Lab 3: Create Jenkins scripted pipeline for Java (Maven build & test)

Full Jenkins Scripted Pipeline for Forex Converter App

GITHUB reference: GitHub - cloud-dev-user/forex-app

Step 1 — Project Structure

```
forex-app/
 ├── Dockerfile
 ├── pom.xml
 ├── rates.csv
 └── src/
     └── main/java/com/example/forex/
         └── ForexConverter.java
     └── test/java/com/example/forex/
         └── ForexConverterTest.java

```

Step 2 — `rates.csv` (Dynamic Forex Rates)

```
currency,rate
USD,0.012
EUR,0.011
AED,0.044
GBP,0.0095
JPY,1.60

```

Step 3 — Java Code

`ForexConverter.java`

```
package com.example.forex;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class ForexConverter {

    private final Map<String, Double> rates = new HashMap<>();

    public ForexConverter(String csvFile) throws IOException {
        try (BufferedReader br = new BufferedReader(new
FileReader(csvFile))) {
            String line;
            br.readLine(); // skip header
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",");
                rates.put(parts[0], Double.parseDouble(parts[1]));
            }
        }
    }

    public double convertFromINR(double amount, String currency) {
```

```
        if (!rates.containsKey(currency)) {
            throw new IllegalArgumentException("Unsupported currency: " + currency);
        }
        return amount * rates.get(currency);
    }

    public double convertToINR(double amount, String currency) {
        if (!rates.containsKey(currency)) {
            throw new IllegalArgumentException("Unsupported currency: " + currency);
        }
        return amount / rates.get(currency);
    }

    public static void main(String[] args) throws Exception {
        if (args.length < 3) {
            System.out.println("Usage: java ForexConverter <csv-file> <amount> <currency>");
            return;
        }

        String csvFile = args[0];
        double amount = Double.parseDouble(args[1]);
        String currency = args[2];

        ForexConverter fx = new ForexConverter(csvFile);

        System.out.println(amount + " INR in " + currency + ": " +
                fx.convertFromINR(amount, currency));
        System.out.println("100 " + currency + " in INR: " +
                fx.convertToINR(100, currency));
    }
}

```

`ForexConverterTest.java`

```
package com.example.forex;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import java.io.IOException;

public class ForexConverterTest {

    @Test
    public void testConvertFromINR() throws IOException {
        ForexConverter fx = new ForexConverter("rates.csv");
        double usd = fx.convertFromINR(1000, "USD");
        assertEquals(12.0, usd, 0.5);
    }

    @Test
    public void testConvertToINR() throws IOException {
        ForexConverter fx = new ForexConverter("rates.csv");
        double inr = fx.convertToINR(12, "USD");
        assertEquals(1000.0, inr, 50.0);
    }

    @Test
    public void testUnsupportedCurrency() throws IOException {
        ForexConverter fx = new ForexConverter("rates.csv");
        assertThrows(IllegalArgumentException.class, () -> fx.convertFromINR(1000, "XYZ"));
    }
}

```

Step 4 — Add `pom.xml`

Inside `pom.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>forex-app</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>
    <name>Forex App</name>
    <description>A simple Forex application</description>
    <url>https://github.com/cloud-dev-user/forex-app</url>

    <properties>
        <java.version>17</java.version>
        <maven.compiler.source>${java.version}
</maven.compiler.source>
        <maven.compiler.target>${java.version}
</maven.compiler.target>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <!-- Spring Boot Starter (Remove if not a Spring Boot
project) -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
            <version>3.2.5</version>
        </dependency>

        <!-- Spring Boot Starter Test (for unit testing) -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <version>3.2.5</version>
            <scope>test</scope>
        </dependency>

        <!-- JUnit Jupiter (for unit testing) -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.2</version>
            <scope>test</scope>
        </dependency>

        <!-- Add your other dependencies here -->
    </dependencies>

    <build>
        <plugins>
            <!-- Compiler plugin -->
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
                    <archive>
                       <manifest>

<mainClass>com.example.forex.ForexConverter</mainClass>
```

```
                    </manifest>
                </archive>
                <source>${java.version}</source>
                <target>${java.version}</target>
            </configuration>
        </plugin>

        <!-- Jacoco plugin (as provided) -->
        <plugin>
            <groupId>org.jacoco</groupId>
            <artifactId>jacoco-maven-plugin</artifactId>
            <version>0.8.8</version>
            <executions>
                <execution>
                    <goals>
                        <goal>prepare-agent</goal>
                    </goals>
                </execution>
                <execution>
                    <id>report</id>
                    <phase>test</phase>
                    <goals>
                        <goal>report</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>

        <!-- Spring Boot Maven plugin (Remove if not a Spring Boot project) -->
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>3.2.5</version>
        </plugin>
    </plugins>
</build>

<repositories>
    <repository>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2/</url>
    </repository>
</repositories>

</project>

```

Step 5 — Dockerfile

```
FROM openjdk:21-jdk-slim-buster
WORKDIR /app
COPY target/forex-app-1.0-SNAPSHOT.jar app.jar
COPY rates.csv rates.csv
ENTRYPOINT ["java", "-cp", "app.jar", "com.example.forex.ForexConverter", "rates.csv"]

```

Step 6 — Jenkins Scripted Pipeline

`Jenkinsfile`

```
node {
    // Parameters for user input
    properties([
        parameters([
```

```
            string(name: 'AMOUNT', defaultValue: '1000', description: 'Amount in INR'),
            choice(name: 'CURRENCY', choices: ['USD', 'EUR', 'AED', 'GBP', 'JPY'], description: 'Target currency')
        ])
    ])

    stage('Checkout') {
        checkout scm
    }

    stage('Build') {
        sh 'mvn clean compile'
    }

    stage('Test & Coverage') {
        sh 'mvn test jacoco:report'
        junit 'target/surefire-reports/*.xml'
        publishHTML(target: [
            reportDir: 'target/site/jacoco',
            reportFiles: 'index.html',
            reportName: 'Jacoco Coverage Report'
        ])
    }

    stage('Package') {
        sh 'mvn package -DskipTests'
        archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
    }

    stage('Run Forex App') {
        sh "java -cp target/forex-app-1.0-SNAPSHOT.jar com.example.forex.ForexConverter rates.csv ${params.AMOUNT} ${params.CURRENCY}"
    }

    stage('Docker Build & Run') {
        sh '''
          docker build -t forex-app .
          docker run --rm forex-app ${AMOUNT} ${CURRENCY}
        '''
    }
}

```

Step 7 — Validation

1. Run the pipeline with **different params** (`AMOUNT=5000` , `CURRENCY=EUR`).
2. Check **Jacoco coverage report** in Jenkins.
3. Verify **Docker image** is built & container runs.
4. Ensure artifacts (`.jar`) are archived.

✅ Key Learnings

- Reading dynamic forex rates from CSV.
- Parameterized Jenkins scripted pipeline.
- Integrating **unit tests + coverage reports**.
- Dockerizing Java app for deployment.
- Full CI/CD workflow from code → test → package → container.