Course1-End Project | DevOps Foundations: Version Control and CI/CD with Jenkins
Jean Fedely Zamor

Insured Assurance.

Objective

This project is to create a GitHub Actions CI/CD pipeline workflow for invoking the deployment of a Java application as a Jenkins job using Tomcat Apache.

Problem Statement

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.

Tools Used in the Project

This project employs several key tools, each with distinct functions. Below are the three main tools used for the implementation:

Jenkins: An open-source automation tool that assists in building, testing, and deploying software projects. It supports CI/CD workflows, enabling teams to automate tasks, quickly identify issues, and accelerate the software release process.

GitHub Actions: A CI/CD tool integrated within GitHub, which automates the software development process. It facilitates efficient code integration and deployment by handling workflow automation directly in GitHub, streamlining the update process for developers.

Apache Tomcat: An open-source web server and servlet container that supports Java servlets and JSPs. It is a critical component for developing and deploying Java-based web applications, making it indispensable for organizations that rely on Java technologies.

Steps to be taken

To implement CI/CD using GitHub Actions and Jenkins, we will first ensure that we complete the necessary installation and meet all prerequisites. Basically, we will start by creating a code repository on GitHub and set up a GitHub Actions pipeline to automate continuous integration. Next, we will configure Apache Tomcat for automated code deployment ensuring that the application can be deployed seamlessly. Additionally, we will Integrate the GitHub Actions pipeline with Jenkins to further automate the deployment processes. Finally, we will validate the automated deployment by invoking the pipeline and confirming that everything functions as expected.

Create a Code Repository on GitHub

We will follow the below steps to create the code repository:

1. Log in to your account, click on the "+" icon at the top right, select "New repository," fill in the repository name and optional description, choose the visibility (public or private).
2. Initialize the repository by adding a “README file.”
3. Click "Create repository."

See screenshot below:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Owner * jeazamor / **Repository name *** Course-end Project 1

Your new repository will be created as Course-end-Project-1.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about **verbose-tribble** ?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

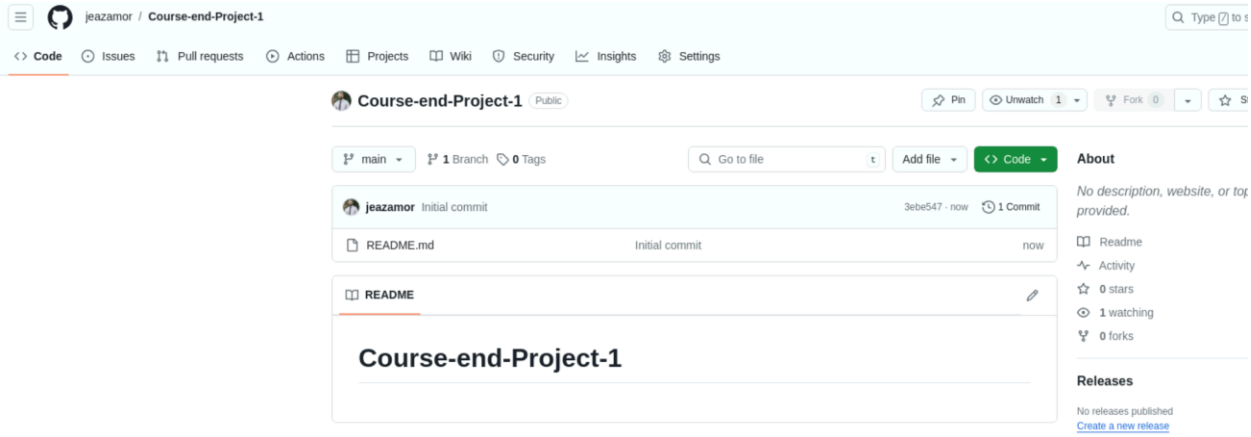
Choose a license
License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

Create repository



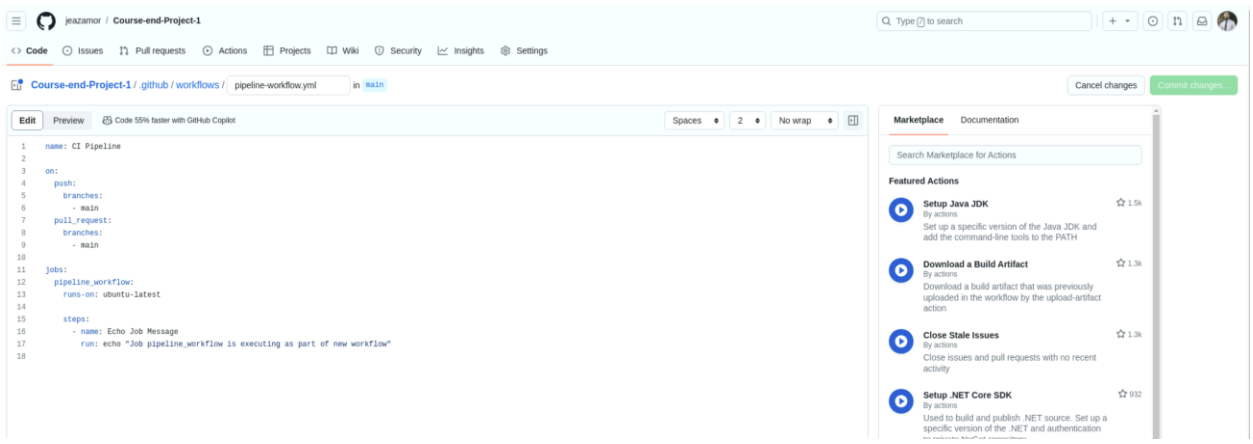
Create a GitHub Actions pipeline to perform continuous integration

We will follow the below steps to Create a GitHub Actions Pipeline for Continuous Integration

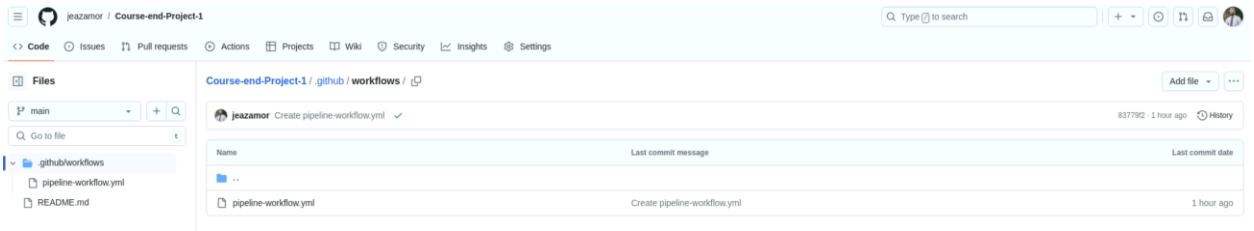
1. Navigate to the GitHub repository created above and create a new workflow by adding a YAML file in the ".GitHub/workflows" directory.
2. Define the pipeline's triggers, such as "push" or "pull request", and specify the jobs and steps required for the CI process, including setting up the environment, installing dependencies, and running tests or build commands.
3. Save the YAML file, which will automatically trigger the workflow according to the defined conditions.
4. Monitor the workflow's progress and results through the Actions tab in the GitHub repository to ensure successful integration.

See screenshots below:

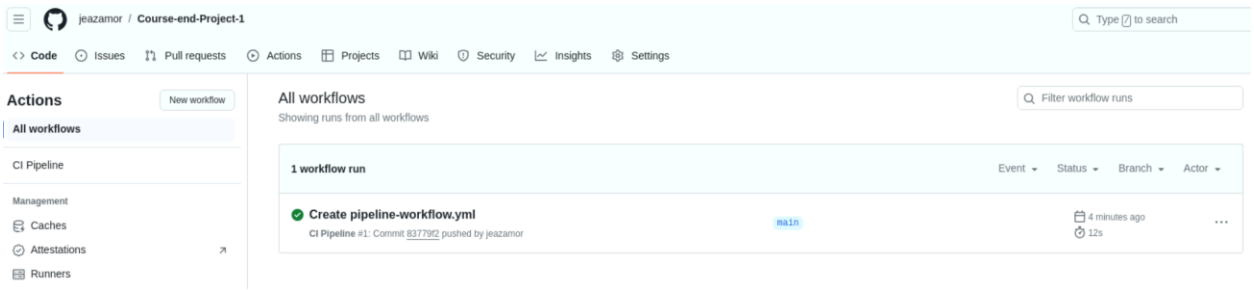
a- **YAML file**



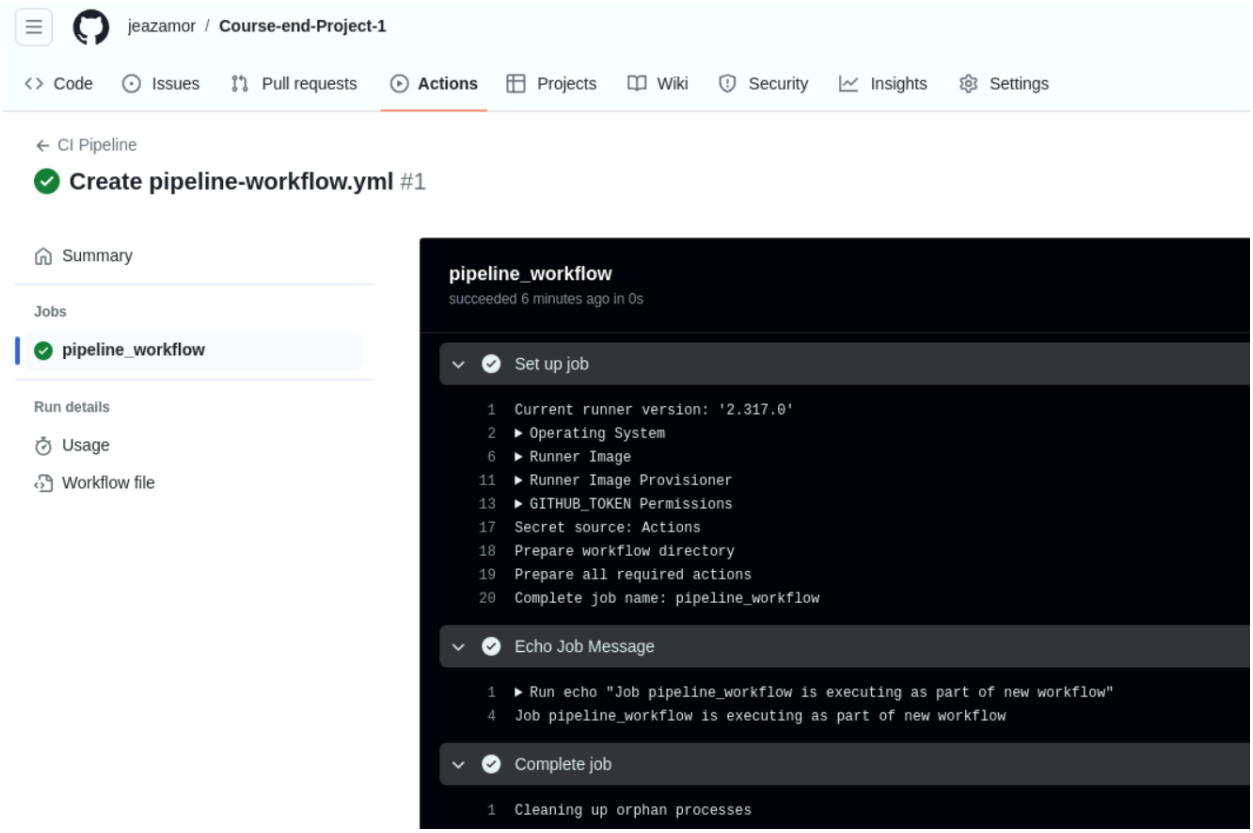
workflow file configured in GitHub Actions



b- **Workflow running successfully**



c- **Workflow's results**



Configure Tomcat Apache for automated code deployment

To configure Tomcat Apache for automated code deployment, we will follow the steps below:

1. Install Tomcat using the package manager with “sudo apt install -y tomcat9 tomcat9-admin”
2. Enable the manager and host-manager applications by editing the “/etc/tomcat9/tomcat-users.xml” to add roles and users with deployment privileges then restart Tomcat to apply changes.
3. As Jenkins already used port 8080, we will need to change the connector port for Tomcat to 9090 by running this command “sudo nano /etc/tomcat9/server.xml”
4. Finally, ensure that remote deployments are permitted by configuring the Tomcat manager with the appropriate roles for remote access.

See screenshot below:

a- **Install Tomcat**

```
File  Edit  View  Search  Terminal  Help
zfedelygmail@ip-172-31-25-71:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
64 packages can be upgraded. Run 'apt list --upgradable' to see them.
zfedelygmail@ip-172-31-25-71:~$ sudo apt install -y tomcat9 tomcat9-admin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tomcat9 is already the newest version (9.0.58-1ubuntu0.1).
tomcat9-admin is already the newest version (9.0.58-1ubuntu0.1).
The following packages were automatically installed and are no longer required:
  docker-ce-rootless-extras libslirp0 slirp4netns
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 64 not upgraded.
zfedelygmail@ip-172-31-25-71:~$
```

b- Configure user/admin in tomcat

```
File  Edit  View  Search  Terminal  Help
  GNU nano 6.2                                                    /etc/tomcat9/tomcat-users.xml
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
              version="1.0">
<!--
  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary.

  Built-in Tomcat manager roles:
    - manager-gui    - allows access to the HTML GUI and the status pages
    - manager-script - allows access to the HTTP API and the status pages
    - manager-jmx    - allows access to the JMX proxy and the status pages
    - manager-status - allows access to the status pages only

  The users below are wrapped in a comment and are therefore ignored. If you
  wish to configure one or more of these users for use with the manager web
  application, do not forget to remove the <!.. ..> that surrounds them. You
  will also need to set the passwords to something appropriate.
-->
<!--
  <user username="admin" password="<must-be-changed>" roles="manager-gui"/>
  <user username="robot" password="<must-be-changed>" roles="manager-script"/>
-->
<!--
  The sample user and role entries below are intended for use with the
  examples web application. They are wrapped in a comment and thus are ignored
  when reading this file. If you wish to configure these users for use with the
  examples web application, do not forget to remove the <!.. ..> that surrounds
  them. You will also need to set the passwords to something appropriate.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
  <user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
  <user username="role1" password="<must-be-changed>" roles="role1"/>
-->
<user username="tomcat" password="password" roles="admin-gui,manager-script"/>

</tomcat-users>

^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute   ^C Location  M-U Undo      M-A Set Mark   M-] To Bracket M-Q Previous  ^B Back      ^← Prev Word ^A Home
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify   ^/ Go To Line M-E Redo      M-6 Copy       ^O Where Was   M-W Next      ^F Forward   ^→ Next Word ^E End
```

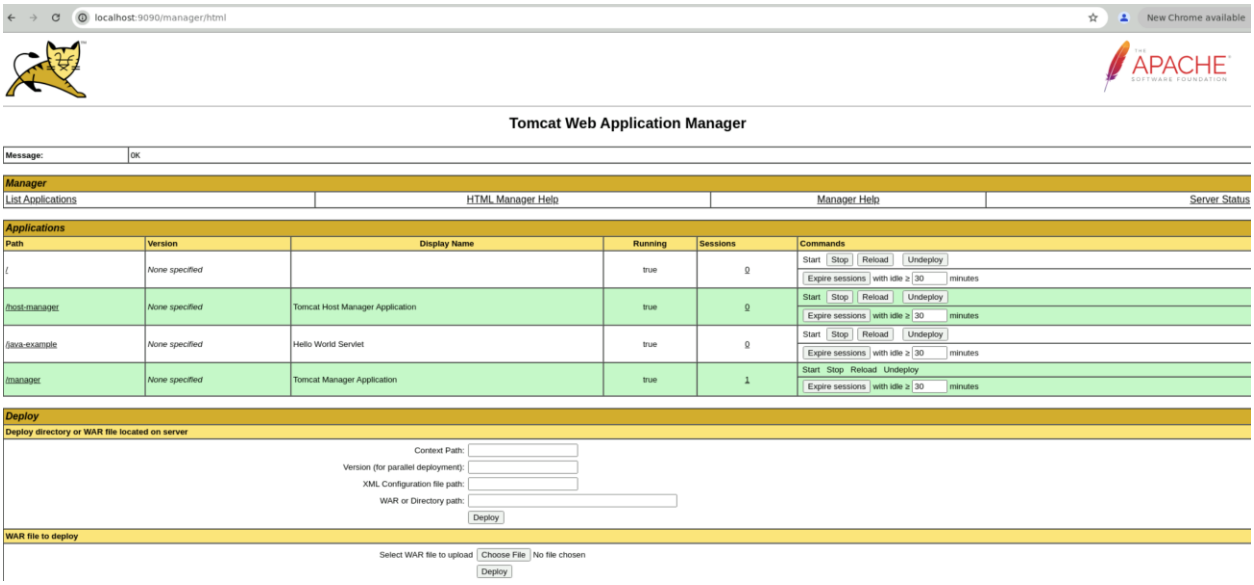
c-Change the connector port of Tomcat

```
zfedelygmail@ip-172-31-25-71: ~
File  Edit  View  Search  Terminal  Help
GNU nano 6.2                    /etc/tomcat9/server.xml
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->


    <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="9090" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"              connectionTimeout=>
               redirectPort="8443" />

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^/ Go To Line
```

d- Restart Tomcat

```
zfedelygmail@ip-172-31-25-71:~$ sudo nano /etc/tomcat9/tomcat-users.xml
zfedelygmail@ip-172-31-25-71:~$ sudo systemctl restart tomcat9
zfedelygmail@ip-172-31-25-71:~$
```

e- Successfully access Tomcat manager



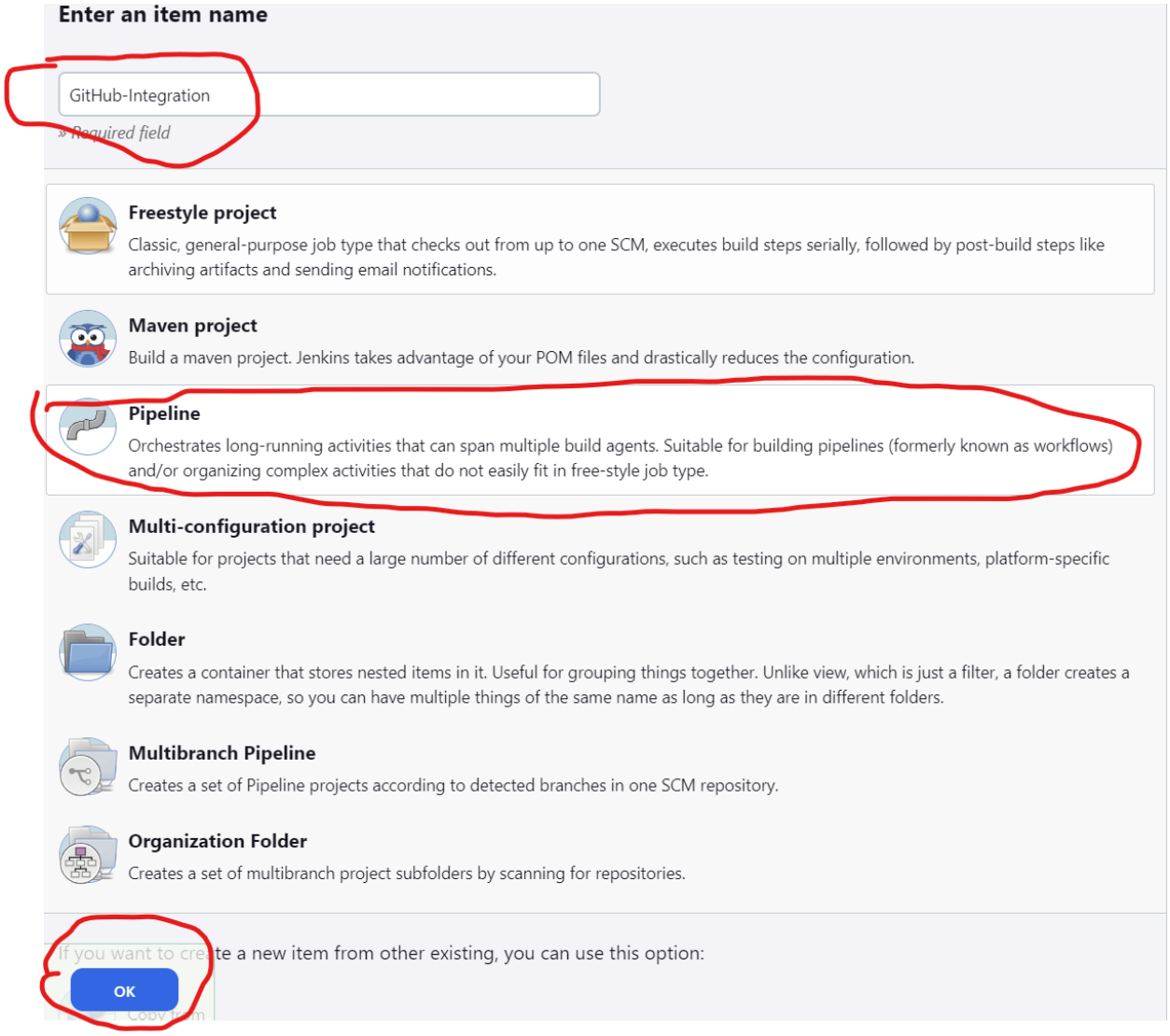
Integrate the GitHub Actions pipeline to invoke the Jenkins pipeline

To integrate a GitHub Actions pipeline with Jenkins pipeline, we will follow the steps below:

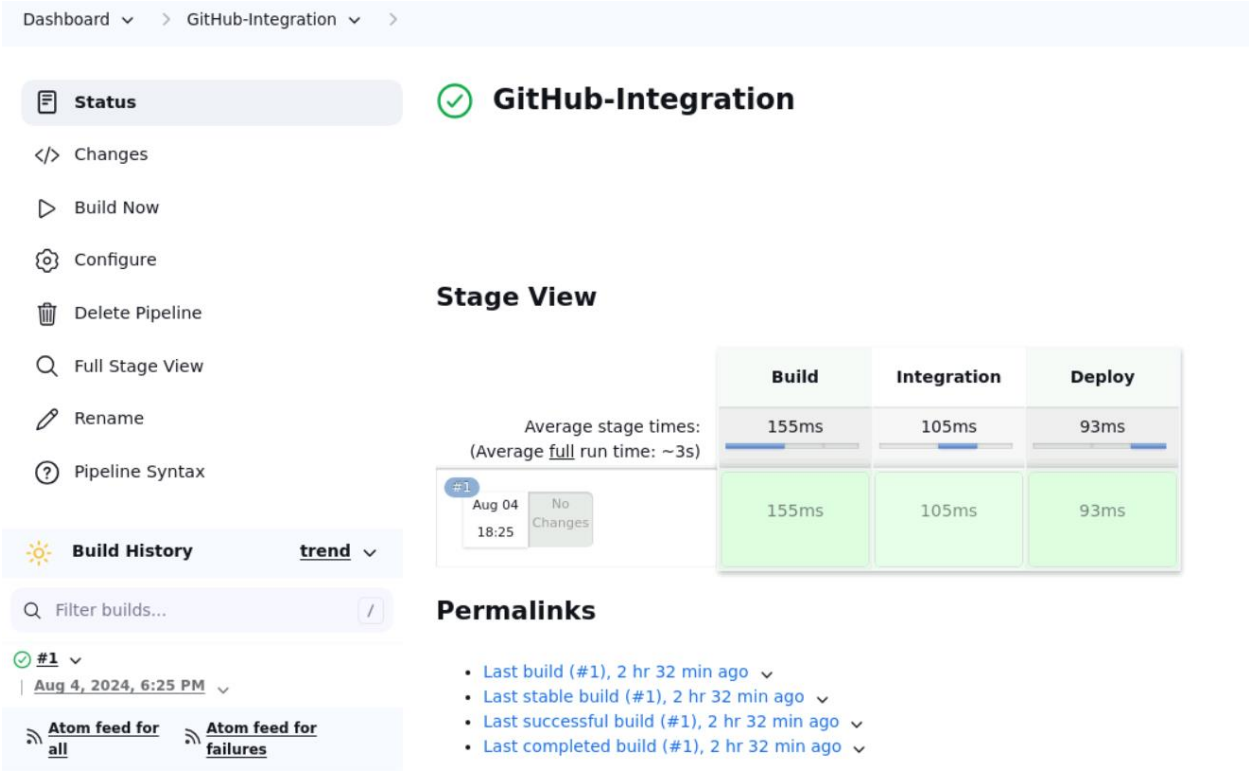
1- Install Jenkins and configure it with the necessary plugins (Git, GitHub, SSH Pipeline Steps, Pipeline)
2- Create a new Jenkins job or pipeline for deployment.
3- Update the GitHub Actions workflow file (pipeline-workflow.yml) to include a job that triggers Jenkins by using “curl” with Jenkins credentials and job details.
4- Finally, in the repository settings under "Secrets and variables" > "Actions," add the necessary GitHub Secrets for “JENKINS_URL”, “JENKINS_USER”, and “JENKINS_TOKEN” to securely access Jenkins.

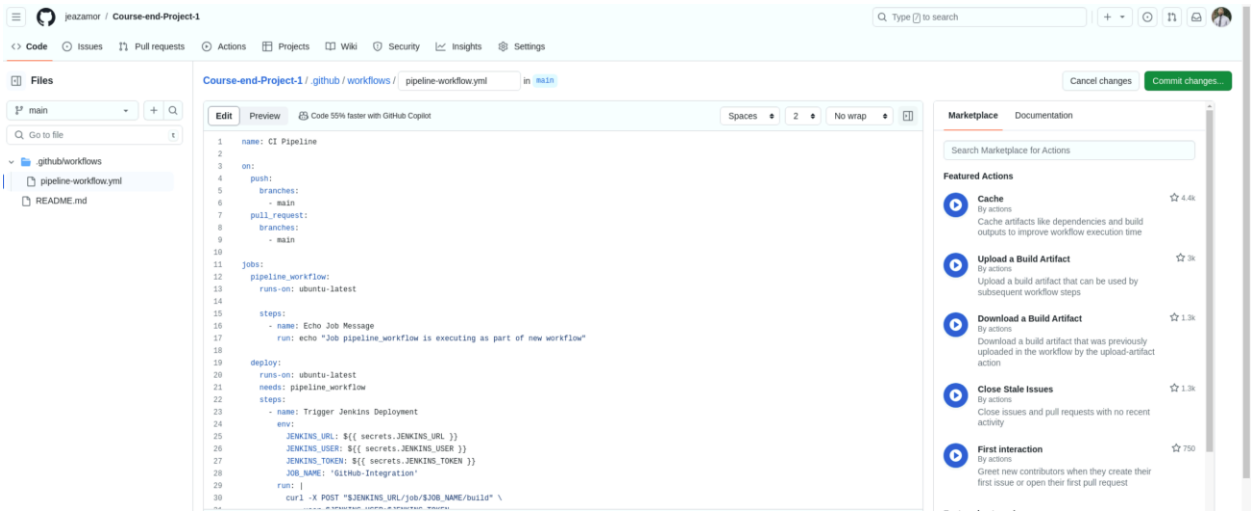
See screenshots below:

Jenkins job creation

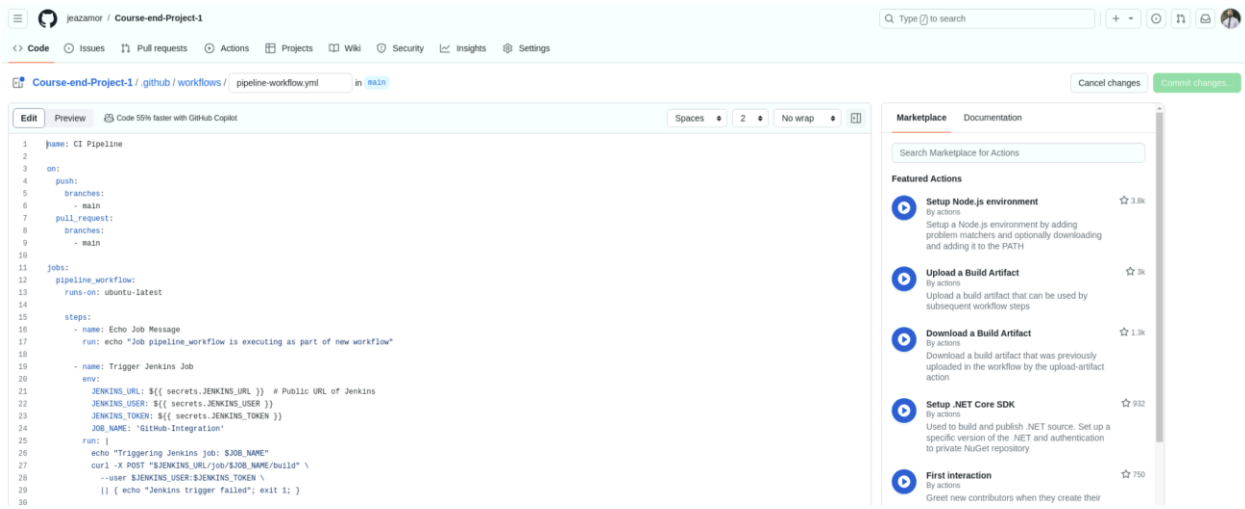


The job successfully completed

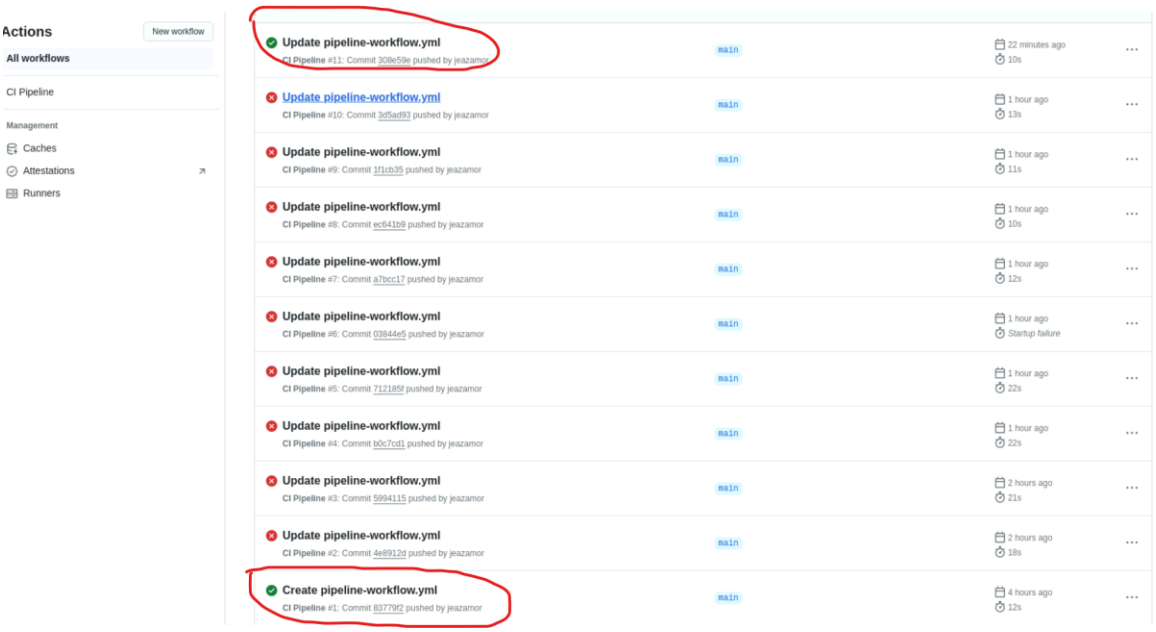




GitHub Actions workflow file update:



yml file finally update after many attempts:

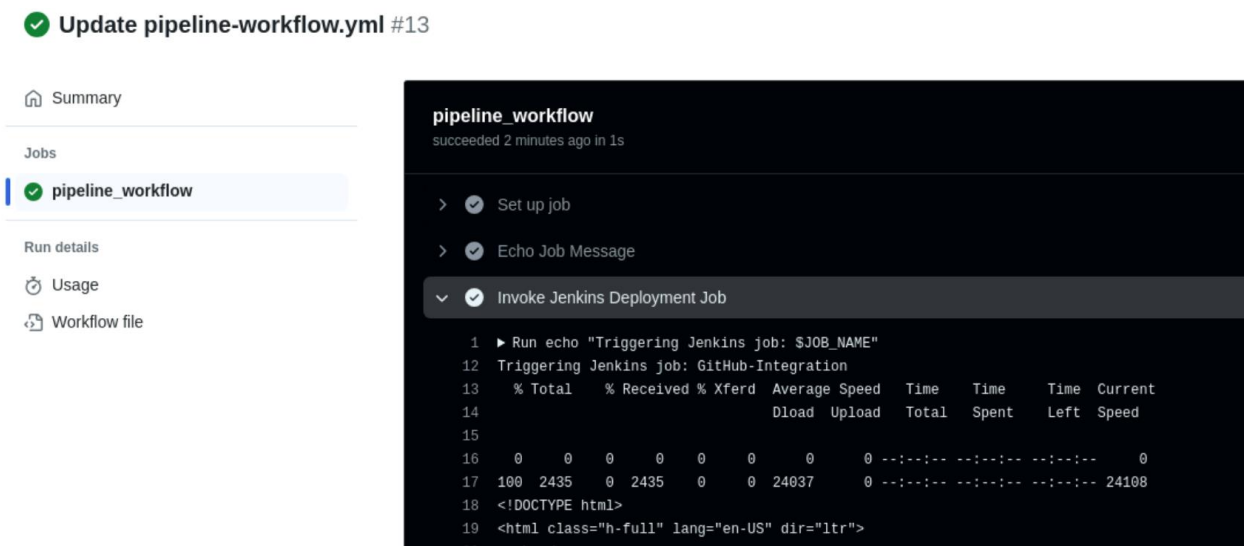


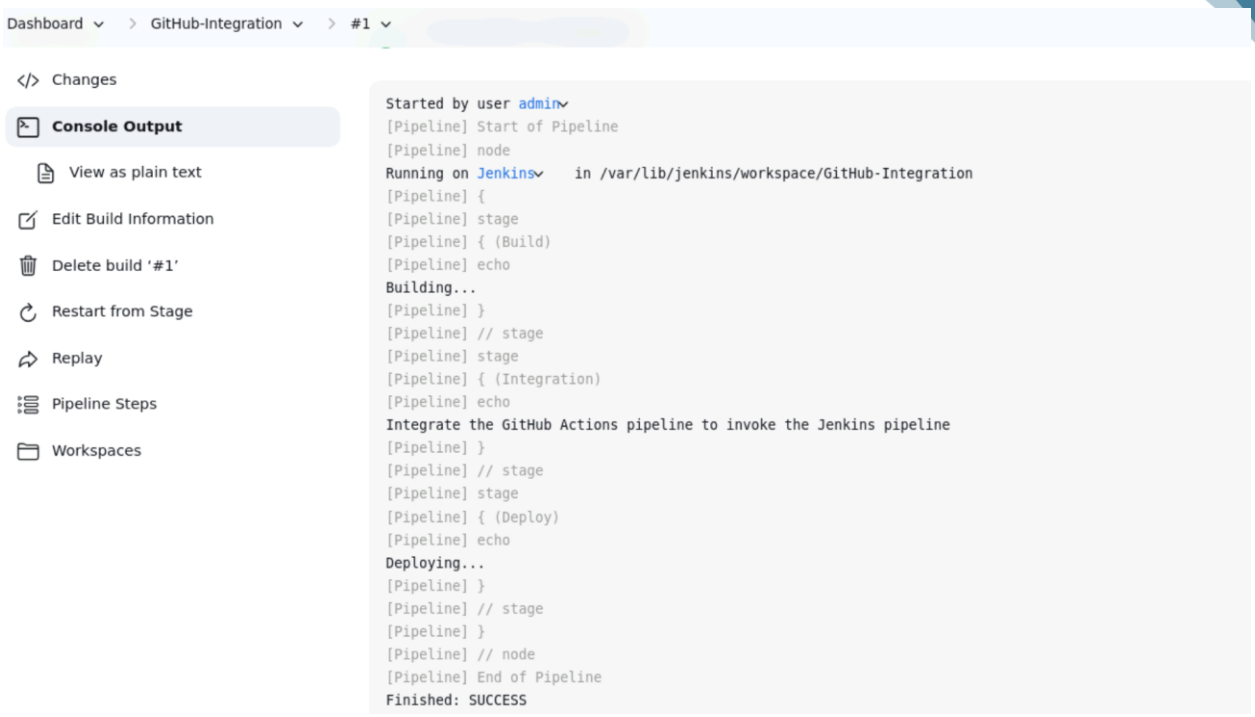
GitHub Secrets

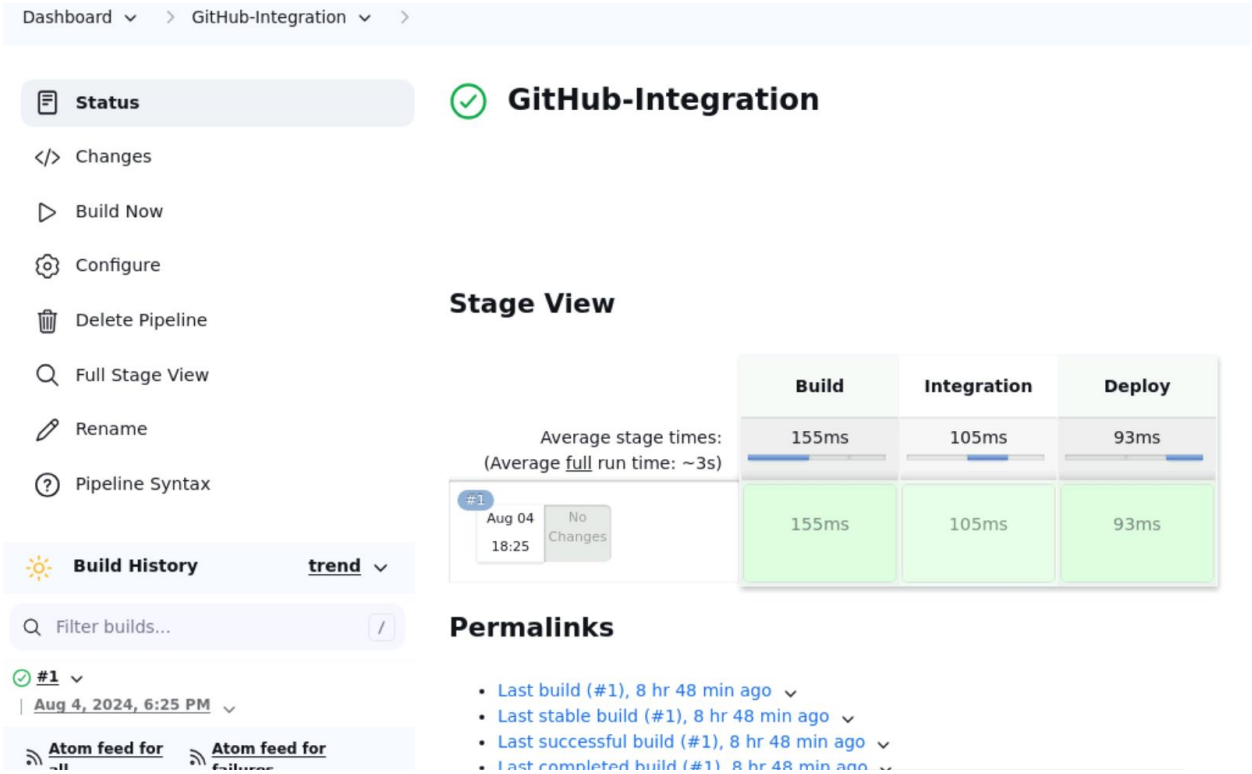
Invoke pipeline to validate automated deployment

To validate automated deployment, we will start by committing and pushing changes to our Java application's main branch. Then, monitor the CI pipeline's execution in the GitHub Actions tab of our repository. After the workflow is complete, we will check the Jenkins dashboard to confirm that the deployment job was triggered. Finally, we will try to access the deployed application on the Tomcat server to ensure that the deployment was successful.

- Update GitHub Actions pipeline-workflow.yml file to include a step that triggers the Jenkins pipeline:




Dashboard
GitHub-Integration
#1
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Restart from Stage
Replay
Pipeline Steps
Workspaces
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/GitHub-Integration
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integration)
[Pipeline] echo
Integrate the GitHub Actions pipeline to invoke the Jenkins pipeline
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS


Dashboard
GitHub-Integration
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
Filter builds...
#1
Aug 4, 2024, 6:25 PM
Atom feed for
Atom feed for
GitHub-Integration
Stage View
Build
Integration
Deploy
Average stage times:
(Average full run time: ~3s)
155ms
105ms
93ms
#1
Aug 04
18:25
No Changes
155ms
105ms
93ms
Permalinks
Last build (#1), 8 hr 48 min ago
Last stable build (#1), 8 hr 48 min ago
Last successful build (#1), 8 hr 48 min ago

Conclusion

This document outlines how to set up a CI/CD pipeline for Insured Assurance using GitHub Actions, Jenkins, and Tomcat. The pipeline automates the build, test, and deployment process for a Java application, deploying it to a Tomcat server for reliable software delivery. It also provides a starting point for future enhancements like automated rollbacks as the company improves its DevOps practices.