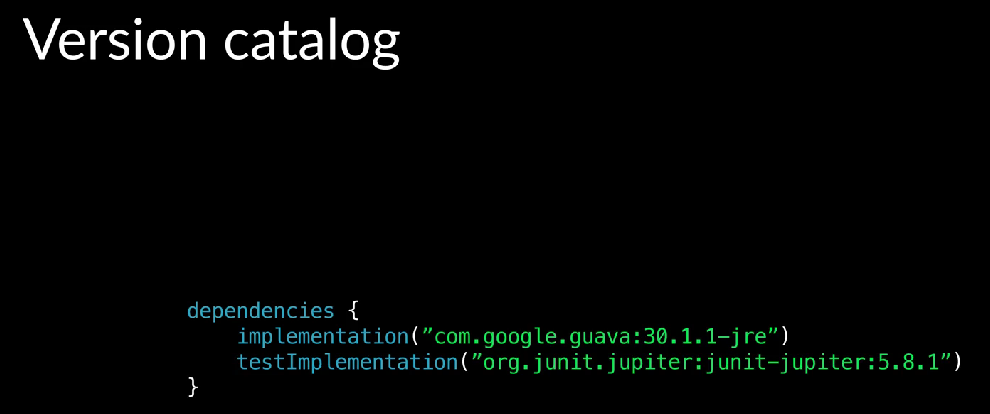
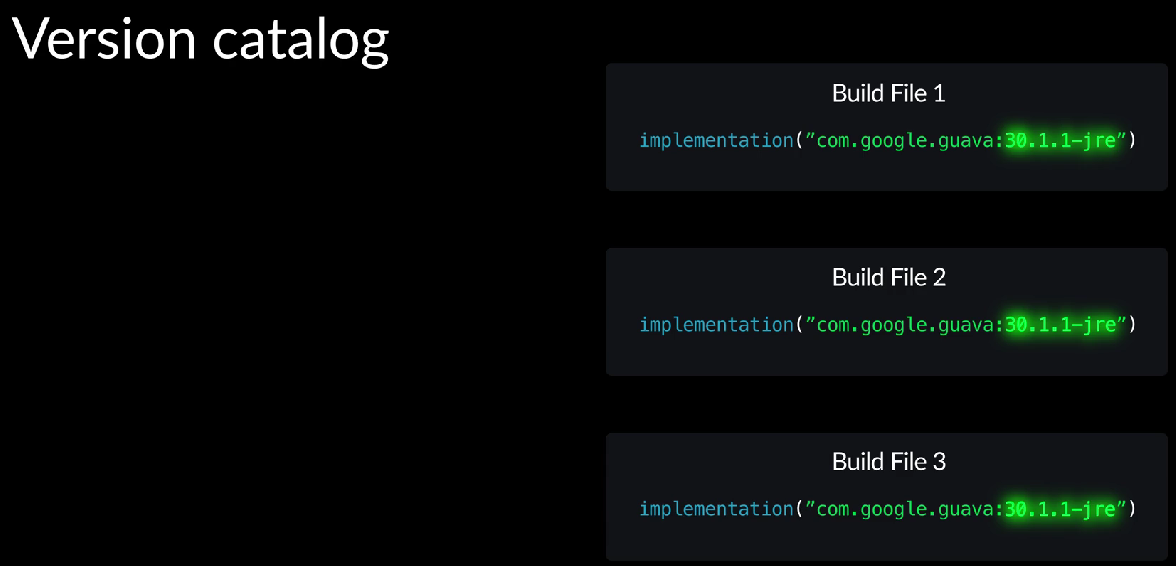
Version catalog – 9:12 minute

Learn how use the version catalog to better manage dependencies especially as your project scales.



As we have seen, you can define library dependencies and build files in the dependency configuration block. If there’s a library dependency that appears in multiple build files, if the version needs to be updated, it would have to be done in multiple locations, which can lead to mistakes or some definitions not being updated.



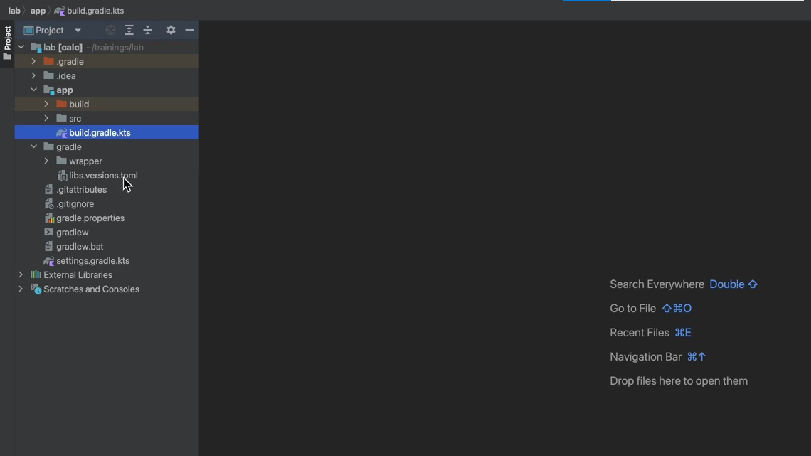
Gradle build tools support declaring libraries and versions in a central location, which can then be referenced in the build files.



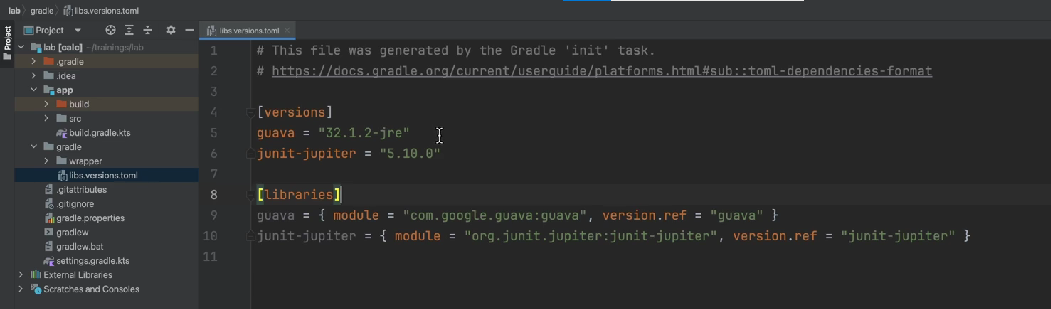
That way, updates need to be made in one place and reduces the chances of mistakes.



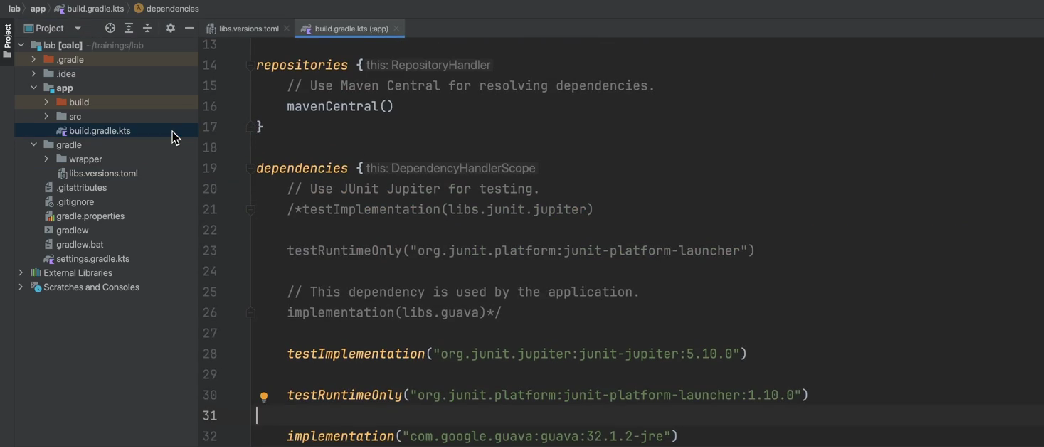
This is known as the version catalog. The default location for the version catalog is in the Gradle directory in a file called libs.versions.toml.



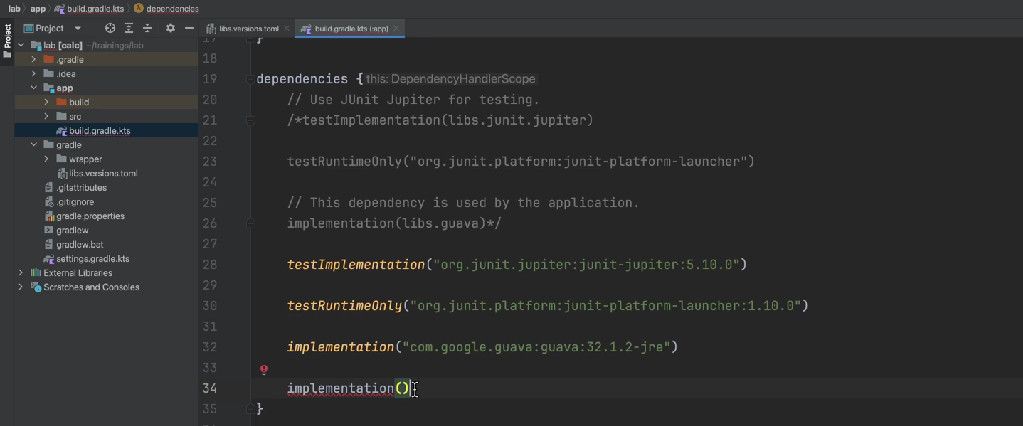
Gradle init created this file for us. There’s a section where you can define the libraries that are dependencies, and a section where you can declare the version for the libraries, which can be referenced by the library section.



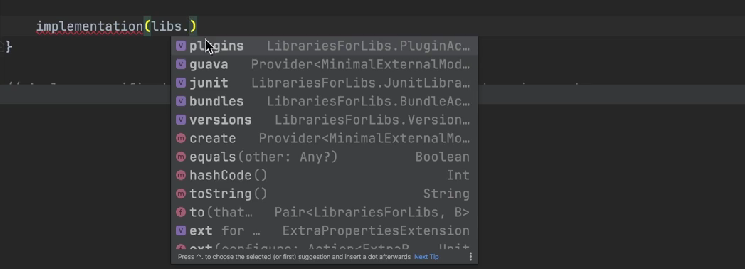
It’s important to understand that only the libraries and versions are declared in the version catalog, not the bucket dependency configurations. That is, we declare what libraries will be used, not how they will be used. Libraries in the version catalog can be referenced in build files. Let’s say we wanted to reference the Guava library defined in the version catalog and added to the implementation bucket dependency configuration for the apps sub project. We can go to the build file,



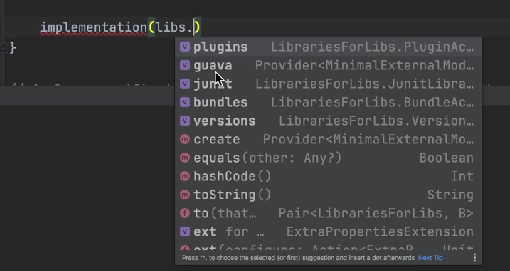
in the dependencies configuration, we can write the bucket dependency configuration, and then reference version catalog using the variable libs.



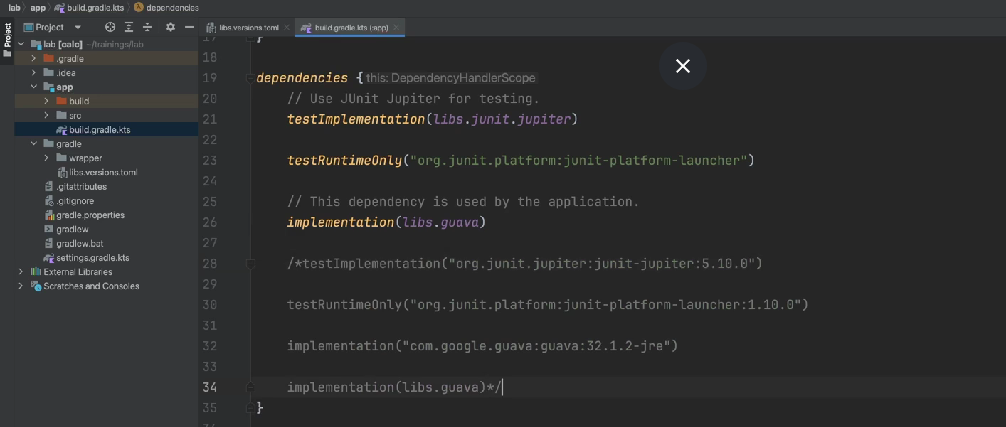
We can see,



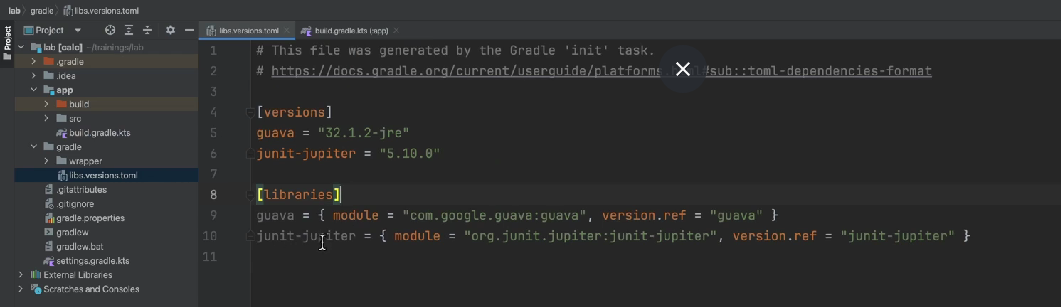
the library is available plus some other things that we’ll go over, and we can add the guava library.



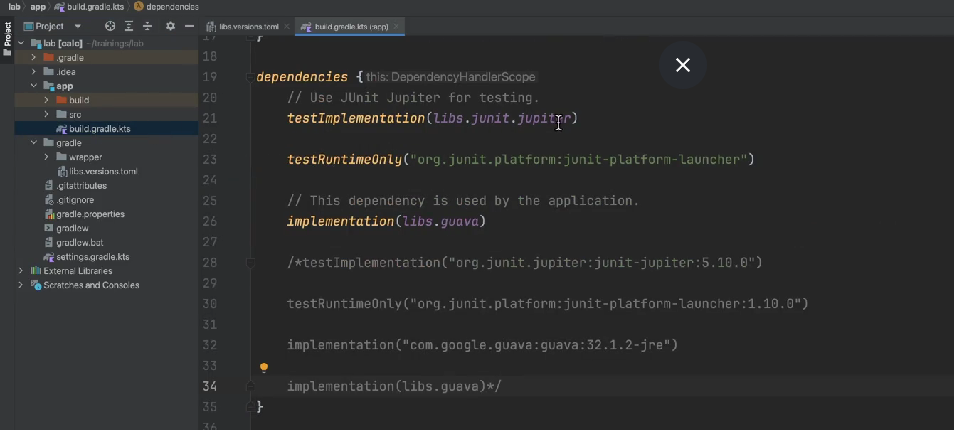
References to libraries in the version catalog are type safe. The reference is through a variable. This is another means of reducing mistakes. We can see the configuration Gradle init created reference the libraries in the version catalog. Let’s uncomment that configuration and comment out the changes that we had made.



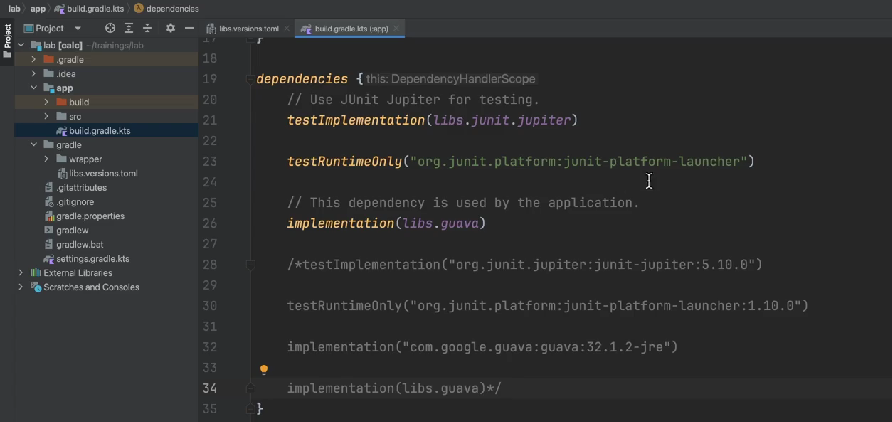
Note, for the Junit Jupiter library, in the version catalog,



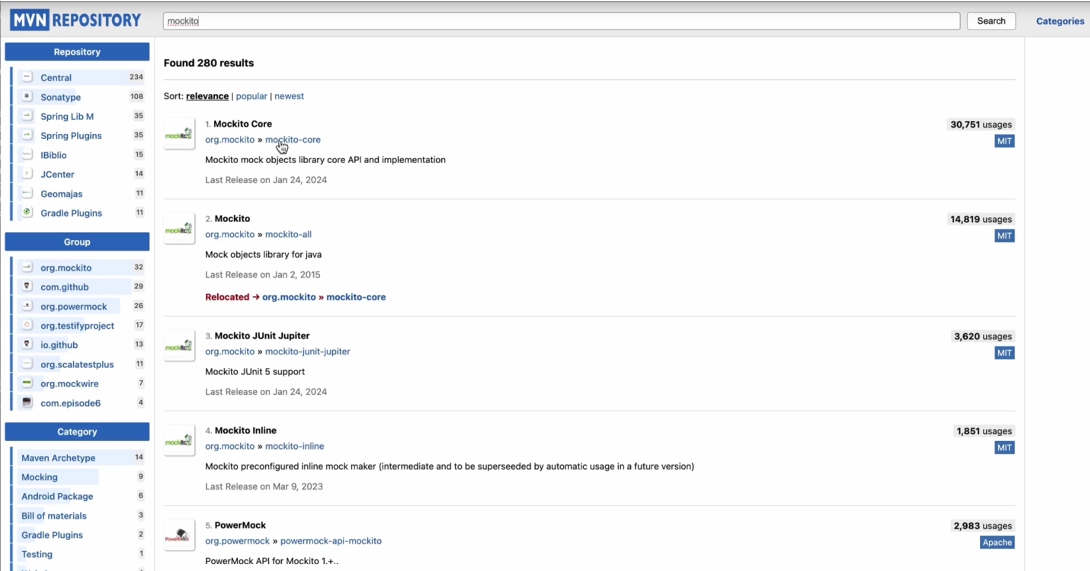
it was dash ( - ) junit Jupiter. Whereas in the build file,

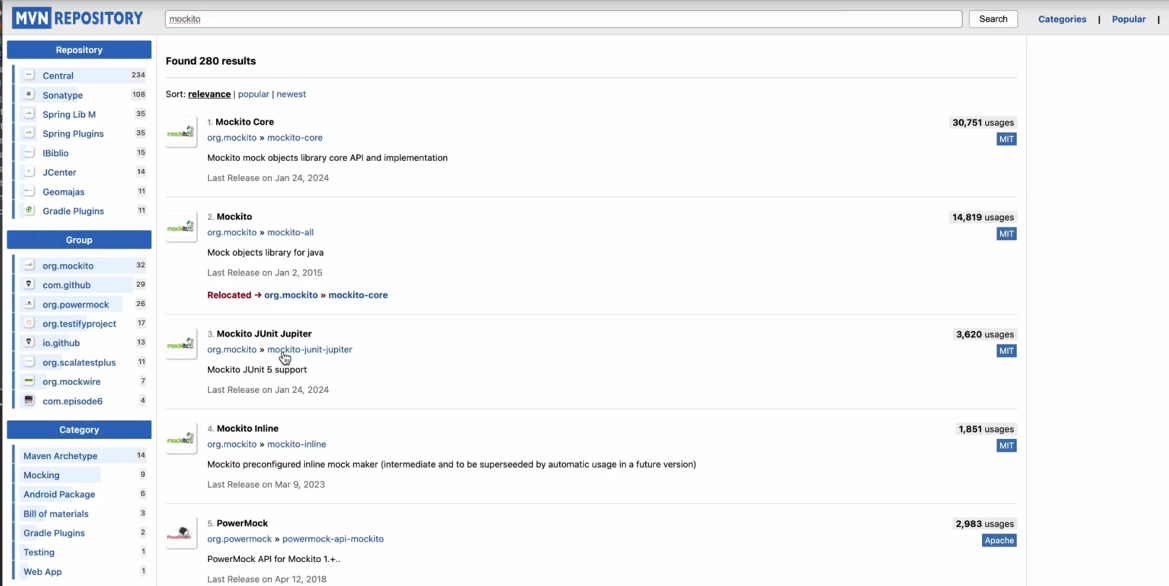


it’s junit.jupiter. The dashes in the version catalog are treated as kind of deliminators, and dots ( **.** ) are used instead in the build file. We’ll go over this again in another example coming up.

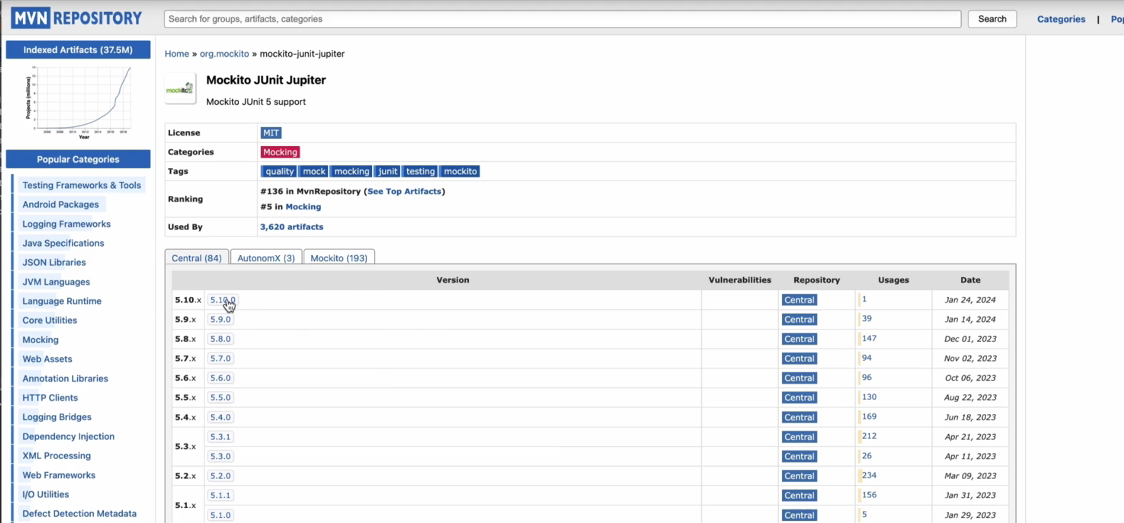


You may also notice the Junit platform launcher does not have a version associated with it. This is a more advanced topic that is covered in future dependency management training. Let’s go over an example of adding a couple of libraries with the version catalog. Mockito is a popular library that provides a mocking framework for testing. Let’s assume we wanted to use two Mockito libraries, Mockito Core and Mockito Junit Jupiter.

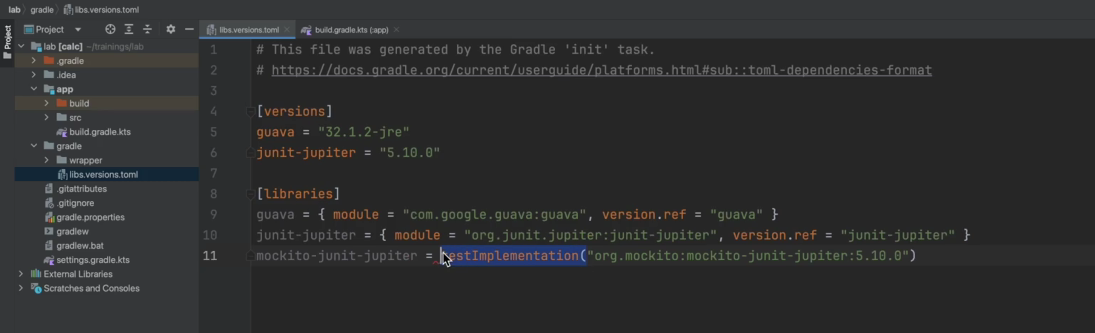


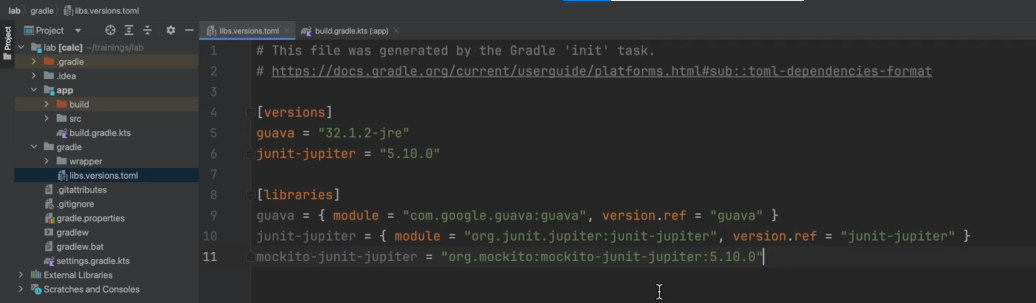


We wanna use the same version for both libraries. 5.10 for the Mockito Core, and also 5.10 for Junit Jupiter.

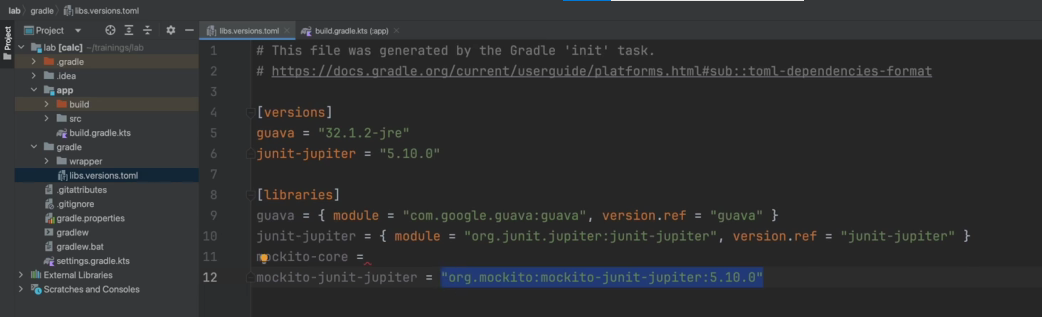


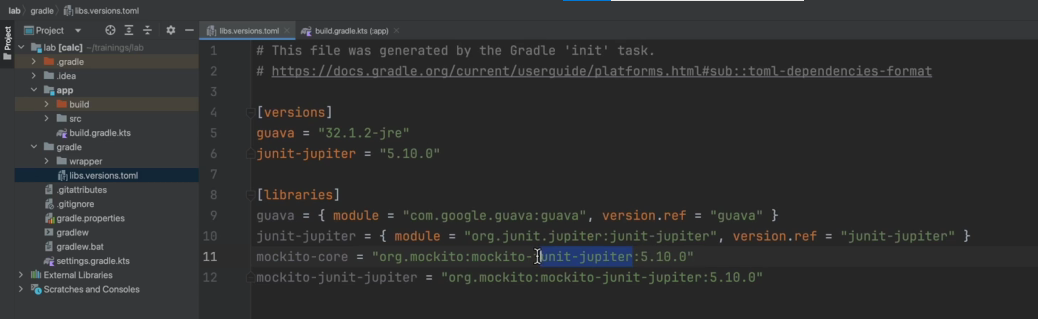
In the version, catalog we can add the Mockito junit Jupiter library. We can remove the dependency configuration,

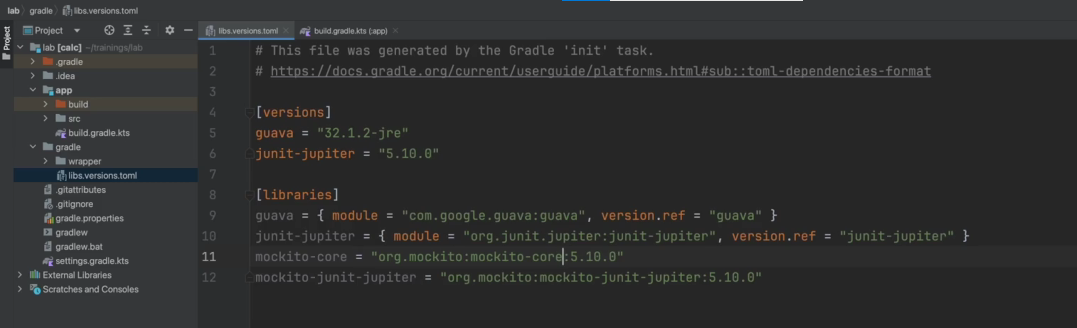




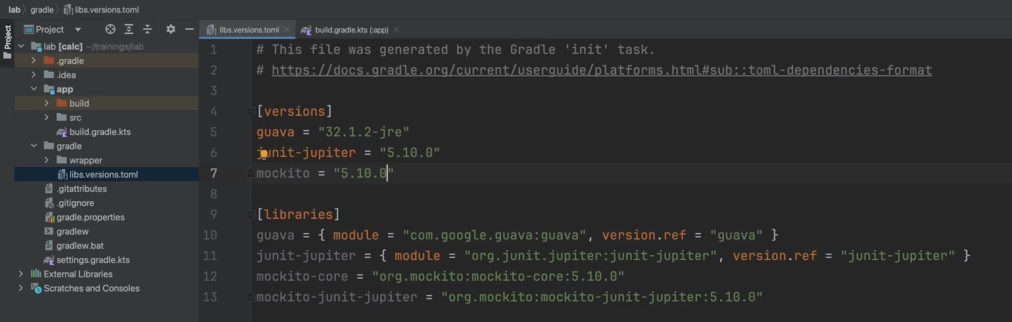
and this single string format is also supported in the version catalog with both the module ID and the version together. Let’s also add the Mockito core library.



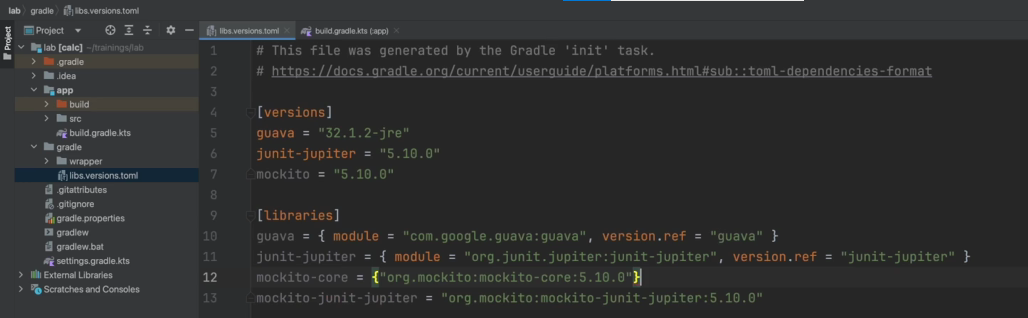


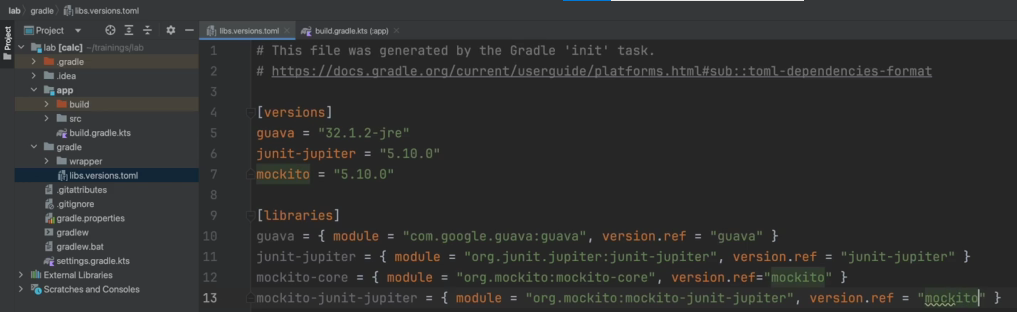


To make things easier to manage and prevent mistakes when updating the version in the future, we can create a version for both Mockito libraries.

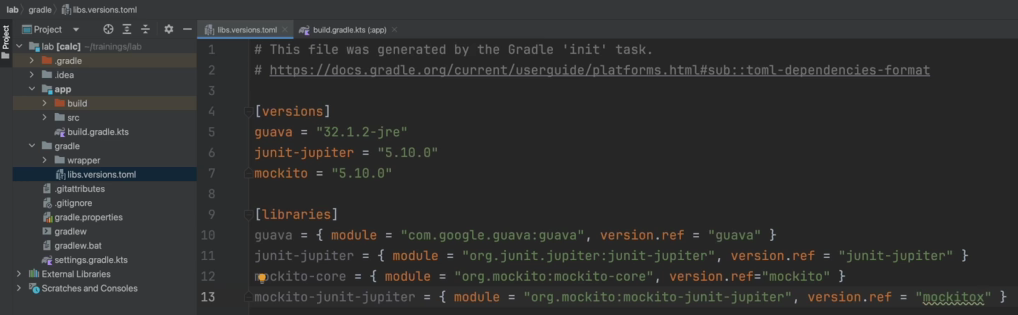


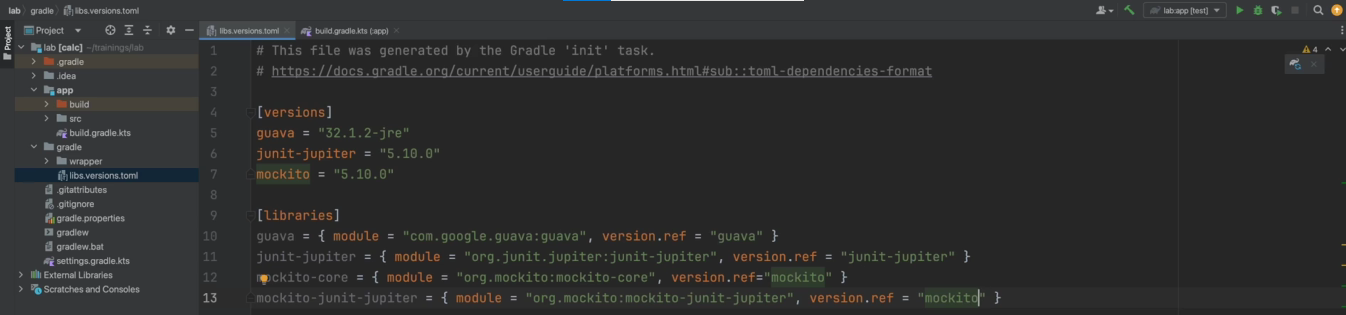
And update the two library declarations to reference the same version.



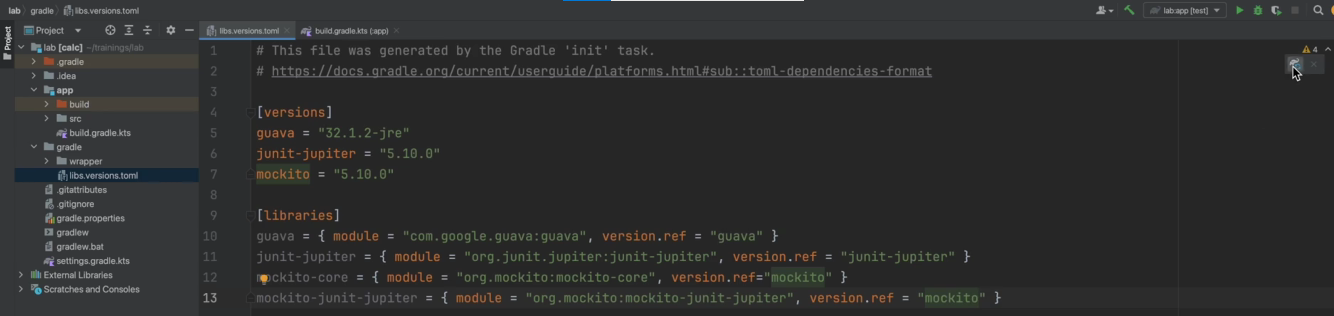


Note that if we make a typo in the version reference, the editor alerts us that there is an issue.

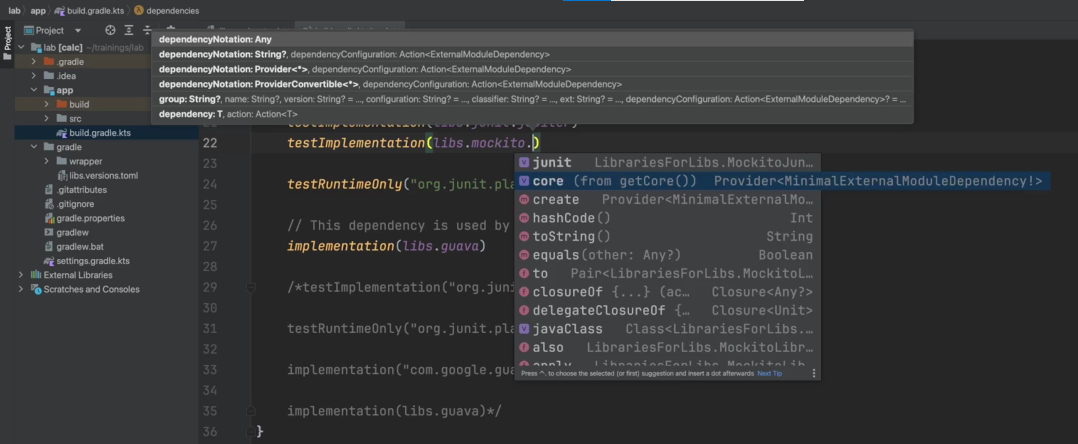




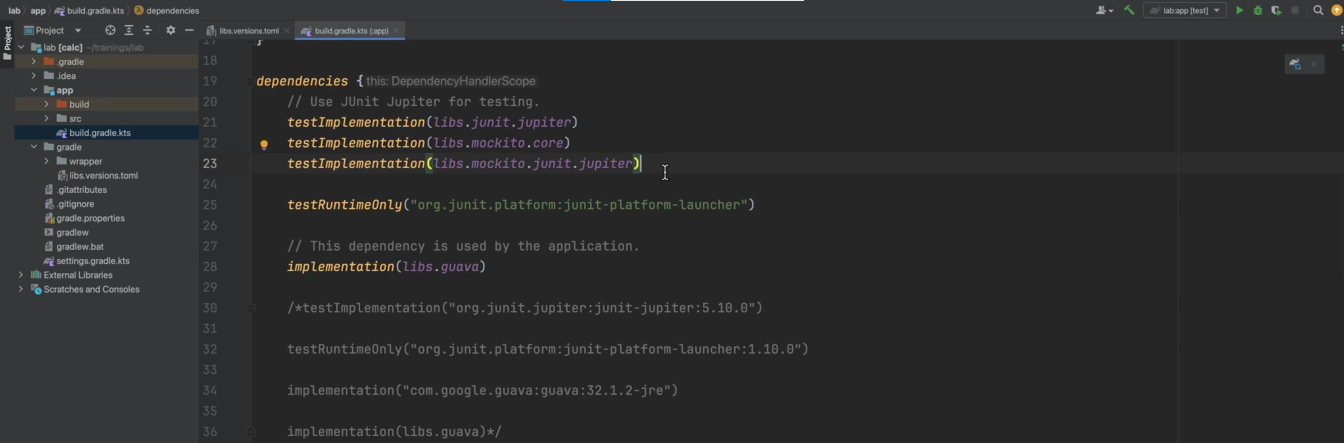
Now we can hit the Gradle refresh icon

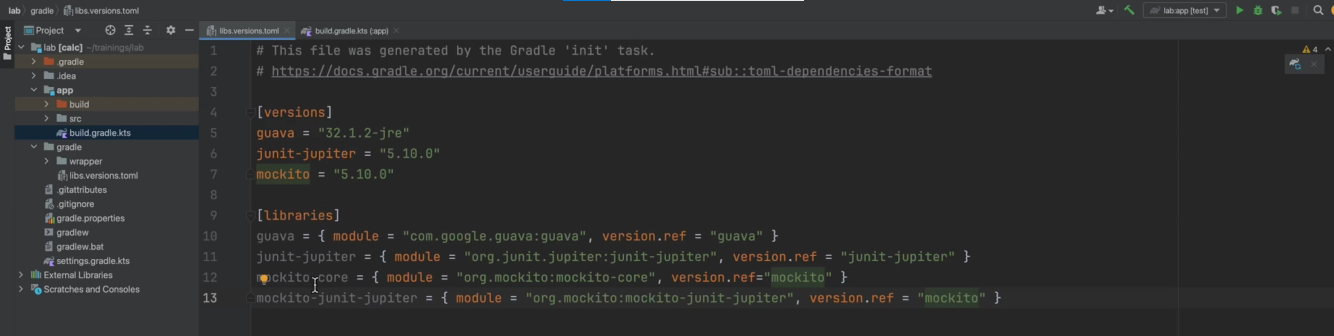


and go to the build file. Here we can add the libraries to the test implementation bucket dependency configuration.

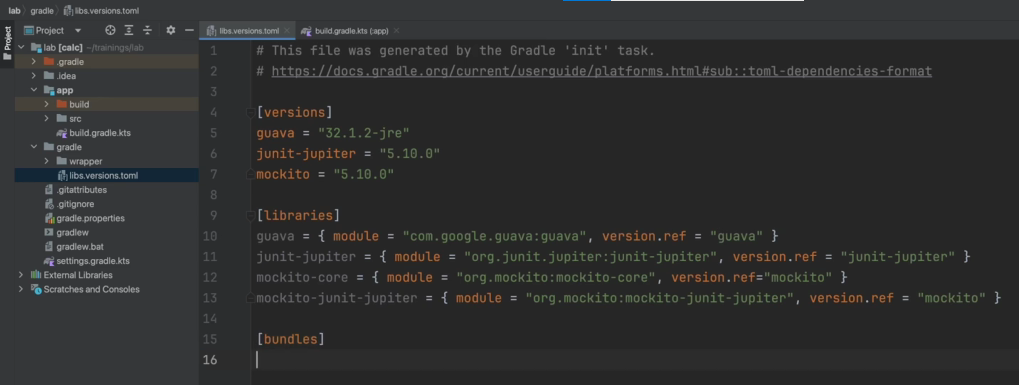


Note again that the dashes in the version catalog are treated as kind of deliminators and the dots ( . ) are used instead in the build file.

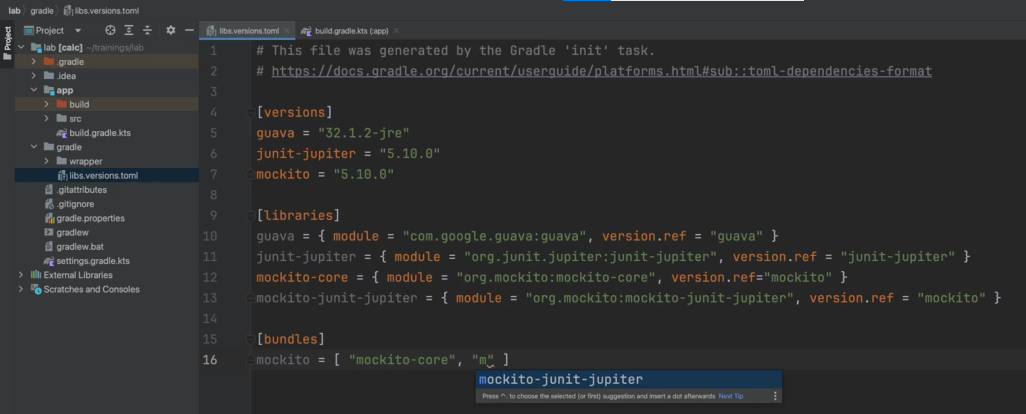




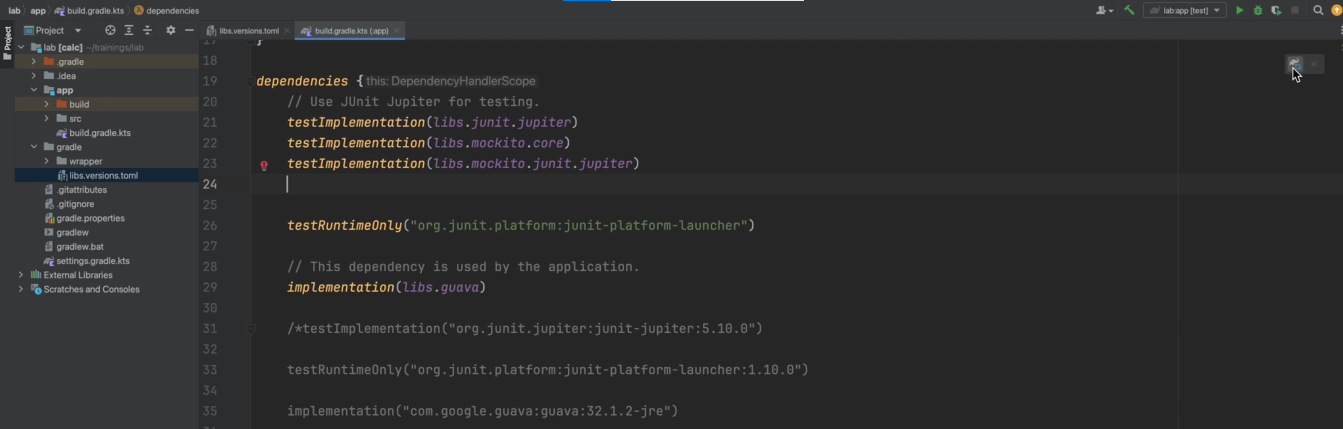
If there were a group of libraries we wanted to always use together, we can make things easier by using bundles. We can go to the version catalog and create a new section called bundles.

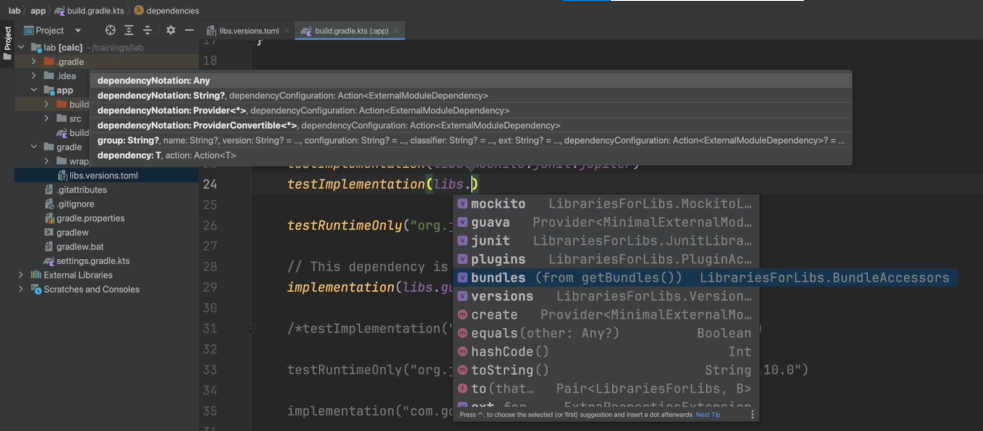


Here we can create a Mockito bundle and reference to two Mockito libraries.

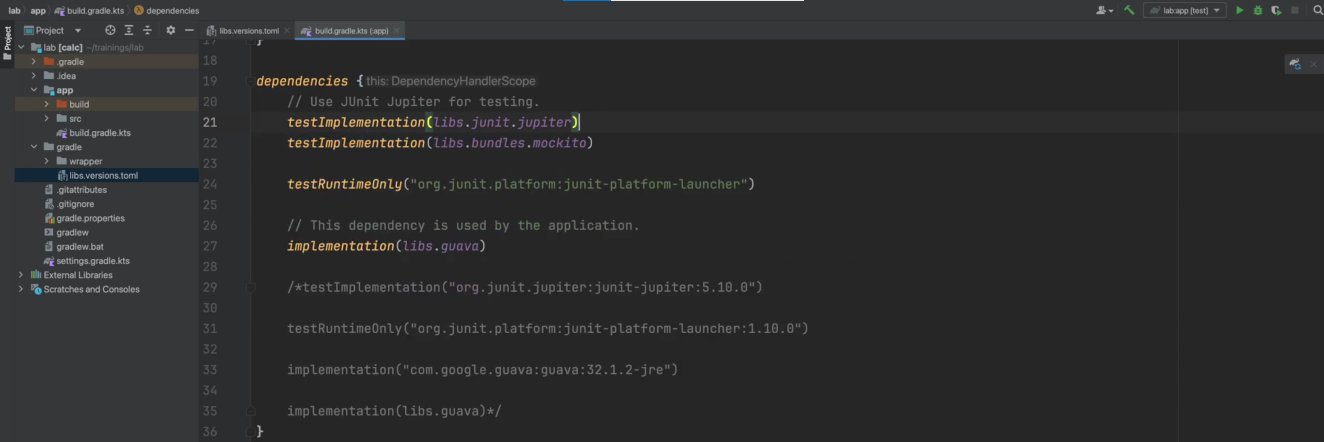


Now we can use the bundle in the build file instead of the two separate declarations. Remember to hit the Gradle refresh icon so the editor sees the updates to the version catalog.

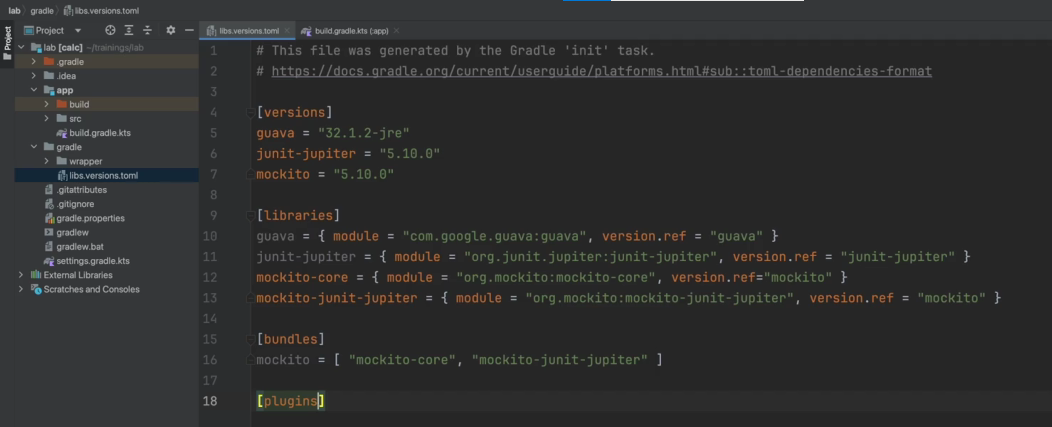




You can also define the plugins and versions you wanna use in version catalog.



They’ll be in a section called plugins.



We won’t be showing an example of that here. It is covered in another training. The version catalog is a new feature in Gradle build tool, so you will encounter older projects that do not use it yet.



The benefits are helpful and it is recommended to use it for new projects and also introduce it in existing projects.