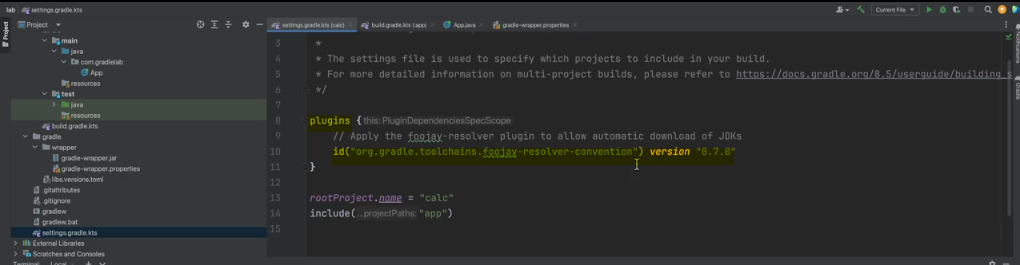
https://dpeuniversity.gradle.com/courses/012de84f-fcd3-45d4-9c4c-284382eb3f3f/activities/9313adb0-d147-4465-bd39-11392cee4756

Plugins – 4:59

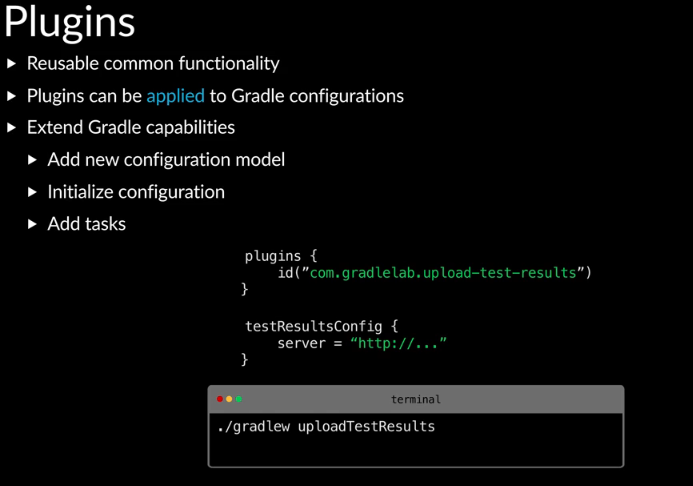
Plugins provide reusable common functionality. When using a plugin, the terminology is that the plugin is being applied. Plugins can be applied to both the settings file as well as sub project build files.



The settings plugin are a more advanced topics, we will focus on plugins applied in build files for this training. It is important to understand that different sub projects can have different plugins apply to them. Sub projects may require a different tasks to build them, hence, different plugins can be applied, which provide the different required tasks. Plugins apply to a project can do three things in general. First, they can add new configuration model for which values can be specified. Second, they may initialize the new configuration to default values. Finally, they can add functionality by making new tasks available, for example, if we apply the plain that allowed us to upload test results to a server, after applying the plugin, there would be a new configuration available that allows us to specify the URL of the server to upload to.



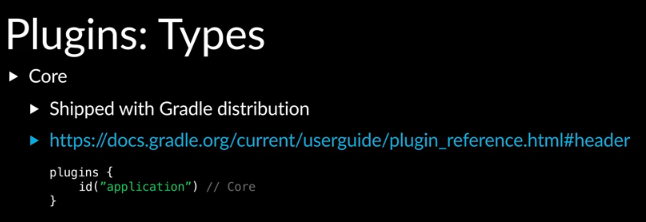
The plugin may also initialize this URL to default location, maybe local host. Finally, the plugin will add a new task that we can run that will perform the upload after reading the server URL configuration.



There are three types of project plugins; core, community and local.

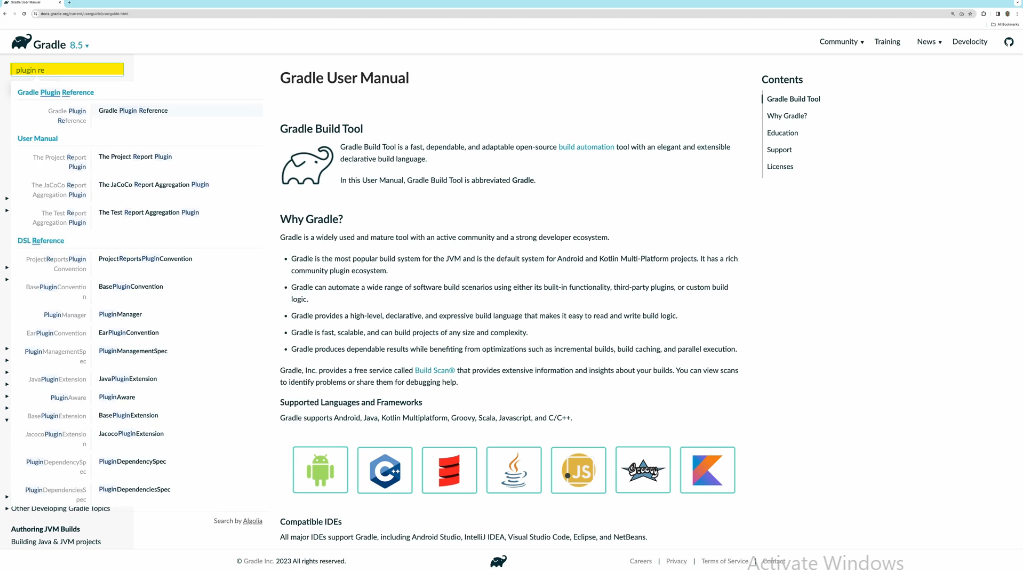


Core plugins are shipped with the Gradle build tool binary.

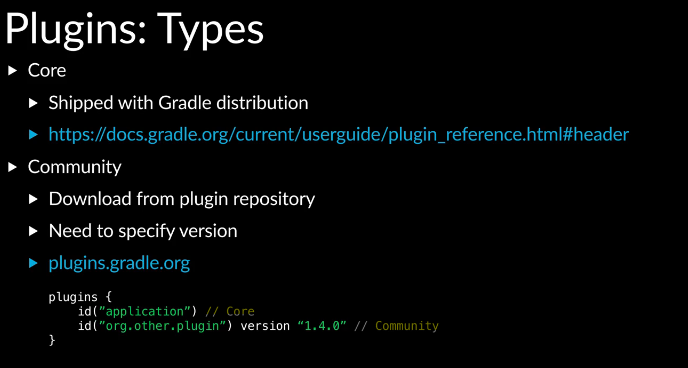


You can refer to the docs to see all the core plugins.

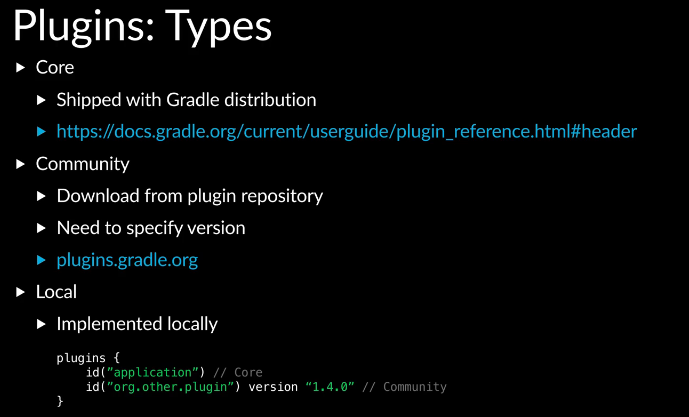
In the search bar, you can type plugin reference.



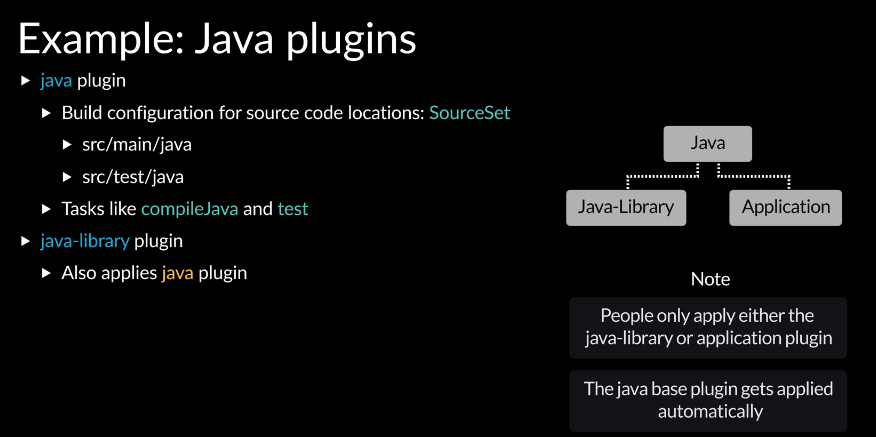
Community plugins are those created by members of the Gradle build tool community and typically are downloaded from a plugin repository.



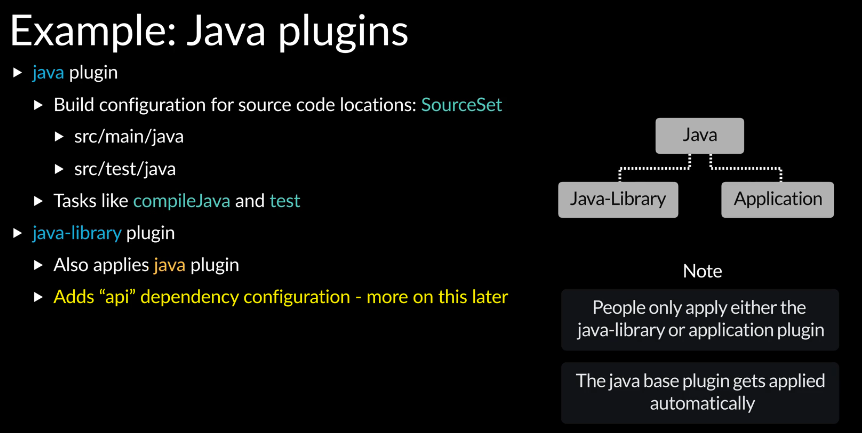
When applying a community plugin, you need to specify which version of the plugin to use. With core plugins, you do not need to specify the version since Gradle build tool will use the version available in the binary. You may also create a plugin for some additional functionality or some common configuration that is specific to a project and does not make sense to publish and share with others. You can check this plugin into version control alongside your project code and use it in the build configuration as a local plugin. The core Java plugins are a good example to look at.



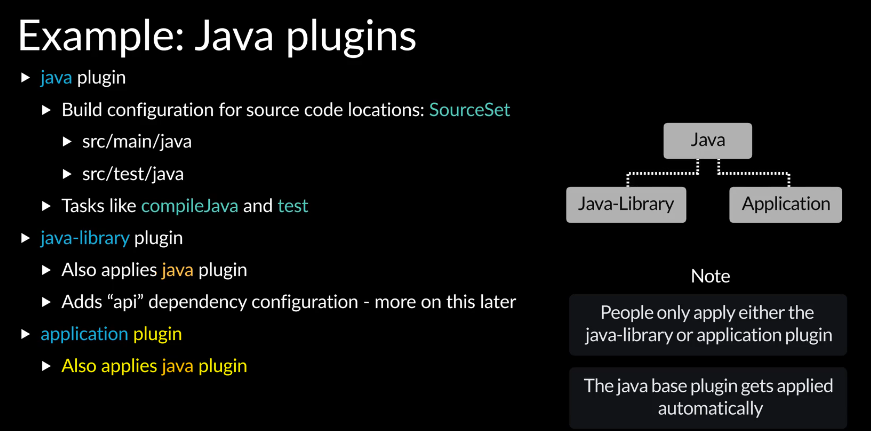
There are three plugins, the base Java plugin, the Java library plugin and the application plugin. The base Java plugin provides a number of new configuration that can be specified such as the location for the source code known as SourceSet. It also initialize some of the configuration with default values, for example, SourceSet is initialized with locations such as source main Java. The base Java plugin also provides a number of tasks to compile and test the code. The Java library plugin builds upon the base Java plugin.



Whenever you apply the Java library plugin, the base Java plugin is automatically applied. You do not need to explicitly apply the base Java plugin. If you’re working on a library that will be published and used by others as a dependency, such as a math library, you will want to apply the Java library plugin. This plugin adds some dependency management configuration options, which we will go over later in the training.

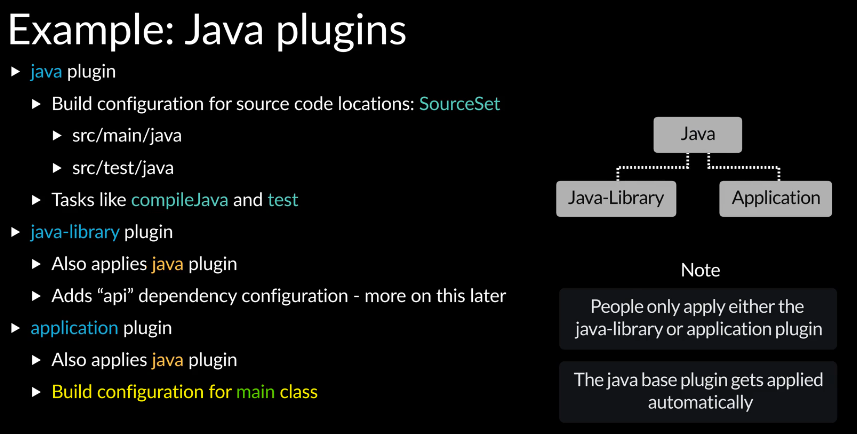


If you’re working on an executable such as a calculator application, you will want to apply the application plugin, which also builds upon the base Java plugin. Whenever you apply the application plugin, the base job or plugin is automatically applied, you do not need to explicitly apply the base Java plugin.

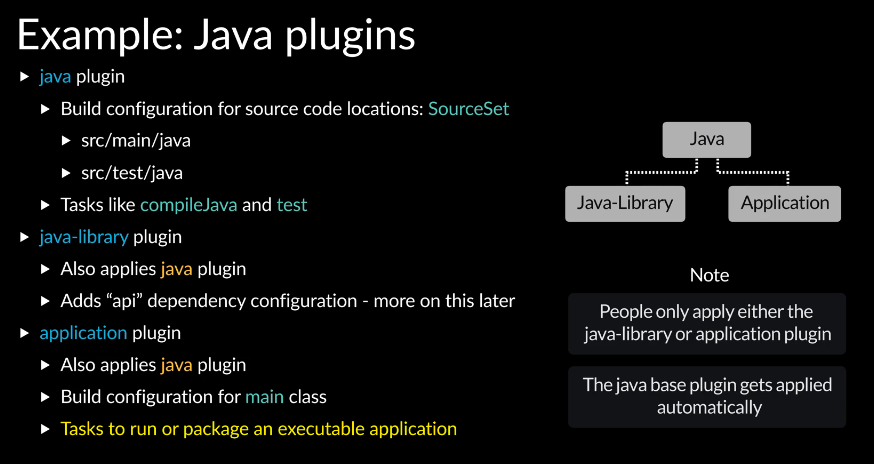


You can see this in the configuration generated by Gradle init. Only the application plugin is applied.

The application plugin also adds additional configuration options for specifying the main task of the executable.



It also adds a task to run the application.



Completed…