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

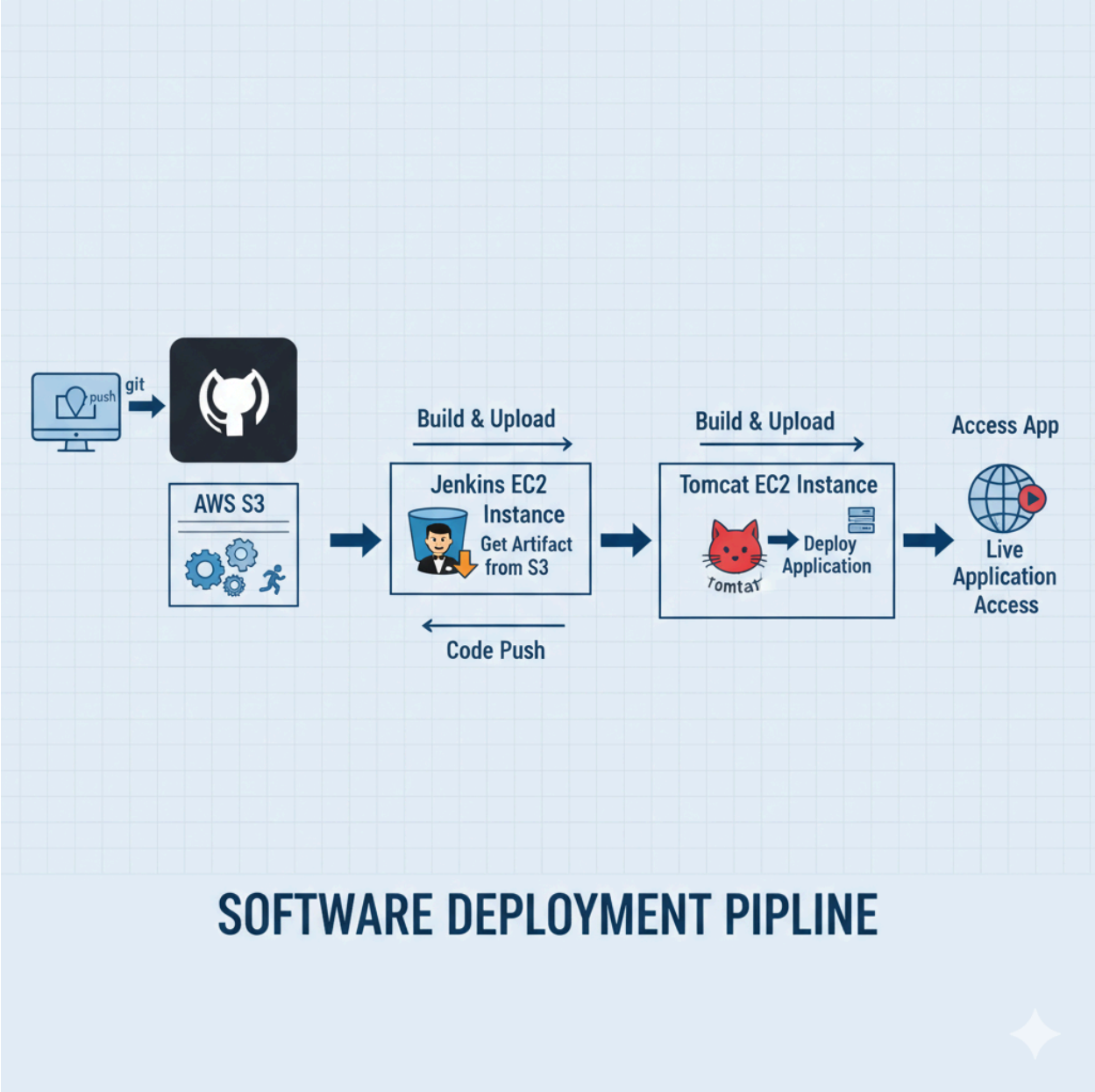
Project Overview

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



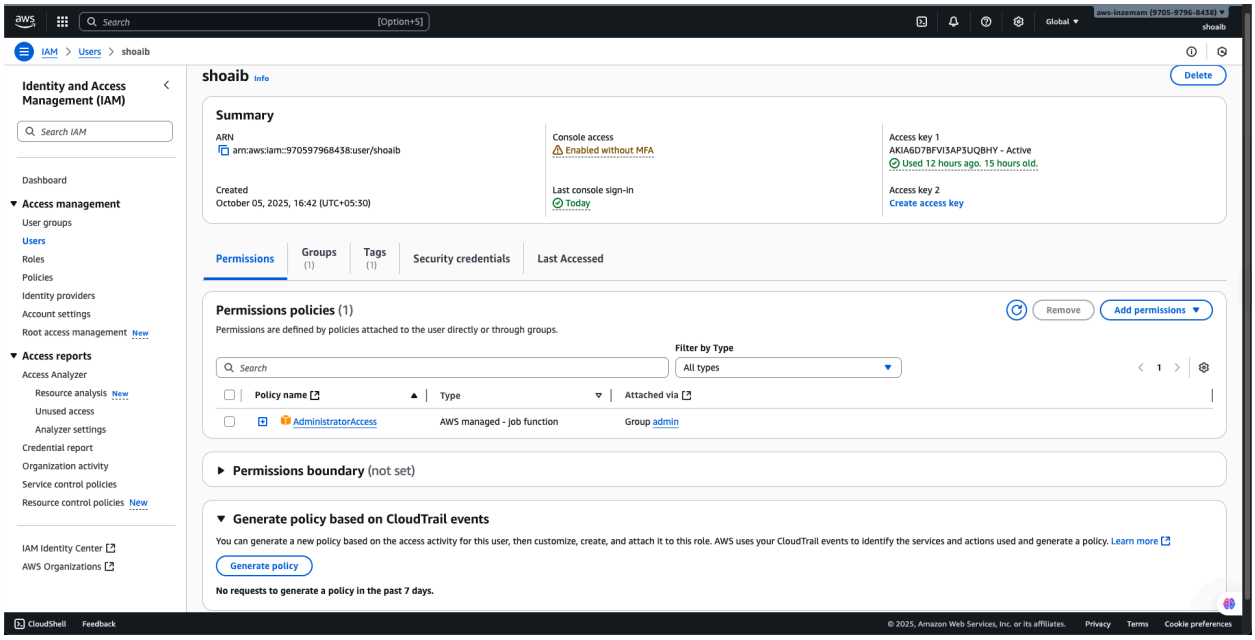
Screenshot: Network diagram showing Jenkins EC2, Tomcat EC2, S3 bucket, GitHub.

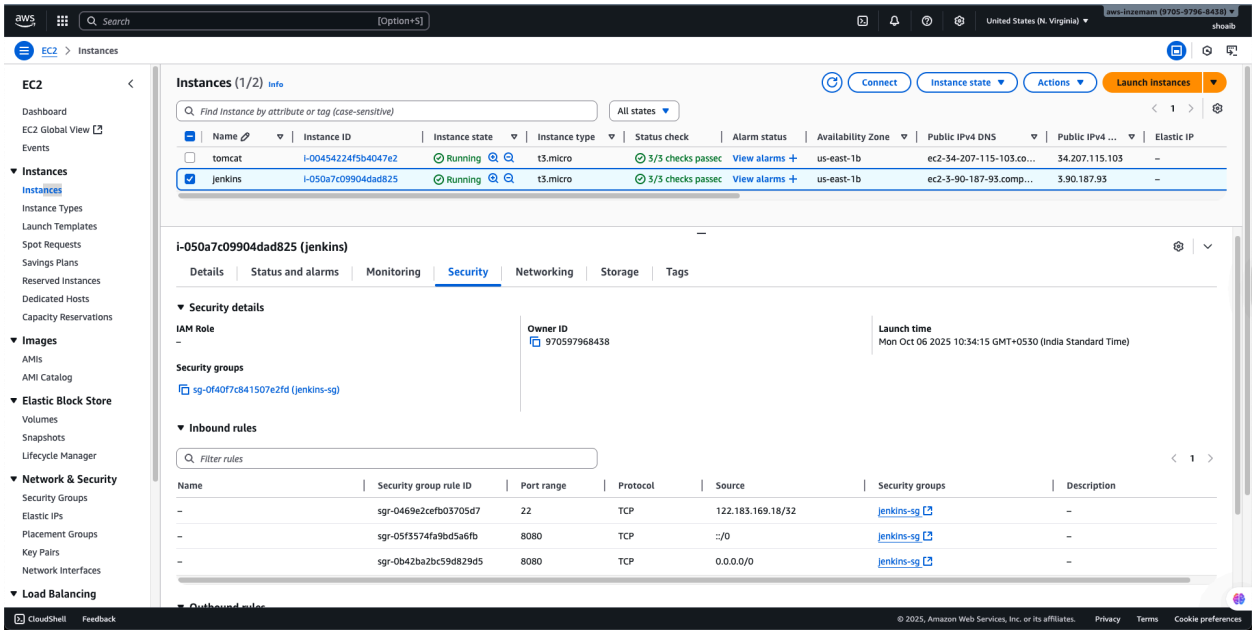
Step-by-Step Setup

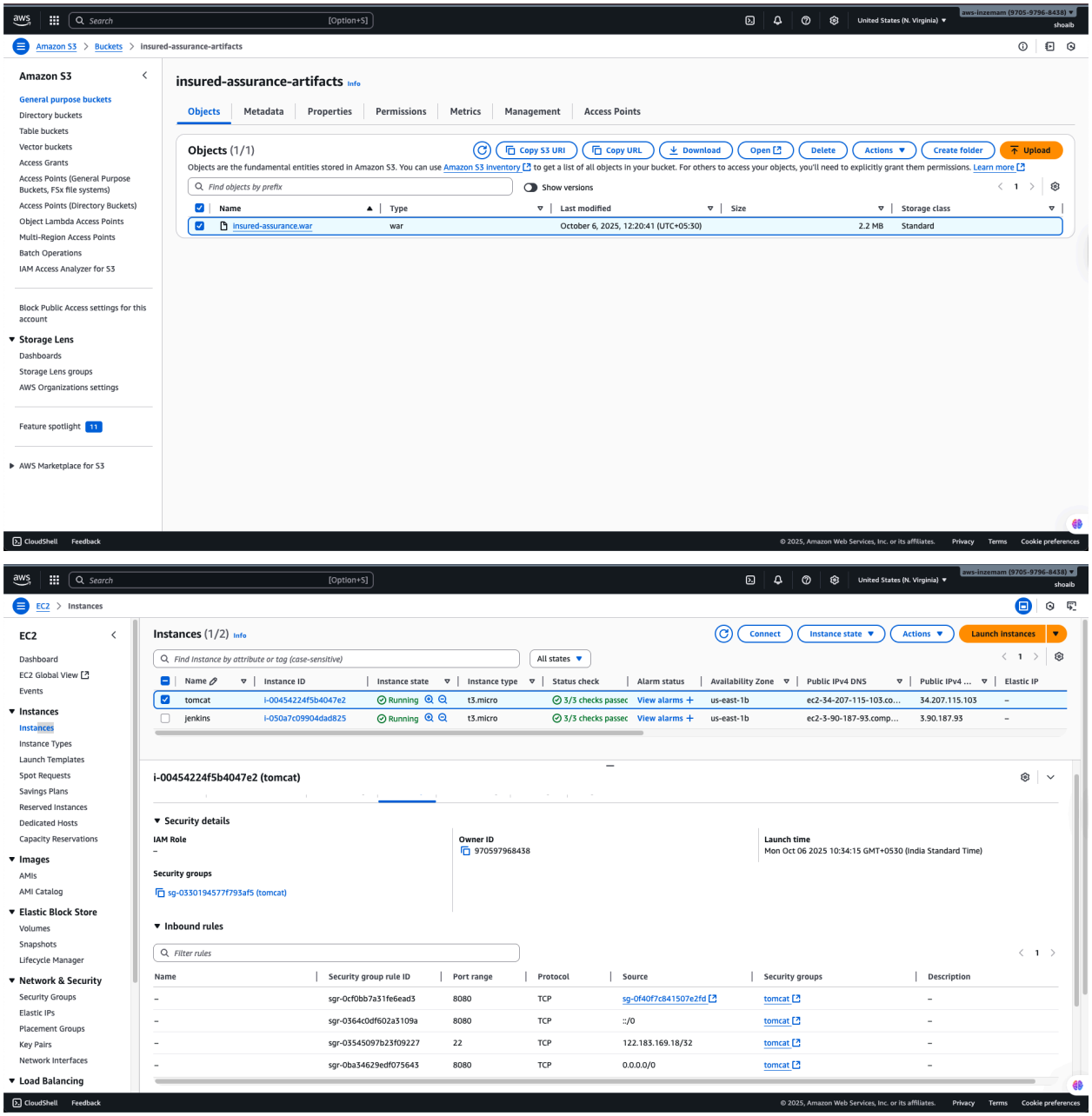
1. AWS Resources

- 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.




Search
[Option+S]
United States (N. Virginia)
aws-inzemam (9705-9796-8438)
shoaib
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Instances (1/2) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elastic IP
tomcat
i-00454224f5b4047e2
Running
t3.micro
3/3 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
Filter rules
Name
Security group rule ID
Port range
Protocol
Source
Security groups
Description
–
sgr-0469e2cefb03705d7
22
TCP
122.183.169.18/32
jenkins-sg
–
–
sgr-05f3574fa9bd5a6fb
8080
TCP
::/0
jenkins-sg
–
–
sgr-0b42ba2bc59d829d5
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



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
```

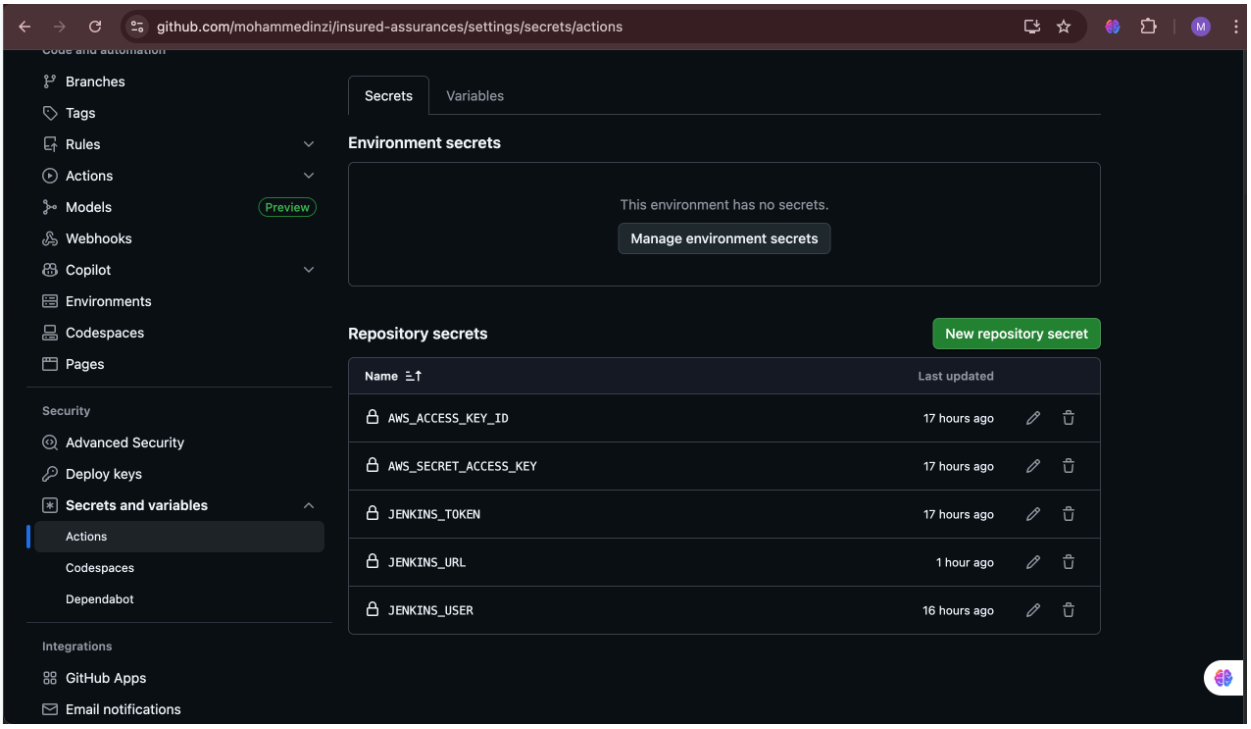
(Alternative: use EC2 console PEM key, SSM, or manual copy as needed)

```
[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

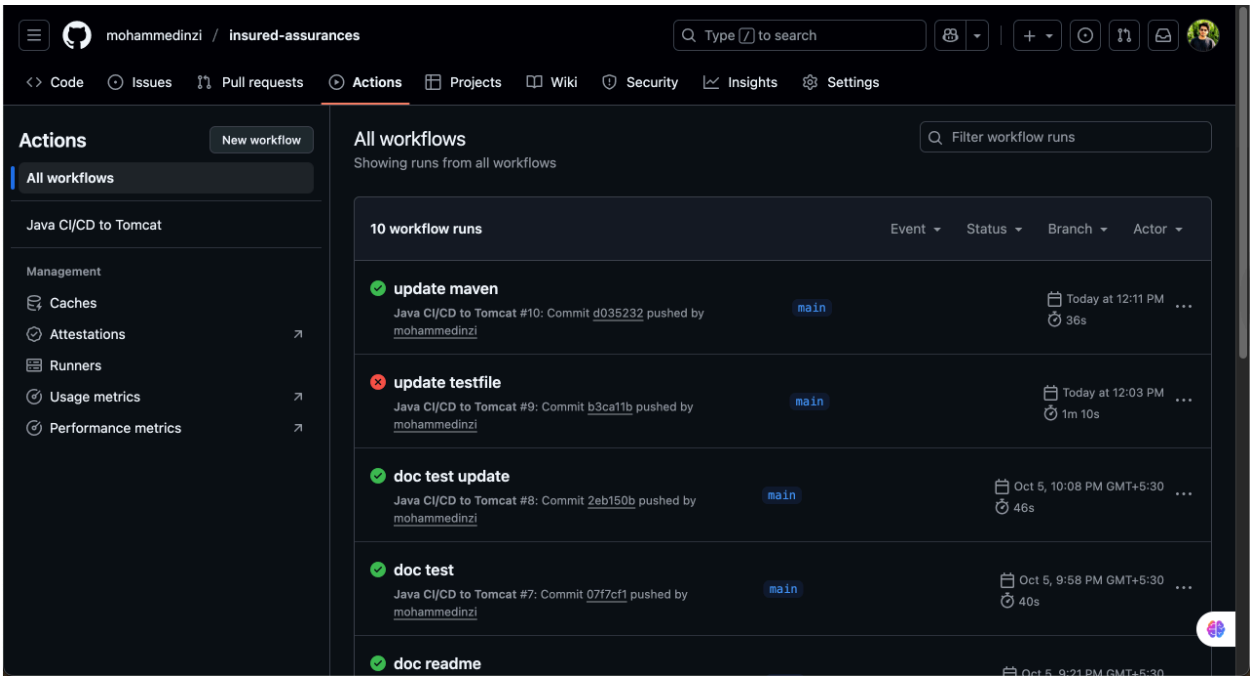
```
name: CI - build war - upload to S3 - trigger Jenkins
on:
  push:
    branches: [main]

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 }}
```



Screenshot: GitHub Actions workflow run log.

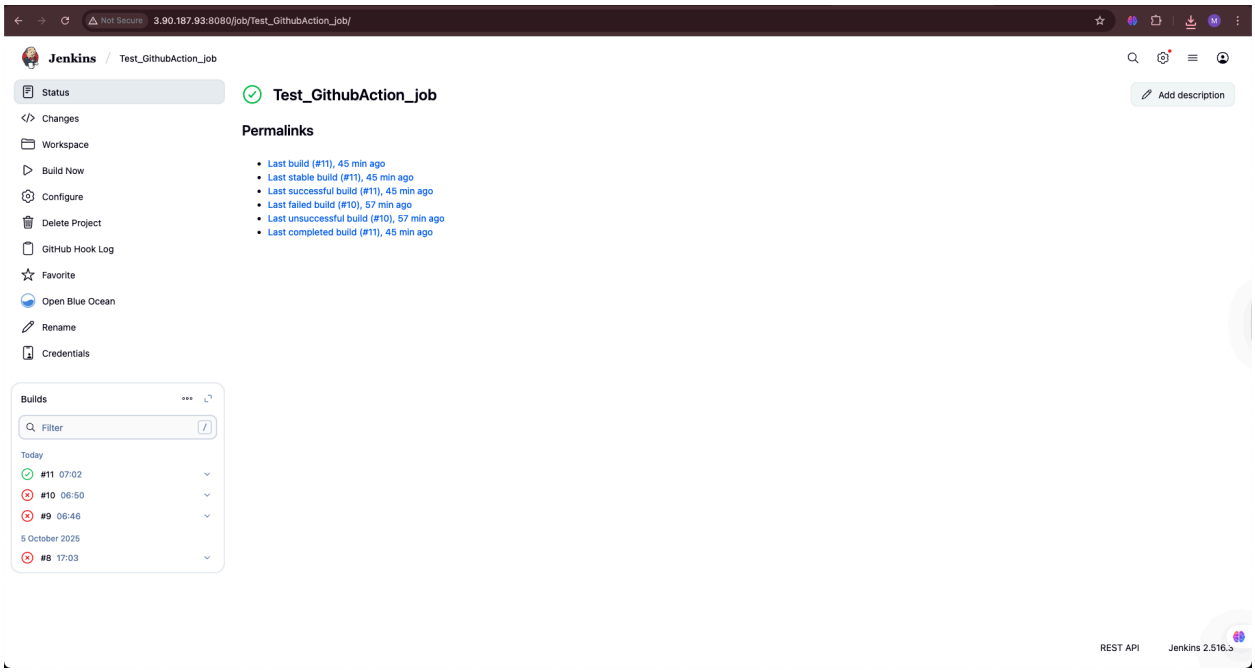
7. Jenkins Job Setup (`TomcatDeployment`)

- 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..."
```

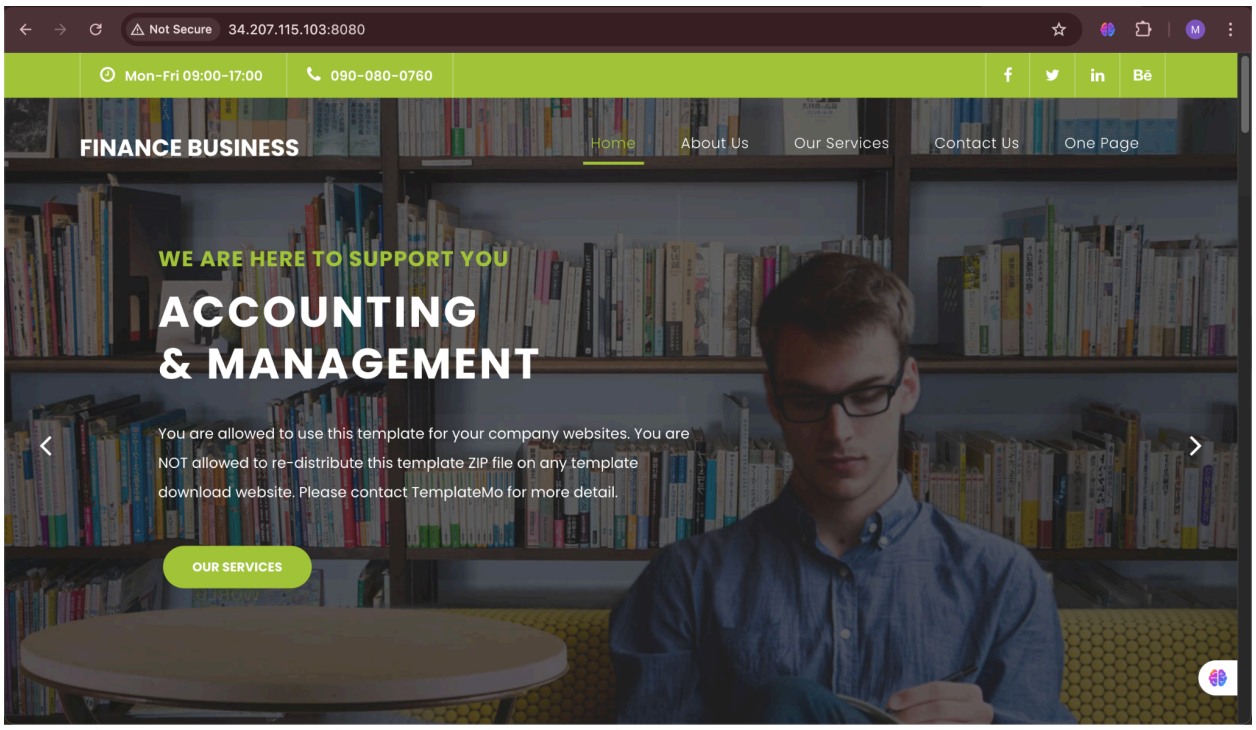
```
scp -i ~/.ssh/tomcat_id_rsa $WORKSPACE/app.war
tomcat@TOMCAT_PUBLIC_IP:/opt/tomcat/webapps/insured-assurance.war
ssh -i ~/.ssh/tomcat_id_rsa tomcat@TOMCAT_PUBLIC_IP "sudo systemctl
restart tomcat"
echo "Deployed to Tomcat done."
```



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.
