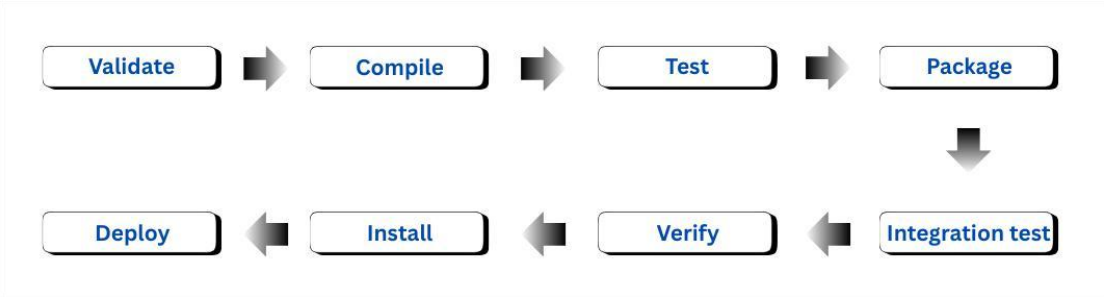
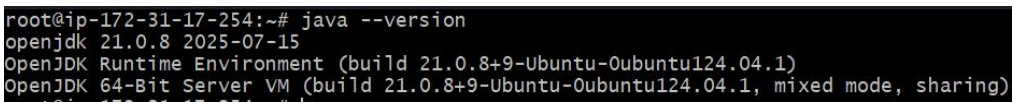
- Source code (java, .net etc) => Compile (javac, Roslyc etc) => Tests (Unit/Integration) => Packaging (jar, war, .exe, msi, .zip etc) => Health checks (Code analysis, Find bugs)
- Build tools:
  - Maven
    - Language : java
    - Build file format : xml
  - Ant
    - Language : java
    - Build file format : xml
  - MsBuild
    - Microsoft build engine is a platform for building applications.
  - Gradle
    - DSL based on Groovy
  - & NANT
    - Windows .net platform
  - Make
    - Builds executable programs and libraries from source code.
- Maven:

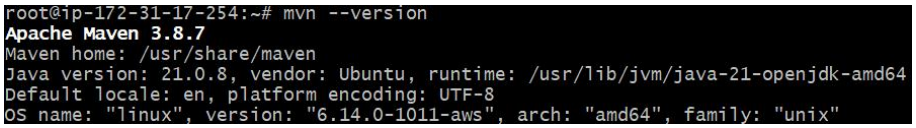


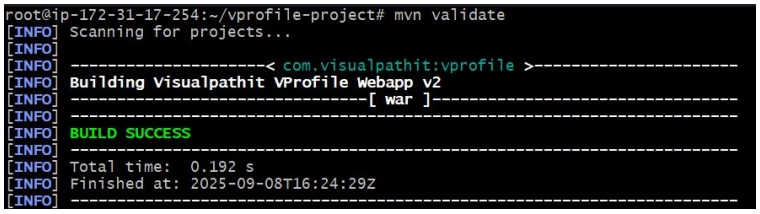
  - Validate:
    - Validate the project is correct and all necessary information is available
  - Compile:
    - Compile the source code of the project
  - Test:
    - Test the compiled source code using a suitable unit testing framework.
    - These tests should not require the code be packaged or deployed.
  - Package:
    - Take the compiled code and package it in its distributable format, such as a JAR.

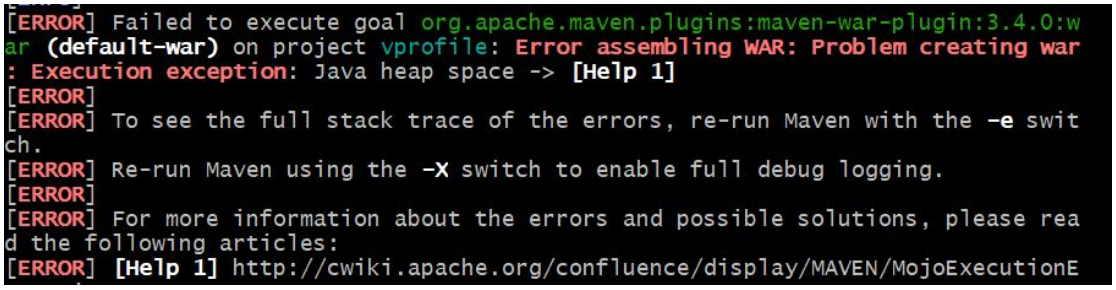
- Verify:
  - Run any checks on results of integration tests to ensure quality criteria are met.
- Install:
  - Install the package into the local repository, for use as a dependency in other projects locally.
- Deploy:
  - Done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

- In context of Java:
  - JDK : Java Development Kit
    - Try executing the command **apt search jdk**, you'll get so many.
    - I am installing **jdk-17** and **jdk-21**.
  - JRE : Java Runtime Environment
    - It is just to run the java application in order to do some development work.

- Steps I followed for **Maven**:
  - **apt search jdk | grep 17** (to find jdk version 17)
  - **apt search jdk | grep 21** (to find jdk version 21)
  - **apt install openjdk-21-jdk -y**
    - Installed jdk version 21.
    - You can verify the version using **java --version** command.
    - 

  - **apt install maven -y**
    - Installed maven.
    - You can verify the version using **mvn --version** command.
    - 

  - Cloned the repo from the github (having java application)
  - **mvn validate**
    - 

  - **mvn test**
    - Trigger unit testcases written by the developers.
    - Generate report inside target folders.
  - **mvn install**
    - When you execute this command, **mvn** will download all the dependencies present inside the **.pom** file.
    - 

    - Its saying some heap space error.
    - **export MAVEN_OPTS="-Xmx1024m"**

- It's telling Maven: "when you run, give your JVM a maximum of 1 GB heap memory."

- Now the build succeed.

- **~/.m2/repository**
  - Here the installed dependencies remain.

```
root@ip-172-31-17-254:~/vprofile-project# ls -a ~/
.  ..  .bashrc  .lesshst  .m2  .profile  .ssh  snap  vprofile-project
root@ip-172-31-17-254:~/vprofile-project# ls -a ~/.m2
.  ..  repository
root@ip-172-31-17-254:~/vprofile-project# ls -a ~/.m2/repository/
.                          classworlds    commons-fileupload  jakarta    log4j
..                         com            commons-io          javax      mysql
backport-util-concurrent   commons-cli    commons-logging     joda-time  net
ch                         commons-codec  io                  junit      org
root@ip-172-31-17-254:~/vprofile-project#
```

- **mvn clean install**
  - It'll delete the target folder (if present) and then start the build process.
  - But it doesn't delete the dependencies.
  - If you want a proper clean installation, then delete everything inside that repository i.e. **rm -rf ~/.m2/repository/*** and then run **mvn clean install**.
  - It'll remove the *target* folder and build the application again.

- I downloaded **jdk-17** version now.
  - **apt install openjdk-17-jdk**
  - Now if I execute **java --version**, it still displaying *version 21* only.

- To switch between **jdk versions**: **update-alternatives --config java**

```
root@ip-172-31-17-254:~/vprofile-project# update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java   2111      auto mode
  1            /usr/lib/jvm/java-17-openjdk-amd64/bin/java   1711      manual mode
* 2            /usr/lib/jvm/java-21-openjdk-amd64/bin/java   2111      manual mode

Press <enter> to keep the current choice[*], or type selection number: 1
update-alternatives: using /usr/lib/jvm/java-17-openjdk-amd64/bin/java to provide /usr/
root@ip-172-31-17-254:~/vprofile-project# java --version
openjdk 17.0.16 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
```

- If you want to use a different version of **mvn** to be used, then you can download the *binary* file from the ***mvn** archive.*
  - https://archive.apache.org/dist/maven/maven-3/3.9.9/binaries/
    - I installed 3.9.9 version of **mvn**
    - I downloaded using wget (I downloaded **zip** file, you can even download **tar.gz** file)
    - Unzipped it and moved it inside */usr/local/bin/mvn3.9* folder.

```
root@ip-172-31-17-254:/tmp# mv apache-maven-3.9.9 /usr/local/bin/maven3.9
root@ip-172-31-17-254:/tmp# ls /usr/local/bin/maven3.9/
LICENSE  NOTICE  README.txt  bin  boot  conf  lib
```

- **/usr/local/bin/mvn3.9/bin** : in this folder, **mvn** command presents.

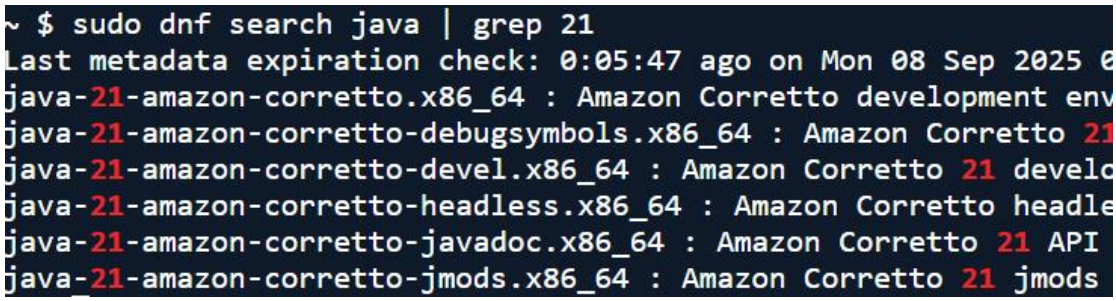
```
root@ip-172-31-17-254:~/vprofile-project# ls /usr/local/bin/maven3.9/bin/
m2.conf  mvn  mvn.cmd  mvnDebug  mvnDebug.cmd  mvnyjp
```

- **/usr/local/bin/mvn3.9/bin/mvn clean install**
  - Executed inside the vprofile-project, now it'll use the installed **mvn** version instead of the *default mvn version*.

- You can open **cloudshell** inside the aws console.
  - Its just a RPM based terminal where you can simulate the things.

- Button is present at the *bottom-left* cornor.

- Difference in Debian and RPM based:
  - In Debian based (ex: ubuntu), packages are usually named as **openjdk-*** or **default-jdk**
    - openjdk-11-jdk
    - openjdk-17-jdk
    - default-jdk
  - In RPM based (ex: centos), packages are usually named as **java-*** (sometimes **java-<version>-openjdk**)
    - java-1.8.0-openjdk
    - java-11-openjdk
    - java-17-openjdk
- **sudo dnf search java | grep 21**
  - Inside cloudshell, used this command to search the **java version 21**packages.



  - When you see *corretto*, means it is from *AWS*.
- **dnf install maven -y**
  - Installed maven as well

DON'T FORGET TO DELETE THIS INSTANCE, OTHERWISE YOU'LL BE CHARGED BY AWS