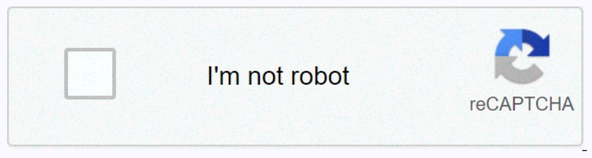

Continue

Android developer latest version

Google is committed to promoting racial equity for black communities. Here's how. New safeguards to protect user privacy that you'll need to support in your app. System changes that can affect your app when it's running on Android 10. APIs for foldable, dark theme, nav by gestures, connectivity, media, NNAPI, biometrics and more. Get Android 10 — Install Android 10 on a Pixel device or set up an emulator. Set up your environment - See the Setup Guide for details. Review the changes — Learn about privacy and behavior changes. See what's new — Learn about the new features and APIs you can use in your app. Test your app —Run through streams and look for issues Update your app—Directing API 29, if possible, testing with users through beta channels or other groups. The content and code samples on this page are subject to the licenses described in the Content License. Java is a registered trademark of Oracle and/or its affiliates. Last update 2020-01-17 UTC. Android 11 contains a variety of great ways to extend your app. Android 11 also includes behavior changes to improve battery life and security and improve user privacy. Some of these behavior changes affect only apps that target Android 11, while others affect all apps when they're running on an Android 11 device, regardless of the target of an SDKVersion app. To develop with the Android 11 APIs and test your app with android 11 behavior changes, follow the instructions on this page to set up the Android 11 SDK in Android Studio and build and run your app on Android 11. Get the latest Preview of Android Studio The Android 11 SDK includes changes that aren't compatible with some older versions of Android Studio. So for the best development experience with the Android 11 SDK, we recommend that you install the latest preview version of Android Studio. Get Android Studio Preview You can compile and test Android 11 apps using Android Studio 3.3 or higher, but some Android 11 SDK users may encounter gradle sync flaws and outdated dependencies warnings. Remember, you can keep your existing version of Android Studio installed because you can install multiple versions side by side. Get the Android 11 SDK After installing and opening Android Studio, install the Android 11 SDK as follows: Click Tools > SDK Manager. On the PLATFORMS SDK tab, select Android 11. On the SDK Tools tab, select Android SDK Build-Tools 30 (or higher). Click OK to start the installation. Changing the build setting of your app to achieve Android 11 gives your app android 11 APIs and lets you fully test your app's compatibility as you prepare to add full support to Android 11. To do this, open your build.gradle module-level file and update the compiledSdkVersion and targetSdkVersion: android { compileSdkVersion 30 defaultConfig { targetSdkVersion 30 } Note: If you're not ready to fully support Android 11, you can still perform app compatibility tests using a depurable app, an Android 11 11 and the compatibility framework, without changing the compiled of your applicationSdkVersion or targetSdkVersion. To learn more about the changes in Android 11 that can affect your app so you can start testing for them, read the following pages: To learn more about the new APIs available on Android 11, read the Features of Android 11 and APIs. Android Studio 4.1 has been released for the stable channel. Download it here. Android Studio 4.2 is currently on Beta channels. Android Studio Arctic Fox | 2020.3.1 is currently on the Canary and Dev channels. For the latest release news, including a list of notable fixes for each release, see also the Release updates. If you encounter any problems using a preview version of Android Studio, please let us know. Your bug reports help improve Android Studio. Android Studio Arctic Fox | 2020.3.1 Updated numbering of the Android Studio version We changed the version numbering system for Android Studio to align more with IntelliJ IDEA, the IDE on which Android Studio is built. In the previous numbering system, this would have been Android Studio 4.3 or version 4.3.0.1. With the new numbering system, it's now Android Studio - Arctic Fox | 2020.3.1 Canary 1, or version 2020.3.1.1. Intellij Old Name Old Version - Number System New - Year System New Version Name 2020.3 4.3 Canary 1 4.3.0.1 2020.3.1.1 Arctic Fox | 2020.3.1 Canary 1 Going forward, see how the Android Studio version number is determined: <Year of= intellij= version=><IntelliJ major= version=>. . <Studio major= version=>. <Studio minor/patch= version=>The first two groups of numbers represent the version of the Intelllj platform on which a specific version of Android Studio is based. For this release, this is version 2020.3. The third number group represents the main version of Studio, starting at 1 and incrementing by one for each major release. The fourth number group represents the Studio minor/patch version, starting at 1 and incrementing by one for each smaller version. We're also giving each major release a version name, incrementing from A to Z based on animal names. This release is called Arctic Fox. We are updating the version numbering for the Android Gradle plugin (AGP) to more closely match the underlying Gradle build tool. Therefore, AGP 7.0 will be the next release after AGP 4.2. For more details, see The version changes in the AGP release notes. Android Studio now uses Gradle Test Runner To improve the overall consistency of test runs, Android Studio now uses Gradle to run all unit tests by default. In many cases, this change will not affect your test workflow in the IDE. For example, when you click the Run command in the context menu (visible when you click the or its corresponding gutter action, Android Studio will use the gradle run setting by default to run unit tests. However, Android Studio no longer recognizes existing Android JUnit execution settings, so you should migrate the Android JUnit execution settings that you</Studio> </Studio> </IntelliJ> </Year> save as project files for gradle execution settings. To create a Gradle test configuration, select the Gradle template by following the instructions in Create a new run/debug configuration. When you created a new setting, it will appear in the Edit Settings dialog box in the Gradle section: If you want to inspect Android JUnit settings that are no longer recognized, you can do one of two things: Open manually saved settings in a text editor. The locations of these files are user-specified, but the files typically appear in <my-app>/.idea/runConfigurations/. Look for temporary settings in <my-app>/.idea/workspace.xml and look under the <component name=RunManager ...=>node. For example: <component name=RunManager selected=Gradle.PlantTest>... <configuration name=PlantTest type=AndroidJUnit factoryname=Android JUnit nameisgenerated=true> <module name=Sunflower.app></module> <useClassPathOnly></useClassPathOnly> <extension name=coverage><pattern><option name=PATTERN value=com.google.samples.sunflower.sunflower.data.*><option name=ENABLED value=true></pattern> </extension><option name=PACKAGE_NAME value=com.google.samples.apps.sunflower.data><option name=MAIN_CLASS_NAME value=com.google.samples.sunflower.data.PlantTest&. gt;<option name=METHOD_NAME value=><option name=TEST_OBJECT value=class><option name=><option name=WORKING_DIRECTORY value=$MODULE_DIR$><method v=2><option name=Android.Gradle.BeforeRunTask enabled=true></method></configuration> Android Plugin Gradle 7.0 No AGP 7.0 Canary 3 , the following settings (or dependency scopes) have been removed: Compile Depending on the use case, this has been replaced by api or implementation. It also applies to *Build variants, for example: debugCompile. as long as it has been replaced by compileOnly. It also applies to *Supplied variants, for example: releaseProvided. apk This has been replaced by runtimeOnly. publish This has been replaced by runtimeOnly. In most cases, the AGP Update Wizard will automatically migrate your project to the new settings. Class change when compiling against the Gradle Android plugin If you're compiling with the Gradle Android plugin, your build classpath may change. Because the Android Gradle plugin now uses api/deployment settings internally, some artifacts can be removed from your build classpath. If you rely on an Android Gradle plugin dependency at compile time, be sure to add it as an explicit dependency. Accessibility scanner for layout editor Android Studio now integrates with the Android Accessibility Test Framework to help you find accessibility issues in your layouts. When using the click the Accessibility Scanner button to start the scanner. The tool also provides suggested fixes for some common issues, such as the lack of content descriptions. The Accessibility Scanner is available from Canary 8. Support for</component></component></my-app></my-app> for</component></component></my-app></my-app> Comporing the Jetpack Compose toolkit provides a modern approach to building your application's UI. The toolkit also brings all the benefits of Kotlin, such as helping you write concise and idiomatic code that is fully interoperable with Java. For the best development experience with Jetpack Compose, you should use the latest version of Android Studio 4.2. That's because when you use Android Studio to develop your app with Jetpack Compose, you can benefit from smart editor features such as New Project templates and the ability to immediately view your UI. To learn more and get started, go to the Jetpack Compose overview. Jetpack Composes Tool Support at Arctic Fox | 2020.3.1 Android Studio now includes additional support for previewing and testing apps that use Jetpack Compose. Comporting the view The following parameters for @Preview methods are already available: showBackground: Turn a background on and off for your preview. backgroundColor: Set a color that is only used on the preview surface. uiMode: This new parameter can take any of the Configuration.UI_* constants and allows you to change the behavior of the visualization to, for example, set it to Night Mode to see how the theme reacts. Interactive visualization In this mode, you can interact with your UI components, click them, and see how the state changes. It's a quick way to get feedback on how your UI reacts and view animations. To enable it, just click on the interactive icon and the preview will change modes. To stop, click the Stop Interactive View in the top toolbar. Deploy to Device Use this feature to deploy a ui snippet to a device. This will help test small parts of your code on the device without having to launch the full app. Click the deployment icon for the device next to the @Preview annotation or at the top of the preview, and Android Studio will deploy this @Preview to your connected device or emulator. Visualization Data Sources API The new Data Sources API lets you generate visualizations from your data. If you have an existing data list or a list of themes, this API allows you to inject as a parameter to the @Preview. HelloWorldProvider class : CollectionPreviewParameterProvider<String>(listOf(Hello World, Привет мир, Hello World, Hola World)) @Preview @Composable Fun HelloWorldPreview(@PreviewParameter(HelloWorldProvider::class) text: String) { MaterialTheme { Text (text = text) } } To enable the above features, the build.grad of your module must contain the following settings: android { ... buildFeatures { compor true } comporOptions { kotlinCompilerExtensionVersion = kotlinCompilerVersion = 1.3.70-dev-withExperimentalGoogleExtensions-20200424 } } Known issues to Compose preview androidx.ui.foundation.The dialog is currently not supported in compose preview. Known issues for Arctic Fox Preview On Linux and macOS machines, update patches are not working for canary 2. To upgrade to canary version 2, download and install the full </String> </String> Android Studio 4.2 Set up app subscription by variant You can now enable or disable app subscription in the Android Gradle plugin by variant. This example demonstrates how to define application signature by variant using the onVariants() method in Kotlin or Groovy: androidComponents { onVariants(selector().withName(fooDebug), { signingConfig.enableV1Signing.set(false) signatureConfig.enableV2Signing.set(true) }) Optimizing gradle builds for JDK 11 When running in Android Studio, The Gradle build tool uses the default JDK version set in Android Studio. In previous versions, JDK 8 was used by default; in 4.2, however, JDK 11 is now the default JDK. This update to JDK 11 impacts the default jvm garbage collector configuration, as JDK 8 uses the parallel garbage collector, while JDK 11 uses the G1 garbage collector. To potentially improve build performance, we recommend testing your Gradle builds with the parallel garbage collector. In gradle.properties set the following: org.gradle.jvmargs=-XX:+UseParallelGC If there are other options already defined in this field, add a new option: org.gradle.jvmargs=-Xmx1536m -XX:+UseParallelGC To measure the build speed with different settings, see Profile of your build. In the CPU profiler, the System Trace feature now includes new metrics to analyze application performance. Events Table The Events table is a new tab on the right side of the analysis pane. This table lists all trace events in the currently selected segment. New track groups and crawls More data is already available in System Trace for app traces deployed on devices running Android 9 or higher. BufferQueue (in the View section) This new track shows the count of application surface bufferqueue bufferqueue (0, 1, or 2). It can help you understand the state of image buffers as they move between the graphical components of Android. For example, a value of 2 means that the application is currently buffered triple, which can result in extra input latency. CPU frequency (in the CPU cores section) In addition to CPU scheduling data, we also include CPU frequency per core. This shows how difficult each core is working and can give you an idea of which cores are large or small in modern mobile processors. The new Process Memory (RSS) shows the amount of physical memory currently in use by the application. Total This is the total amount of physical memory currently in use by your process. In Unix-based systems, this is known as Resident Set Size, and is the combination of all memory used by anonymous allocations (those supported by the swap file), file mappings (files that are loaded into memory one page at a time), and shared memory allocations processes and asperifed by a variety of mechanisms). For Windows developers, Resident Set Size is analogous to the size of the working set. Allocated This counter tracks the amount of physical memory currently used by normal process memory allocations. These are allocations that are anonymous (not supported by private file(not shared). File mappings This counter tracks how much physical memory is being used by any process-owned file mappings. Shared This counter tracks how much physical memory is being used to share memory between this process and other processes in the system. R8 retrace now available in command-line tools Available in version 4.0 of the command-line tools, the R8 retrace is a standalone tool to get the original stack trace from an obfuscated stack trace. You can download this package with the SDK manager, which installs r8 redo android_sdk/cmdline-tools. Alternatively, you can download the standalone command-line tool pack. For usage information, see R8 redo in the user guide. New Layout Inspector update action Introduced in Android Studio 4.0, layout inspector is designed for real-time inspection of your running app's UI stack. However, you may not always want layout inspector to immediately reflect what's happening in your application, as you may want to inspect a snapshot of your app's layout at any given time or minimize the performance impact of live updates in your application. Pause live updates and update the screenshot in layout inspector. To manually upload a snapshot of your app's UI data, first disable the Live Updates option. You can then click the Refresh button to take a new photo of the UI stack for inspection. Layout Inspector now remembers your preference to keep live updates turned on or off between sessions. Support for the Android Gradle plugin for Jetpack Compose Starting with Android Gradle Plugin 4.2 Canary 13, only jetpack compose compiler 1.0.0-alpha-04 and higher will be supported. Upgrade wizard for AGP From Android Studio 4.2 Canary 5, an upgrade wizard for the Gradle Android plugin can help you update the AGP version for your project. Built on top of existing AGP update functionality, this tool guides you through updates/refactorings throughout the project and includes a preview of the updates to help prevent possible break changes before performing the AGP update. Support for Safe Args Safe Args is a Gradle plugin that generates simple classes of objects and constructors for safe type navigation and access to any associated arguments. Android Studio 4.2 Canary 9 or higher includes special support when working with Safe Args, as described below: Autocompletions for Directions, Args and the various building classes Support for safe Java plugins and Kotlin Args source navigation for bank inspector analysis of XML The Database Inspector includes some improvements to help you write and execute your custom SQL statements. When you open the inspector and open a New query tab, you should notice a larger, resizable editor surface for the author and format your queries, as shown below. In addition, we now provide a history of your previous queries. When you click the Show query history button, you should see a list of queries queries previously ran against the currently selected database. Click a query in the list to see a preview of the full query in the editor, and press Enter to copy it to the editor. Then click Run to execute the statement. Offline mode In earlier versions of Android Studio, disconnection of an application process when using the Database Inspector resulted in the inspector and its data closing. In Android Studio 4.2 Canary 8 or higher, we've added the ability to continue inspecting your app's databases after a process disconnection, making it easier to debug your app after an accident. When a disconnect occurs, the Database Inspector downloads your databases and makes them available to you in offline mode. When offline, you can open tables and run queries. Keep in mind that when you reconnect to a live application process, the Database Inspector returns to the way of life and shows only the data that is on the device. That is, the data shown in offline mode does not persist when you reconnect to an application process. Because of this, the Database Inspector does not allow you to edit or execute modification statements while in offline mode. New removable configuration for feature modules The Android Gradle 4.2 plugin uses bundletool 1.0.0, which introduces a behavior change for applications using feature modules: Any feature module specified as dist:install-time that is not explicitly marked as dist:removable will become non-removable by default. This new configuration optimizes the fusion of installation time modules with the base module, potentially improving application performance for some applications. For more information about this new configuration, see the dist:removable tag documentation in the feature module manifestation documentation. This section provides a summary of new features and changes in Android Studio 4.2. Android Plugin Gradle 4.2 Behavior change for gradle.properties Files From AGP 4.2, it is no longer possible to override the Gradle properties of subprojects. In other words, if you declare a property in a gradle.properties file in a subproject instead of the root project, it is ignored. As an example, in previous versions, AGP would read values from projectDir/gradle.properties, projectDir/app/gradle.properties, projectDir/library/gradle.properties, etc. For application modules, if the same Gradle property were present in both projectDir/gradle.properties and projectDir/app/gradle.properties, the value of projectDir/app/gradle.properties would take precedence. In AGP 4.2, this behavior has changed, and the AGP will not load gradle.properties values into subprojects (for example, projectDir/app/gradle.properties). This change reflects gradle's new behavior and supports the For more information about setting values in gradle.properties files, see the Gradle documents. Java language version 8 by default Starting in version 4.2, AGP will use the Java 8 language level by default. Java 8 provides access to a number of newer language features, including lambda expressions, method references, and static Methods. For the full list of supported features, see the java 8 documentation. To maintain the old behavior, specify Java 7 explicitly at your build.gradle.kts or build.gradle file module level: android // build.gradle.kts { ... compileOptions { sourceCompatibility = JavaVersion.VERSION_1_7 targetCompatibility = JavaVersion.VERSION_1_7 } // For KotlinOptions projects { jvmTarget = 1.6 } } // android build.gradle { ... compileOptions { sourceCompatibility JavaVersion.VERSION_1_7 targetCompatibility JavaVersion.VERSION_1_7 } // For KotlinOptions projects { jvmTarget = 1.6 } } New JVM resource compiler A new JVM feature compiler in the Android Gradle 4.2 plugin tool replaces parts of the AAPT2 feature compiler, potentially improving build performance, especially on Windows machines. Starting with the Canary 7 release, the new JVM resource compiler is enabled by default. The v3 and v4 signature now supports Android Gradle Plugin 4.2 now supports APK v3 and APK v4 signature formats. To enable one or both of these formats in your build, add the following properties to your build.gradle module level or file build.gradle.kts: // android build.gradle.kts { signatureConfigs { config { ... enable V3Signing(true) enableV4Signing (true) } // android build.gradle { ... signatureConfigs { config { ... enable V3Signing true enableV4Signing true } } } The APK v4 signature allows you to quickly deploy large APKs using the ADB incremental APK installation on Android 11. This new flag takes care of the APK signing step in the deployment process. Improved instrumentation testing From Android Studio 4.2 Canary 1, instrumentation tests can now be run on multiple devices in parallel and investigated using a specialized instrumentation test results dashboard. By using this panel, you can determine whether tests are failing due to API level or hardware properties. Testing your app at a wide variety of API levels and form factors is one of the best ways to ensure that all users have a great experience when using your app. To take advantage of this feature: Select Modify the Device Set in the drop-down menu of the target device (in the top center of the IDE). Select the target devices and click OK. Select Multiple Devices from the target device drop-down menu and run your tests. To see the test results in the Run panel, go to View > Windows Tool > Run. The new test results pane lets you filter your test results by status, device, and API level. In addition, you can sort each column by clicking the header. By clicking an individual test, you can view device records and information individually for each device. Apply changes To help you be more As you iterate into your app, we've made the following enhancements to apply changes to devices running Android 11 or higher: Support for additional code changes For devices running Android 11 or higher, you can now add static end primitive fields and then deploy those changes to your running app by clicking apply apply Changes or Apply changes and restart activity. You can now also add features and then deploy these changes to your running app on Android 11 devices by clicking Apply changes and restart activity. ANDROID_SDK_HOME deprecated environment variable The ANDROID_SDK_HOME variable is deprecated and has been replaced by ANDROID_PREFS_ROOT. For more information, see Emulator Environment Variables. Known issues for viewing 4.2 This section describes the current known issues in Android Studio 4.2 Preview. Native memory profile: The launch of the profiling application disabled the native Profile Creation memory at application startup has been disabled. This option will be enabled in an upcoming release. As a workaround, you can use the perfetto command-line profiler to capture startup profiles. Studio does not start after installing Canary 8 After upgrading Android Studio to 4.2 Canary 8, the IDE may not be initial for certain users who set custom VM options in the .vmoptions file. To work around this problem, we recommend commenting on 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\AndroidStudioPreview4.2\studio64.exe.vmoptions macOS ~/Library/Application Support/Google/AndroidStudioPreview4.2/studio.vmoptions Linux ~/.config/Google/AndroidStudioPreview4.2/studio64.vmoptions ~/.config/Google/AndroidStudioPreview4.2/studio64.vmoptions