Maven Interview Questions & Answers

**What Is Maven?**

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

**What Does It Mean When You Say Maven Uses Convention Over Configuration?**

Maven uses Convention over Configuration which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration details.

**What Are The Aspects Maven Managed?**

**Maven provides developers ways to manage following:**

Builds

Documentation

Reporting

Dependencies

SCMs

Releases

Distribution

mailing list

**How Do You Know The Version Of Mvn You Are Using?**

**Type the following command :**

mvn --version

**What Is Pom?**

POM stands for Project Object Model. It is fundamental Unit of Work in Maven. It is an XML file. It always resides in the base directory of the project as pom.xml. It contains information about the project and various configuration details used by Maven to build the project(s).

**What Information Does Pom Contain?**

**POM contains the some of the following configuration information −**

project dependencies.

plugins.

goals.

build profiles.

project version.

developers.

mailing list.

**What Is Maven Artifact?**

An artifact is a file, usually a JAR that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR.

Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string. The three together uniquely identify the artifact. A project's dependencies are specified as artifacts.

**Name The 3 Build Lifecycle Of Maven?**

**The three build lifecycles are:**

clean:cleans up artifacts created by prior builds.

default (or build):This is used to build the application.

site: generates site documentation for the project.

**What Is Maven Build Lifecycle?**

A Build Lifecycle is a well defined sequence of phases which define the order in which the goals are to be executed. Here phase represents a stage in life cycle.

**What Is The Command To Quickly Build Your Maven Site?**

**Type the command−**mvn site

**What Would The Command Mvn Clean Do ?**

This command removes the target directory with all the build data before starting the build process.

**What Are The Phases Of A Maven Build Lifecycle?**

**Following are the phases:−**

**validate** − validate the project is correct and all necessary information is available.

**compile** − compile the source code of the project.

**test** − test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

**package** − take the compiled code and package it in its distributable format, such as a JAR.

**integration-test** − process and deploy the package if necessary into an environment where integration tests can be run.

**verify** − run any checks to verify the package is valid and meets quality criteria.

**install** − install the package into the local repository, for use as a dependency in other projects locally.

**deploy** − done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

**What Is A Goal In Maven Terminology?**

A goal represents a specific task which contributes to the building and managing of a project. It may be bound to zero or more build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

**What Would This Command Do Mvn Clean Dependency:copy-dependencies Package?**

This command will clean the project, copy the dependencies and package the project (executing all phases up to package).

**What Phases Does A Clean Lifecycle Consist?**

**The clean lifecycle consists of the following phases:**

pre-clean.

clean.

post-clean.

**What Phases Does A Site Lifecycle Consist?**

**The phases in Site Lifecycle are:−**

pre-site

site

post-site

site-deploy

**What Is Build Profile?**

A 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.

**What Are Different Types Of Build Profiles?**

**Build profiles are of three types :**

Per Project − Defined in the project POM file, pom.xml.

Per User − Defined in Maven settings xml file (%USER\_HOME%/.m2/settings.xml).

Global − Defined in Maven global settings xml file (%M2\_HOME%/conf/settings.xml)

**How Can You Activate Profiles?**

**A Maven Build Profile can be activated in various ways** −

Explicitly using command console input.

Through maven settings.

Based on environment variables (User/System variables).

OS Settings (for example, Windows family).

Present/missing files.

**What Is A Maven Repository?**

A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

**1. Explain what is Maven ? How does it work ?**

Maven is a project management tool. It provides the developer a complete build lifecycle framework. On executing Maven commands, it will look for POM file in Maven; it will run the command on the resources described in the POM.

**2. List out what are the aspects does Maven Manages ?**

Maven handles following activities of a developer

• Build  
• Documentation  
• Reporting  
• Dependencies  
• SCMs  
• Releases  
• Distribution  
• Mailing list

**3. Mention the three build lifecycle of Maven ?**

• Clean: Cleans up artifacts that are created by prior builds

• Default (build): Used to create the application

• Site: For the project generates site documentation

**4. Explain what is POM ?**

In Maven, POM (Project Object Model) is the fundamental unit of work. It is an XML file which holds the information about the project and configuration details used to build a project by Maven.

**5. Explain what is Maven artifact ?**

Usually an artifact is a JAR file which gets arrayed to a Maven repository. One or more artifacts a maven build produces such as compiled JAR and a sources JAR.

Each artifact includes a group ID, an artifact ID and a version string.

**6. Explain what is Maven Repository ? What are their types ?**

A Maven repository is a location where all the project jars, library jars, plugins or any other particular project related artifacts are stored and can be easily used by Maven.

Their types are local, central and remote.

**7. Why Maven Plugins are used ?**

Maven plugins are used to

• Create a jar file  
• Create war file  
• Compile code files  
• Unit testing of code  
• Documenting projects  
• Reporting

**8. List out the dependency scope in Maven ?**

The various dependency scope used in Maven are:

• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project

• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime

• Runtime: This tells that the dependency is not needed for compilation but is required during execution

• Test: It says dependency is available only for the test compilation and execution phases

• System: It indicates you have to provide the system path

• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM’s section

**9. Mention how profiles are specified in Maven ?**

Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

**10. Explain how you can exclude dependency ?**

By using the exclusion element, dependency can be excluded.

**11. Mention the difference between Apache Ant and Maven ?**

Apache Ant Maven

• Ant is a toolbox – Maven is a framework  
• Ant does not have formal conventions like project directory structure – Maven has conventions  
• Ant is procedural; you have to tell to compile, copy and compress – Maven is declarative ( information on what to make & how to build)  
• Ant does not have lifecycle; you have to add sequence of tasks manually – Maven has a lifecycle  
• Ant scripts are not reusable – Maven plugins are reusable

**12. In Maven what are the two setting files called and what are their location ?**

In Maven, the setting files are called settings.xml, and the two setting files are located at

• Maven installation directory: $M2\_Home/conf/settings.xml

• User’s home directory: ${ user.home }/ .m2 / settings.xml

**13. List out what are the build phases in Maven ?**

Build phases in Maven are

• Validate  
• Compile  
• Test  
• Package  
• Install  
• Deploy

**14. List out the build, source and test source directory for POM in Maven ?**

• Build = Target  
• Source = src/main/java  
• Test = src/main/test

**15. Where do you find the class files when you compile a Maven project ?**

You will find the class files ${basedir}/target/classes/.

**16. Explain what would the “jar: jar” goal do ?**

jar: jar will not recompile sources; it will imply just create a JAR from the target/classes directory considering that everything else has been done.

**17. List out what are the Maven’s order of inheritance ?**

The maven’s order of inheritance is

• Parent Pom  
• Project Pom  
• Settings  
• CLI parameters

**18. For POM what are the minimum required elements ?**

The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version.

**19. Explain how you can produce execution debug output or error messages ?**

To produce execution debug output you could call Maven with X parameter or e parameter.

20. Explain how to run test classes in Maven ?

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”

**21. What is a MOJO ?**

A MOJO stands for Maven plain Old Java Object. Each MOJO is an executable goal in Maven, and a plugin is a distribution of one or more related MOJOs.

**22. What does it mean when you say Maven uses Convention over Configuration ?**

Maven uses Convention over Configuration which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration details.

**23. What are the main features of Maven ?**

Some of the main features of Maven are:

**Simple:** Maven provides simple project setup that is based on best practices.

**Fast:** You can get a new project or module started in a few seconds in Maven.

**Easy to learn:** Maven usage and commands are easy to learn across all projects. Therefore ramp up time for new developers coming onto a project is very less.

**Dependency management:** Maven provides superior dependency management including automatic updates and transitive dependencies.

**Multiple Projects:** You can easily work with multiple projects at the same time by using Maven.

**Large Library:** Maven has a large and growing repository of libraries and metadata to use out of the box.

**Extensible:** Maven supports the ability to easily write plugins in Java or scripting languages for extending its core functionality.

**Instant:** Maven is online and it provides instant access to new features with very less configuration.

**24. What are the main advantages of Maven ?**

Maven has a long list of advantages for Software development. Some of the main advantages are:

**Common Project Structure:** By using Maven, every developer has a common project structure that helps in understanding the code as well as developing new features in a new project.

**Modular Design:** Maven promotes modular design that divides a complex project into multiple modules that are easier to manage. By using Maven, it is easier to manage multiple modules for build, test, release etc.

**Centralized Dependency Management:** With Maven, each developer does not have to include the jars separately in each project or module. Maven provides a centralized dependency management that can help improve efficiency of software development.

**Fewer Decisions:** With Maven a developer has to make fewer decisions about things unrelated to software development work. The project structure comes ready with Maven, dependency management is a uniform approach and build/release are handled by Maven. So a developer can focus on core work of developing software.

**25. Why do we say “Maven uses convention over configuration” ?**

Convention over configuration is a Software Design Paradigm that decreases the number of decisions made by a software developer, without losing flexibility.

In Maven, there are many conventions for setting up the project, building the artifacts, running unit tests and releasing the code. These conventions lead to common process for Software development.

In case of other tools, there are a lot of configuration options are present. But most of the time, a developer uses same set of configuration options. So it is better to make these as a default options. Maven uses default options from best practices and provides right conventions for Software development.

**26. What are the different types of profile in Maven ? Where will you define these profiles ?**

In Maven, we can have following types of Profile:

**Per Project**  
It is defined in the POM itself (pom.xml).

**Per User**  
We can define it in the Maven-settings (%USER\_HOME%/.m2/settings.xml).

**Global**  
It is defined in the global Maven-settings (${maven.home}/conf/settings.xml).

**Profile descriptor**  
Descriptor is located in project basedir (profiles.xml) (It is not supported in Maven 3.0)

**27. What are the differences between Ant and Maven ?**

Key differences between Ant and Maven are:

⦁ Ant is a Java library and command line toolbox for build process. Maven is a framework for many aspects of software development like- project setup, compile, build, documentation etc.  
⦁ Ant does not have any conventions for project structure or build processes. Maven has conventions for setting up project structure as well as for build processes.  
⦁ Ant is based on procedural programming. We have to write code for compilation build, copy etc. Maven is based on declarative programming. We have to just configure it for our project setup and programming.  
⦁ Ant does not impose any lifecycle. We need to create the sequence of tasks manually. Maven has a lifecycle for software build processes. There are well-defined phases that we can use in Maven.  
⦁ Ant scripts are not reusable in multiple projects. Maven has plugins that are reusable across multiple projects.

**28. What is the difference between a Release version and SNAPSHOT version in Maven ?**

A SNAPSHOT version in Maven is the one that has not been released.

Before every release version there is a SNAPSHOT version. Before 1.0 release there will be 1.0-SNAPSHOT.

If we download 1.0-SNAPSHOT today then we may get different set of files than the one we get on downloading it yesterday. SNAPSHOT version can keep getting changes in it since it is under development.

But release version always gives exactly same set files with each download.

**29. How will you verify if Maven is installed on Windows ?**

To check this, type mvn –version in cmd prompt of Windows. This will give you the version of Maven installed on Windows.

? **30. How does Maven looks for a dependency or resource**

It refers to the settings.xml to look for the repositories to look for the resource. First It looks into the configured local repository, then it looks into the configured Remote repositories. If the resource is still not found , it looks it within maven repository central i.e repo1.maven.org. If its still not found, it throws the exception saying “Unable to find resource in repository central”.

**31. What is maven repository central ?**

Its the repository provided by Maven. In case your POM specify the dependencies and its not available in the configured local and the remote repository. It then looks for the resource in Maven Central. Maven provides most of the generic dependency resources at this remote location.

**32. What would you do if you have to add a jar to the project using Maven ?**

If its already there in Maven local repository, We can add that as a dependency in the project pom file with its Group Id, Artifact Id and version.

We can provide additional attribute SystemPath if its unable to locate the jar in the local repository.

If its not there in the local repository, we can install it first in the local repository and then can add it as dependency.

**33. Have you ever had problem getting your projects in eclipse refreshed after you made changes in the Pom files ?**

Yes, It happens many times but I would usually perform mvn eclipse:eclipse and this would resolve the project refresh problems.

**34. What is the difference between compile and install ?**

Compile compiles the source code of the project

whereas

Install installs the package into the local repository, for use as a dependency in other projects locally

**35. What is a cyclic dependency ?**

A has dependency of B, B has dependency of C and C has dependency of A.

With Maven 2 , came transitive dependency wherein in above scenario, C will acts as a dependency of A as if this dependency has been defined directly in A but the negative side is that if it leads to cyclic dependency , it creates problems.

36. What is a Test Dependency Scope in Maven ?

This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases. This scope is not transitive.

**37. What is the POM packaging in Maven ?**

Pom packaging is simply a specification that states the primary artifact is not a war or jar, but the pom.xml itself.

**38. Difference between Maven Project and Module ?**

Maven Module has a Parent whereas Project doesnt. when we add the parent section to the pom file, it adds the module section to the parent project pom file. When we execute mvn compile / install, it basically checks that module section of the parent to identify all the modules that needs to be compiled first.

**39. How to specify the sequence in which sub modules needs to be built ?**

By specifying the modules in the same sequence in the parent pom.

**40. What is the best practice configuration usage for files – pom.xml or settings.xml ?**

The best practice guideline between settings.xml and pom.xml is that configurations in settings.xml must be specific to the current user and that pom.xml configurations are specific to the project.

**41. How can I change the default location of the generated jar when I command “mvn package” ?**

By default, the location of the generated jar is in ${project.build.directory} or in your target directory. We can change this by configuring the outputDirectory of maven-jar-plugin.

**42. How would you see the version of Maven** ?

mvn –version

**43. How can we see Dependencies for the project and where exactly they are defined ?**

Using

mvn dependency:tree

**44. What are the benefits of transitive depency in Maven ?**

Transitive dependencies allows to avoid specifying the libraries that are required by the project which are specified in other dependent projects – Remote or Local.

**45. What is the difference between Maven, Ant and Jenkins ?**

Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.

**46. Have you heard of Ban Duplicate Classes Maven enforcer plugin ? What is its use ?**

Yes , we have been using this plugin with our projects and its purpose is to warn and stop the Build if there are duplicates of the same package and class are being carried either directly or through transitive dependencies. the duplicate could be coming through different types of dependencies or through different versions of the same dependency. Its purpose is to make sure that there is only one copy thats being used at compile time and runtime and hence shouldnt later result in runtime problems.

**47. How to tackle duplicate classes in maven build ?**

The simplest way is to ignore them if Maven enforcer plugin is complaining about it but it may lead to runtime problems later.

We can do the dependency:tree to see from where these duplicate ones are coming and hence can exclude the duplicate one.

**48. What is Archetype ?**

Archetype is a Maven plugin which has the task of creating a maven project structure.

**49. What means SNAPSHOT in Maven ?**

SNAPSHOT is a type of version that indicates a current deployment copy. Maven checks during each build for a new SNAPSHOT version in the remote repository.

**50. What is the difference between version and SNAPSHOT ?**

Maven will download always the specified version. In case of SNAPSHOT Maven wil download the latest SNAPSHOT.

**1. What Is Gradle Framework?**

Gradle is an open source build automation system that builds based on the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration.

**2. Advantages Of Using Gradle?**

Gradle combines both Ant and Maven, taking the best from both of these frameworks; Flexibility from Ant tool and convention over configuration, dependency management and plugins from Maven.

Gradle provides support for multi-project builds.

**3. What Is Gradle Wrapper?**

A wrapper is a batch script and it is one of the ways to perform Gradle build. When executed the first time, it automatically downloads Gradle and then initiate the build.

It helps to setup Gradle workspace quickly for first-time users (Zero installation) and also ensure all the developers use the same version of Gradle.

**4. Why Gradle Is Preferred Over Other Build Tools?**

Gradle build script is written using groovy API which has the syntax similar to Java so it is easy to understand.

Gradle supports ant tasks, ivy and Maven repositories for dependency management. It also has a maven Pom.xml converter to Gradle build script.

It is open source.

provides strong support for multi project builds.

supports build cache.

**5. What Is The Gradle Build Script File Name?**

build.gradle.

**6. What Is The Latest Version Of Gradle Available In The Market?**

Gradle build tool latest version is 3.5 as of May 2017. The version on your local Gradle installation can be checked using gradle -v command.

**7. How Do You Run Gradle Build?**

Execute Gradle build using gradle command.

**8. What Are The Core Components Of Gradle Build Script?**

Project and task are the core compoents. Groovy organizes projects as a list of tasks.

To view the list of available projects, use the command gradle projects, for the tasks list the command is gradle tasks.

**9. How Do You Find Gradle Project Dependencies?**

Use the Gradle command gradle dependencies that lists the dependencies of the selected project. It includes both direct and transitive dependencies.

**10. The Main Difference Between Maven Build.xml And Build.gradle Script?**

Maven build.xml is xml document that includes start and end tags. Build.gradle is a Groovy script which has syntax similar to Java.

**11. How Do I Check Out And Build The Framework?**

Framework uses a Gradle-based build system. In the instructions below, ./gradlew is invoked from the root of the source tree and serves as a cross-platform, self-contained bootstrap mechanism for the build.

**Prerequisites**

Git and JDK 8 update 20 or later

Be sure that your JAVA\_HOME environment variable points to the jdk1.8.0 folder extracted from the JDK download.

**Check out sources**

git clone git@github.com:spring-projects/spring-framework.git

**Import sources into your IDE**

Run ./import-into-eclipse.sh or read import-into-idea.md as appropriate.

Note: Per the prerequisites above, ensure that you have JDK 8 configured properly in your IDE.

**Install all spring-\* jars into your local Maven cache**

./gradlew install

**Compile and test; build all jars, distribution zips, and docs**

./gradlew build

... and discover more commands with ./gradlew tasks. See also the Gradle build and release FAQ.

**Contributing**

Pull requests are welcome; see the contributor guidelines for details.

**Staying in Touch**

Follow @SpringCentral as well as @SpringFramework and its team members on Twitter. In-depth articles can be found at The Spring Blog, and releases are announced via our news feed.

**License**

The Spring Framework is released under version 2.0 of the Apache License

**12. How Long Should A Build Take? When Running `./gradlew Build` For The First Time, It Is Likely That The Gradle Wrapper Script Will Need To Download Gradle For You. Gradle Distributions Run Around 30mb, So This Step Will Be Connection-speed Dependent?**

You will also need to download all of Spring's compile- and test-time dependencies. This currently runs around 120MB. Keep in mind here that almost all of these dependencies are optional at runtime for applications that use Spring. It's only when building from source that you actually need them all!

Once you've bootstrapped a Gradle distribution and downloaded dependencies, you won't need to do it again; they are cached in your $HOME/.gradle directory. A complete ./gradle clean build at this point will take between 5 and 15 minutes depending on your clock and disk speed. A solid state drive makes a huge difference here!

As is also mentioned below in the 'tips' section, you'll want to break yourself of any habits executing the clean task during normal development iterations. Gradle has excellent incremental build support, meaning that once you've built Javadoc, compiled, run test, etc, Gradle won't execute these tasks again unless the inputs for those tasks (e.g. .java files) have changed. As a general rule, just run ./gradle build or ./gradle test (without clean) to keep things snappy.

Also, consider running with the -a flag to avoid evaluating other subprojects you depend on. For example, if you're iterating on changes in spring-webmvc, cd into the spring-webmvc directory and run ../gradlew -a build to tell gradle to evaluate and build only that subproject.

**13. How Do I Configure The Gradle Daemon To Speed Up Builds?**

The Gradle daemon helps greatly in eliminating startup overhead. This feature may potentially be enabled by default in the future, but in the meantime you need to instruct Gradle to launch the daemon process. This can be achieved by passing the --daemon flag to gradle at the command line, by exporting a GRADLE\_OPTS environment variable that includes -Dorg.gradle.daemon=true, or by adding org.gradle.daemon=true to the gradle.properties file in your gradle user home directory (e.g., ~/.gradle/gradle.properties).

If you are building against JDK 9 and using the Gradle daemon, you may encounter an Unrecognized VM option error which halts the build. To avoid this error, you can add org.gradle.jvmargs=-XX:MaxMetaspaceSize=1024m -Xmx1024m to the gradle.properties file in your gradle user home directory. See also GRADLE-3256 for details.

**14. Why Are Compile-time Warnings Suppressed? You'll Notice That `build.gradle` Includes The Following Line?**

[compileJava, compileTestJava]\*.options\*.compilerArgs = ['-Xlint:none']

This tells Gradle to suppress all warnings during compilation. The reason for this is that the framework currently has many warnings, most of which are related to generics usage -- particularly raw type warnings -- e.g. using Class instead of Class<?>. This is an artifact switching to Java 5 in Spring 3. As with the Javadoc warnings mentioned above, committers are encouraged to fix these warnings whenever possible. Once the bulk of them are eliminated, we can switch to -Xlint:all. In the meantime, it's just creates unnecessary noise in the build output.

**15. How Do I Perform A Milestone, Rc, Or Ga Release?**

The steps are simple, and almost everything is done via the Bamboo and Artifactory UIs.

**One-time setup**

Configure your CI build plan to use the Artifactory Maven 3 or Artifactory Gradle tasks as appropriate. For "Deployer Username", use "buildmaster" (password on request).

**Steps at a glance**

Stage the release into the libs-staging-local repository

Verify and test the staged artifacts

Promote the release to libs-milestone-local (or libs-release-local as appropriate).

Merge release branch

Announce the release

**Steps in detail**

**Stage the release**  
  
The Artifactory Bamboo plugin mentioned above also includes sophisticated Release Management capabilities. This feature allows for publishing releases directly from CI, including creating a release branch and/or tag; incrementing the project version; and publishing to the libs-staging-local, libs-milestone-local or libs-release-local repositories as appropriate.  
  
To access this feature, click on the "Default Job" for the Spring 3.2.x build plan, where you'll see a link to "Artifactory Release Management". Fill out the form fields there and click "Build and Release to Artifactory". Typical values -- in this case for a milestone release.

**Verify staged artifacts**  
  
When the staging build and release process is complete, you can navigate to the associated build record in Artifactory to verify that all modules were published as expected

**Promote the release**  
  
When verification is complete, return to the build in Bamboo from which you staged the release and click 'Default Job' and 'Artifactory' at the top, below the Job status bar. Make sure you have the side-bar shown in order to see this.

**Merge the release branch**  
  
At this point, the release is complete and successful, so the release branch should be merged back into master.

**Announce the release!**  
  
At this point, announcements may be made and users may consume the released artifacts by adding http://wisdomjobs/libs-milestone-local to their build scripts.

**16. What About Publishing Artifacts To Maven Central?**

This allows for maximum convenience for the majority of Spring users, given that most users have Maven-based builds and Maven resolves artifacts by default from Maven Central.

The preferred way of releasing artifacts to Maven Central is via Sonatype's Nexus server at oss.sonatype.org (OSO). This is explained in detail in Sonatype's OSS usage guide.

The Spring Artifactory repository has been customized with a "nexus-push" plugin that allows for automatic propagation of builds from Artifactory to the Nexus server at OSO for publication into Maven Central.

All Spring projects -- that is, all projects having groupid org.springframework -- can publish to OSO under the shared 'springsource' account. This has already been set up in the nexus-push plugin, so there's no additional setup necessary at OSO, even for new projects.

The Artifactory Bamboo plugin supports use of the nexus-push plugin through it's UI. Step 3 of the the FAQ entry above on publishing releases described the process for promoting a build out of staging. If the build is a GA release, simply choose the 'Push to Nexus' option, and select 'libs-release-local' as the target repository.

**17. When Will I Be Able To Play With This?** Right now. See the samples README for details on how to get started.

**18. What's The Overall Roadmap?**

a subsequent milestone release of Gradle Script Kotlin will ship with the forthcoming Gradle 3.0. We'll be posting further information about the roadmap to Gradle Script Kotlin 1.0 GA soon.

**19. Is Using Groovy For My Build Scripts Deprecated?**

No. Gradle's Groovy support is not deprecated, and will continue to be supported.

**20. Will Existing Plugins Still Work When I Write My Build Logic In Kotlin?**

Yes, they will continue to work without a problem. Gradle plugins can be developed in any JVM language and they interoperate well with each other as well as with build scripts written in Kotlin or Groovy.

**21. Do I Have To Use Intellij Idea When Using Kotlin For Gradle?**

No. Although JetBrains is the company behind IDEA and also the inventor and driving force behind Kotlin, JetBrains is also committed to providing Kotlin support for Eclipse. The IDE support for writing Gradle build logic in Eclipse will actually improve with Kotlin once we integrate the Eclipse Kotlin support into Buildship, the Gradle plugin for Eclipse.

**22. In What Language Should I Develop My Plugins?**

You can develop your plugins in any JVM language, but as part of this effort, we are working on making Kotlin the language of choice for developing Gradle plugins. Stay tuned!