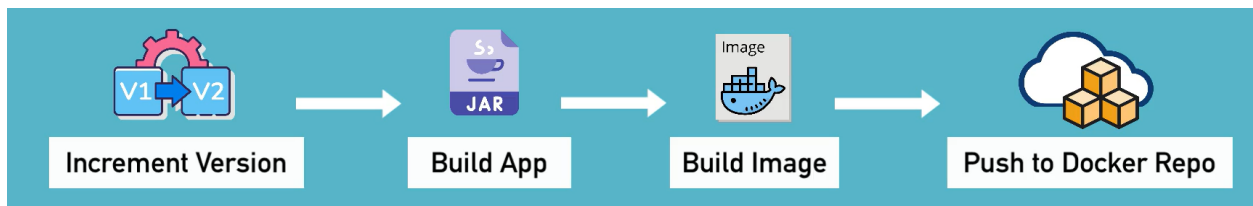


# Demo Project: Dynamically Increment Application version in Jenkins Pipeline

This project demonstrates how to:

1. Increment the application version dynamically in a Java Maven project.
2. Use the updated version in Docker image tagging.
3. Push the Docker image to DockerHub.
4. Commit the updated version back to the GitLab repository.
5. Avoid commit loops caused by Jenkins triggers.



## 1. Increment Application version on Jenkins with JAVA MAVEN Project

Why this is needed:

## 2. Docker Image Version

## 3. Update Dockerfile

Verification:

## 4. Commit version bump from Jenkins to Repository

Common Issue: Authentication Error

## 5. Avoid Commit Loops

Solution: Ignore Jenkins Commits

## 6. Verify the Setup

# 1. Increment Application version on Jenkins with JAVA MAVEN Project

To dynamically increment the application version before building, modify the **Jenkinsfile** as follows:

```
pipeline {
  agent any
  tools {
    maven 'maven-3.9'
  }
  stages {
    stage('increment version') {
      steps {
        script {
          echo 'incrementing app version...'
          sh 'mvn build-helper:parse-version versions:set \
            -DnewVersion=\\${parsedVersion.majorVersion}.\\${parsedVersion.minorVersion}.\\${parsedVersion.nextIncrementalVersion} \
              versions:commit'

          def matcher = readFile('pom.xml') =~ '<version>(.)</version>'
          def version = matcher[0][1]

          env.IMAGE_NAME = "$version-$BUILD_NUMBER"
        }
      }
    }
    stage('build app') {
```

## Why this is needed:

- Incrementing the version ensures that the generated JAR file reflects the updated version for every build.

## 2. Docker Image Version

To use the updated version in Docker image tagging:

1. Update the `increment version` stage to extract the version and set a Docker image tag:

```

pipeline {
  agent any
  tools {
    maven 'maven-3.9'
  }
  stages {
    stage('increment version') {
      steps {
        script {
          echo 'incrementing app version...'
          sh 'mvn build-helper:parse-version versions:set \
            -
            DnewVersion=\\${parsedVersion.majorVersion}.\\${parsedVersion.minorVersion}.\\${parsedVersion.nextIncrementalVersion} \
            versions:commit'
        }
      }
    }
  }
}

```

2. Add the `build image` stage:

```

stage("build image") {
  steps {
    script {
      echo "building the docker image..."
      withCredentials([usernamePassword(credentialsId: 'docker-hub-repo',
passwordVariable: 'PASS', usernameVariable: 'USER')) {
        sh "docker build -t eduardobautistamaciell/demo-app:${IMAGE_NAME} ."
        sh 'echo $PASS | docker login -u $USER --password-stdin'
        sh "docker push eduardobautistamaciell/demo-app:${IMAGE_NAME}"
      }
    }
  }
}

```

## 3. Update Dockerfile

1. Ensure the **Dockerfile** dynamically picks the JAR file from the target folder:

```

FROM amazoncorretto:8-alpine3.17-jre

EXPOSE 8080

COPY ./target/java-maven-app-*.jar /usr/app/

WORKDIR /usr/app

CMD java -jar java-maven-app-*.jar

```

## Verification:

- Test the pipeline and check Jenkins logs to confirm the incremented version.
- Verify the pushed Docker image in DockerHub.

## 4. Commit version bump from Jenkins to Repository

To avoid starting each build with the original version, commit the updated version back to the repository:

```
stage('commit version update'){
    steps {
        script {
            withCredentials([usernamePassword(credentialsId: 'gitlab-credentials',
passwordVariable: 'PASS', usernameVariable: 'USER')]){
                sh 'git config --global user.email "jenkins@example.com"'
                sh 'git config --global user.name "jenkins"'
                sh 'git status'
                sh 'git branch -a'
                sh 'git branch'
                sh 'git config --list'
                sh "git remote set-url origin https://${USER}:${PASS}@gitlab.com/twn-devops-
projects/jenkins/java-maven-app.git"
                sh 'git add .'
                sh 'git commit -m "ci: version bump"'
                sh 'git remote -v'
                sh 'git push origin HEAD:jenkins-jobs'
            }
        }
    }
}
```

## Common Issue: Authentication Error

### Error:

```
fatal: unable to access 'https://@gitlab.com/twn-devops-projects/jenkins/java-maven-
app.git/': URL using bad/illegal format or missing URL
```

### Root Cause:

The @ character in the password causes parsing issues during authentication.

### Solution:

- URL-encode the password:

```
def encodedPassword = URLEncoder.encode(PASS, 'UTF-8')
```

```
sh "git remote set-url origin <https://${USER}:${encodedPassword}@gitlab.com/><your-group>/<your-repo>.git"
```

This ensures that special characters in the password are properly handled.

```
stage('commit version update'){
    steps {
        script {
            withCredentials([usernamePassword(credentialsId: 'gitlab-credentials',
passwordVariable: 'PASS', usernameVariable: 'USER')]){
                def encodedPassword = URLEncoder.encode(PASS, 'UTF-8')
                sh 'git config --global user.email "jenkins@example.com"'
                sh 'git config --global user.name "jenkins"'
                sh 'git status'
                sh 'git branch -a'
                sh 'git branch'
                sh 'git config --list'
                sh "git remote set-url origin
https://${USER}:${encodedPassword}@gitlab.com/twn-devops-projects/jenkins/java-maven-app.git"
                sh 'git add .'
                sh 'git commit -m "ci: version bump"'
                sh 'git remote -v'
                sh 'git push origin HEAD:jenkins-jobs'
            }
        }
    }
}
```

## 5. Avoid Commit Loops

When Jenkins commits version updates, webhooks may trigger new builds, causing an endless loop.

### Solution: Ignore Jenkins Commits



#### 1. Install the Plugin:

- Go to **Manage Jenkins** → **Plugins** and install **Ignore Committer Strategy**.

#### 2. Configure the Multibranch Pipeline:

- Navigate to **Pipeline Configuration** → **Build Strategies**.
- Add the **Ignore Committer Strategy** and enter the Jenkins email:

```
jenkins@example.com .
```

 **Ignore Committer Strategy** ? 

List of author emails to ignore ?

☒ Allow builds when a changeset contains non-ignored author(s) ?

## 6. Verify the Setup

### 1. Pipeline Logs:

- Ensure that the application version is incremented and pushed.

Stage View

	Declarative: Checkout SCM	Declarative: Tool Install	increment version	build app	build image	deploy	commit version update
Average stage times: (Average full run time: ~30s)	1s	99ms	6s	8s	6s	275ms	5s
#22 Dec 29 06:15 2 commits	1s	86ms	6s	8s	5s	213ms	4s
#21 Dec 29 06:12 2 commits	1s	105ms	5s	8s	5s	197ms	5s
#20 Dec 29 05:56 1 commit	1s	70ms	5s	9s	6s	363ms	7s

- Verify the Docker image in DockerHub.



```
pom.xml x
pom.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5      <modelVersion>4.0.0</modelVersion>
6
7      <groupId>com.example</groupId>
8      <artifactId>java-maven-app</artifactId>
9      <version>1.1.3</version>
10
11     <build>
12         <plugins>
```