Maven Java build tool

START HERE

`mvn compile`
Compile main sources

`mvn test`
Compile and run unit tests

`mvn package`
Build the jar or war

`mvn clean install`
Rebuild and install locally

`mvn verify`
Run everything incl. integration tests

`mvn -DskipTests package`
Build without running tests

`mvn dependency:tree`
See resolved dependencies

`./mvnw <args>`
Use the project's pinned Maven

validate check POM → **compile** main sources → **test** unit tests → **package** jar / war → **verify** integration tests → **install** to ~/.m2 → **deploy** to remote repo

Default lifecycle: asking for a phase runs all phases before it

CONCEPTS

Name a phase and Maven runs every earlier phase too: mvn package also compiles and tests.

Goals are plugin:goal (dependency:tree). Phases are lifecycle steps (package) that run the goals bound to them.

BUILD

`mvn compile`
Compile main sources

`mvn test`
Run unit tests

`mvn package`
Build the artifact into target/

`mvn verify`
Run integration tests and checks

`mvn install`
Copy artifact to local repo ~/.m2

`mvn clean`
Delete the target/ directory

`mvn clean install`
Fresh build, then install locally

`mvn deploy`
Publish to the remote repository

BUILD (cont.)

`mvn test-compile`
Compile test sources only

OPTIONS

`-DskipTests`
Compile tests but do not run them

`-Dmaven.test.skip=true`
Skip compiling and running tests

`-o / --offline`
Use only the local repository

`-q / --quiet`
Print errors only

`-X / --debug`
Debug output

`-e / --errors`
Show full stack traces

`-U / --update-snapshots`
Force re-check of SNAPSHOT deps

`-T <n>`
Build modules in parallel with n threads

`-T 1C`
One thread per CPU core

`-B / --batch-mode`
Non-interactive, no colours (CI)

`-D<key>=<value>`
Set a system property

MULTI-MODULE

`mvn -pl <module>`
Build only this module

`mvn -pl <module> -am`
Also build what it depends on

`mvn -pl <module> -amd`
Also build modules depending on it

`mvn -pl '!<module>'`
Exclude a module

`mvn -rf :<module>`
Resume the reactor from a module

`mvn versions:set -DnewVersion=<v>`
Set version in all modules
Needs the versions plugin; undo with versions:revert, accept with versions:commit.

Select modules by path (-pl app/web) or by :artifactId.

DEPENDENCIES

`mvn dependency:tree`
Print the resolved dependency tree

`mvn dependency:tree -Dincludes=<g>:<a>`
Find who pulls in a library

`mvn dependency:analyze`
Find unused or undeclared deps

DEPENDENCIES (cont.)

`mvn dependency:copy-dependencies`
Copy dependency jars to target/dependency

`mvn dependency:go-offline`
Prefetch everything for offline builds

`mvn versions:display-dependency-updates`
List newer dependency versions

Scopes: compile (default), provided, runtime, test. Only compile and runtime go into the package.

INSPECT AND DEBUG

`mvn help:effective-pom`
Show the POM after inheritance

`mvn help:effective-settings`
Show merged settings.xml

`mvn help:active-profiles`
List active profiles

`mvn help:evaluate -Dexpression=<expr> -q -DforceStdout`
Print one POM value

`mvn -v`
Show Maven, Java and OS versions

INSPECT AND DEBUG (cont.)

`mvn -Dtest=<Class>#<method>`
Run one test method

`mvn -Dtest='<Class>*'`
Run tests matching a pattern

`mvn -Dmaven.surefire.debug test`
Wait for a debugger on port 5005

PROFILES AND PROPERTIES

`mvn -P<id>`
Activate a profile (-Pa,b for several)

Profiles also switch on by JDK, OS, property or file; check with help:active-profiles.

WRAPPER AND PROJECT

`mvn wrapper:wrapper`
Add mvnw and mvnw.cmd to the project

`./mvnw <args>`
Run with the pinned Maven version

`mvn archetype:generate`
Create a project from a template

WRAPPER AND PROJECT (cont.)

Where things live

Path	Holds
`pom.xml`	project model
`~/.m2/settings.xml`	user settings, mirrors, servers
`~/.m2/repository`	local artifact cache
`target/`	build output
`.mvn/`	wrapper and jvm.config

Wrapper settings are in .mvn/wrapper/maven-wrapper.properties; commit .mvn/ and mvnw.

RECIPES

Fast CI build

```
mvn -B -ntp -T 1C clean verify
```

Rebuild one module and its dependencies

```
mvn -pl services/api -am clean install -DskipTests
```

Find and read a version conflict

```
mvn dependency:tree -Dverbose \
  -Dincludes=com.google.guava
mvn help:effective-pom | less
```

Run a single test in a module

```
mvn -pl app -Dtest=UserServiceTest#login \
  -Dsurefire.failIfNoSpecifiedTests=false test
```