
🚀 End-to-End CI/CD Pipeline with GitHub Actions, SonarQube, Nexus, Tomcat & Slack

---

📘 1. Introduction

This guide provides a **complete CI/CD pipeline** for Java-based web applications using **GitHub Actions** as the automation engine.

The workflow integrates:

- **SonarQube** → Code quality analysis
- **Maven** → Build & package WAR files
- **Nexus Repository** → Artifact management
- **Apache Tomcat** → Application deployment
- **Slack** → Real-time deployment notifications

🧭 2. CI/CD Pipeline Flow

Developer Commit → GitHub Actions → SonarQube → Maven Build → Nexus → Tomcat → Slack Notification

🔄 Pipeline Stages

Stage	**Description**	**Tool**
1️⃣ **Checkout Code**	Pull latest code from GitHub	GitHub
2️⃣ **SonarQube Analysis**	Run static code quality checks	SonarQube
3️⃣ **Build Artifact**	Create WAR file using Maven	Maven
4️⃣ **Upload Artifact**	Push WAR to Nexus repo	Nexus
5️⃣ **Deploy to Tomcat**	Force-clean and deploy new WAR	Apache Tomcat
6️⃣ **Verify Deployment**	Check if app is reachable	curl
7️⃣ **Notify Slack**	Send success/failure messages	Slack Webhook

---

⚙️ 3. Infrastructure Requirements

Component	Purpose	Example
Source Repository	Application code	https://github.com/mrtechreddy/Java-Web-Calculator-App
SonarQube Server	Code analysis	http://3.139.87.2:9000
Nexus Repository	Artifact storage	http://18.226.34.227:8081/repository/maven-releases/
Tomcat Server	Application hosting	http://18.216.0.11:8080/manager/text
Slack Channel	CI/CD notifications	#ci-cd-updates

🔐 4. Credentials Setup

Store all credentials securely as **GitHub Secrets** under
Repository → Settings → Secrets → Actions → New Repository Secret.

Secret Name	Description	Example
SONAR_HOST_URL	SonarQube URL	http://3.139.87.2:9000
SONAR_TOKEN	SonarQube Access Token	squ_0cf181e458a690f26fdfdd1d8ac4a12abac9c866
NEXUS_USER	Nexus username	admin
NEXUS_PASS	Nexus password	admin123
TOMCAT_USER	Tomcat Manager username	admin
TOMCAT_PASS	Tomcat Manager password	admin123

Secret Name	Description	Example
SLACK_WEBHOOK_URL	Slack Webhook URL	https://hooks.slack.com/services/T000/B000/XXXX

🧩 5. Repository Folder Structure

```
Java-Web-Calculator-App/
│
├── src/
│   ├── main/
│   │   ├── java/
│   │   └── webapp/
│
├── pom.xml
└── .github/
    └── workflows/
        └── ci-cd.yml   ← GitHub Actions Pipeline
```

🧾 6. GitHub Actions Workflow File

📁 **Path:** .github/workflows/ci-cd.yml

Below is your **final working pipeline YAML**, combining build, deploy, and Slack notifications.

```
name: Java CI/CD - SonarQube, Nexus, Tomcat (Full Clean Deploy + Slack Alerts)


on:
  push:
    branches:
      - main
  workflow_dispatch:
```

```
jobs:
  build-deploy:
    runs-on: ubuntu-latest

    env:
      SONAR_HOST_URL: ${{ secrets.SONAR_HOST_URL }}
      SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
      NEXUS_URL: http://18.226.34.227:8081
      NEXUS_REPO: maven-releases
      NEXUS_GROUP: com/web/cal
      NEXUS_ARTIFACT: webapp-add
      TOMCAT_URL: http://18.216.0.11:8080/manager/text
      SLACK_WEBHOOK_URL: ${{ secrets.SLACK_WEBHOOK_URL }}

    steps:
      - name: 📦 Checkout Code
        uses: actions/checkout@v4

      - name: ☕ Setup Java 17
        uses: actions/setup-java@v4
        with:
          distribution: 'temurin'
          java-version: '17'

      - name: 🧰 Cache Maven Dependencies
        uses: actions/cache@v4
        with:
```

```
    path: ~/.m2
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-

- name: 🔍 Run SonarQube Analysis
  run: |
    mvn clean verify sonar:sonar \
      -DskipTests \
      -Dsonar.projectKey=JavaWebCalculator \
      -Dsonar.host.url=${{ secrets.SONAR_HOST_URL }} \
      -Dsonar.login=${{ secrets.SONAR_TOKEN }}

- name: ⚙️ Build WAR File
  run: |
    mvn clean package -DskipTests
    echo "✅ Build completed!"
    ls -lh target/*.war

- name: ⬆️ Upload WAR to Nexus
  env:
    NEXUS_USER: ${{ secrets.NEXUS_USER }}
    NEXUS_PASS: ${{ secrets.NEXUS_PASS }}
  run: |
    set -e
    WAR_FILE=$(ls target/*.war | head -1)
    VERSION="0.0.${{ github.run_number }}"
    echo "📦 Uploading WAR to Nexus version $VERSION ..."
```

```
          curl -u ${NEXUS_USER}:${NEXUS_PASS} --upload-file "$WAR_FILE" \
            "${{ env.NEXUS_URL }}/repository/${{ env.NEXUS_REPO }}/${{ env.NEXUS_GROUP }}/${{ env.NEXUS_ARTIFACT }}/${VERSION}/${{ env.NEXUS_ARTIFACT }}-${VERSION}.war"
          echo "✅ Uploaded successfully."

      - name: 🚀 Deploy WAR to Tomcat (Force Clean)
        env:
          TOMCAT_USER: ${{ secrets.TOMCAT_USER }}
          TOMCAT_PASS: ${{ secrets.TOMCAT_PASS }}
          NEXUS_USER: ${{ secrets.NEXUS_USER }}
          NEXUS_PASS: ${{ secrets.NEXUS_PASS }}
        run: |
          set -e
          APP_NAME="webapp-add"
          cd /tmp; rm -f *.war

          echo "🔍 Fetching latest WAR from Nexus..."
          DOWNLOAD_URL=$(curl -s -u ${NEXUS_USER}:${NEXUS_PASS} \
            "${{ env.NEXUS_URL }}/service/rest/v1/search/assets?repository=${{ env.NEXUS_REPO }}" \
            | grep -oP '"downloadUrl"\s*:\s*"[^"]*webapp-add-[0-9.]+\.war' | tail -1 | cut -d'"' -f4)

          if [ -z "$DOWNLOAD_URL" ]; then
            echo "❌ No WAR found in Nexus!"
            exit 1
          fi
          echo "⬇️ Downloading WAR: $DOWNLOAD_URL"
          curl -u ${NEXUS_USER}:${NEXUS_PASS} -O "$DOWNLOAD_URL"
          WAR_FILE=$(basename "$DOWNLOAD_URL")

          echo "🧹 Undeploying old application..."
          curl -u ${TOMCAT_USER}:${TOMCAT_PASS} "${{ env.TOMCAT_URL
}}/undeploy?path=/${APP_NAME}" || true
          sleep 5

          echo "🧼 Cleaning Tomcat cache directories..."
          curl -u ${TOMCAT_USER}:${TOMCAT_PASS} "${{ env.TOMCAT_URL
}}/expire?path=/${APP_NAME}" || true
          curl -u ${TOMCAT_USER}:${TOMCAT_PASS} "${{ env.TOMCAT_URL
}}/reload?path=/${APP_NAME}" || true

          echo "🚀 Deploying fresh WAR to Tomcat..."
          curl -u ${TOMCAT_USER}:${TOMCAT_PASS} --upload-file "$WAR_FILE" \
            "${{ env.TOMCAT_URL }}/deploy?path=/${APP_NAME}&update=true"

          echo "✅ Deployment completed successfully!"

      - name: 🔎 Verify Deployment
        id: verify
        run: |
          sleep 15
          STATUS=$(curl -o /dev/null -s -w "%{http_code}"
http://18.216.0.11:8080/webapp-add/)
          if [ "$STATUS" -eq 200 ]; then
            echo "✅ App is live with the latest code!"
          else
            echo "⚠️ App returned status $STATUS"
            exit 1
          fi

      - name: ✅ Notify Slack (Success)
        if: success()
        run: |
          curl -X POST -H 'Content-type: application/json' \
          --data "{
            \"text\": \":white_check_mark: *Deployment Successful!*\n
            *Repository:* ${GITHUB_REPOSITORY}\n
            *Branch:* ${GITHUB_REF_NAME}\n
            *Commit:*
<https://github.com/${GITHUB_REPOSITORY}/commit/${GITHUB_SHA}|${GITHUB
_SHA:0:7}>\n
            *Run:*
<${GITHUB_SERVER_URL}/${GITHUB_REPOSITORY}/actions/runs/${GITHUB_R
UN_ID}|View Workflow>\n
            *App URL:* http://18.216.0.11:8080/webapp-add/\"
          }" \
          ${{ secrets.SLACK_WEBHOOK_URL }}

      - name: ❌ Notify Slack (Failure)
        if: failure()
        run: |
          curl -X POST -H 'Content-type: application/json' \
          --data "{
            \"text\": \":x: *Deployment Failed!*\n
```

```
            *Repository:* ${GITHUB_REPOSITORY}\n
            *Branch:* ${GITHUB_REF_NAME}\n
            *Commit:* <https://github.com/${GITHUB_REPOSITORY}/commit/${GITHUB_SHA}|${GITHUB_SHA:0:7}>\n
            *Run:* <${GITHUB_SERVER_URL}/${GITHUB_REPOSITORY}/actions/runs/${GITHUB_RUN_ID}|View Workflow>\"
          }" \
          ${{ secrets.SLACK_WEBHOOK_URL }}
```

📡 7. Slack Integration Setup

Step 1: Create a Slack App

- Visit https://api.slack.com/apps
- Click **Create New App → From Scratch**
- Choose your workspace
- Enable **Incoming Webhooks**
- Add new webhook to a desired channel
- Copy the Webhook URL

Step 2: Add to GitHub Secrets

In repository secrets:

SLACK_WEBHOOK_URL=https://hooks.slack.com/services/XXX/YYY/ZZZ

Step 3: Test Message

When a deployment finishes, you'll see messages like:

✅ Success

:white_check_mark: Deployment Successful!

Repository: mrtechreddy/Java-Web-Calculator-App

Branch: main

Commit: 5c17d8a

App URL: http://18.216.0.11:8080/webapp-add/

❌ Failure

:x: Deployment Failed!

Repository: mrtechreddy/Java-Web-Calculator-App

Branch: main

Commit: 5c17d8a

🧠 8. Triggering the Pipeline

There are two ways:

1. **Automatic:** On each push to main.
2. **Manual:**
   - Navigate to Actions → Java CI/CD - SonarQube, Nexus, Tomcat (Full Clean Deploy + Slack Alerts)
   - Click **"Run workflow"**

✅ 9. Verification Checklist

Check	Expected Result
Build	WAR generated under /target
SonarQube	Analysis report visible
Nexus	New version stored
Tomcat	New WAR deployed
App URL	HTTP 200 OK
Slack	Notification sent

⚠️ 10. Troubleshooting

Issue	Cause	Solution
Old app still loads	Tomcat cache not cleared	Added /undeploy and /reload

Issue	Cause	Solution
401 Unauthorized (Tomcat)	Manager credentials incorrect	Update tomcat-users.xml
Slack not sending	Webhook invalid	Regenerate Slack webhook
SonarQube step fails	Server not reachable	Check host and token
Nexus upload 400	Duplicate artifact version	Increment version automatically

---

📊 11. Improvements & Add-Ons

Feature	Description
SonarQube Quality Gate	Add check to fail pipeline if quality < 80%
Rollback Feature	Deploy previous version from Nexus
Multiple Environments	Add staging/production workflows
Slack Formatting	Include emoji, colors, or build duration
Parallel Jobs	Split build, test, and deploy stages

🧾 12. Summary

Component	Role
GitHub Actions	CI/CD engine
Maven	Build automation
SonarQube	Code quality
Nexus	Artifact repository
Tomcat	Deployment server
Slack	Notifications

🧱 13. Expected Output

Once the workflow runs successfully:

1. GitHub shows ✅ green status.
2. New artifact appears in Nexus.
3. Tomcat hosts the latest build.
4. Slack sends a confirmation message.
5. Browser shows updated code in app.
