
Apache
maven

Build management

- It is the process of compiling and assembling a software system
- Build automation is the act of scripting or automating a wide variety of tasks
  - Compiling source code
  - Packing binaries
  - Running automated tests
  - Deploying to production system
  - Creating documentation

Advantages of Build Automation

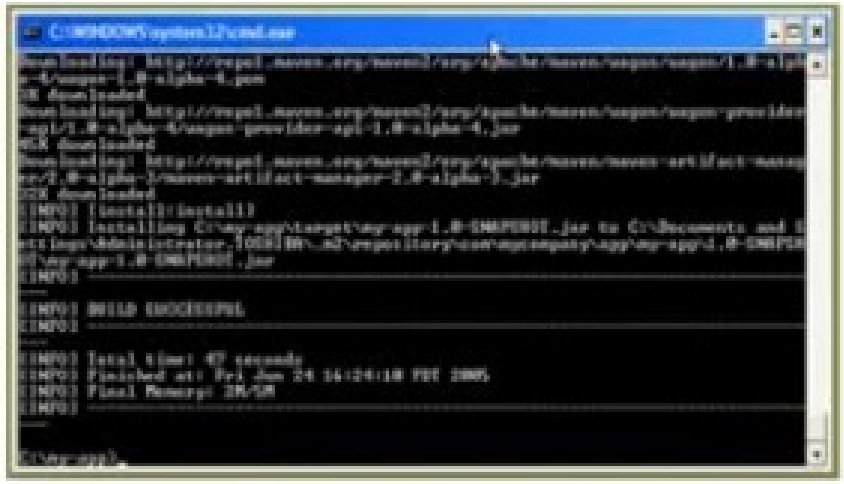
- The advantages of build automation to software development projects include
- Eliminate redundant tasks
- Accelerate the compile and link processing
- Improve product quality
- Minimize "bad builds"
- Eliminate dependencies on key personnel
- Have history of builds and releases in order to investigate issues
- Save time and money - because of the reasons listed above

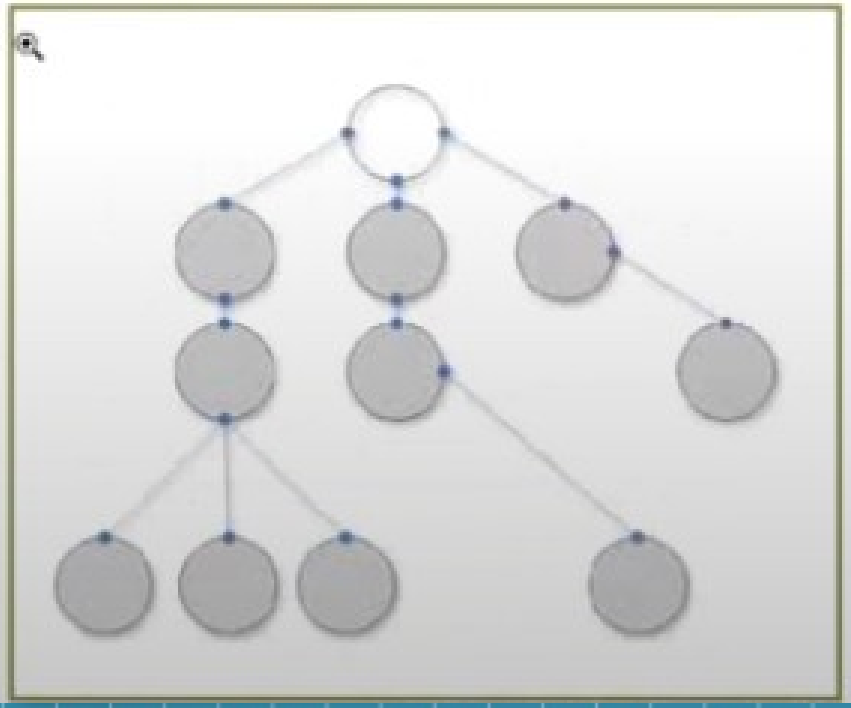
Characteristics of MAVEN

- Maven is more than just Build Tool
- Maven was built considering certain objectives
- Maven Provides:
  - Easy Build Process
  - Uniform Build System
  - Quality Project Information
  - Guidelines for Best Practices Development
- Achieved Characteristics:
  - Visibility
  - Reusability
  - Maintainability
  - Comprehensibility "Accumulator of Knowledge"

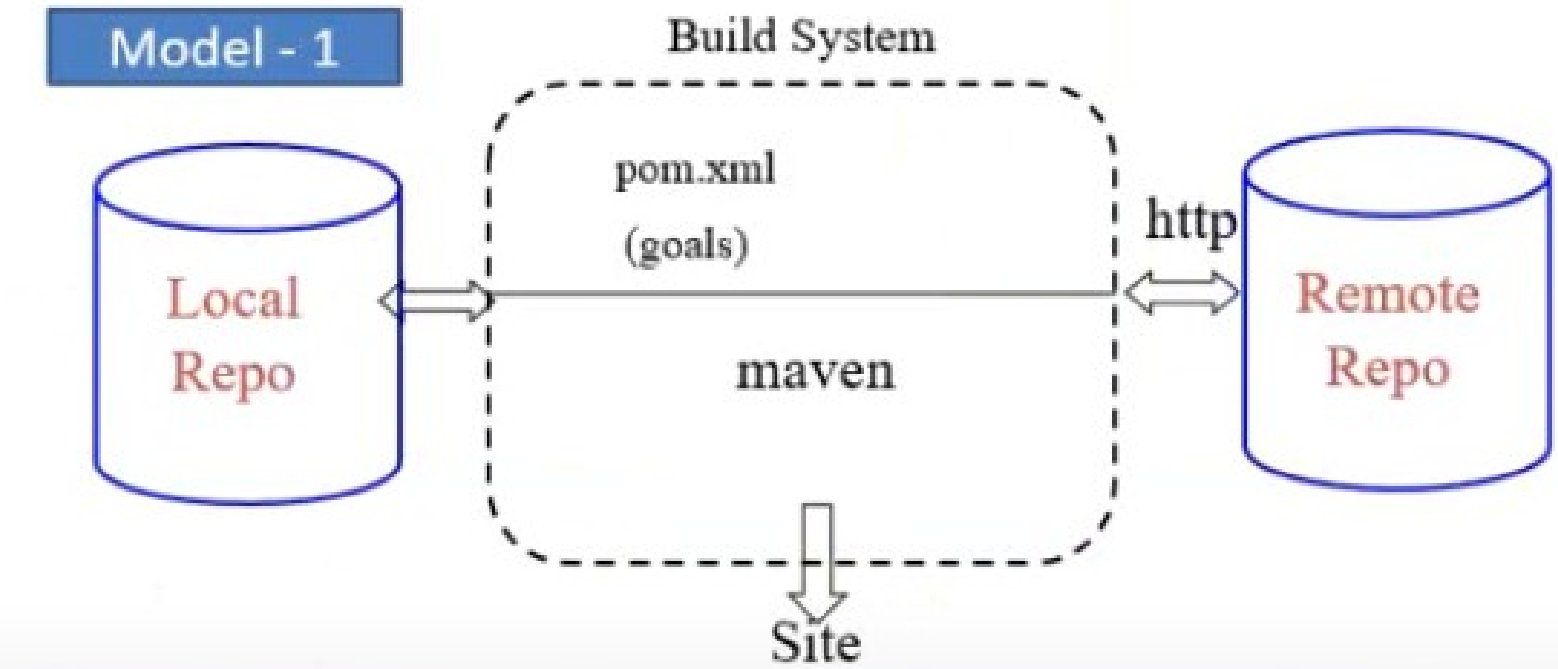
Main Features of MAVEN

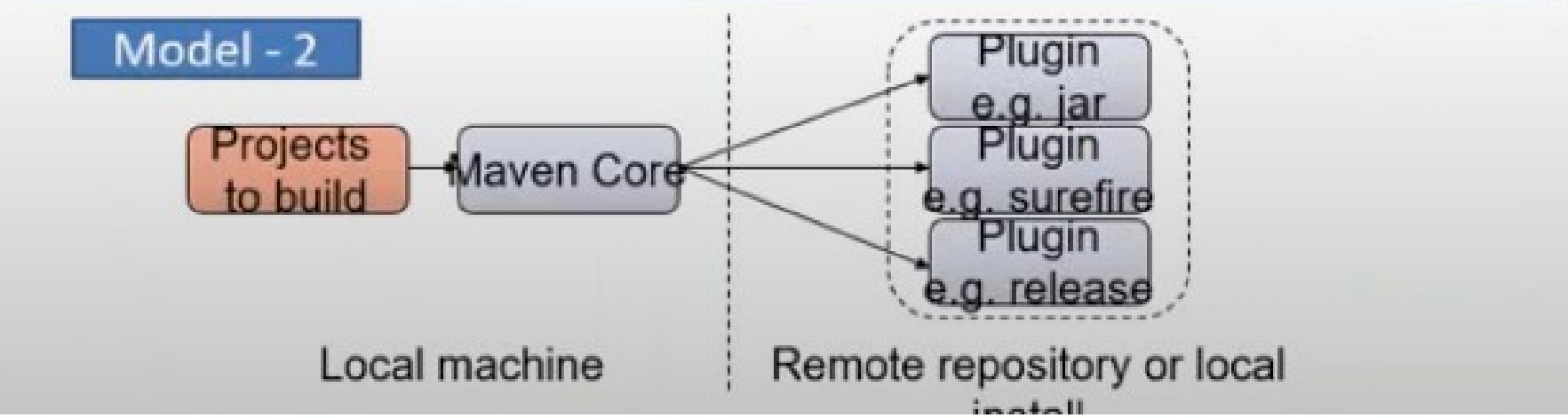
- Build-Tool
- Dependency Management Tool
- Documentation Tool

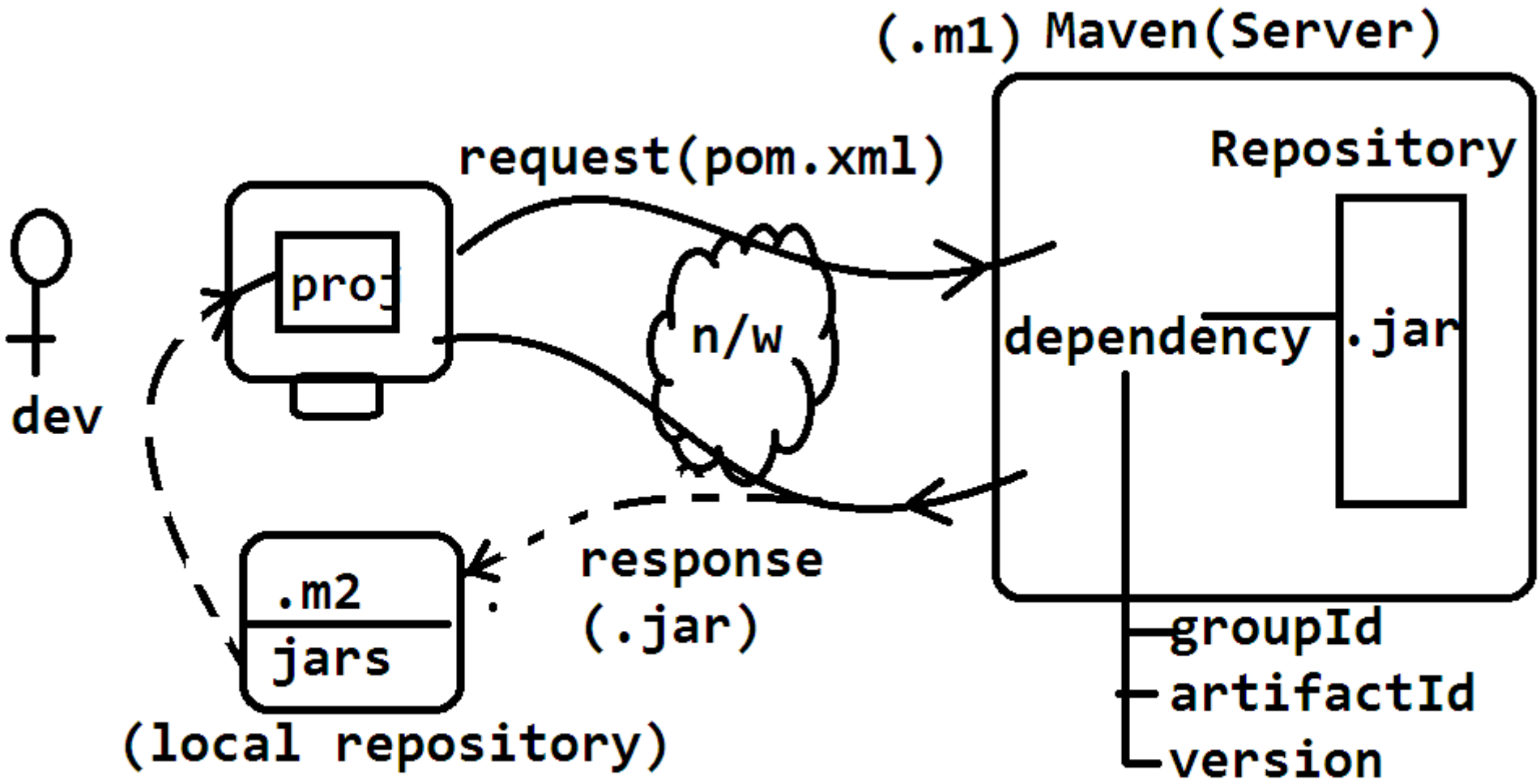




Overview of Simple Architecture






(.m1) Maven(Server)
Repository
request(pom.xml)
proj
dev
n/w
dependency
.jar
.m2
jars
response
(.jar)
groupId
artifactId
version
(local repository)

Maven Build Lifecycle

- A Maven build follow a lifecycle
- Default lifecycle
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - Install
  - deploy
- There is also a Clean, Site lifecycle

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

Standard Directory Layout

src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/assembly	Assembly descriptors
src/main/config	Configuration files
src/main/scripts	Application/Library scripts
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/filters	Test resource filter files
src/site	Site
LICENSE.txt	Project's license
NOTICE.txt	Notices and attributions required by libraries that the project depends on
README.txt	Project's readme

Project Name (GAV)

- Maven uniquely identifies a project using:
  - groupID: Arbitrary project grouping identifier (no spaces or colons)
    - Usually loosely based on Java package
  - artfiactId: Arbitrary name of project (no spaces or colons)
  - version: Version of project
    - Format {Major}.{Minor}.{Maintenance}
    - Add '-SNAPSHOT ' to identify in development
- GAV Syntax: groupId:artifactId:version
- Build type identified using the "packaging" element
- Tells Maven how to build the project
- Example packaging types:
  - pom, jar, war, ear, custom
  - Default is jar

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-training</artifactId>
    <groupId>org.lds.training</groupId>
    <version>1.0</version>
    <packaging>jar</packaging>
</project>
```

Maven Repositories

- Dependencies are downloaded from repositories
  - Via http
- Downloaded dependencies are cached in a local repository
  - Usually found in ${user.home}/.m2/repository
- Repository follows a simple directory structure
  - {groupId}/{artifactId}/{version}/{artifactId}-{version}.jar
  - groupId '.' is replaced with '/'
- Maven Central is primary community repo
  - http://repo1.maven.org/maven2

POM

- What is POM?

  POM Stands for Project Object Model

  As a fundamental unit of work in Maven, POM is an XML file that contains information about project and configuration details used by Maven to build the project"
- Describes a project
  - Name and Version
  - Artifact Type
  - Source Code Locations
  - Dependencies
  - Plugins
  - Profiles (Alternate build configurations)
- Uses XML by Default
  - Not the way Ant uses XML

Project Object Model (POM)

- Metadata: Location of Directories, Developers/Contributors, Dependencies, Repositories
- Dependencies (Transitive Dependencies), Inheritance, and Aggregation
- Key Elements
  - Project
  - Model Version
  - Group ID
  - Packaging
  - Artifact ID
  - Version
  - Name
  - URL
  - Description

Maven Plugin management

- Maven is actually a plugin execution framework where every task is actually done by plugins
- A plugin generally provides a set of goals and which can be executed using following syntax:

  % mvn [plugin-name]:[goal-name]

  % mvn compiler:compiler

Plugin Types

Build plugins : They execute during the build and should be configured in the <build/> element of pom.xml

Reporting plugins : They execute during the site generation and they should be configured in the <reporting/> element of the pom.xml

- Plugins are specified in pom.xml using plugins element.
- Each plugin can have multiple goals.
- You can define phase from where plugin should starts its processing using its phase element. You can configure tasks to be executed by binding them to goals of plugin.
- That's it, Maven will handle the rest. It will download the plugin if not available in local repository

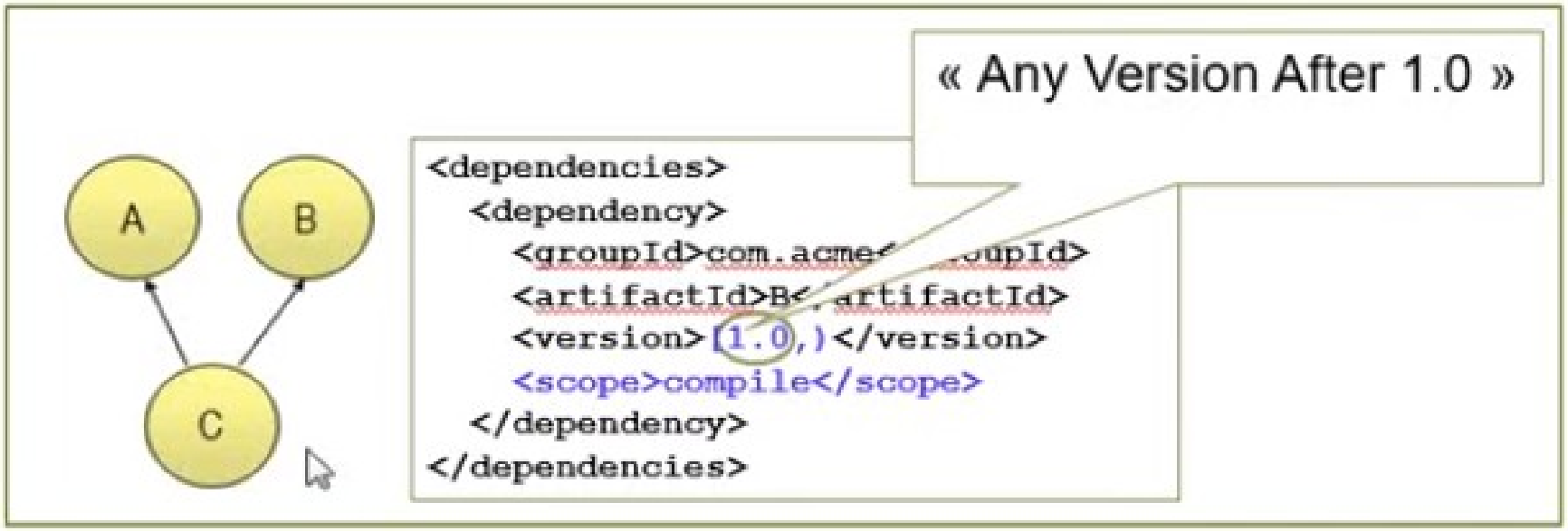
```
<project>
 <build>
  <plugins>
   <plugin>
   # what is the Plugin GAV
   # when the plugin has to be invoked
   # what the plugin has to do
   </plugin>
   <plugin>
   # what is the Plugin GAV
   # when the plugin has to be invoked
   # what the plugin has to do
   </plugin>
  </plugins>
 </build>
</project>
```

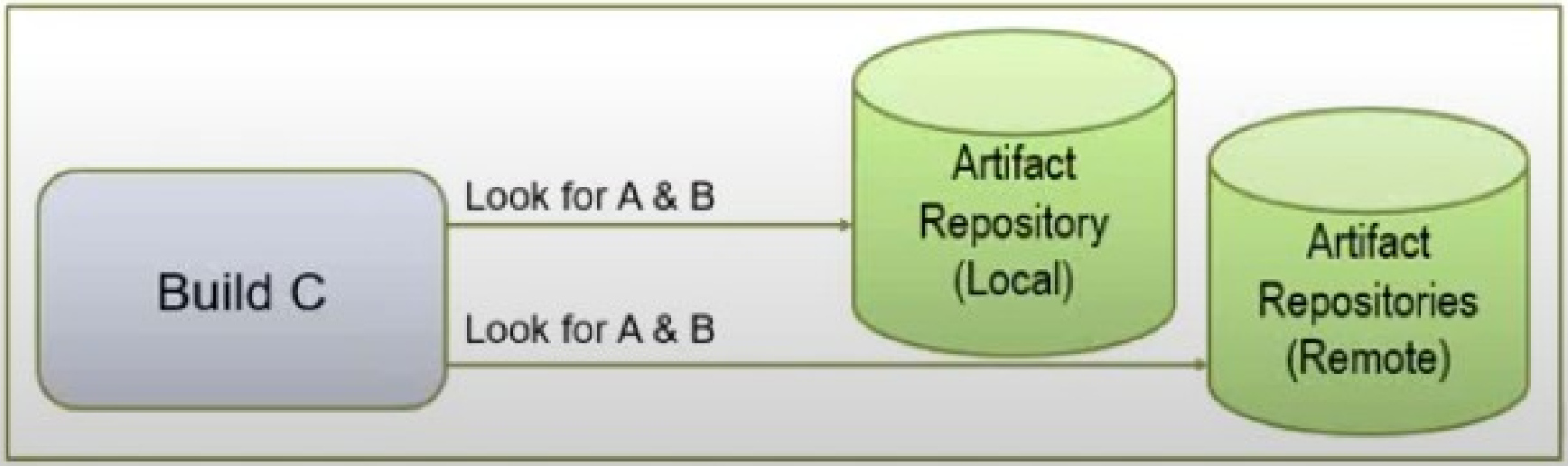
```
<project>
 <build>
  <plugins>
   <plugin>
     # what is the Plugin GAV
    <groupid> XXX </groupid>
    <artifactid> YYY </artifactid>
    <version> 123 </version>
    <executions>
     <execution>
       # when the plugin has to be invoked
       <phase> compile </phase>
       <goals>
        <goal>test</goal>
       </goals>
         # what the plugin has to do
     </execution>
    </executions>
   </plugin>
  </plugins>
 </build>
</project>
```

Maven Profile
==========
* Build profile is a set of configuration values which can be used to set or override default values of Maven build
* Using a build profile, you can customize build for different environments such as Production v/s Development environments.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.companyname.projectgroup</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
   <profiles>
     <profile>
         <id>test</id>
         <build>
         <plugins>
       <plugin>
              <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-antrun-plugin</artifactId>
                  <version>1.1</version>
                  <executions>
                    <execution>
                         <phase>clean</phase>
                         <goals>
                           <goal>run</goal>
                         </goals>
                         <configuration>
```

Dependency Management





Maven SNAPSHOTS

- A large software application generally consists of multiple modules and it is common scenario where multiple

teams are working on different modules of same application

- For ex Demo2 team uses Demo.jar
- Now if Demo team builds a new jar
  - Demo should inform every time when they release an updated code
  - Demo2 have to update their pom.xml to get the latest Demo.jar

What is SNAPSHOT?

SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

Multi Module Projects

- Maven has 1st class multi-module support
- Each maven project creates 1 primary artifact
- A parent pom is used to group modules
- To run a particular module alone

$ mvn clean –pl <modulename>

```
<project>
  <groupId>EBU</groupId>
  <artifactId>Parent-module</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
    <modules>
        <module>Child-jar</module>
        <module>child-war</module>
    </modules>
</project>
```