

CI/CD Pipeline Project Documentation (AWS, Jenkins, GitHub Actions, Tomcat)

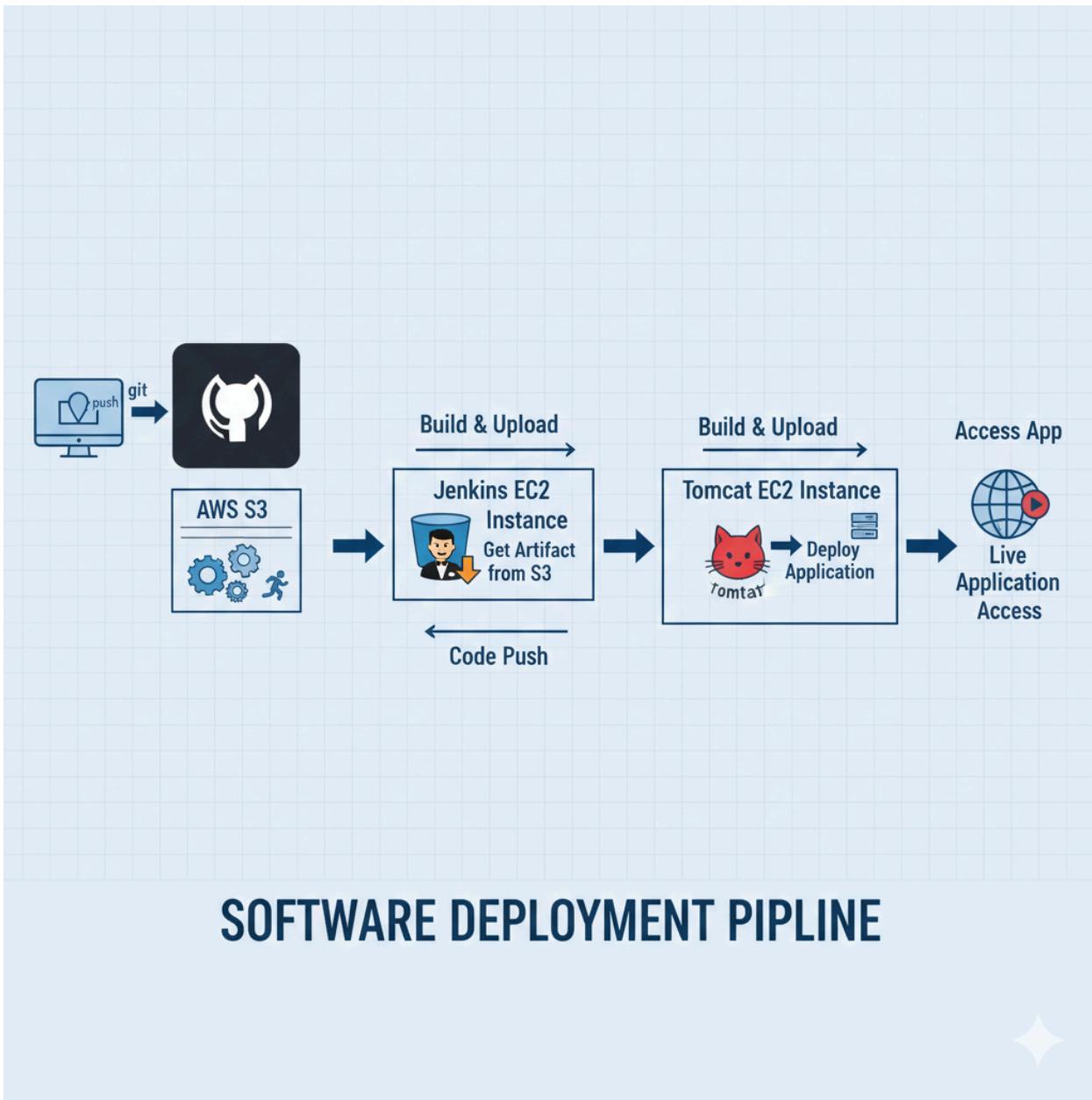
Project Overview

This project implements a CI/CD pipeline following robust, reusable standards suitable for public repositories and FAANG-style architectures. The solution:

- Uses AWS Free Tier with EC2, S3, IAM.
- Automates build, test, artifact upload, and deployment for web applications.
- Employs GitHub Actions for CI and Jenkins for CD to Apache Tomcat.

Architecture

- **AWS EC2:**
 - Jenkins instance (public IP with security group open for SSH and Jenkins port).
 - Tomcat instance (private or public IP, security group open for SSH from Jenkins, Tomcat port restricted).
- **AWS S3:** Private bucket for WAR artifacts.
- **IAM:** User with limited S3 permissions.
- **Security:** All secrets are handled via encrypted Jenkins credentials and GitHub repository secrets.
- **Git-repo:** <https://github.com/mohammedinzi/insured-assurances.git>



Screenshot: Network diagram showing Jenkins EC2, Tomcat EC2, S3 bucket, GitHub.

Step-by-Step Setup

1. AWS Resources

- Create a private S3 bucket (e.g., `insured-assurance-artifacts-uniqueid`) for build artifacts.

- Launch two EC2 instances:
 - Jenkins: t2.micro, Amazon Linux 2 or Ubuntu 22.04.
 - Tomcat: t2.micro, matching OS.
- Assign Elastic IPs for public access if needed (demo usage).
- Configure Security Groups:
 - Jenkins SG: Open port 22 to your IP; 8080 for Jenkins.
 - Tomcat SG: Open 22 from Jenkins only; 8080 for your IP/demo/public.

The screenshot shows the AWS IAM User Details page for a user named 'shoailb'. The user was created on October 05, 2025, at 16:42 UTC+05:30. The ARN is arn:aws:iam::970597968438:user/shoailb. The user has console access enabled without MFA. There is one attached policy, 'AdministratorAccess', which is an AWS managed - job function attached via the 'admin' group. The user has two access keys: one active and one used 12 hours ago, 15 hours old. The user has no groups or tags attached. The 'Permissions' tab is selected. The 'Permissions policies' section shows the attached policy. The 'Permissions boundary' section is set to '(not set)'. The 'Generate policy based on CloudTrail events' section indicates no requests to generate a policy in the past 7 days.

Searched for: [Option+S]

United States (N. Virginia) aws-inzemam (9705-9796-8438) shoalb

EC2 Instances (1/2) Info

Instances (1/2) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
tomcat	i-00454224f5b4047e2	Running	t3.micro	2/3 checks passed	View alarms	us-east-1b	ec2-34-207-115-103.co...	34.207.115.103	-
jenkins	i-050a7c09904dad825	Running	t3.micro	3/3 checks passed	View alarms	us-east-1b	ec2-3-90-187-93.comp...	3.90.187.93	-

i-050a7c09904dad825 (jenkins)

Details | Status and alarms | Monitoring | **Security** | Networking | Storage | Tags

▼ Security details

IAM Role - Owner ID: 970597968438 Launch time: Mon Oct 06 2025 10:34:15 GMT+0530 (India Standard Time)

Security groups: sg-0f40f7c841507e2fd (jenkins-sg)

▼ Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0469e2cefb03705d7	22	TCP	122.183.169.1/32	jenkins-sg	-
-	sgr-05f3574fa9bd5a6fb	8080	TCP	::/0	jenkins-sg	-
-	sgr-042ba2bc59d829d5	8080	TCP	0.0.0.0/0	jenkins-sg	-

Outbound rules

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Amazon S3

General purpose buckets

- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight: **CloudWatch Metrics Insights**

AWS Marketplace for S3

cloudShell Feedback

EC2 Instances

Instances (1/2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
tomcat	i-00454224f5b4047e2	Running	t3.micro	0/3 checks passed	View alarms +	us-east-1b	ec2-54-207-115-103.co...	34.207.115.103	-
jenkins	i-050a7c09904dad825	Running	t3.micro	0/3 checks passed	View alarms +	us-east-1b	ec2-3-90-187-93.comp...	3.90.187.93	-

i-00454224f5b4047e2 (tomcat)

Security details

IAM Role: -

Owner ID: 970597968438

Launch time: Mon Oct 06 2025 10:34:15 GMT+0530 (India Standard Time)

Security groups: sg-0330194577f793af5 (tomcat)

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
-	sgr-0cf0bb7a51fe6ead3	8080	TCP	sg-0f40f7c841507e2fd	tomcat	-
-	sgr-0364c0df602a3109a	8080	TCP	::/0	tomcat	-
-	sgr-03545097b23f09227	22	TCP	122.183.169.18/32	tomcat	-
-	sgr-0ba34629edf075643	8080	TCP	0.0.0.0/0	tomcat	-

CloudShell Feedback

Screenshot: EC2 console showing instance details and security group rules and s3 bucket.

2. Jenkins Installation and Configuration

Install Jenkins on EC2:

bash

```
sudo yum update -y
```

```
sudo amazon-linux-extras install java-openjdk11 -y
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum install jenkins git -y
sudo systemctl enable --now jenkins
sudo systemctl status jenkins
```

- Create an admin user and generate Jenkins API token.
- Add necessary Jenkins plugins: Maven Integration, GitHub, Publish Over SSH.

Screenshot: Jenkins initial setup, plugin installation, API token creation.

3. Tomcat Installation and Service Configuration

Install Tomcat:

```
bash
sudo yum update -y
sudo amazon-linux-extras install java-openjdk11 -y
sudo useradd -m -U -d /opt/tomcat tomcat
cd /opt
sudo curl -O
https://downloads.apache.org/tomcat/tomcat-9/v9.0.xx/bin/apache-tomcat-9.0.xx.tar.gz
sudo tar xzf apache-tomcat-9.0.xx.tar.gz
sudo ln -s apache-tomcat-9.0.xx tomcat
sudo chown -R tomcat /opt/apache-tomcat-9.0.xx
```

- Create a systemd service for Tomcat:

```
bash
sudo tee /etc/systemd/system/tomcat.service <<EOF
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target
```

```

[Service]
Type=forking
User=tomcat
Group=tomcat
Environment="JAVA_HOME=/usr/lib/jvm/jre"
Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.pid"
ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
Restart=on-failure

[Install]
WantedBy=multi-user.target
EOF

```

```

sudo systemctl daemon-reload
sudo systemctl enable --now tomcat
sudo systemctl status tomcat

```

Screenshot: Tomcat running via systemd, Tomcat manager interface.

4. SSH Key Setup for Jenkins → Tomcat Deployment

- On Jenkins EC2:

```

bash
sudo su - jenkins
ssh-keygen -t rsa -b 4096 -f ~/.ssh/tomcat_id_rsa -N ""

```

- Copy the public key to Tomcat's `.ssh/authorized_keys` using EC2 .pem file or SSM if password not known:

```

bash
ssh-copy-id -i ~/.ssh/tomcat_id_rsa.pub tomcat@TOMCAT_PUBLIC_IP

```

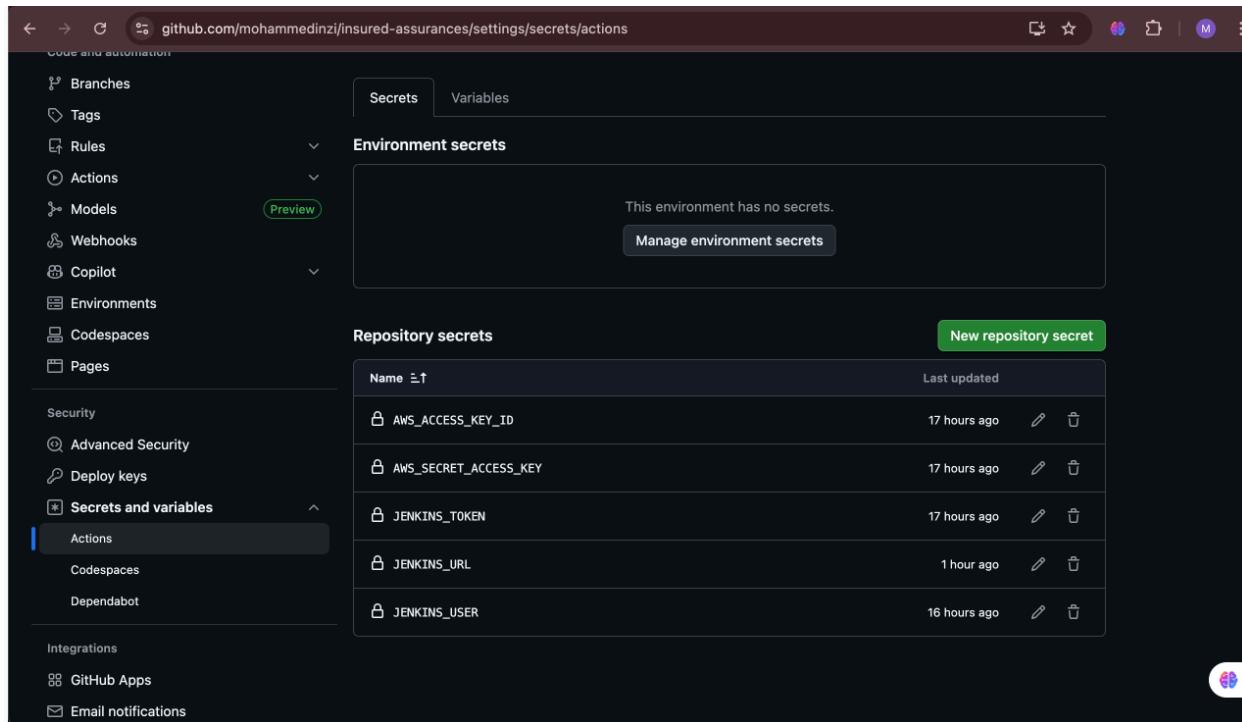
(Alternative: use EC2 console PEM key, SSM, or manual copy as needed)

```
[root@ip-172-31-26-71 ~]# curl -u jenkins:Jenkins@123 http://34.207.115.103:8080/manager/text/list
OK - Listed applications for virtual host [localhost]
/:running:0:ROOT
/examples:running:0:examples
/host-manager:running:0:host-manager
/manager:running:0:manager
/docs:running:0:docs
[root@ip-172-31-26-71 ~]# curl -u jenkins:Jenkins@123 "http://34.207.115.103:8080/manager/text/undeploy?path=/"
OK - Undeployed application at context path [/]
[root@ip-172-31-26-71 ~]#
```

Screenshot: SSH connectivity test from Jenkins to Tomcat.

5. GitHub Secrets and Repository

- Add secrets to project repository:
 - AWSACCESSKEYID
 - AWSSECRETACCESSKEY
 - AWSREGION
 - S3BUCKET
 - JENKINSURL
 - JENKINSUSER
 - JENKINSTOKEN
- Structure repository:
 - README.md, .github/workflows/maven.yml, src/, pom.xml for Maven webapp, webapp directory for HTML/CSS/JS.



Screenshot: GitHub repository settings with secrets configured.

6. GitHub Actions Workflow

Sample `.github/workflows/maven.yml`:

```
text
name: CI - build war - upload to S3 - trigger Jenkins
on:
  push:
    branches: [main]

env:
  WARNAME: insured-assurance.war

jobs:
  build-and-trigger:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Set up Java 11
```

```
uses: actions/setup-java@v3
with:
  distribution: temurin
  java-version: 11
- name: Cache Maven packages
  uses: actions/cache@v4
  with:
    path: ~/.m2/repository
    key: ${{ runner.os }}-maven-${{ hashFiles('pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-
- name: Build with Maven
  run: mvn -B clean package
- name: Configure AWS credentials
  uses: aws-actions/configure-aws-credentials@v2
  with:
    aws-access-key-id: ${{ secrets.AWSACCESSKEYID }}
    aws-secret-access-key: ${{ secrets.AWSSECRETACCESSKEY }}
    aws-region: ${{ secrets.AWSREGION }}
- name: Upload WAR to S3
  run: |
    ARTIFACTKEY=releases/${{ github.sha }}-${{ env.WARNAME }}
    aws s3 cp target/${{ env.WARNAME }} s3://${{ secrets.S3BUCKET }}/${{ ARTIFACTKEY }}
    echo "s3://${{ secrets.S3BUCKET }}/${{ ARTIFACTKEY }}" >
artifactpath.txt
- name: Create presigned URL (1 hour)
  id: presign
  run: |
    ARTIFACTPATH=$(cat artifactpath.txt)
    PRESIGNEDURL=$(aws s3 presign ${ARTIFACTPATH} --expires-in
3600)
    echo "url=${PRESIGNEDURL}" >> $GITHUB_OUTPUT
- name: Trigger Jenkins job
  uses: appleboy/jenkins-action@v0.3.0
  with:
    url: ${{ secrets.JENKINSURL }}
    user: ${{ secrets.JENKINSUSER }}
```

```

token: ${ secrets.JENKINSTOKEN }
job: TomcatDeployment
parameters: |
  PRESIGNEDURL=${ steps.presign.outputs.url }

```

The screenshot shows the GitHub Actions workflow run log for the 'insured-assurances' repository. The left sidebar displays project management options like Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main area lists 10 workflow runs, each with a status indicator (green checkmark for success), the name of the action, the event (e.g., Java CI/CD to Tomcat), the commit hash, the pushed-by user (mohammedinzi), the branch (main), the timestamp (e.g., Today at 12:11 PM, Oct 5, 10:08 PM GMT+5:30), and the duration (e.g., 36s, 1m 10s). The runs are: 1. update maven (success), 2. update testfile (failure), 3. doc test update (success), 4. doc test (success), and 5. doc readme (success).

Screenshot: GitHub Actions workflow run log.

7. Jenkins Job Setup (TomcatDeployment)

- Type: Freestyle project.
- Add String Parameter: **PRESIGNEDURL**
- Build Step: Execute Shell

bash

```

#!/bin/bash
set -euo pipefail
echo "Downloading WAR from presigned URL..."
wget -O $WORKSPACE/app.war "$PRESIGNEDURL"
echo "Copying WAR to Tomcat server..."

```

```

scp -i ~/.ssh/tomcat_id_rsa $WORKSPACE/app.war
tomcat@TOMCAT_PUBLIC_IP:/opt/tomcat/webapps/insured-assurance.war
ssh -i ~/.ssh/tomcat_id_rsa tomcat@TOMCAT_PUBLIC_IP "sudo systemctl
restart tomcat"
echo "Deployed to Tomcat done."

```

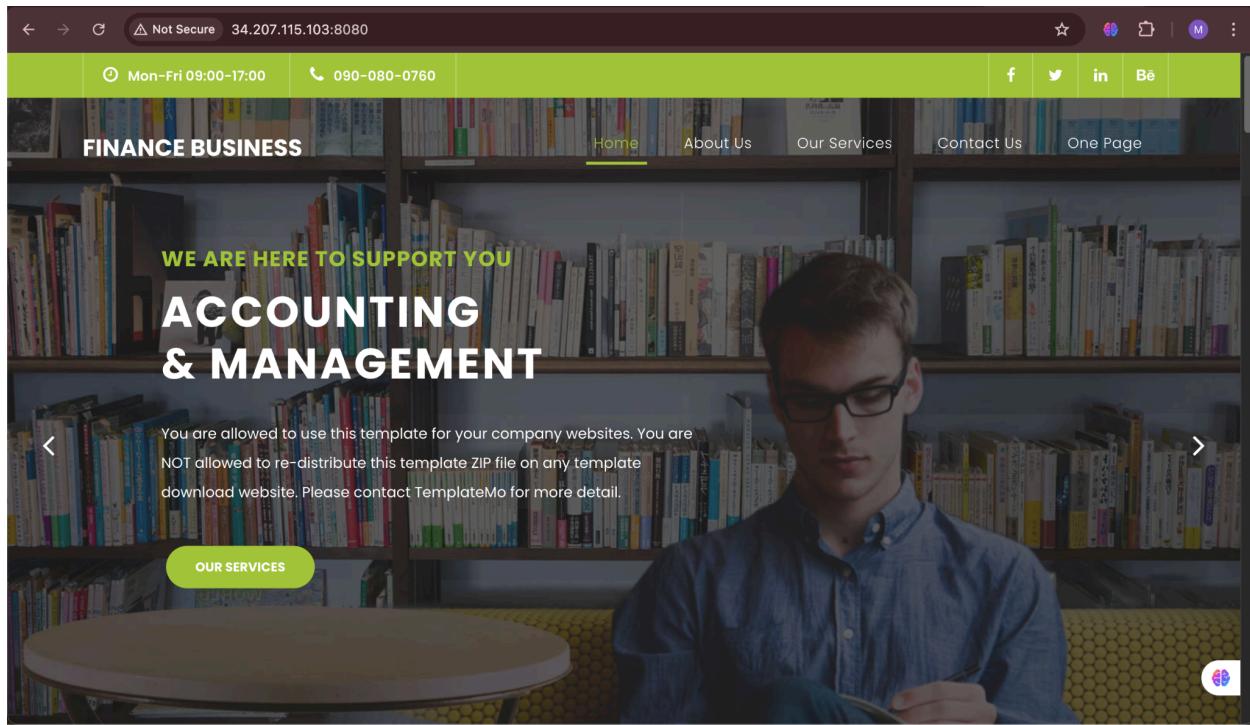
The screenshot shows the Jenkins interface for the 'Test_GithubAction_job' project. The left sidebar contains links for Status, Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, Favorite, Open Blue Ocean, Rename, and Credentials. The main content area displays the build history under the 'Builds' section. A table lists builds from today and October 5, 2025. The build #11 (07:02) is marked with a green checkmark and labeled 'Last successful build'. Other builds are marked with red circles and labeled 'Last completed build'.

Build	Status	Time
#11	Success	07:02
#10	Failure	06:50
#9	Failure	06:46
#8	Failure	17:03

8. Validation and Testing

- Push update to GitHub **main** branch.
- Verify:
 - GitHub Actions runs and completes successfully (build, S3 upload, Jenkins trigger).
 - Jenkins job triggered and shows download/copy/restart logs.
 - S3 contains the expected WAR artifact.
 - Browser loads app at http://TOMCAT_PUBLIC_IP:8080/insured-assurance.
 - Health endpoints respond (HTTP 200).

- Troubleshoot using logs from GitHub Actions and Jenkins console.



Screenshot: Application deployed and running on Tomcat (browser view).

9. Optional Improvement Areas

- Dedicated deploy user on Tomcat with permissions.
- Blue/Green deployment by deploying to `/v2` then swapping context.
- Add Slack notifications for success/failure in Jenkins.
- Store all secrets in environments and credentials managers only.
- Use HTTPS for Jenkins/Tomcat; OIDC for GitHub → AWS credentialing.

10. Troubleshooting

- **Build failures:** Ensure `pom.xml` is present for Maven builds.

- **Jenkins SSH failures:** Check private key permissions, authorized_keys on Tomcat, and security group rules.
- **S3 authentication errors:** Confirm AWS credentials provided in Jenkins and GitHub secrets.
- **Pipeline deployment errors:** Check WAR upload step and instance disk space; verify Tomcat logs for app context issues.

11. Final Checklist

1. S3 bucket exists with artifacts uploaded.
 2. GitHub secrets configured.
 3. EC2 instances running (Jenkins and Tomcat).
 4. Jenkins job setup with correct shell script and parameters.
 5. GitHub Actions workflow file validated and working.
 6. SSH connectivity established between Jenkins and Tomcat.
 7. App runs successfully post-deployment from browser.
-