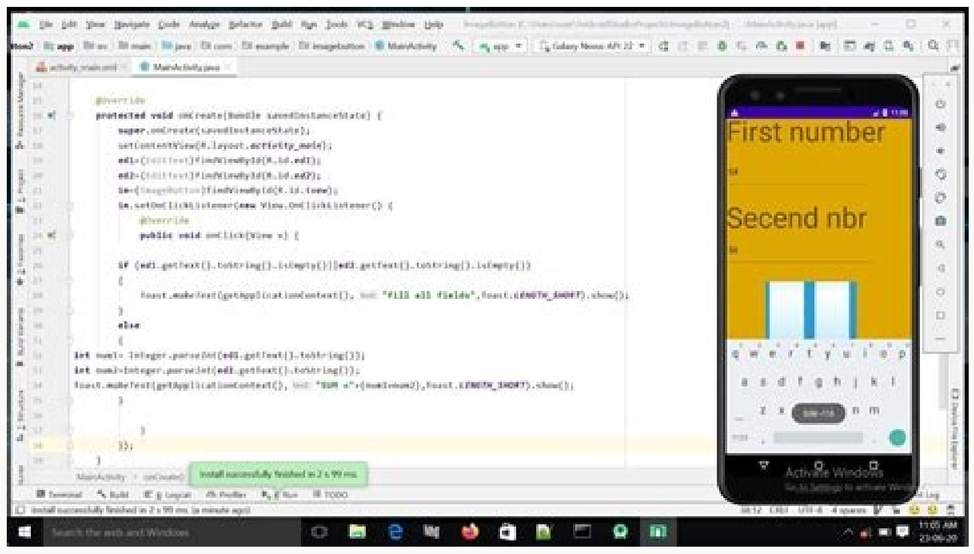
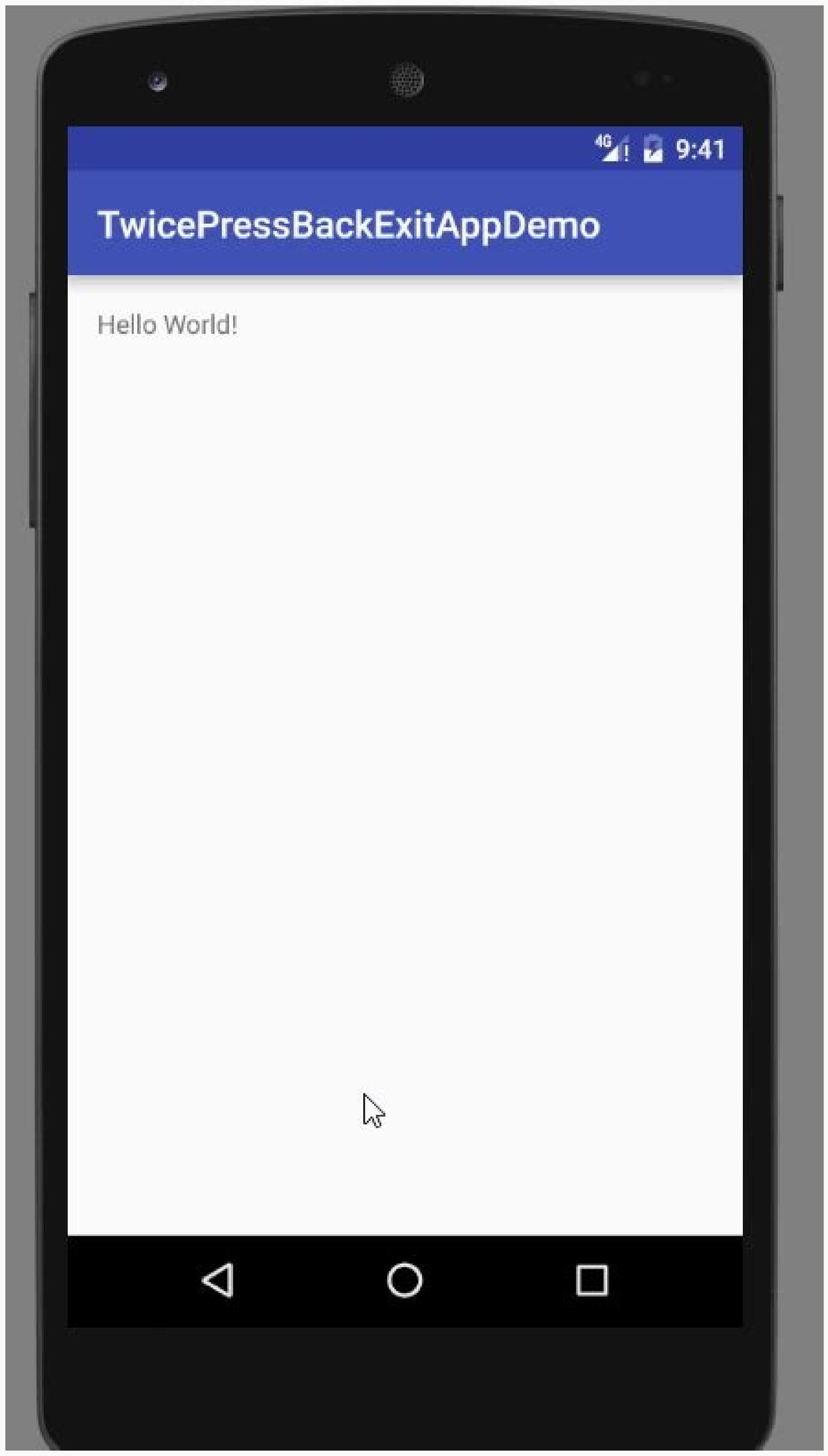
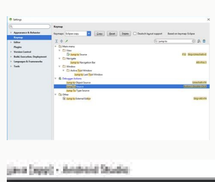
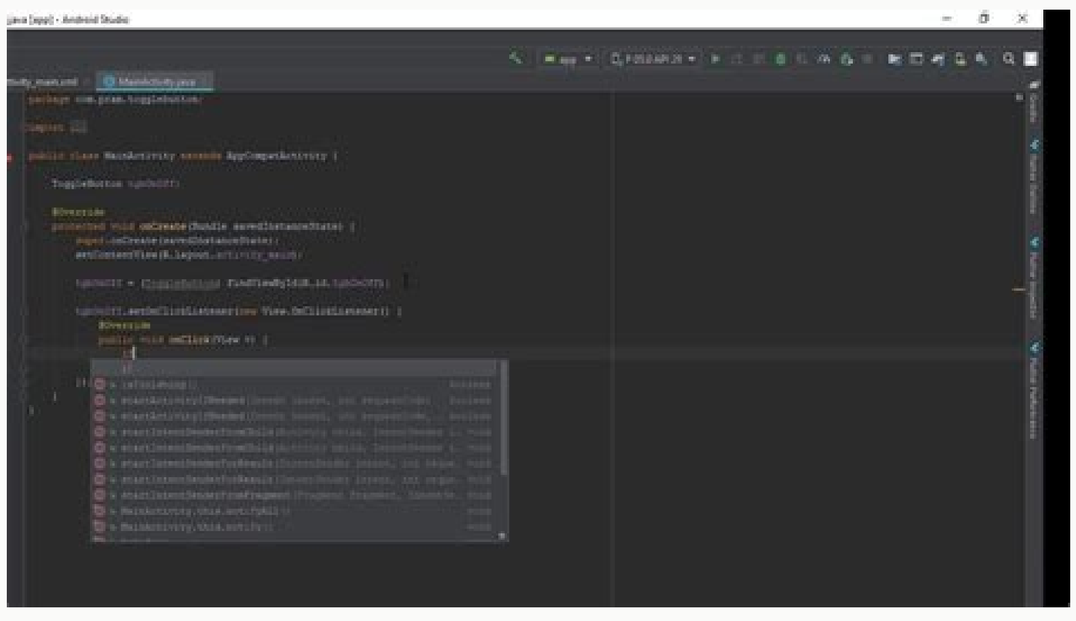
Double click android studio

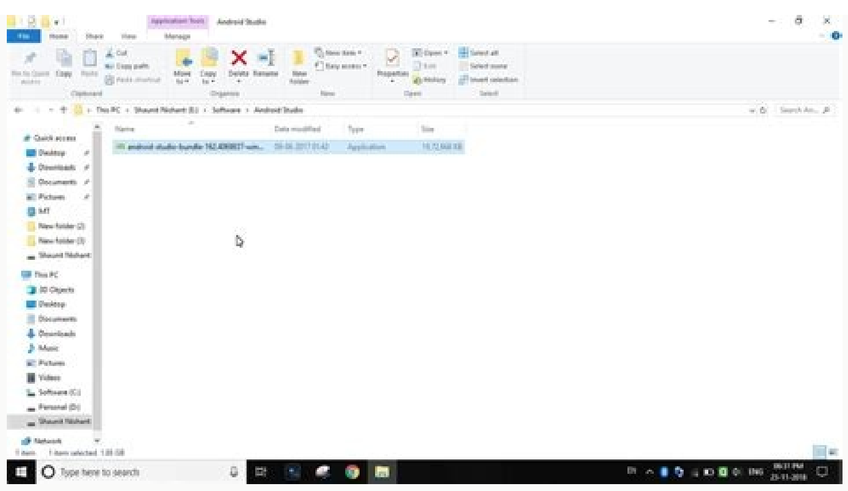
Continue


First number
Secend nbr
Activate Windows


9:41
TwicePressBackExitAppDemo
Hello World!




java [app] - Android Studio



What is double-click. Android studio detect double click. Double click vs double-click. Double click time. How to double click on android. Android studio prevent double click. Double click method. Android studio double click button.

This page tracks known issues with Android Studio Dolphin | 2021.3.1 and Android Gradle plugin 7.3.0. If you experience an issue not already included here, please report a bug. Upgrade to preview: Each release of Android Studio and the Android Gradle plugin aims to improve stability and performance, and add new features. To experience the benefits of upcoming releases now, download and install Android Studio Preview. Known Issues with Android Studio This section describes known issues that exist in the latest stable version of Android Studio. Error when rendering Compose Preview Starting with Android Studio Chipmunk, if you're seeing java.lang.NoSuchFieldError: view_tree_saved_state_registry_owner or java.lang.ClassNotFoundException: androidx.savedstate.R$id in the issues panel, make sure to include a debugImplementation dependency to androidx.lifecycle:lifecycle-viewmodel-savedstate in your module. If you're seeing java.lang.NoSuchFieldError: view_tree_lifecycle_owner in the issues panel, make sure to include a debugImplementation dependency to androidx.lifecycle:lifecycle-runtime in your module. If you're seeing java.lang.NoClassDefFoundError: Could not initialize class androidx.customview.poolingcontainer.PoolingContainer or java.lang.NoClassDefFoundError: androidx/customview/poolingcontainer/PoolingContainerListener in the issues panel, make sure to include a debugImplementation dependency to androidx.customview:customview-poolingcontainer in your module. Error when using different passwords for key and keystore Starting with version 4.2, Android Studio now runs on JDK 11. This update causes an underlying behavior change related to signing keys. When you navigate to Build > Generate Signed Bundle / APK and attempt to configure app signing for an app bundle or an APK, entering different passwords for the key and keystore may result in the following error: Key was created with errors: Warning: Different store and Key passwords not supported for PKCS12 Key stores To work around this issue, enter the same password for both the key and keystore. Android Studio doesn't start after installing version 4.2 Studio tries to import previous .vmoptions and sanitize them to work with the garbage collector used by JDK 11. If that process fails, the IDE may not start for certain users who set custom VM options in the .vmoptions file. To work around this issue, we recommend commenting out custom options in .vmoptions (using the "#" character). The .vmoptions file can be found in the following locations: Windows C:\Users\YourUserName\AppData\[Local|Roaming]\Google\AndroidStudio4.2\studio64.exe.vmoptions macOS ~/Library/Application Support/Google/AndroidStudio4.2/studio.vmoptions Linux ~/.config/Google/AndroidStudio4.2/studio64.vmoptions If Studio still doesn't start after trying this workaround, see Studio doesn't start after upgrade below. Apps using Database Inspector crash on Android 11 emulator Apps using the Database Inspector may crash when running on the Android 11 emulator, with an error like the following appearing in logcat: Fatal signal 11 (SIGSEGV), code 1 (SEGV_MAPERR) To fix this issue, upgrade your Android 11 emulator to version 9 or higher by navigating to Tools > SDK Manager. In the SDK Platforms tab, check the box labeled Show Package Details and select revision 9 or higher of the Android 11 emulator. Studio doesn't start after upgrade If Studio doesn't start after an upgrade, the problem may be due to an invalid Android Studio configuration imported from a previous version of Android Studio or an incompatible plugin. As a workaround, try deleting (or renaming, for backup purposes) the directory below, depending on the Android Studio version and operating system, and start Android Studio again. This will reset Android Studio to its default state, with all third-party plugins removed. For Android Studio 4.1 and later: Windows: %APPDATA%\Google\AndroidStudio Example: C:\Users\your_user_name\AppData\Roaming\Google\AndroidStudio4.1 macOS: ~/Library/Application Support/Google/AndroidStudio Example: ~/Library/Application Support/Google/AndroidStudio4.1 Linux: ~/.config/Google/AndroidStudio and ~/.local/share/Google/AndroidStudio Example: ~/.config/Google/AndroidStudio4.1 and ~/.local/share/Google/AndroidStudio4.1 For Android Studio 4.0 and earlier: Windows: %HOMEPATH%\.AndroidStudio\config Example: C:\Users\your_user_name\.AndroidStudio3.6\config macOS: ~/Library/Preferences/AndroidStudio Example: ~/Library/Preferences/AndroidStudio3.6 Linux: ~/.AndroidStudio/config Example: ~/.AndroidStudio3.6/config Note that the configuration directory for Canary and Beta releases of Android Studio is PreviewX.Y instead of X.Y for the . For example, Android Studio 4.1 Canary builds use AndroidStudioPreview4.1, instead of the AndroidStudio4.1 directory that is used for Release Candidates and Stable releases. Compilation issue in Kotlin multiplatform projects Compilation errors may arise in Kotlin MPP code due to missing symbols. Upgrading your Kotlin plugin to version 1.4 should resolve this issue. Key mapping conflicts on Linux On Linux, certain keyboard shortcuts conflict with default Linux keyboard shortcuts and those of popular window managers, such as KDE and GNOME. These conflicting keyboard shortcuts may not work as expected in Android Studio. More information about this issue (including potential workarounds) can be found in IntelliJ's bug tracker. Small UI text on Chrome OS On Chrome OS, text might appear much smaller than in previous releases. To work around this issue, do the following: Open the Settings window by clicking File > Settings Navigate to Appearance & Behavior > Appearance. Select Use custom font. Increase the font size. In the Settings window, navigate to Editor > Font. Increase the font size. Click OK. Code editing This section describes known issues related to the code editor. Frozen keyboard input - "iBus" problems on Linux There are some known interactions between the iBus daemon on Linux and Android Studio. In some scenarios, the IDE stops responding to keyboard input or starts inputting random characters. This bug is triggered by some missing synchronization between iBus and XLib + AWT, and has already been reported upstream to JetBrains and iBus. There are three current workarounds for this issue: Workaround 1: Force iBus into synchronous mode. Before starting Android Studio, run the following on the command line: $ IBUS_ENABLE_SYNC_MODE=1 ibus-daemon -xrd Workaround 2: Disable iBus input in Android Studio. To disable iBus input for Android Studio only, run the following on the command line: $ XMODIFIERS= ./bin/studio.sh This workaround only disables input methods for Android Studio, not any other applications you may be running. Note that if you restart the daemon while Android Studio is running (for example, by running ibus-daemon -rd), you effectively disable the input methods for all other applications and may also crash Android Studio's JVM with a segmentation fault. Workaround 3: Double-check the shortcut bindings to make sure that the Next input shortcut is not set to Control+Space, since this is also the code completion shortcut in Android Studio. Ubuntu 14.04 (Trusty) makes Super+Space the default shortcut, but settings from previous versions may still be around. To check your shortcut bindings, run ibus-setup on the command line to open the IBus Preferences window. Under Keyboard Shortcuts, check the Next input method. If it is set to Control+Space, change it to Super+Space, or another shortcut of your choice. Project configuration This section describes known issues related to project configuration and Gradle sync. Gradle Sync Failed: Broken Pipe The issue is that the Gradle daemon is trying to use IPv4 instead of IPv6. Workaround 1: On Linux, put the following in your ~/.profile or ~/.bash_profile: export _JAVA_OPTIONS="-Djava.net.preferIPv6Addresses=true" Workaround 2: in Android Studio's vmoptions file, change the line -Djava.net.preferIPv4Addresses=true to -Djava.net.preferIPv6Addresses=true For more information, see the Networking IPv6 User Guide. "peer not authenticated" errors from Gradle sync or SDK Manager The root cause of these errors is a missing certificate in $JAVA_HOME/jre/lib/certificates/cacerts. To resolve these errors, proceed as follows: If you're behind a proxy, try to connect directly. If the direct connection works, then in order to connect via the proxy you may need to use keytool to add the proxy server's certificate to the cacerts file. Re-install a supported, unmodified JDK. There's a known issue affecting Ubuntu users, which results in an empty /etc/ssl/certs/java/cacerts. To work around this issue, execute the following on the command line: sudo /var/lib/dpkg/info/ca-certificates-java.postinst configure Deploying This section describes known issues related to deploying your app to a connected device. [Mac OS only] Incremental updates are not applied due to an issue with Gradle file watching on projects saved under /System/Volumes/Data Gradle issue 18149 affects Android Gradle Plugin versions 7.0 and higher because they require Gradle version 7.0 and higher. Starting in Gradle 7.0, file watching is enabled by default. If you are working on Mac OS and your project is saved under /System/Volumes/Data, Gradle file watching will not properly track file changes. This will cause the Build System to not see any file changes and it will therefore not update the APK(s). The incremental deployment code will then do nothing because the local APK state is the same as on the device. To work around this issue you should move your project's directory to your user directory, that is, under /Users/username. The Build System will then be properly notified about file changes by Gradle file watching and incremental changes will be successfully applied. Android Emulator HAXM on macOS High Sierra The Android Emulator on macOS High Sierra (10.13) requires HAXM 6.2.1+ for best compatibility and stability with macOS. However, macOS 10.13 has a more involved process to install kernel extensions such as HAXM. You need to manually allow the kernel extension itself to be installed as follows: First, attempt to install the latest version of HAXM from the SDK Manager. In MacOS, go to System Preferences > Security and Privacy. If you see an alert that System software from developer "Intel Corporation Apps" was blocked from loading, click Allow: For more information and workarounds, see this Apple webpage and issue 62395878. Apply Changes This section describes known issues that are related to Apply Changes. New app name not applied If you rename your app and then try to apply that change, the updated name might not be reflected. To work around this issue, click Run to re-deploy your app and see your changes. Issue in Android Runtime throws error If you're using a device that runs Android 8.0 or 8.1, you might encounter "VERIFICATION_ERROR" messages when trying to apply certain types of changes (especially if you're using Kotlin). This message is caused by an issue with the Android Runtime that is fixed in Android 9.0 and higher. Although the issue causes Apply Changes to fail, you can still Run your app again to see your changes. However, we recommend that you upgrade the device to Android 9.0 or greater. Debugging and testing This section describes known issues related to debugging and testing your app. JUnit tests missing resources in classpath when run from Android Studio If you have specific resource folders in your Java modules, then those resources won't be found when running tests from the IDE. Running tests using Gradle from the command line will work. Executing the Gradle check task from the IDE will also work. See issue 64887 for more details. This issue occurs because as of IntelliJ 13, which requires that you only have a single folder as the classpath. IntelliJ's builder copies all resources into that build folder, but Gradle doesn't copy over the resources. Workaround 1: Run the Gradle check task from the IDE rather than running a unit test. Workaround 2: Update your build script to manually copy resources into the build folder. See comment #13 for more information. Running JUnit tests may compile the code twice When creating a new project, the template JUnit configuration might be created with two "Before launch" steps: Make and Gradle-aware Make. This configuration is then propagated to all created JUnit run configurations. To fix the issue for the current project, click Run > Edit Configurations and change the default JUnit configuration to only include the Gradle-aware Make step. To fix the issue for all future projects, click File > Close Project. You should see the welcome screen. Then click Configure > Project Defaults > Run Configurations and change the JUnit configuration to only include the Gradle-aware Make step. Some test run configurations don't work Not all run configurations that are available when right-clicking a test method are valid. Specifically, the following configurations are not valid: Gradle run configurations (which have a Gradle logo as the icon) don't work. JUnit run configurations (which have an icon without the green Android) don't apply to instrumentation tests, which cannot be run on the local JVM. Android Studio also remembers the run configuration created in a given context (for example, right-clicking a specific class or method), and will not offer to run in a different configuration in the future. To fix this, click Run > Edit Configurations and remove the incorrectly-created configurations. Adding Java breakpoints while debugging native code While your app is paused at a breakpoint in your native code, the Auto and Dual debuggers may not immediately recognize new Java breakpoints that you set. To avoid this issue, add Java breakpoints either before starting a debug session or while the app is paused on a Java breakpoint. For more information, see issue 229949. Stepping out of the native debugger While using the Auto or Dual debugger to debug Java and native code, if you step into a native function from your Java code (for example, the debugger pauses execution at a line in your Java code that calls a native function and you click Step Into) and you want to return to your Java code, click Resume Program (instead of Step Out or Step Over). Your app process will still be paused, so click Resume Program in the your-module-java tab to resume it. For more information, see issue 224385. Native debugger hangs while loading libraries While using the native debugger for the first time after upgrading to Android Studio 4.2 and higher, the native debugger may stop responding while loading libraries from the Android device. This issue is a sticky problem that continues to happen even if you stop and restart the debugger. To fix this problem, delete the LLDB cache at $USER/.lldb/module-cache/. Native debugger crashes with "Debugger process finished with exit code 127" This error occurs on Linux-based platforms when starting the native debugger. It indicates that one of the libraries required by the native debugger is not installed on the local system. The name of the missing library may be already printed in the idea.log file. If not, you can use a terminal to navigate to the Android Studio installation directory and execute the bin/lldb/bin/LLDBFrontend --version command line to learn which libraries are missing. Typically, the missing library is ncurses5 as some recent Linux distributions have already upgraded to ncurses6. Profilers This section describes known issues with the Profilers. Native Memory Profiler: Profiling not available during app startup The Native Memory Profiler is currently unavailable during app startup. This option will be available in an upcoming release. As a workaround, you can use the Perfetto standalone command-line profiler to capture startup profiles. Timeout errors in CPU Profiler You may experience "Recording failed to stop" errors in the Android Studio CPU Profiler when you select the Sample Java Methods or Trace Java Methods configurations. These are often timeout errors, especially if you see the following error message in the idea.log file: Wait for ART trace file timed out The timeout errors tend to affect traced methods more than sampled methods and longer recordings more than shorter recordings. As a temporary workaround, it may be helpful to try shorter recordings to see if the error disappears. If you experience timeout issues with the Profiler, please file a bug that includes the make/model of your device(s) and any relevant entries from idea.log and logcat. When using Platform Tools 29.0.3, native debugging and the Android Studio Profilers might not work properly, and you might see either "AdbCommandRejectedException" or "Failed to connect port" in the idea.log file when you select Help > Show Log. Upgrading the Platform Tools to 29.0.4 or higher fixes both issues. To upgrade the Platform Tools, do the following: Open the SDK Manager from Android Studio by clicking Tools > SDK Manager or click SDK Manager in the toolbar. Click the checkbox next to Android SDK Platform-Tools so it shows a checkmark. A download icon should appear in the left column. Click Apply or OK. Plugin prevents Build Output window from working Using the CMake simple highlighter plugin prevents content from appearing in the Build Output window. The build runs and the Build Output tab appears, but there is no output printed (issue #204791544). Installation order prevents launch Installing a newer version of Android Studio before an older version might prevent the older version from launching. For example, if you first install the canary version of Android Studio, and then try to install and launch the stable version, the stable version might not launch. In cases like this, you must clear the cache to get the stable (older) version to launch. On macOS, to clear the cache delete the Library/ApplicationSupport/Google/AndroidStudioversion_number directory. On Windows, to clear the cache use Disk Cleanup. Known issues with the Android Gradle Plugin This section describes known issues that exist in the latest stable version of the Android Gradle plugin. Not all dynamic-feature library dependencies are lint checked When running lint with checkDependencies = true from an app module, dynamic-feature library dependencies aren't checked unless they're also app dependencies (issue #191977888). As a workaround, the lint task can be run on those libraries. Signing file named with carriage return characters JAR signing (v1 scheme) does not support file names containing carriage return characters (issue #63885809). Modifying variant outputs at build time might not work Using the Variant API to manipulate variant outputs is broken with the new plugin. It still works for simple tasks, such as changing the APK name during build time, as shown below: // If you use each() to iterate through the variant objects, // you need to start using all(). That's because each() iterates // through only the objects that already exist during configuration time— // but those object don't exist at configuration time with the new model. // However, all() adapts to the new model by picking up object as they are // added during execution. android.applicationVariants.all { variant -> variant.outputs.all { outputFileName = "${variant.name}-${variant.versionName}.apk" } } However, more complicated tasks that involve accessing outputFile objects no longer work. That's because variant-specific tasks are no longer created during the configuration stage. This results in the plugin not knowing all of its outputs up front, but it also means faster configuration times. manifestOutputFile is no longer available The processManifest.manifestOutputFile() method is no longer available, and you get the following error when you call it: A problem occurred configuring project ':myapp'. Could not get unknown property 'manifestOutputFile' for task ':myapp:processDebugManifest' of type com.android.build.gradle.tasks.ProcessManifest. Instead of calling manifestOutputFile() to get the manifest file for each variant, you can call processManifest.manifestOutputDirectory() to return the path of the directory that contains all generated manifests. You can then locate a manifest and apply your logic to it. The sample below dynamically changes the version code in the manifest: android.applicationVariants.all { variant -> variant.outputs.all { output -> output.processManifest.doLast { // Stores the path to the maifest. String manifestPath = "$manifestOutputDirectory/AndroidManifest.xml" // Stores the contents of the manifest. def manifestContent = file(manifestPath).getText() // Changes the version code in the stored text. manifestContent = manifestContent.replace('android:versionCode="1"', String.format('android:versionCode="%s"', generatedCode)) // Overwrites the manifest with the new text. file(manifestPath).write(manifestContent) } } } Issues with AGP 7.3.0 AIDL support and Kotlin 1.7.x Using AGP 7.3.0 with KAPT in Kotlin 1.7.x causes the AIDL source sets for specific build variants to be removed. You can still use the other AIDL source sets, including those of main/, build types, product flavors, and combinations of product flavors. If you need to use the variant-specific AIDL source sets, continue to use Kotlin 1.6.21. Fixed known issues This section describes known issues that have been fixed in a recent release. If you are experiencing any of these issues, you should update Android Studio to the latest stable or preview version. Fixed in Android Studio 2021.1.1 Missing lint output: There is no lint text output printed to stdout when the lint task is UP-TO-DATE (issue #191897708). Fixed in AGP 7.1.0-alpha05. Problems with unit testing an app project that uses the Hilt plugin: The unit test classpath contains the non-instrumented app classes, which means Hilt does not instrument the app classes to handle dependency injection when running unit tests (issue #213534628). Fixed in AGP 7.1.1. Fixed in Android Studio 2020.3.1 Lint exceptions in Kotlin projects: Kotlin projects that set checkDependencies = true may encounter null pointer exceptions or errors (issue #158777858). Fixed in Android Studio 4.2 IDE freezes on macOS Big Sur: Android Studio 4.1 might freeze when you open a dialog. Fixed in Android Studio 4.1 Restart to apply memory settings from previous version of IDE: After updating Android Studio, you need to restart Android Studio to apply any memory settings migrated from an earlier version of the IDE. Manifest class with custom permission strings is no longer generated by default: If you want to generate the class, set android.generateManifestClass = true. Fixed in Android Studio 3.6 APK installation error on LineageOS: Deploying your app to devices running certain versions of LineageOS or CyanogenMod might fail and throw an INSTALL_PARSE_FAILED_NOT_APK exception. On Android Studio 3.6 Beta 1 and higher, the IDE handles this exception by performing a full app install when you deploy your app to LineageOS or CyanogenMod devices, which might result in longer deploy times. Fixed in Android Studio 3.5.2 Broken XML code style: When editing XML code, the IDE applied an incorrect code style when you selected Code > Reformat Code from the menu bar. Fixed in Android Studio 3.3.1 Out of memory errors when scanning C++-based projects: When Gradle scans a project that has C++ code in more than one location on the same drive, the scan includes all directories below the first common directory. Scanning a large number of directories and files may lead to out of memory errors. For more information on this issue, read the bug associated with the issue.