Ah, now I understand your situation more clearly — and this is a **very insightful question**. Let's break it down:

**🧩 What Happened the First Time (When It Worked)**

* You **connected to the Jenkins server** (probably via SSH).
* You ran sudo update-alternatives --config java and selected **JDK 8**.
* Then you ran your **Maven build**, and it worked — because your **current shell session** was using JDK 8.
* Jenkins was already **running**, likely started with **JDK 11**, which is fine — Jenkins itself can run on JDK 11, and your build used JDK 8 because of the system default.

✅ **Why it worked**: Jenkins was already running, and your shell session used JDK 8 for Maven.

**🔁 What Happened the Next Day (When It Didn't Work)**

* You **powered off** the server.
* When the server restarted, **Jenkins tried to start again**.
* But now, the system default Java was **JDK 8**, and **Jenkins might not be compatible with JDK 8** (especially newer versions of Jenkins require JDK 11+).
* So Jenkins **failed to start** because it was trying to run on JDK 8.

❌ **Why it failed**: Jenkins needs JDK 11 to run, but the system default was now JDK 8.

**✅ How to Fix This Properly**

You need to **separate the Java version used to run Jenkins** from the one used for Maven builds.

**1. Run Jenkins with JDK 11**

Edit the Jenkins service file to explicitly use JDK 11

sudo nano /etc/default/jenkins

Or for systemd:

sudo systemctl edit jenkins

Add:

JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

Then restart Jenkins:

sudo systemctl daemon-reexec

sudo systemctl restart jenkins

**2. Use JDK 8 in Your Pipeline for Maven**

In your pipeline, set JDK 8 like this:

pipeline {

    agent any

    environment {

        JAVA\_HOME = '/usr/lib/jvm/java-8-openjdk-amd64'

        PATH = "${env.JAVA\_HOME}/bin:${env.PATH}"

    }

    stages {

        stage('Build') {

            steps {

                sh 'java -version'

                sh 'mvn clean install'

            }

        }

    }

}