[Maven Commands] (CheatSheet)

1. Getting Started

- `mvn -v`: Display Maven version.
- `mvn archetype:generate`: Generate a new project.

2. Building Projects

- `mvn compile`: Compile the source code of the project.
- `mvn test-compile`: Compile the test source code.
- `mvn test`: Run tests using a suitable unit testing framework.
- `mvn package`: Take the compiled code and package it in its distributable format, such as a JAR.
- `mvn verify`: Run any checks to verify the package is valid.
- `mvn install`: Install the package into the local repository, for use as a dependency in other projects.
- `mvn deploy`: Done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

3. Cleaning up

- `mvn clean`: Clean up after the build process and remove the target directory.

4. Project Testing

- `mvn surefire-report:report`: Generate a report based on the results of the unit tests.
- `mvn failsafe:integration-test`: Run integration tests using the Failsafe plugin.

5. Working with Dependencies

- `mvn dependency:tree`: Display the dependency tree.
- `mvn dependency:analyze`: Analyze the dependencies of this project and determine which are: used and declared; used and undeclared; unused and declared.

By: Waleed Mousa

- `mvn dependency:get`: Download a single artifact and optionally its dependencies.
- `mvn dependency:resolve`: Resolve all dependencies necessary for a build.

6. Project Information

- `mvn help:describe`: Display help information on Maven, including available goals.
- `mvn site`: Generate a site for the current project.
- `mvn project-info-reports:dependencies`: Generate a report of the dependencies for the project.

7. Lifecycle Phases

- `mvn validate`: Validate the project is correct and all necessary information is available.
- `mvn initialize`: Initialize build state, e.g., set properties or create directories.
- `mvn generate-sources`: Generate any source code for inclusion in compilation.
- `mvn process-sources`: Process the source code, for example to filter any values.
- `mvn generate-resources`: Generate resources for inclusion in the package.
- `mvn process-resources`: Copy and process the resources into the destination directory, ready for packaging.
- `mvn pre-integration-test`: Perform actions required before integration tests are executed.
- `mvn post-integration-test`: Perform actions required after integration tests have been executed.
- `mvn prepare-package`: Perform any operations necessary to prepare a package before the actual packaging.
- `mvn pre-site`: Perform actions required before generating the site.
- `mvn post-site`: Perform actions required after generating the site.
- `mvn site-deploy`: Deploy the generated site documentation to the specified web server.

8. Release Management

- `mvn release:prepare`: Prepare for a release in SCM.

By: Waleed Mousa

- `mvn release:clean`: Clean up after a release preparation.
- `mvn release:perform`: Perform a release from SCM.
- `mvn release:rollback`: Rollback a previous release.

9. Advanced Usage

- `mvn dependency:purge-local-repository`: Remove the project dependencies from the local repository.
- `mvn help:effective-pom`: Display the effective POM as an XML for this build, with the active profiles factored in.
- `mvn versions:set`: Set the project version.
- `mvn archetype:crawl`: Catalog the Maven project archetypes.

10. Profiles and Properties

- `mvn -P<profile_id> ...`: Activate a specific build profile.
- `mvn -D<property>=<value> ...`: Set a system property.

11. Plugins and Goals

- `mvn <plugin>:<goal>`: Execute a specific plugin goal.
- `mvn <pluginPrefix>:<goal>`: Execute a goal using the shorthand plugin prefix.

12. Skipping Tests

- `mvn install -DskipTests`: Compile and install the package, but skip running tests.
- `mvn install -Dmaven.test.skip=true`: Skip compiling and running tests.

13. Parallel Builds

- `mvn -T 1C clean install`: Run builds with one thread per CPU core.

14. Troubleshooting

- `mvn help:effective-settings`: Display the calculated settings for the project, useful for troubleshooting.
- `mvn dependency:resolve -Dclassifier=sources`: Download sources for all dependencies.

By: Waleed Mousa

15. Setting up Multi-Module Projects

- `mvn archetype:generate -DgroupId=<group-id> -DartifactId=<artifact-id> -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`: Create a simple project structure.

16. Reactor Options

- `mvn -pl <projects>`: Build specified reactor projects instead of all projects.
- `mvn -am`: Also make all projects required by the project list.
- `mvn -amd`: Also make dependents of the projects listed.

17. Snapshot Handling

- `mvn versions:use-latest-versions`: Move to latest versions of dependencies.
- `mvn versions:commit`: Confirm changes made by the versions plugin.
- `mvn versions:revert`: Revert changes made by the versions plugin.

18. Managing Plugins

- `mvn plugin:help`: Display help information on maven-plugin.
- `mvn <plugin>:<goal> -X`: Debug a plugin execution.
- `mvn compiler:help`: Display help information on maven-compiler-plugin.

19. Running Specific Phases and Goals

- `mvn <phase> <phase> ...`: Run specific ordered life cycle phases (e.g., `mvn clean install`).
- `mvn <plugin>:<goal> <plugin>:<goal> ...`: Run specific plugin goals.

20. Interactive Mode

- `mvn --batch-mode`: Run in non-interactive (batch) mode (useful for scripts).

21. Offline Mode

By: Waleed Mousa

- `mvn --offline`: Work offline. Useful if you have all dependencies cached locally and want to avoid checking remote repositories.

22. Project Documentation

- `mvn javadoc:javadoc`: Generate javadoc for the project.
- `mvn javadoc:test-javadoc`: Generate javadoc for the test sources of the project.

23. Creating and Using Archetypes

- `mvn archetype:create-from-project`: Create an archetype from the current project.
- `mvn archetype:generate -DarchetypeCatalog=local`: Generate a project from a local archetype catalog.

24. Miscellaneous Commands

- `mvn -N`: Non-recursive. Do not recurse into sub-projects.
- `mvn -C`: Fail the build if checksums don't match.
- `mvn -fae`: Fail at end; do not stop at the first failure.
- `mvn -nsu`: Suppress SNAPSHOT updates.

25. Customizing Builds

- `mvn install -Dmaven.test.failure.ignore=true`: Ignore test failure and continue the build process.

26. Environment Details

- `mvn help:system`: Display system information useful for debugging Maven itself.

27. Maven Wrapper

- `mvn -N io.takari:maven:wrapper`: Install the Maven Wrapper for the project, allowing it to use a predefined Maven version without needing to install Maven separately.

28. Signing Artifacts

By: Waleed Mousa

- `mvn gpg:sign-and-deploy-file`: Sign and deploy an artifact to the remote repository.

29. Performance Optimization

- `mvn -T 4`: Build with 4 threads. Adjust number according to your machine's CPU cores for parallel builds.

30. Handling Large Projects

- `mvn dependency:go-offline`: Download everything needed to go offline.

31. Advanced Reactor Options

- `mvn -rf :<project>`: Resume reactor from the specified project.

32. Monitoring Build Progress

- `mvn -B`: Run in batch mode (non-interactive) with less output - useful for CI servers.

33. Adjusting Logging Level

- `mvn -q`: Quiet logging - only show errors.
- `mvn -X`: Debug mode, provide full debug output.

34. Handling Plugins and Dependencies

- `mvn dependency:purge-local-repository -DreResolve=false`: Clean your local repository of project dependencies.

35. Custom Build Profiles

- `mvn -P<profile-id>`: Activate certain build profiles for different build configurations.

36. Version Control and SCM

- `mvn scm:checkin`: Check in changes to your version control system.
- `mvn scm:update`: Update sources from your version control system.

By: Waleed Mousa

37. Enhancing the Build Process

- `mvn enforcer:enforce`: Enforce certain rules to fulfill before proceeding with the build.