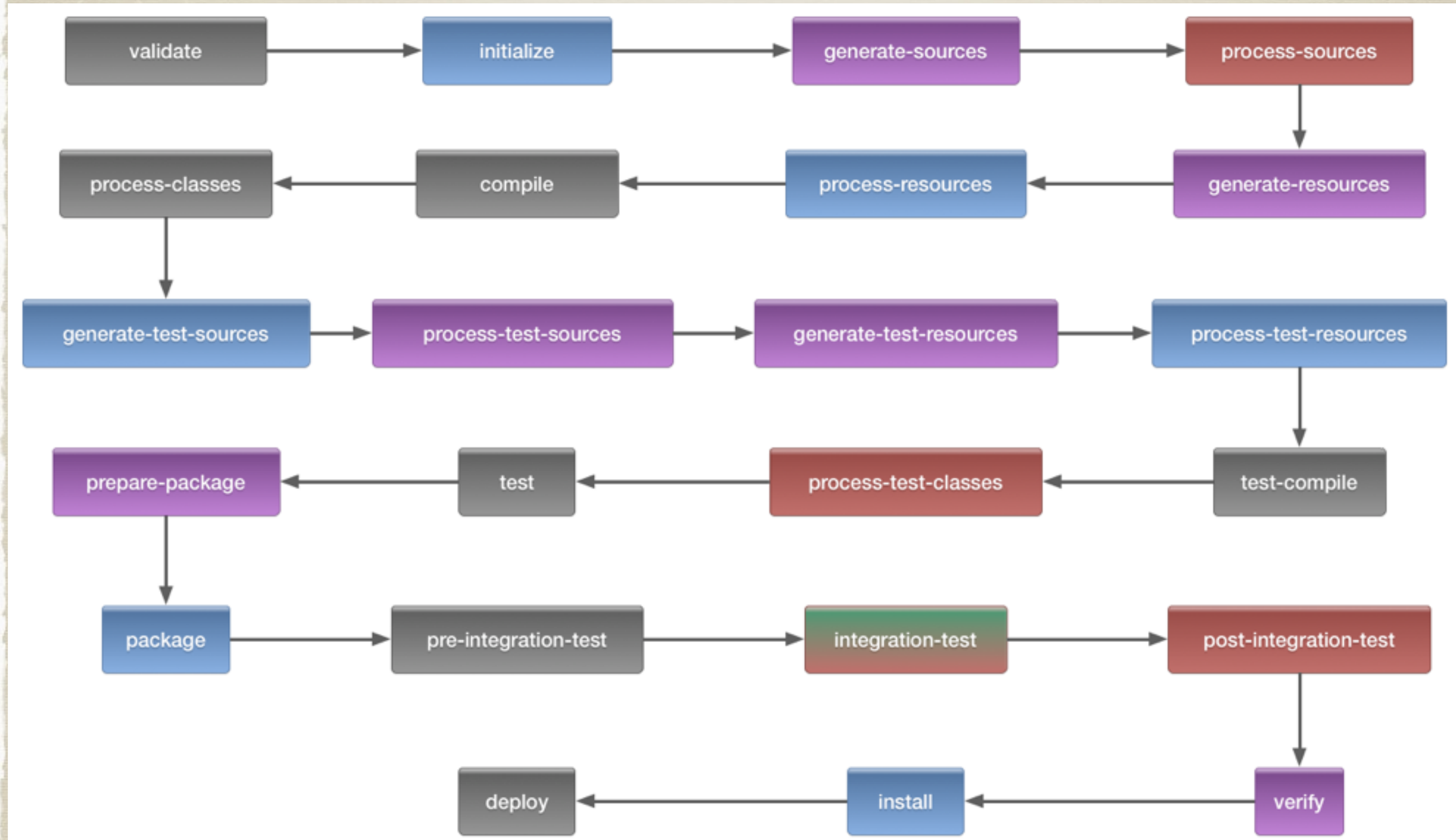
Maven 初探

Zizhi Zhan

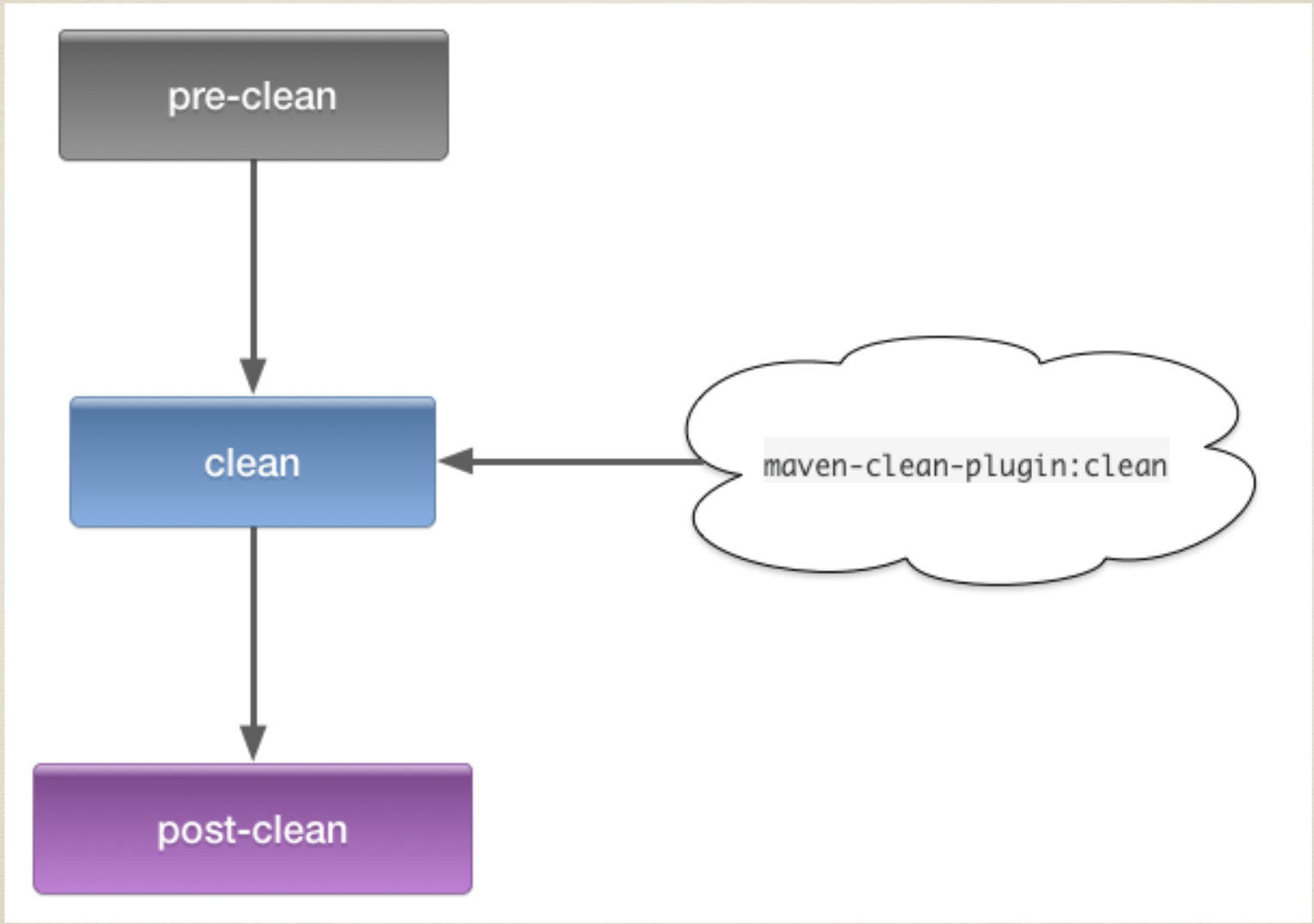
Maven Lifecycle

* Default Lifecycle
* Clean Lifecycle
* Site Lifecycle

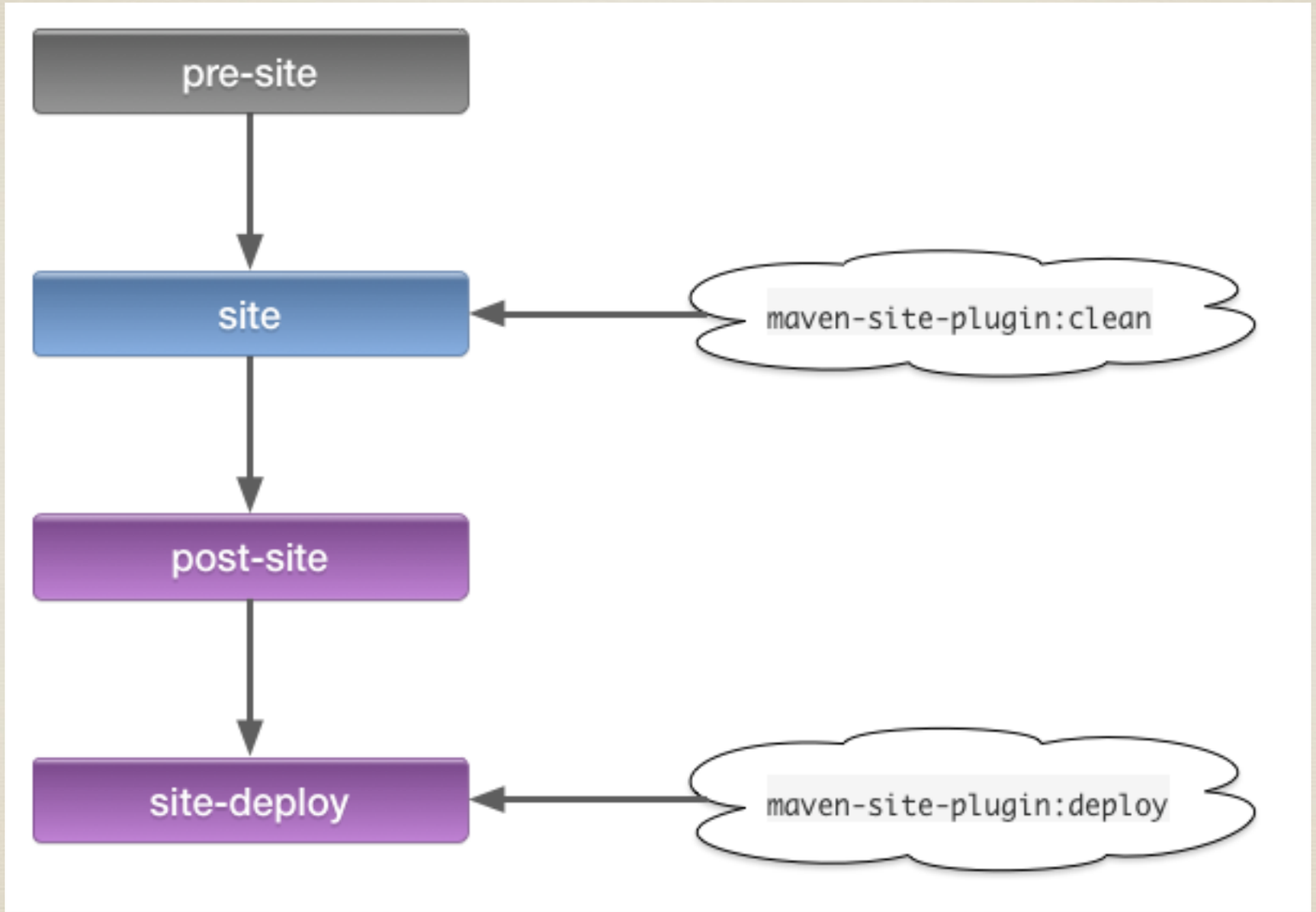
Default Lifecycle



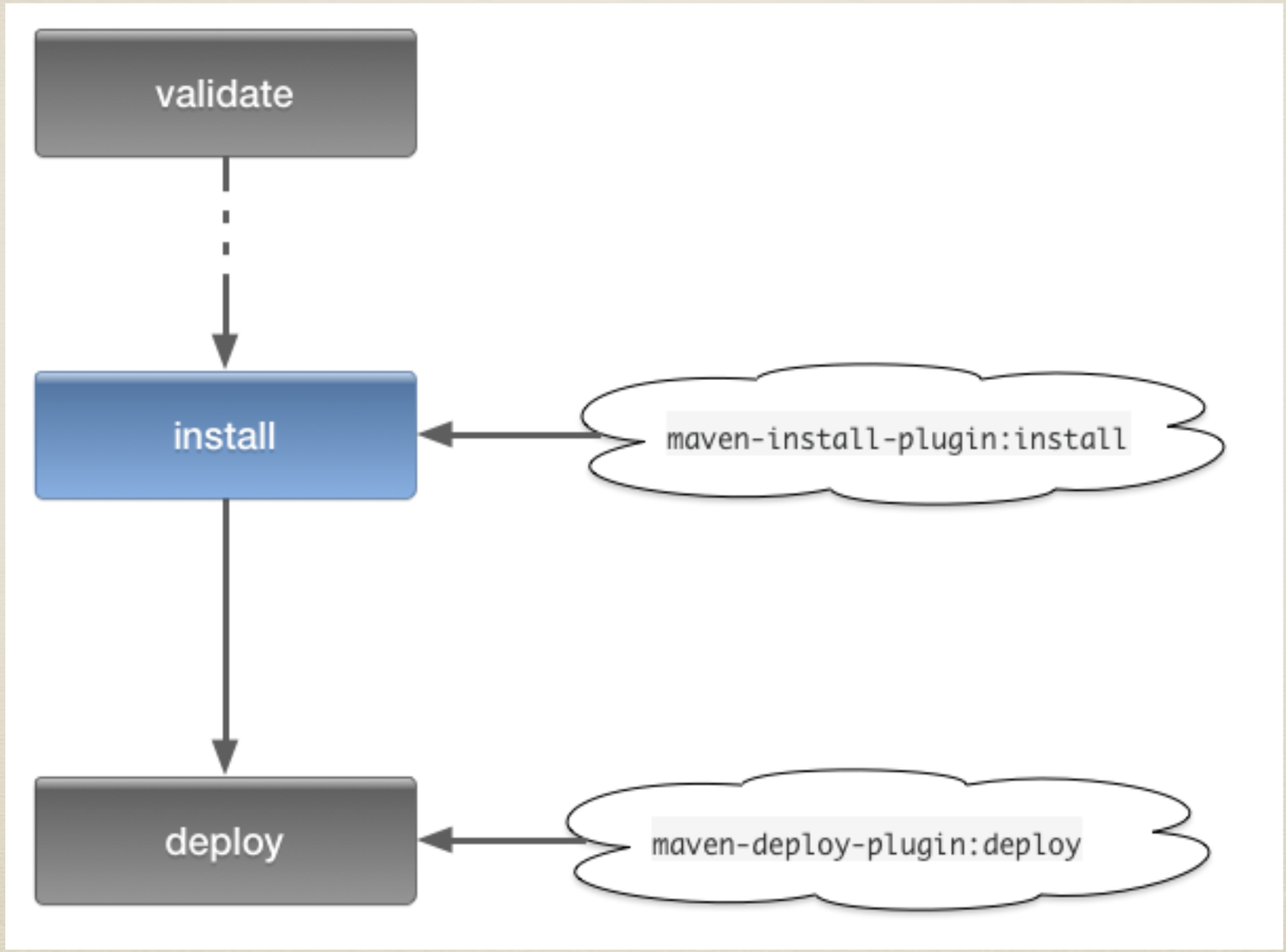
Clean Lifecycle



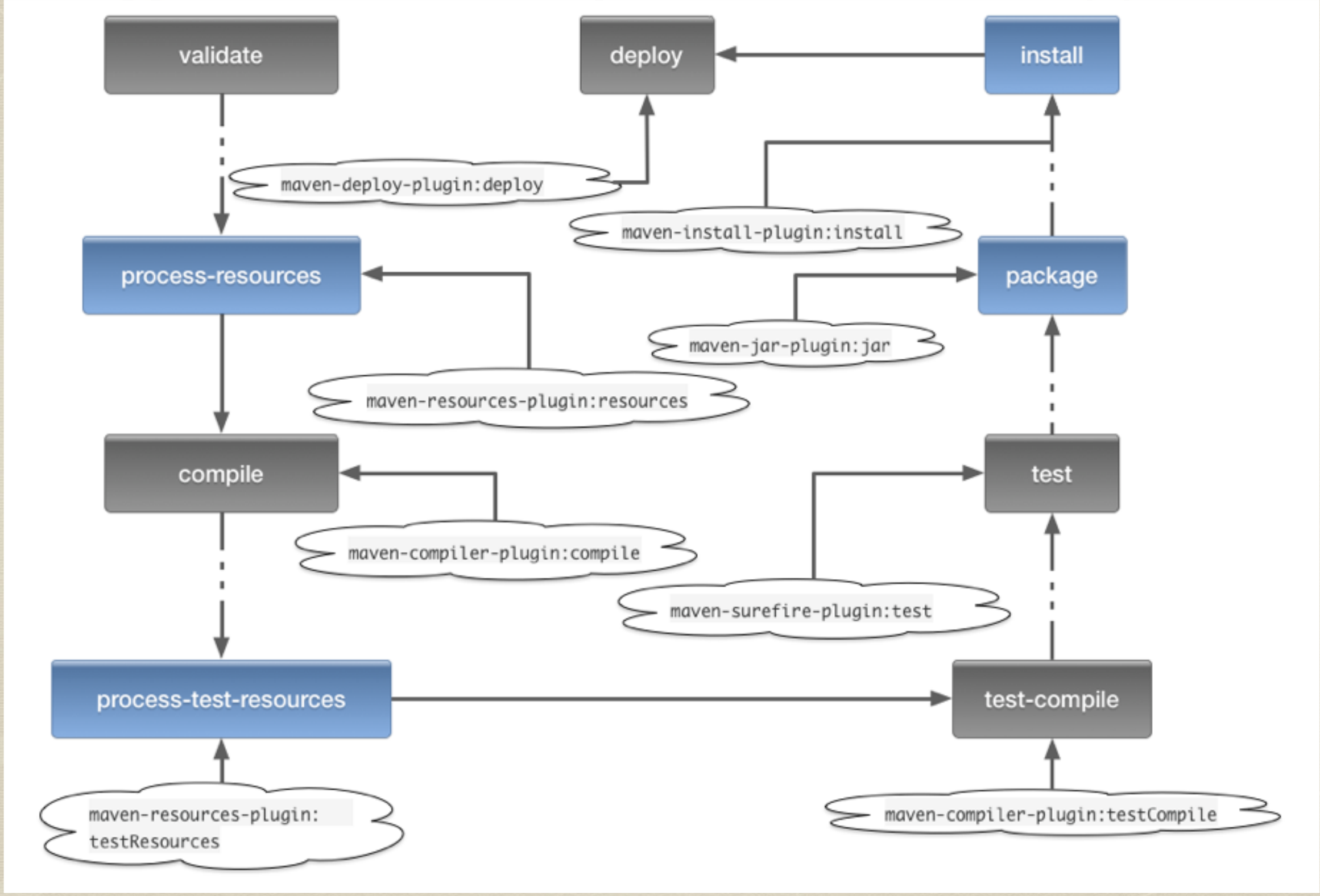
Site Lifecycle



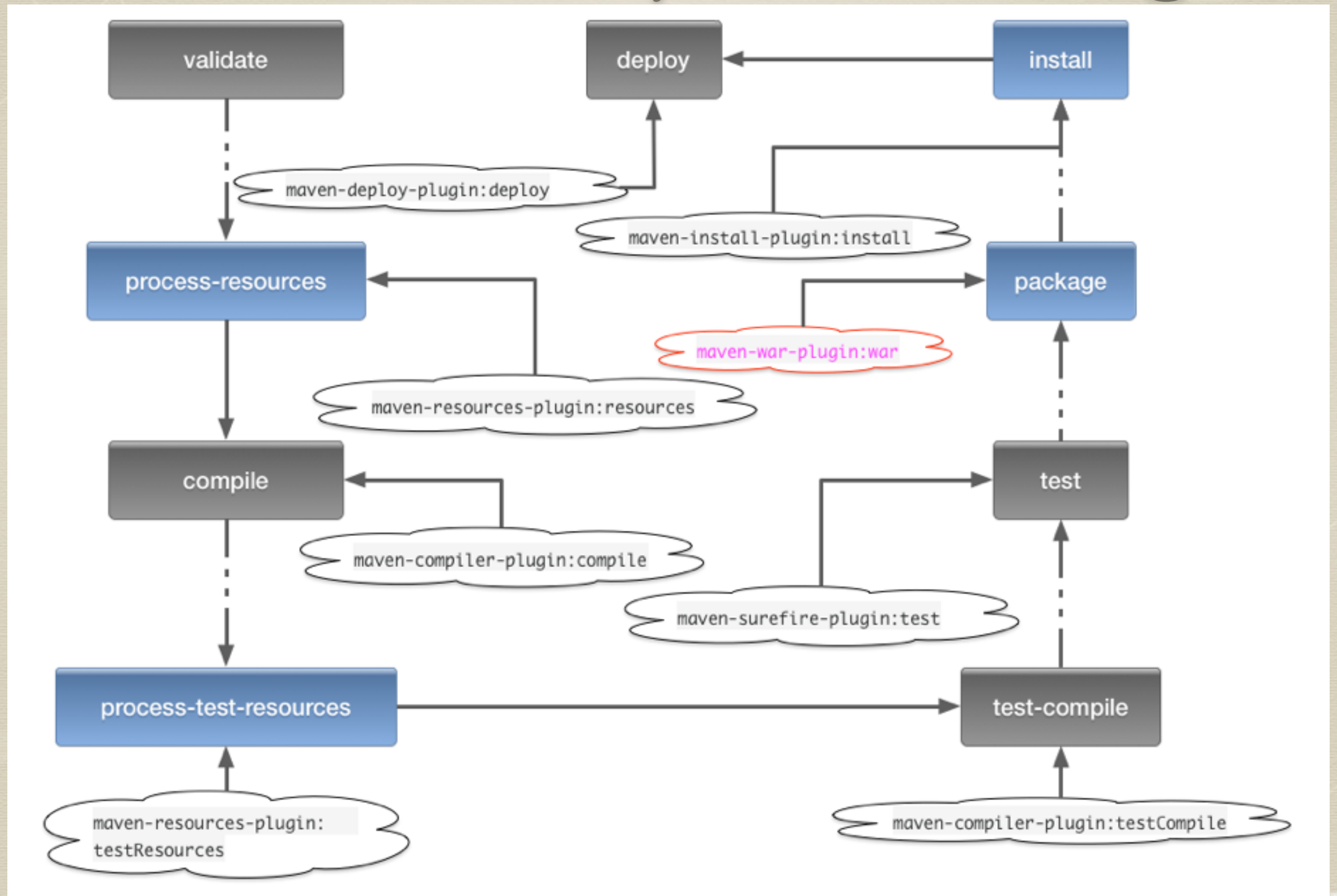
POM Lifecycle Binding



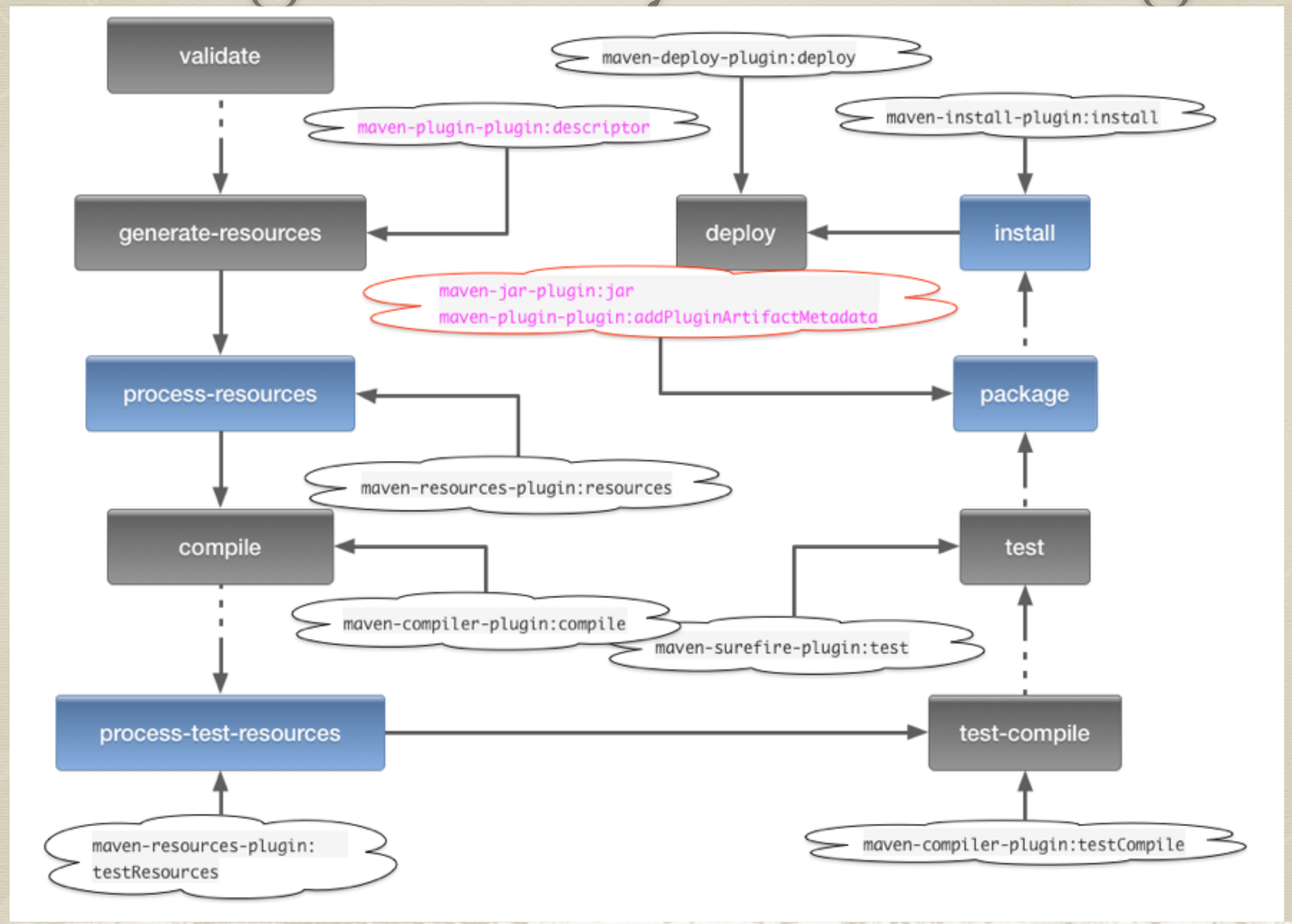
JAR Lifecycle Binding



WAR Lifecycle Binding



Plugin Lifecycle Binding



Where define Maven Components

Please check the following config files

core maven components

maven-core-3.2.1.jar!/META-INF/plexus/components.xml

Artifact Handler for handle different project type, such as pom, jar, ear, war, par.

maven-core-3.2.1.jar!/META-INF/plexus/artifact-handlers.xml

Binding plugins to default lifecycle for different project type.

maven-core-3.2.1.jar!/META-INF/plexus/default-bindings.xml

PlexusContainer is a IoC container to load the components

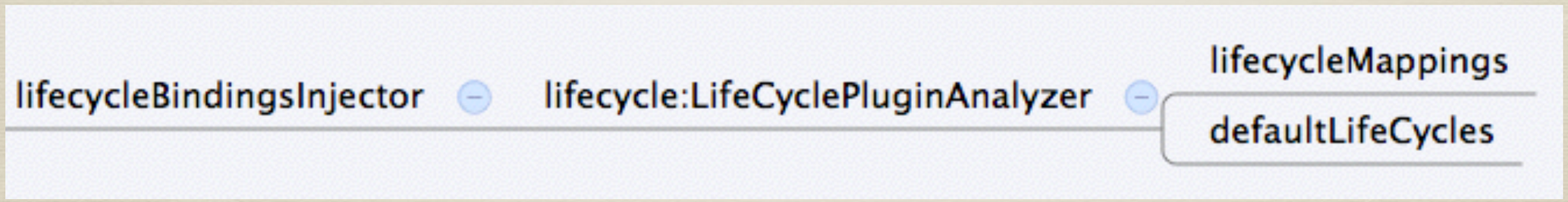
LifecycleBindingsInjector

LifecycleBindingsInjector#injectLifecycleBindings

Injects plugin executions induced by lifecycle bindings into the specified model.

LifeCyclePluginAnalyzer#getPluginsBoundByDefaultToAllLifecycles

It is going to take the project packaging and find all plugin in the default lifecycle and create fully populated Plugin objects, including executions with goals and default configuration taken from the plugin.xml inside a plugin.



code base on maven 3.2.1

Maven bootstrap

${JAVA_HOME}/bin/java -classpath ${M2_HOME}/boot/plexus-classworlds-*.jar -Dclassworlds.conf=${M2_HOME}/bin/m2.conf -Dmaven.home=${M2_HOME} org.codehaus.plexus.classworlds.launcher.Launcher install

plexus-classwords is a class loader framework

Check the code in DefaultClassRealmManager

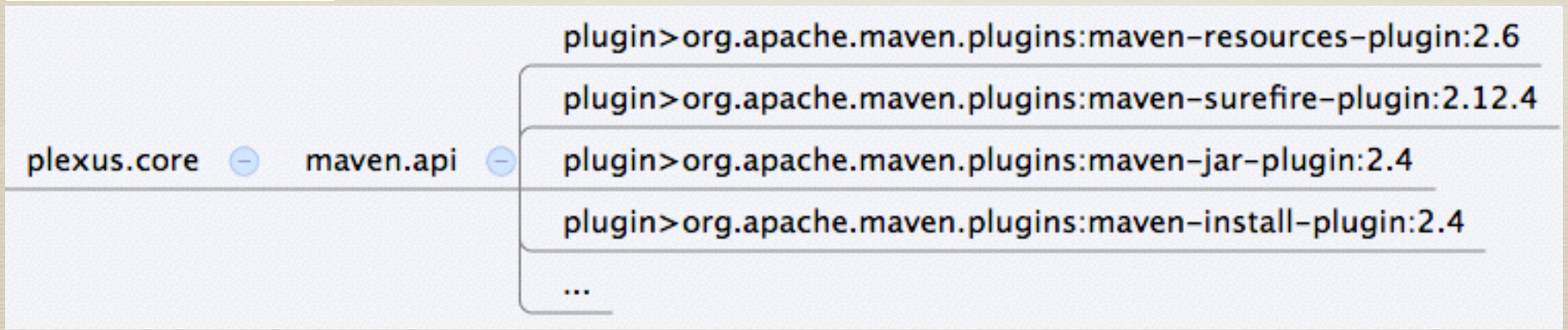
m2.conf

```
main is org.apache.maven.cli.MavenCli from plexus.core

set maven.home default ${user.home}/m2

[plexus.core]
optionally ${maven.home}/lib/ext/*.jar
load       ${maven.home}/lib/*.jar
load       ${maven.home}/conf/logging
```

Classloader Hierarchy



Maven Convention

* All the maven model will inherit from maven-model-builder-#{maven.version}.jar!/org/apache/maven/model/pom-#{pom.version}.xml
  check the code in DefaultSuperPomProvider and DefaultModelBuilder#getSuperModel
* Load the user settings file in ~/.m2/settings.xml
  check the code in MavenCli
* Maven default PluginGroup[org.apache.maven.plugins, org.codehaus.mojo]
  check the code in DefaultMavenExecutionRequestPopulator
* Maven default local repository ~/.m2/repository
  check the code RepositorySystem#defaultUserLocalRepository
* Maven plugin metadata [maven-metadata-local.xml, maven-metadata-#{repo.id}.xml]
* If there is no pom.xml in current directory, it will use maven-core-#{maven.version}.jar!/org/apache/maven/project/standalone.xml

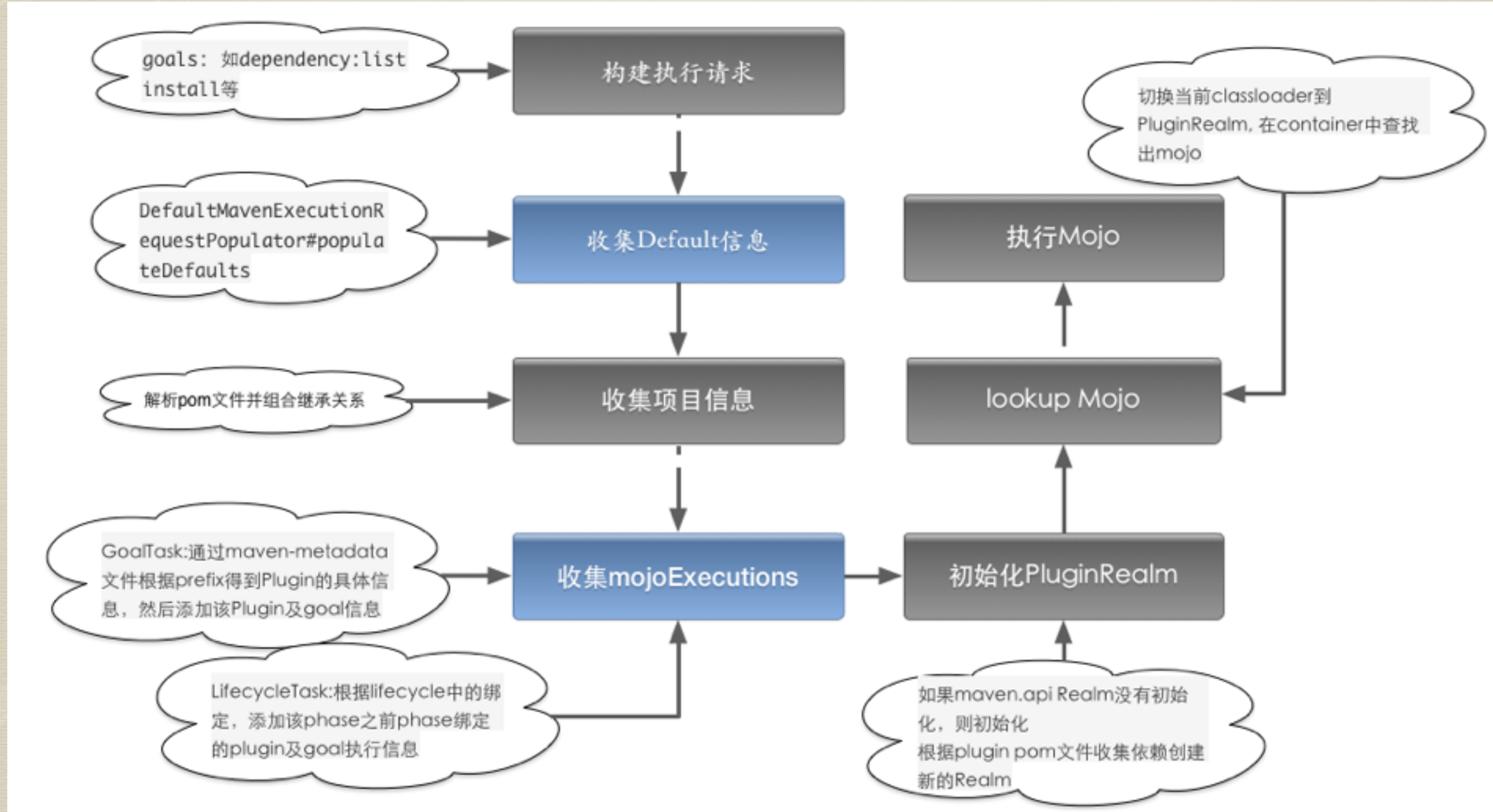
Plugin Locate

* 1. 如果是goal task，则先把prefix映射成Plugin坐标@see DefaultPluginPrefixResolver https://maven.apache.org/guides/introduction/introduction-to-plugin-prefix-mapping.html

* 根据项目模型(包括superModel)中的定义，确定该插件version.

* 确定该Plugin本地位置，如果没有找到则从远端库中去下载，最后确定该Plugin本地位置。

How mojo execute



Thanks