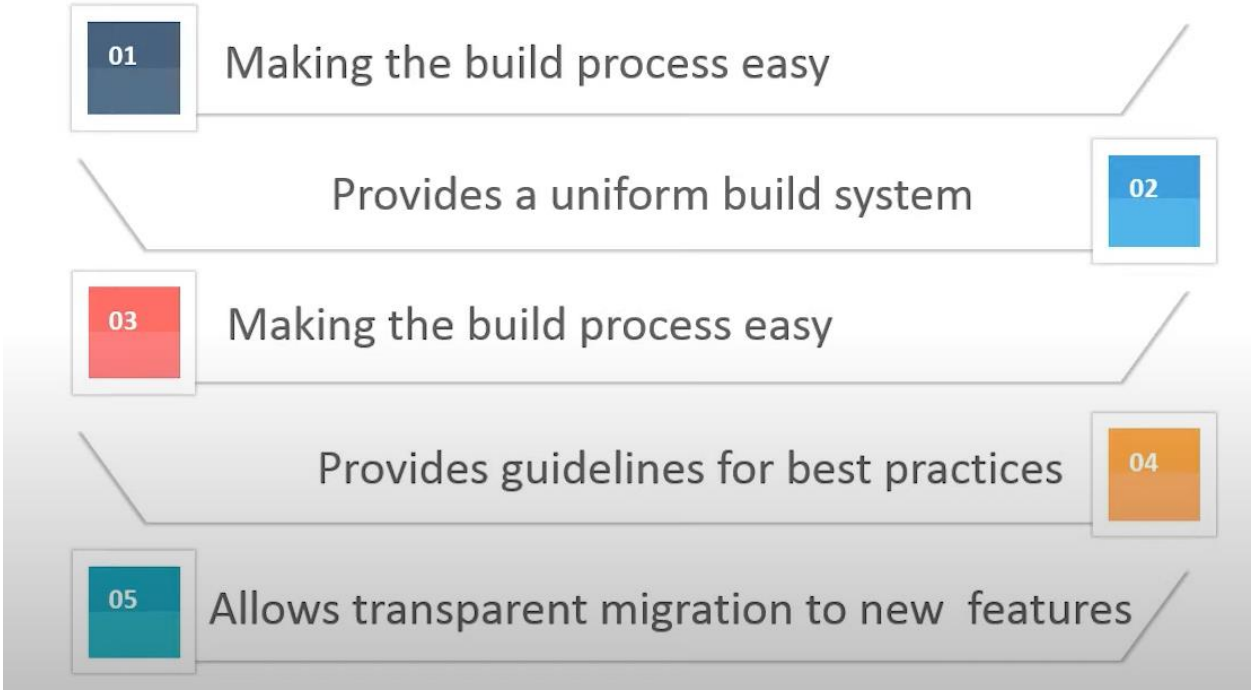
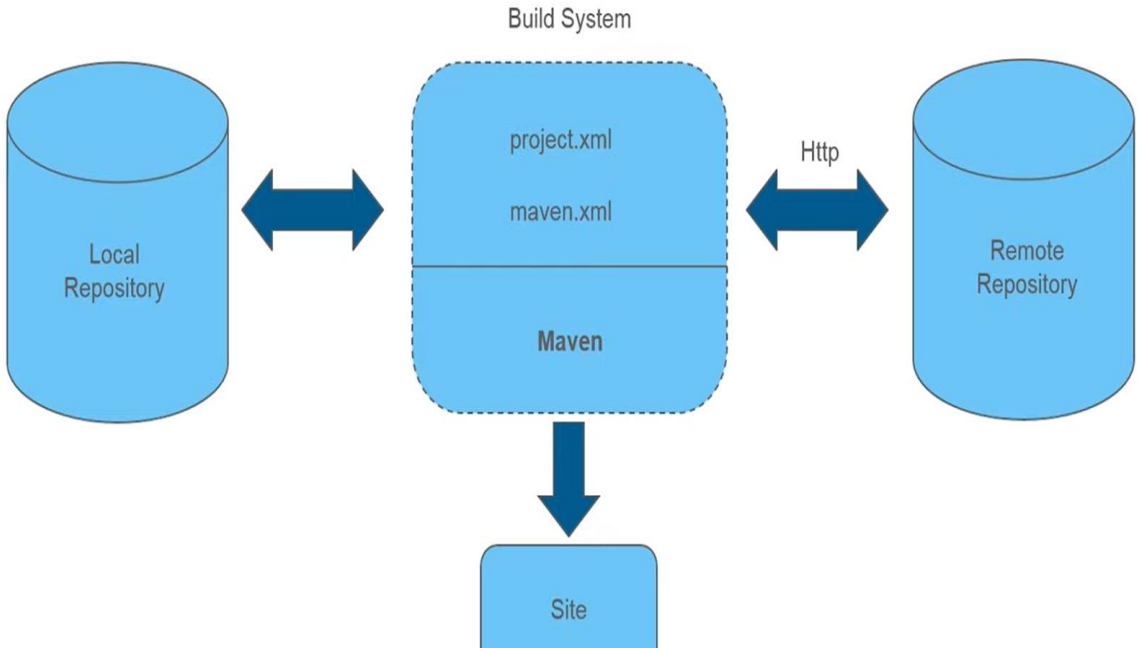
Maven Advantages



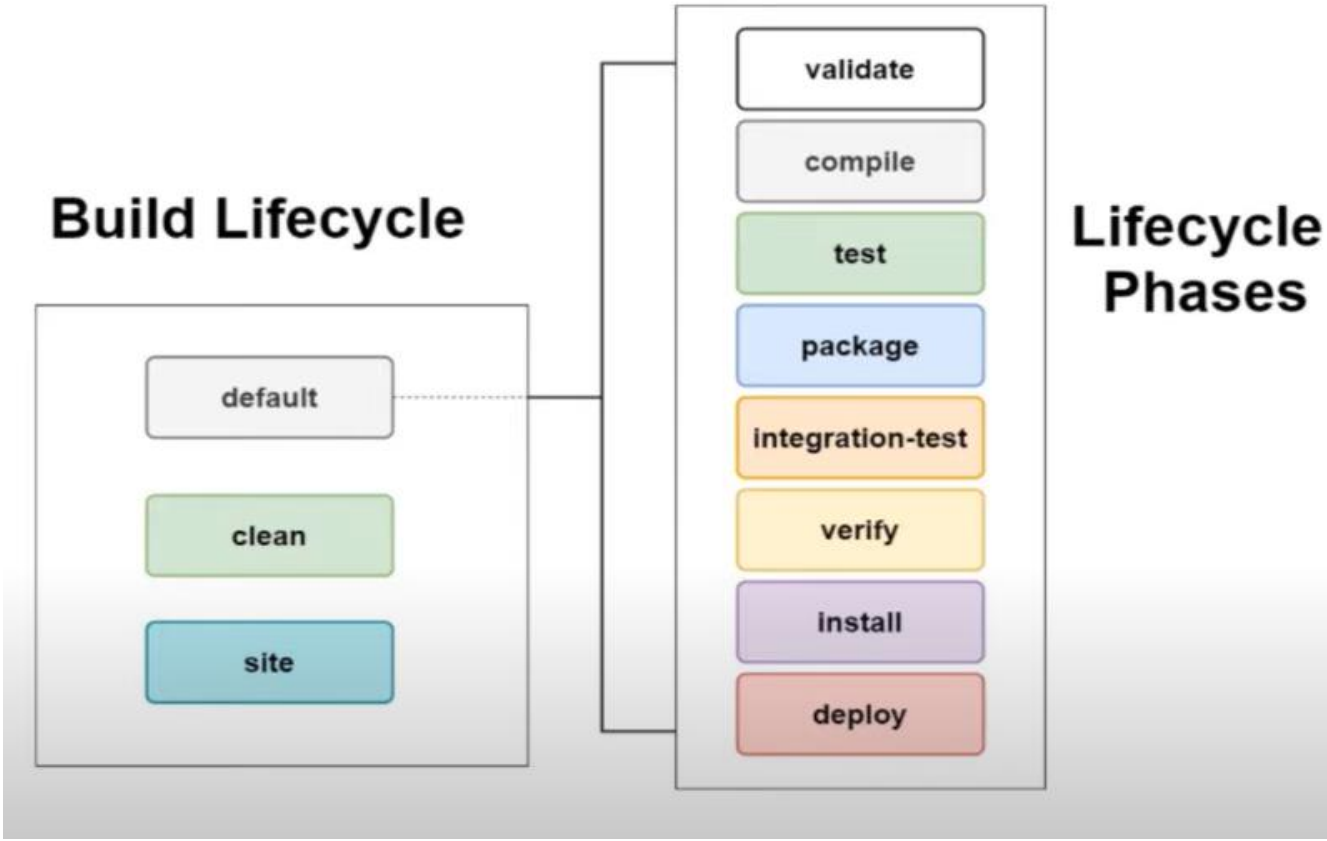
Maven Architecture

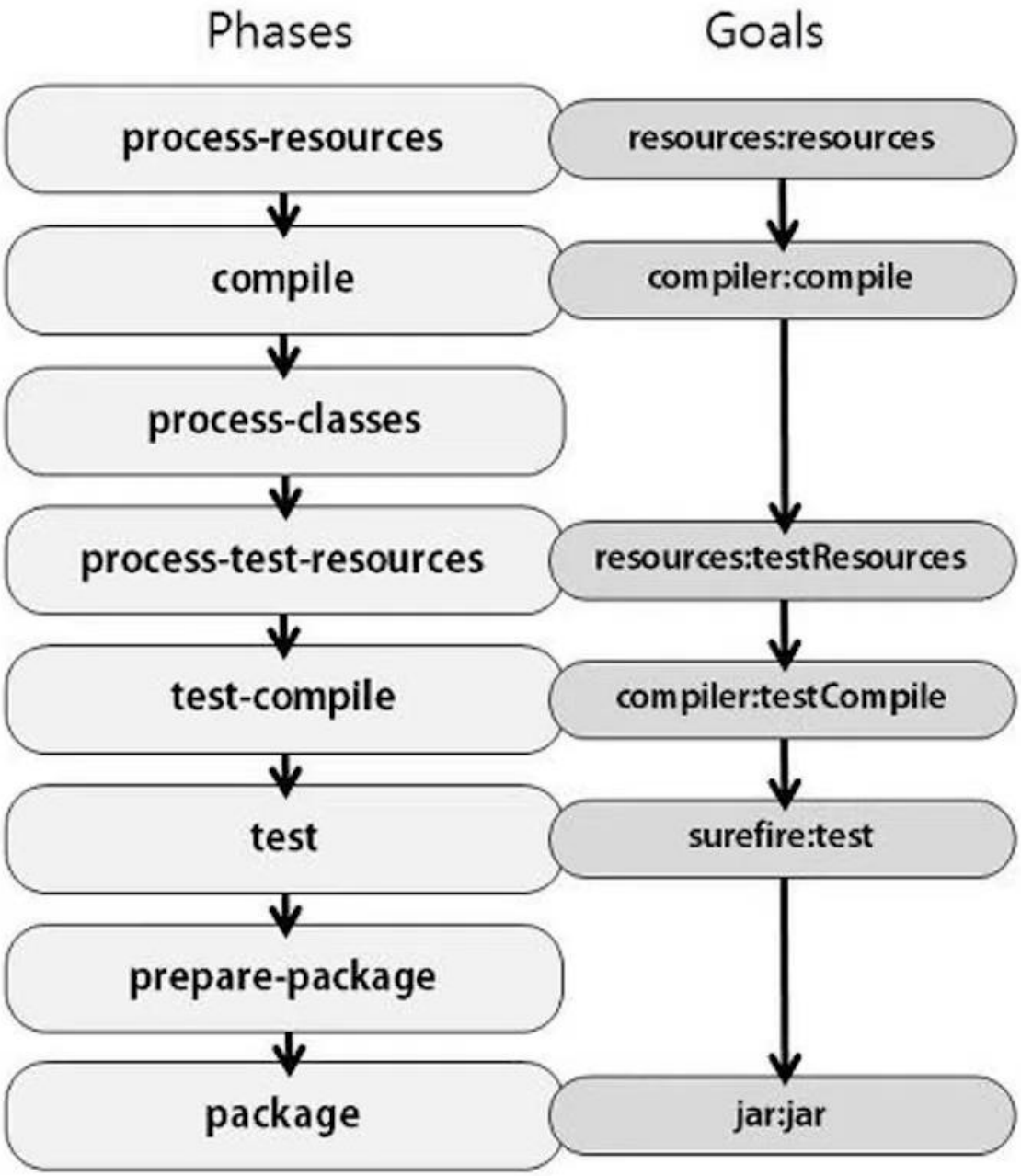


Clean Lifecyle
pre-clean
clean
post-clean

Default Lifecyle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	package
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	install
generate-test-resources	deploy
processs-test-resources	

Site Lifecyle
pre-site
site
post-site
site-deploy




Phases
Goals
process-resources
resources:resources
compile
compiler:compile
process-classes
process-test-resources
resources:testResources
test-compile
compiler:testCompile
test
surefire:test
prepare-package
package
jar:jar

Maven Plugins

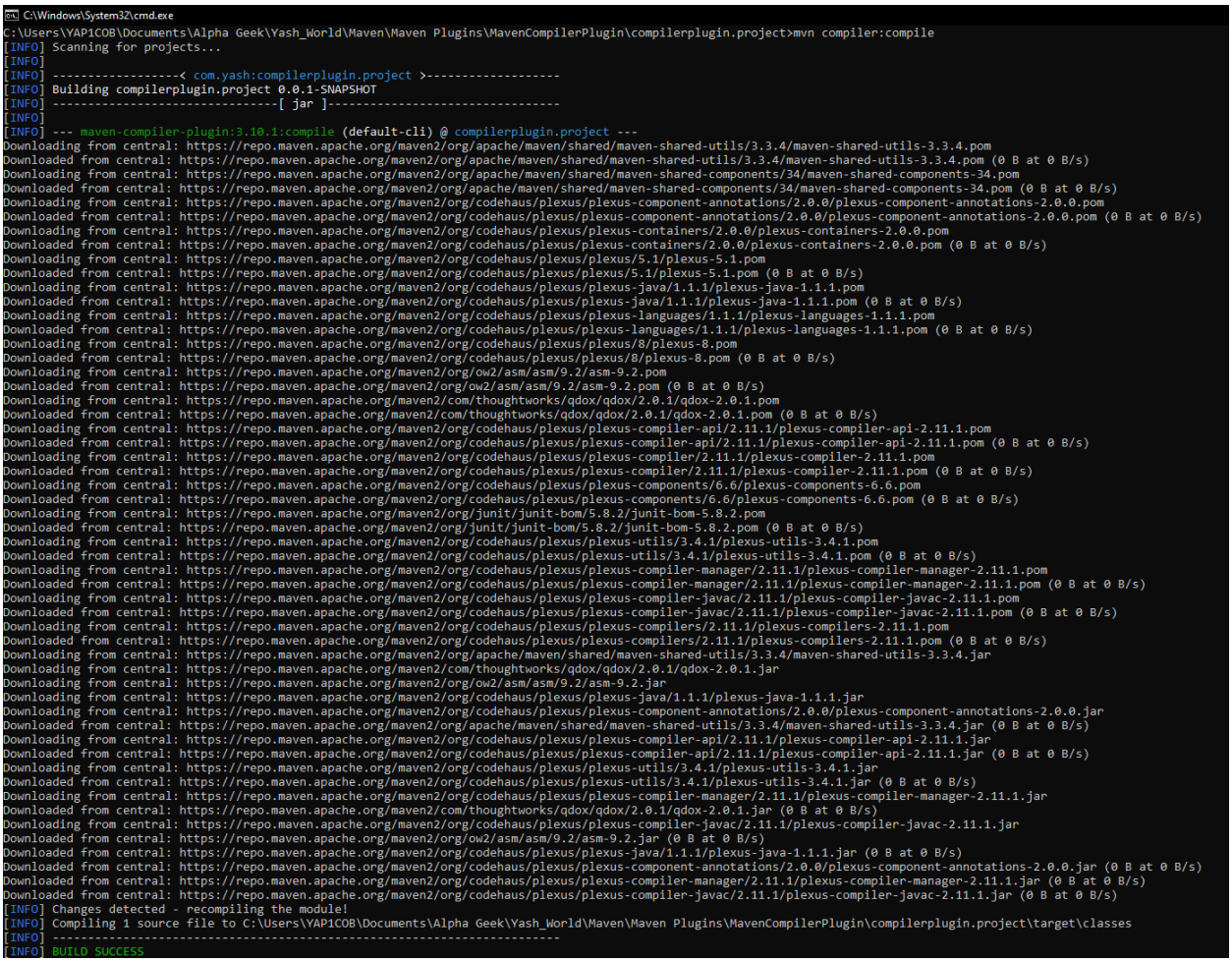
- Plugins enable us to run the lifecycle phases in our Maven Project
- Each Plugin is associated with a GOAL, which is linked to the lifecycle phase
- Plugins can be defined inside <plugin> section, under the <build> tag

Maven Compiler Plugin

- Compiles out java files, similar to running javac <java-class-name>

mvn compiler: compile

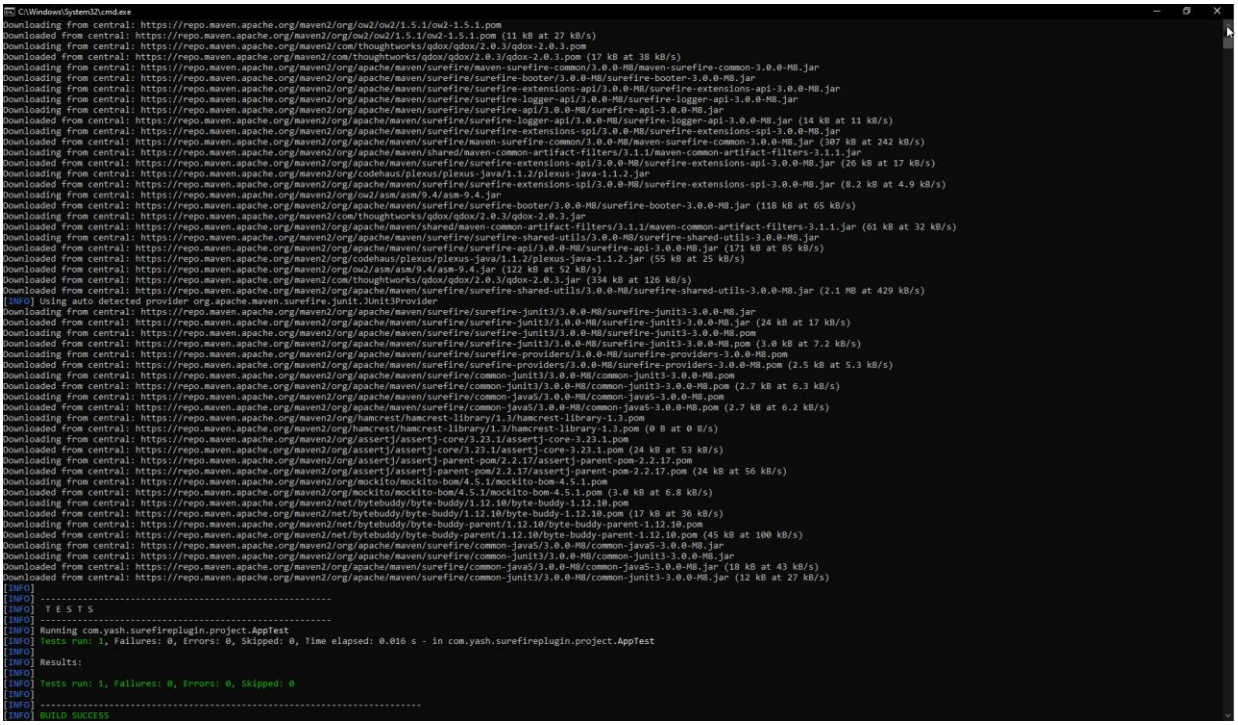
mvn compiler: testCompile



Maven Surefire Plugin

Runs Unit Test inside our Project and also generates Test Reports

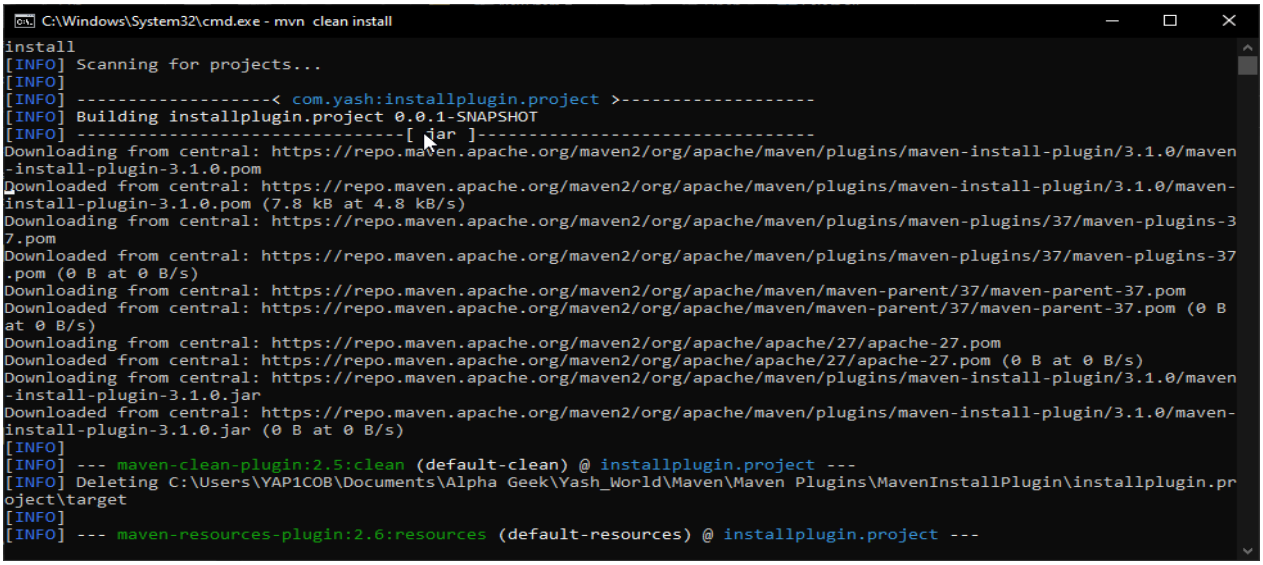
mvn clean test



Maven Install Plugin

Packages source code into an artifact and installs it into the **Local Repository**

mvn clean install



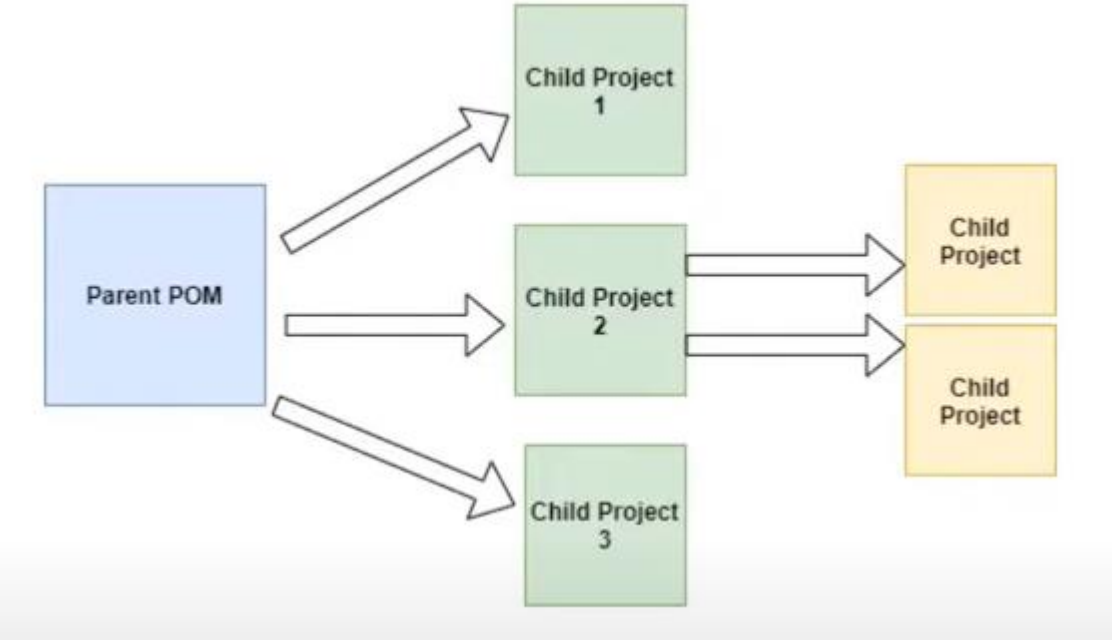
Maven Deploy Plugin

Deploys the created Artifact into the **Remote Repository**

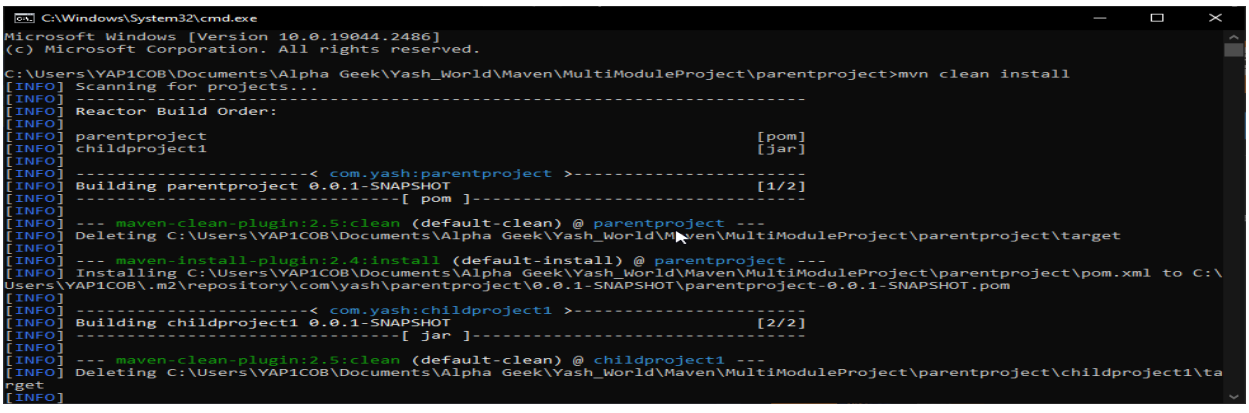
mvn clean deploy

Maven Multi-Module Project

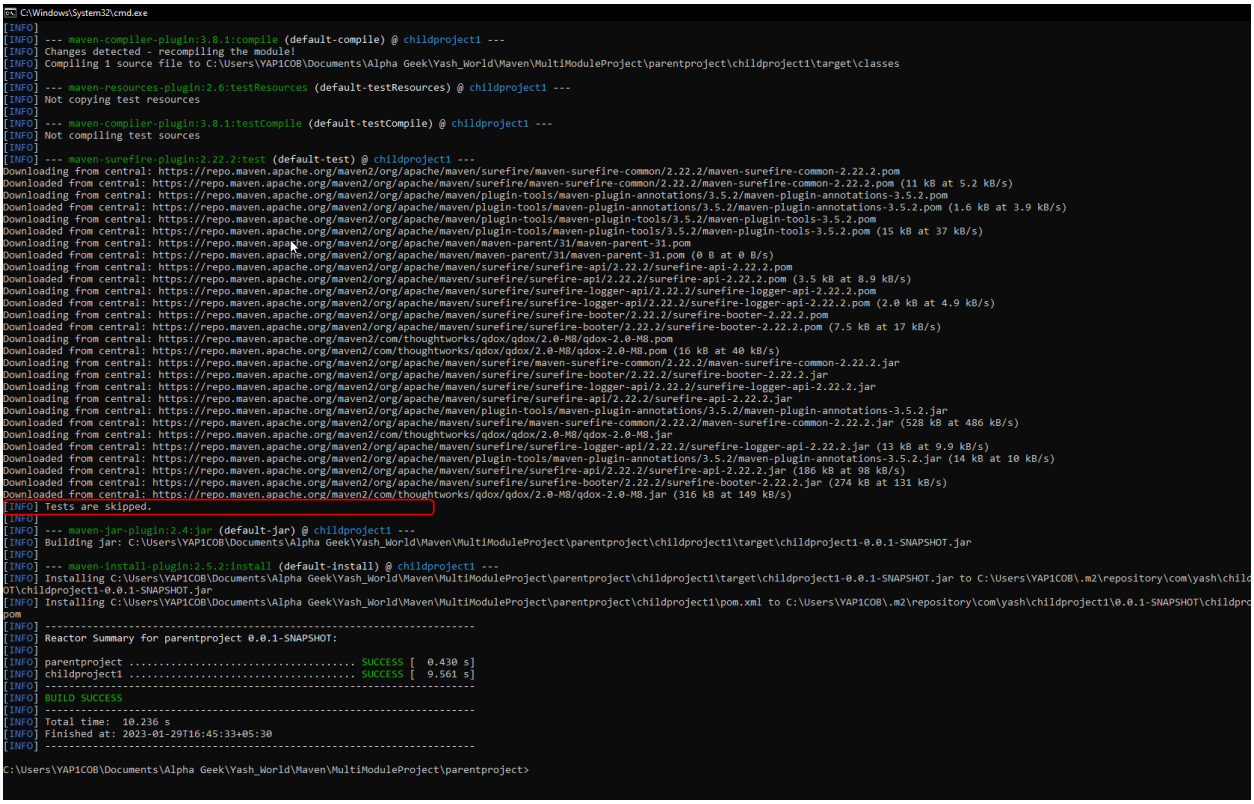
We have a parent project (parent POM) that contains different sub-projects (sub-modules), each of which is again a normal Maven Project.



The parent POM usually encapsulates the other child's and that's why it is packaged as a POM instead usual packaging format of JAR.



Profiles

Profiles can be used in Maven to create customized build configurations. This means customizing the behavior of our builds based on specific conditions.

mvn -Pskip-test clean install