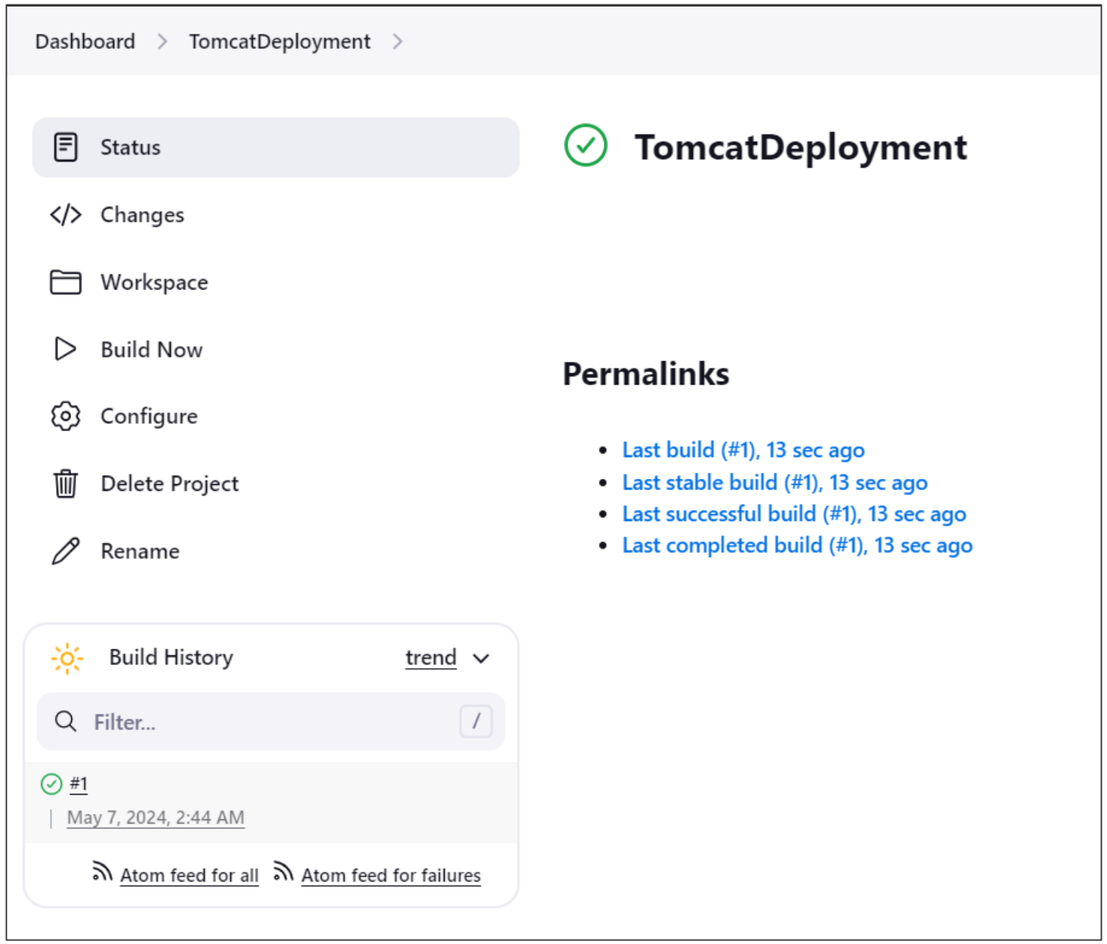
Output Screenshots

Jenkins job being invoked from the GitHub Actions pipeline for continuous deployment:



Output Screenshots

Deployment of the Java application on Tomcat by invoking Jenkins job from a GitHub Actions pipeline:



Thank you