Lab 6: Break pipeline (missing dependency, wrong stage) → Debug & fix

Step 1 — Project Context

We continue using the **Forex Converter multi-module Maven app** (core + service modules).

Pipeline stages:

- Checkout
- Build
- Unit Test
- Integration Test
- Package
- Deploy

Step 2 — The Broken Jenkinsfile

```
node {
    stage('Checkout') {
        checkout scm
    }

    stage('Build') {
        // ❌ Error 1: Wrong JDK version (forces compilation error)
        withEnv(["JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64", "PATH+JAVA=$JAVA_HOME/bin"]) {
            sh 'mvn clean install -DskipTests'
        }
    }

    stage('Unit Test') {
        // ❌ Error 2: Flaky test execution (parallel without config)
        sh 'mvn test -T 2'
        junit '**/target/surefire-reports/*.xml'
    }

    stage('Integration Test') {
        // ❌ Error 3: Wrong Maven profile
        sh 'mvn verify -Pintegration-test'
    }

    stage('Package') {
        // ❌ Error 4: Wrong artifact path
        archiveArtifacts artifacts: '**/build/*.jar', fingerprint: true
    }

    stage('Deploy') {
        // ❌ Error 5: No repo config in pom.xml
        sh 'mvn deploy'
    }

    stage('Optional Chaos') {
        // ❌ Error 6: Delete target folder mid-build to simulate workspace corruption
        sh 'rm -rf target || true'
    }
}

```

Step 3 — What Will Break (Realistic Failures)

1. **Build fails** → `invalid source release: 11` (wrong JDK).
2. **Unit test flaky** → random test failures due to parallel execution.
3. **Integration test skipped** → wrong profile name.
4. **Package fails** → no JAR found in `build/`.
5. **Deploy fails** → `Could not find server settings`.
6. **Workspace corrupted** → `target/classes not found`.
7. **Circular dependency (optional)** → manually add one between modules → `dependency cycle detected`.
8. **Failing test (optional)** → modify `ForexConverterTest` to assert wrong result → pipeline red.
9. **Network issue simulation (optional)** → clear `.m2/repository` → Maven download errors.
10. **Coverage enforcement (optional)** → add JaCoCo → pipeline fails if coverage < 80%.

Step 4 — Debugging Process (Step by Step)

🔧 Fix 1: Wrong JDK

Switch JAVA_HOME to JDK 11:

```
withEnv(["JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64", "PATH+JAVA=$JAVA_HOME/bin"]) {
    sh 'mvn clean install -DskipTests'
}

```

🔧 Fix 2: Flaky Unit Tests

Add Surefire plugin config in `pom.xml`:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>3.0.0</version>
  <configuration>
    <parallel>classes</parallel>
    <threadCount>2</threadCount>
    <useUnlimitedThreads>false</useUnlimitedThreads>
  </configuration>
</plugin>

```

🔧 Fix 3: Wrong Profile

Correct `-P integration-tests` (plural).

🔧 Fix 4: Artifact Path

Change to `**/target/*.jar`.

🔧 Fix 5: Deploy Repo

Add local repo in `pom.xml`:

```xml
<distributionManagement>
  <repository>
    <id>local-repo</id>
    <url>file://${project.basedir}/local-repo</url>
  </repository>
</distributionManagement>
```

🔧 Fix 6: Workspace Corruption

Add cleanup stage at start:

```groovy
stage('Cleanup') {
    deleteDir()
}
```

🔧 Fix 7: Circular Dependency

Remove offending dependency from one module’s POM.

🔧 Fix 8: Failing Test

Fix assertion in `ForexConverterTest`.

🔧 Fix 9: Network Issue

Rebuild Maven repo:

```
mvn dependency:go-offline
```

🔧 Fix 10: Coverage Enforcement

Add JaCoCo plugin:

```xml
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.8</version>
  <executions>
    <execution>
      <goals><goal>check</goal></goals>
      <configuration>
        <rules>
          <rule>
            <element>BUNDLE</element>
            <limits>
              <limit>
                <counter>INSTRUCTION</counter>
                <value>COVEREDRATIO</value>
                <minimum>0.80</minimum>
              </limit>
            </limits>
          </rule>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>

```

Step 5 — Corrected Jenkinsfile (Final)

```
node {
    stage('Cleanup') {
        deleteDir()
    }

    stage('Checkout') {
        checkout scm
    }

    stage('Build') {
        withEnv(["JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64", "PATH+JAVA=$JAVA_HOME/bin"]) {
            sh 'mvn clean install -DskipTests'
        }
    }

    stage('Unit Test') {
        sh 'mvn test'
        junit '**/target/surefire-reports/*.xml'
    }

    stage('Integration Test') {
        sh 'mvn verify -P integration-tests'
    }

    stage('Package') {
        archiveArtifacts artifacts: '**/target/*.jar', fingerprint: true
    }

    stage('Deploy') {
        sh 'mvn deploy -DskipTests'
    }
}

```

✅ Learning Outcome

- Debug **complex CI/CD failures** (not just typos).
- Handle **flaky builds, env mismatches, missing deps, test failures, repo config errors**.
- Build **real production troubleshooting mindset**.

Jenkins Pipeline Troubleshooting Matrix

Error / Issue	Symptom (What You See in Jenkins Logs)	Root Cause (Why It Happens)	Fix (How to Resolve)	Prevention (Best Practice)
Wrong JDK version	`invalid source`	Jenkins using JDK 8, code	Set `JAVA_HOM`	Always define JDK tool in

	`release: 11` / `Unsupported class file major version`	requires JDK 11+	`E` to JDK 11 in pipeline	Jenkins → `Manage Jenkins → Global Tool Config`
Flaky Unit Tests	Tests fail randomly, sometimes pass	Parallel test execution without Surefire config	Add `<parallel>` config to `maven-surefire-plugin`	Use thread-safe parallelism config; avoid shared state in tests
Wrong Maven Profile	`Unknown lifecycle phase 'integration-test'`	Typo: profile `integration-tests` misnamed as `integration-test`	Use correct profile name `-P integration-tests`	Document and standardize profile names
Wrong Artifact Path	`No artifacts found matching pattern`	Archiving from `build/` instead of `target/`	Change to `**/target/*.jar`	Always verify build output path (`mvn package`)

Deploy Failure	`Could not find server settings` / `Failed to deploy artifacts`	No `<distributionManagement>` config in POM	Add repo config pointing to local or Nexus repo	Maintain Maven `settings.xml` + CI repo configs
Workspace Corruption	`target/classes not found` / build fails mid-way	Target folder deleted during pipeline	Add `stage('Cleanup') { deleteDir() }` at start	Always clean workspace before builds
Circular Dependency	`dependency cycle detected`	Module A depends on Module B, and B depends on A	Remove circular reference in one POM	Run `mvn dependency:tree` to catch cycles early
Failing Test Case	`Expected 80 but got 75`	Wrong test assertion	Fix test logic	Code review + pre-merge test runs
Network Issue	`Could not resolve artifact …`	`.m2/repository` wiped or network unstable	Run `mvn dependency:go-offline` before build	Cache Maven repo in Jenkins agents
Coverage Failure	`Rule violated: COVEREDRATIO < 0.80`	JaCoCo threshold not met	Improve tests or lower threshold temporarily	Enforce 80%+ coverage standard in CI/CD