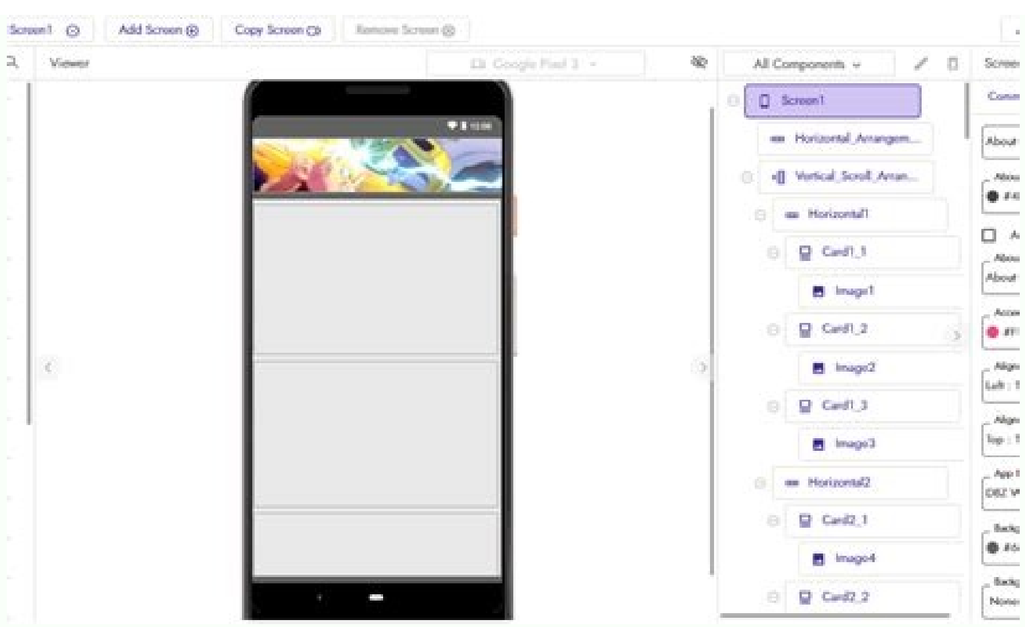
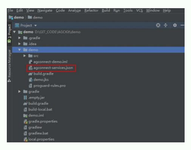
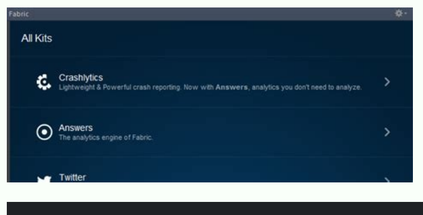
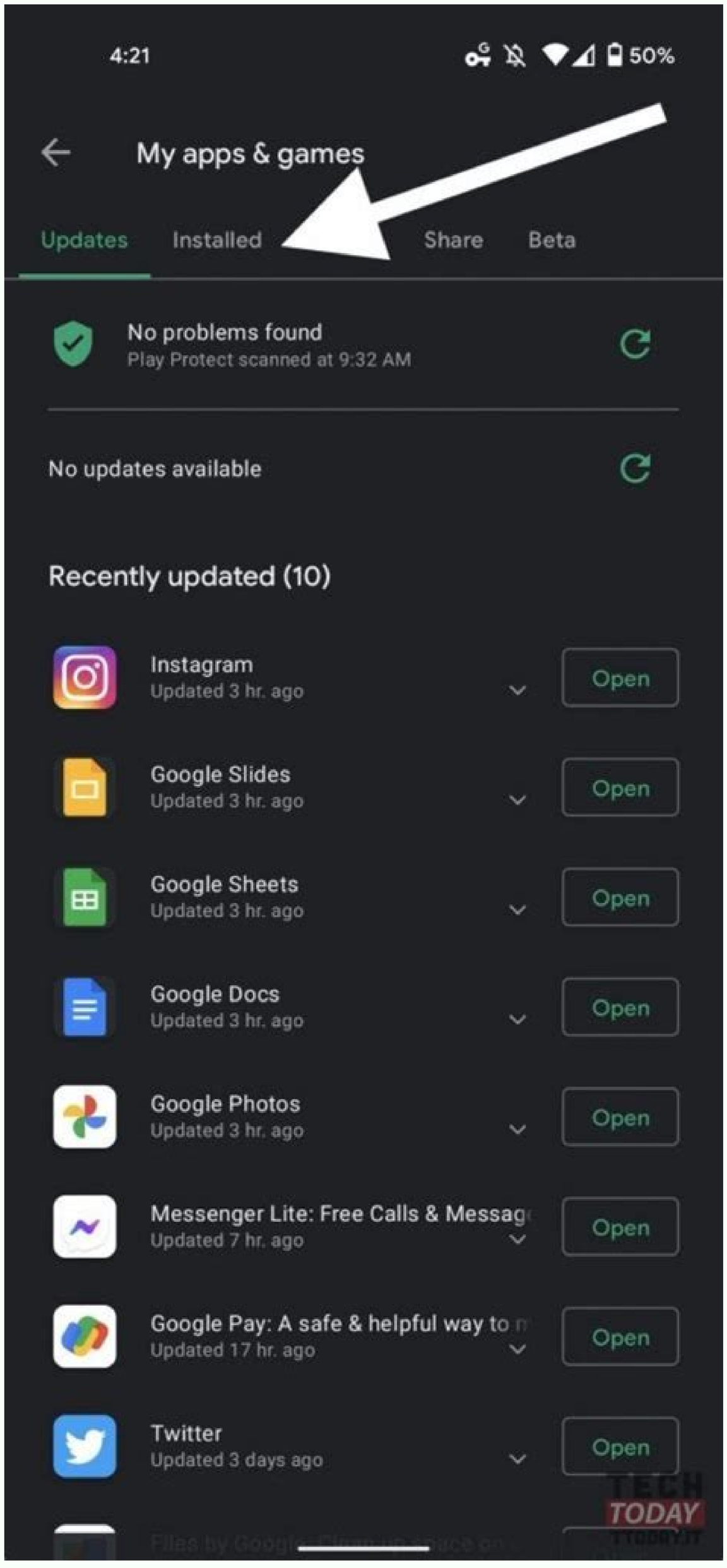
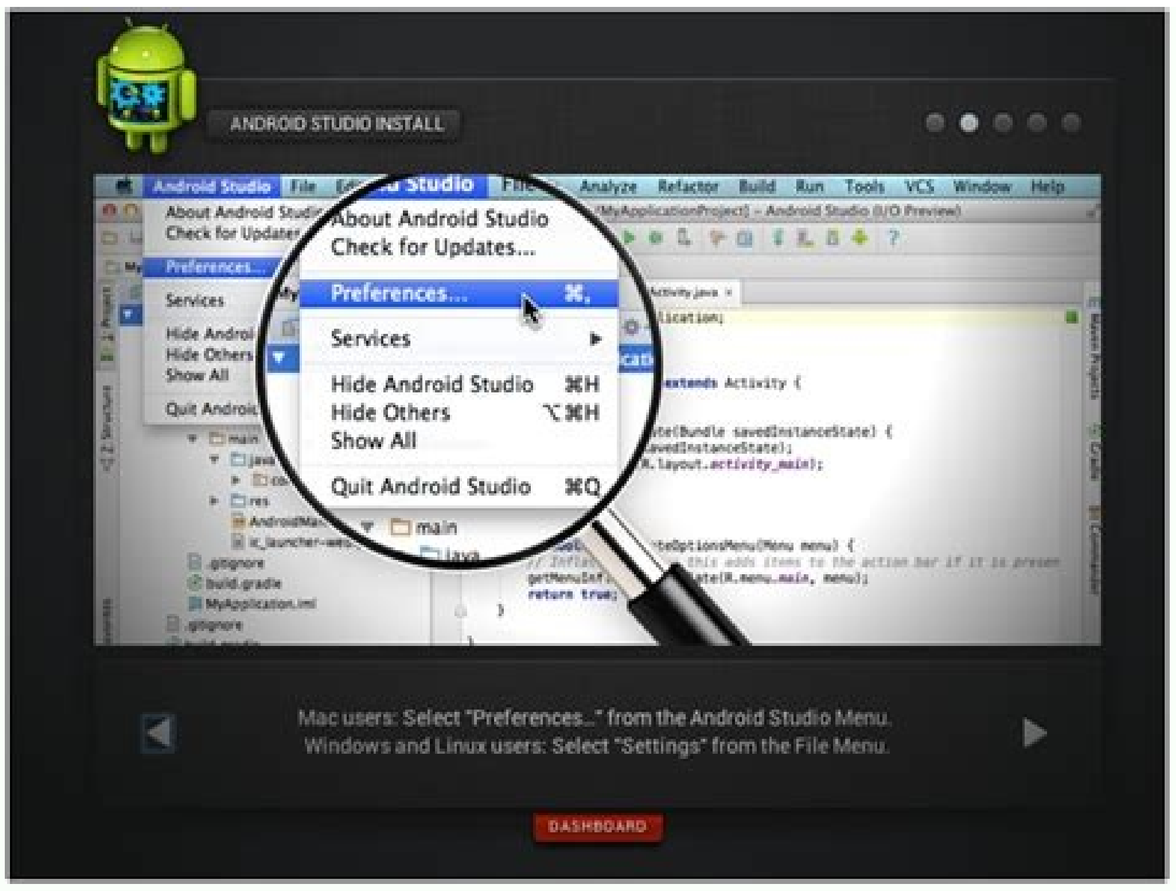
Android studio app crashes on button click


Screen1
Add Screen
Copy Screen
Remove Screen
Viewer
Google Pixel 3
All Components
Screen1
Horizontal_Arrangem...
Vertical_Scroll_Arran...
Horizontal1
Card1_1
Image1
Card1_2
Image2
Card1_3
Image3
Horizontal2
Card2_1
Image4
Card2_2




Fabric
All Kits
Crashlytics
Lightweight & Powerful crash reporting. Now with Answers, analytics you don't need to analyze.
Answers
The analytics engine of Fabric.
Twitter


4:21
50%
My apps & games
Updates
Installed
Share
Beta
No problems found
Play Protect scanned at 9:32 AM
No updates available
Recently updated (10)
Instagram
Updated 3 hr. ago
Open
Google Slides
Updated 3 hr. ago
Open
Google Sheets
Updated 3 hr. ago
Open
Google Docs
Updated 3 hr. ago
Open
Google Photos
Updated 3 hr. ago
Open
Messenger Lite: Free Calls & Messag
Updated 7 hr. ago
Open
Google Pay: A safe & helpful way to
Updated 17 hr. ago
Open
Twitter
Updated 3 days ago
Open
TECH TODAY



App crashes on launch android studio.

This page contains a list of new features introduced in the preview versions of Android Studio. Preview builds provide early access to the latest Android Studio features and improvements. You can download these preview versions here. Please let us know if you encounter any problems using Android Studio Preview. Your bug reports help improve Android Studio. For the latest news on Android Studio previews, including a list of important fixes in each preview, see Version Update on the Android Studio blog. Current Android Studio versions The following table lists the current Android Studio versions and their respective channels. Android Studio Dolphin Channel Version | 2021.3.1 Stable Android Gradle Plugin 7.3.1 Stable Android Studio Electric Eel | 2022.1.1 Beta Android Studio Flamingo | 2022.2.1 Canary Compatibility with Android Gradle Plugin Previews Each preview version of Android Studio is released with a corresponding version of the Android Gradle Plugin (AGP). Preview versions of Studio should work with any compatible stable version of AGP. However, if you are using AGP Preview, you must use the corresponding Studio Preview (e.g. Android Studio Chipmunk Canary 7 with AGP 7.2.0-alpha07). If you try to use other versions (e.g. Android Studio Chipmunk Beta 1 with AGP 7.2.0-alpha07) the synchronization will fail and you will be prompted to update to the appropriate AGP version. For a detailed log of the deprecated and removed Android Gradle Plugin API, see Android Gradle Plugin API Updates. Android Studio Electric Zuss | 2022.1.1. Below are the new features of Android Studio Electric Eel. Live Editing In Android Studio Electric Eel, you can use Live Editing to deploy real-time code changes to your composable emulator or device. You don't have to wait for the build or deployment to finish, so you can work faster when building your app. Note: Android Studio ElectricCanary 10 and later contain a major fix for Kotlin functions overloaded with calls of the wrong return type. Attention. Starting with Electric Eel Canary 9, if your project is using version 1.2 or later, Live Editing no longer requires you to reset your change composition via code. Only the first code change made to this file will reset the theme. Subsequent editions do not reset the composition. Live Edit is available in Electric Eel Canary as an experimental feature. To enable it manually, go to File > Preferences > Editor > Live Edit (Android Studio > Preferences > Editor > Live Edit on Mac). Live Edit focuses on UI and UX related code changes. Live Edit does not currently support changes such as updating method signatures, adding new methods, or changing the class hierarchy. Live Edit is an experimental feature under development, so unstable performance may occur. If you run into a problem, please let us know so we can work to resolve it. You can also see a list of open issues here. For more information, see Live Editing in Android Studio. SDK Stats View dependency stats from the new Google Play SDK Index, a public portal for information about popular dependencies or SDKs. If a specific version of a library has been marked as deprecated by its author, an appropriate Lint warning will be displayed when viewing the dependency definition. This allows you to detect and update dependency issues during development instead of publishing your app to the Play Console later. More information on this new tool can be found on the Android developer blog here. Firebase Crashlytics app quality statistics Starting with Android Studio Electric Eel, you can view and act on app crash data from Firebase Crashlytics directly in the IDE. This integration pulls stack traces and crash statistics from Crashlytics into the App Quality Insights window in the new IDE, so you don't have to switch back and forth between the browser and the IDE. Developmentcan benefit from key features including the following: Show highlighted lines in code if they have associated Crashlytics event data. See the stack trace for the main failures and click on the stack trace to jump to the relevant lines of code. See a statistical summary of the most common crashes and minor events, for example, grouped by device manufacturer and Android version. Filter events by severity, time, and application version. Get the link in your browser, which will open the Crashlytics dashboard page with more information about the event. By integrating Android Studio and Crashlytics, you can code and solve big crash problems in one place. This advanced development environment helps you monitor your app's performance and minimize disruption to users. If you have any problems with this feature, please report a bug. If you're new to Crashlytics and want to learn more about its offerings, check out Firebase Crashlytics. Getting Started To view Crashlytics data in Android Studio, you need to set up Firebase and Crashlytics in your app project. Here's how: Open the Firebase Assistant in Android Studio by choosing Tools > Firebase, click Crashlytics, and then follow the instructions to add Firebase and Crashlytics to your project. For more information about the Firebase Assistant workflow, see the Firebase Android Getting Started Guide. If you've already added Firebase and Crashlytics to your app, sign in to your developer account in the IDE by clicking the avatar icon . After logging in, click on the App Quality Insights window. You should see the Issues, Stack Trace, and Details panels populated with messages from Crashlytics. Navigating Crashlytics Data The App Quality Insights window in Android Studio contains extensive data that provides an overview of the events your users experience and links that help you quickly navigate to the sources of those events. If your project has multiple application modules, make sure that the module you want to view event data in isis selected from the module drop-down menu. Use the Application ID drop-down menu to select the application ID you want to analyze. For example, while you are working on a debug version of your application in the IDE, you might want to view event data for the production version of the application that your users are installing. The Issues panel shows the most popular events that Crashlytics has data for, sorted by the most important crashes. The example stack trace panel shows information about the last event you clicked on in the issues panel. You can view the event's stack trace and click the stack trace to jump to the appropriate lines in the code base. There is also information about the type of device affected, its Android version and the time of the event, as well as a link to the event in the Firebase Crashlytics dashboard. The dashboard is useful when you want to examine additional stack traces, trends, and user logs. The dashboard breaks down the number of crashes by device type and Android version, so you can see which user groups are most affected. It also shows which versions of the app have crashed and how many users have been affected. Use severity filters to select or deselect fatal or non-fatal events. Use time range and application version filters to refine specific subsets of events. Not only can you switch from stack trace to code, but also from code to stack trace: Android Studio now highlights the lines of code that are crashing so you can easily identify and fix them. Hovering over the highlighted line of code will bring up a pop-up window showing the affected event, its frequency, and the number of devices affected. You can click Open in App Quality Insights in App Quality Insights to go to the event details in the App Quality Insights window. Resizable Emulator Now you can test your application on different screen sizes and with a single resizable emulator. Testing in a single resizable emulator doesn't just allow for speedchanges in various interfaces, but also supports smoother development by saving the computing resources and memory that would be needed to manage separate virtual devices. To use the resizable emulator, you need Android emulator version 31.1.3 or later (you can update the version in Tools > SDK Manager). To create a Resizable Android Virtual Device (AVD), select the Resizable Phone Hardware Profile (Experimental) in the device creation process, download the Android Tiramisu image, and follow the instructions to create the AVD. When deploying your application to a scalable emulator, you can quickly switch between many common device types using the display mode drop-down menu in the emulator toolbar. The emulator's screen size is resized so you can easily test your app on different screen sizes and densities. Emulated Bluetooth Now you can discover and connect two virtual Bluetooth phone emulators. Virtual Bluetooth allows you to test whether an application can detect and access a Bluetooth adapter in a scalable way. This feature is available on Android emulator 31.3.8 and later with system T image (API 33). In the future, we plan to add more support for creating virtual prototype peripherals such as beacons and heart rate monitors, as well as testing the integration of your Bluetooth capabilities. Device Mirroring You can now mirror your device from the Running Devices window in Android Studio Electric Eel. By streaming your device's screen directly into Android Studio, you can perform common actions such as rotating the screen, changing the volume, or locking/unlocking your device directly from the IDE itself. Device mirroring is available in the Electric Eel Canary channel as an experimental feature. To enable it manually, go to File > Preferences > Experimental (Android Studio > Preferences > Experimental on Mac) and check the box next to Device mirroring. First, make sure you're connected to your device. All the devices you arewill be reflected in the tabs in the Running Devices window, which can be opened by going to View > Tools Window > Running Devices. When you deploy an application or test to a connected device, the Running Devices window will automatically open and display the mirrored device. Privacy Notice When device mirroring is enabled, Android Studio automatically starts device mirroring for each connected and paired device. This can lead to information disclosure for devices connected via adb tcpip, as mirrored information and commands are sent over an unencrypted channel. Android Studio also uses an unencrypted channel to communicate with the adb server, so mirroring information can be intercepted by other users on the host computer. Logcat Update Android Studio Electric Eel has the new version of Logcat enabled by default, which makes it easier to analyze, query and monitor logs. This is the most significant update to the tool since its inception. We encourage you to send feedback. Android Gradle Plugin Targets JVM Bytecode 11 As of Android Gradle Plugin 7.4.0-alpha04, AGP ships with JVM bytecode 11. This means that if you compile with AGP or custom Writing lint checks, you need to start targeting JVM bytecode 11. You can do by including the following in your module-level build.gradle file: sourceCompatibility="11" targetCompatibility="11" AGP Upgrade Assistant reports upgrade and restore functionality The AGP Upgrade Wizard now includes an update message. This message describes the steps taken and whether the upgrade was successful or not. This includes an action to roll back the changes made by the Update Assistant if there are problems building or testing the project after the update. Parallel import of projects Starting with Android Studio Electric Eel Canary 6, the IDE imports projects in parallel when using Gradle 7.4.2 or higher and Android Gradle7.2.0 or later. In particular, when Android Studio activates Gradle synchronization, information describing the projects contained in your build is created in parallel. This should result in faster sync times, especially for larger projects. Benchmarks show that the time required to build Gradle models for a very large project (with 3500 Gradle sub-projects) is reduced by 50% from 10 to 5 minutes. Android Desktop Virtual Device Now Available You can now test your app on desktop computers like Chromebooks using the Android Desktop Virtual Device (AVD). Users often interact with apps differently on devices with large screens, and a desktop AVD lets you see how your app performs in that environment. Here are some unique features for you to try. Resize apps: Resize apps by dragging the edge of the window. Manage windows freely: place the application in different places on the computer screen and minimize, maximize and resize the application window. Notice: Verify that the notices are displayed correctly when downloading from the desktop system tray. For more information about desktop AVDs and how to include them in the testing process, see Desktop AVD in Android Studio on the ChromeOS Developer Blog. Starting with Android Studio Electric Eel Canary 10, the Build Analyzer provides a summary of the time spent downloading dependencies and a detailed view of downloads in the repository. You can use this information to determine if an unexpected download dependency is negatively impacting build performance. This is especially important during incremental builds where artifacts should not be constantly downloaded. Specifically, you can use this information to identify configuration issues, such as the use of dynamic dependency versions that cause unexpected downloads. Additionally, if you see a large number of failed requests for a particular repository, this may indicate that the repository needs to be removed or moved lower in the repository configuration. SystemRecompose Highlights Android Studio Electric Eel highlights your recomposites so you can see where in the UI your comps are being recomposed. The highlight shows the composable gradient overlay in the layout inspector image area and gradually fades out to give you an idea of where in the UI to find the most composable composable element. If a composite is composed faster than another composite, the first composite gets a stronger gradient overlay color. Using Build Preview with Different Devices In Android Studio Electric Eel, you can edit the device parameter in the preview annotation to define component configurations on different devices. Autocomplete If the device parameter is an empty string (@Preview(device = ""), you can invoke autocomplete by pressing Ctrl+Space. You should be able to set values for each parameter. Press Enter to switch or press Esc to cancel. Validation If you specify an invalid value, the declaration is underlined in red and a solution may be available (Alt + Enter (â¥ + â on macOS) > Replace with ..... The validation tries to find the solution that matches the am Next comes Preview selector You can use the preview selector to edit your configurations, click the settings icon next to the preview to launch it Android Studio Electric Eel's Universal Issues Panel allows you to view all design tools in the Issues Panel provided In the Tools window, go to View > Tools window > Issues Visual Enhancement Starting with Electric Eel Canary 1, Android Studio will automatically run your layout to check for visual lint issues on different screens and different device sizes, if there is a problem, you can see it in the problems panel, which is designed to clean all problems in design tools from your layout e.g show u.written in Views or Compose. Layout Preview Starting with Android Studio Electric Eel Beta 1, you can see instant updates when changes are made. Android Studio Electric Eel Known Issues The following are Android Studio Electric Eel known issues. Projects using JDK 1.8 do not sync after upgrading to AGP 7.4.0. If your project is configured to use Gradle with JDK 1.8, upgrading to AGP 7.4.0 will cause a sync error. This is due to an incompatibility between AGP 7.4.0 and JDK 1.8. The error message is as follows: No matching variant of com.android.tools.build:gradle:7.4.0-rc01 was found. To resolve this issue, follow these steps: Update the project to use AGP 7.3.1. Set your project's Gradle JDK to JDK 11. Here's how to do it: Open the Preferences or Preferences dialog. On Windows or Linux, choose File > Preferences from the menu bar. On macOS, choose Android Studio > Preferences from the menu bar. Go to Build, Execution, Deployment > Build Tools > Gradle. Set the Gradle JDK field to JDK 11. Update the project to use AGP 7.4.0. Your project should now sync successfully. Flamingo Android Studio | 2022.2.1. Below are the new features in Android Studio Flamingo. IntelliJ IDEA 2022.2 platform update Android Studio Flamingo Canary 1 includes IntelliJ IDEA 2022.2 updates that improve the IDE experience. For more information about the changes, see the IntelliJ IDEA 2022.2 release notes. Automatic connection to foreground process in Layout Inspector Layout Inspector now automatically connects to applications on virtual or physical devices. In particular, the layout inspector automatically adds debuggable processes running in the foreground of the connected device. If you have feedback about this feature, please submit a bug report. Network Inspector Motion Capture Starting with Android Studio Flamingo Canary 1, Network Inspector displays all motion data for the entire timeline by default. You can select a range in the timeline to see only the movement within that rangeYou can also create and manage rules to help you test your application's behavior when faced with different responses, such as status codes, response headers, and content. The rules specify what responses are to be captured and how to modify them before they reach the application. You can choose which rule you want to enable or disable by checking the Active box next to each rule. Rules are automatically saved each time you change them. To get started, open the Rules tab in Web Inspector and click + to create a new rule. In the Rule Details pane, name the new rule, and in the Source subsection, provide information about the origin of the response you want to capture. The URL in the rules table should be updated based on the changes made to the response source. All fields in this subsection are optional. In the Response subsection, you can change the response before sending it to your app. For example, you can set a rule that fires responses with a specific status code and modify that status code. In the Header rules subsection, you can create multiple subrules that add or change headers in the response. When creating multiple header rules, use the up and down arrows at the top of the rules table to reorder the header rules. The order affects the modified response header because the header rules are applied in the order they are listed. Click + in the Header Rules section to get started. To add a header, enter a header name and value in the Add new header section. To edit a header, go to the Edit Existing Header tab and enter the header name or value you want to find. Enter the header title or the value you want to replace it with. Editing the response content You can also create subrules to edit the response content. You can select Find and replace the body part that replaces the first part; or you can replace the entire body contentChange your whole body. Similar to header rules, you can create multiple content rules that are applied in the order they appear in the table. RenderScript flag default value change Starting with AGP 8.0.0-alpha02, the default value of the buildFeatures.renderScript flag has been changed from true to false. If you want to enable RenderScript compilation, set the flag to true. This build feature should only be enabled in modules that use it. For help modifying your code to support these changes, use the AGP Upgrade Assistant. Important change: A namespace is required in the module-level build.gradle file. As of AGP 8.0.0-alpha03, you must set the namespace in the module-level build.gradle file instead of the manifest file. You can start using DSL namespace properties from AGP 7.3. See Namespace Configuration for more information. For help migrating to a new DSL namespace, use the AGP Upgrade Assistant (Tools > AGP Upgrade Assistant). When migrating to the DSL namespace, be aware of the following issues: In previous versions of AGP, the test namespace was derived from the root namespace or application ID, in some cases incorrectly. AGP Upgrade Assistant will block the upgrade if it detects that your project's root namespace and test namespace are the same. If the update is blocked, the test namespace must be changed manually and the source code changed accordingly. After changing the test namespace, the code may compile, but the instrumented tests will fail at runtime. This can happen if the instrument test source code references a resource defined in both the androidTest source and the application sources. For more information, see comment no. 19 for issue 191813691. Android Studio bundled with JDK 17 Starting with Android Studio Flamingo Canary 3, Studio IDE comes bundled with JDK 17. If Android Studio is configured to use the built-in JDK, new projects will use the latest stable version of the Android Gradle plugin and JDK 17 However, existing projects canand you may need to manually install the JDK to a compatible version. See Installing the JDK version for more information. AIDL flag default value change As of AGP 8.0.0-alpha04, the default value of the buildFeatures.aidl flag has been changed from true to false. If you want to enable this feature, make sure it is explicitly set to true. This build feature should only be enabled in modules that use it. Use the AGP Upgrade Assistant (Tools > AGP Upgrade Assistant) to help you adapt your code to handle this change. App Quality Statistics Android Studio Flamingo Canary 5 and later updates introduce several new App Quality Statistics features to help you focus on high-priority issues and collaborate with the development team. New Filters and Search Filtering To help you identify the most important issues, you can now filter by the following attributes. Each filter is sorted by the number of events, so you can see where the most events are happening. Android platform version Device make and model Crashlytics signal: Signal icons also appear next to the respective problems in the Problems panel, so you can see the counters and signals side by side. An issue is considered resolved if it was closed in the past and reappeared in a new version of the application. Application Version: This filter now includes a higher-level Playback Tracks filter that can be used to automatically select Production, Open Source, Closed, and/or Internal Test versions. It's also now possible to search filters with multiple options, so you can customize your view more quickly without having to scroll through all the menu options. Note and close issues directly from Android Studio. To make it easier for you to collaborate with your teammates, you can now do the following directly from the Application Quality Insights window: Close issues. To close the job, click the Close button on the stack trace taskbar. Recently closed issues are displayed in the Crossout Issues panel. You can reopen questions that were recently closedback to the button. However, if you refresh the Application Quality Insights window, closed issues will no longer be visible. Read and attach issue notes so they are visible in the Firebase console and to your teammates. To write a note about an issue, select the issue and open the Notes panel. You must have write permission for the Crashlytics project in order to write notes. Issues with notes are indicated by the Notes icon in the Issues area. If you're new to App Quality Insights and want to learn more, check out the previous release notes. Offline learning with limited functionality. Starting with Android Studio Flamingo Canary 8, you can perform certain actions in the Offline App Quality Statistics window. If you make a new request, eg. for example, by clicking Refresh and Android Studio can't contact Crashlytics, you can log out from the App Quality Statistics window. In this mode, you can continue to explore issues and current events from the cached data. Some features eg. B. Changing some filter options or closing questions is not available. To try to connect to Crashlytics again and get online again, click Reconnect. Known Issues The following are known issues with the Application Quality Insights preview features. Invalid data for apps not published to the Play Console. For applications that are not associated with at least one play track, incorrect data such as application versions and number of events may be displayed when the play track is removed from the drop-down list. However, song playback versions of apps must display accurate information. We are currently working on a fix for this issue. Filtered searches are case sensitive and limited in scope. The Filtered Search action is case sensitive. It also only looks for top-level fields like "Google" and not sub-fields like "Google". B. "Pixel 5". We are currently working on a fix for this issue. Use Firebase Test Lab devices with Gradle managed devices starting with AGP.You can run large-scale automated instrumental tests on Firebase Test Lab devices using devices managed by Gradle. Test Lab allows you to run tests simultaneously on many different Android devices, both physical and virtual. These tests take place in remote Google data centers. With the support of Gradle Managed Devices, the build system can now fully manage test execution on these Test Lab devices based on the configurations in your project's Gradle files. Note: For Firebase Test Lab usage and associated costs (if applicable), see Test Lab Usage Tiers, Limits, and Pricing. Getting Started with Gradle-Managed Firebase Test Lab Devices To get started with Gradle-managed Firebase Test Lab devices, complete the following steps: To create a Firebase project, go to the Firebase Console and click Project Add. Follow the instructions to name your project. To install the Google Cloud CLI, follow the instructions in gcloud CLI installation. Configure your on-premises environment. Link to your Firebase project in gcloud: gcloud config set project firebase-project-name Authorize API access with your credentials: gcloud auth application-default login Optional: add your project as a restricted project. This step is only required if you have exceeded your free trial limit. gcloud auth application-default set-quota-project firebase-project-name Enable required APIs. On the Google Developers Console API Library page, enable the Cloud Testing API and the Cloud Tool Results API by typing the names of these APIs in the search box at the top of the console and clicking Enable API on each API's overview page. Configure your Android project. Add the Firebase Test Lab plugin to your top-level build.gradle file: plugins { ... id 'com.google.firebase.testlab' version '0.0.1-alpha01' use false } Custom device types in Gradle activate. Properties file: android.experimental.testOptions.managedDevices.customDevice=true Add the Firebase Test Lab plugin to build.gradle at the module levelplugins { ... id 'com.google.firebase.testlab' } Building and running tests on a Firebase Test Lab appliance managed by Gradle In a module-level build, you can specify a Firebase Test Lab appliance that Gradle will use to test your app's .gradle file . The following sample code creates a Pixel 2 with API level 30 as a Gradle-managed Test Lab device named device1. Android { testOptions { ManageDevices { devices { device 1 (com.google.firebase.testlab.gradle.ManagedDevice) { device = "Pixel2" apiLevel = 30 } } } } } Tip. To find out which Test Lab devices are currently available, follow the instructions for available equipment in the test lab. To run tests using configured Gradle-managed test lab devices, use the following command. deviceName is the name of the device configured in the Gradle build script, for example device1, and BuildVariant is the build variant of the app you want to test. Note that Gradle does not run tests in parallel and does not support other Google Cloud CLI configurations for Test Lab devices. On Windows: gradlew devicenameBuildVariantAndroidTest On Linux or macOS: ./gradlew devicenameBuildVariantAndroidTest The test output contains the path to the HTML file containing the test report. You can also import test results into Android Studio for further analysis by clicking Run > Test History in the IDE. New Android SDK Upgrade Assistant Android Studio Canary 6 introduces the Android SDK Upgrade Assistant, a new tool that helps you upgrade the targetSdkVersion or API level your app is exposed to. The Android SDK Update Assistant will guide you through the targetSdkVersion update process level by level. At each stage of the migration, it highlights key changes and how to fix them. Important! Effective November 1, 2022, most apps and app updates must target Android 12 (API level 31) or higher to meet Google Play's target API level. To open the Android SDK Update Assistant, go to Tools > Android SDK Update Assistant. In the assistant panel, selectThe API level to which you want to upgrade the instructions. For best results, update the targetSdkVersion values one level at a time. The Android SDK Upgrade Guide contains only the most important changes you need to know about at each API level. For a full list of changes, see the Android release summaries. This feature is under active development. Please report a bug if you have feedback. Android Studio Flamingo Canary 6 includes new templates for creating a project or module. By default they use Compose Material 3 templates unless specified as a view template. As a best practice for building Android apps, we recommend using Compose Material 3 templates (eg Empty Activity). For more information, see Creating Material 3. To view the templates, open the New Project or Create New Module wizard by selecting File > New > New Project or New Module. Online editing now has two modes: manual and automatic. In manual mode, code changes are applied every time you manually save with Ctrl + S (Command + S for macOS). If you update a feature you create in automatic mode, the changes will be applied to your device or emulator while those changes are being made. To select the mode in which you want to run edits in real time, go to File > Preferences in the menu bar (or in macOS Android Studio > Preferences), click Editor > Live Edits, and check the Push Edits Manually or Push Edits Automatically check box. Starting with Android Studio Flamingo Canary 6, live editing is enabled by default. To disable this feature, go to File > Preferences (or Android Studio > Preferences on macOS), click Editor > Live Editing, and check the None box. Gradle Release Catalog Support Android Studio Flamingo Canary 7 introduces support for Gradle release catalogs, a feature that allows you to manage dependencies in one place and share dependencies between modules or projects. Android Studio now makes it easier to configure versioningthanks to editor suggestions and integration with the project structure dialog. If you're new to Gradle build directories, check out the documentation first to get started. Code completion and navigation Android Studio provides code completion by editing the version directory in TOML file format or by adding a dependency on the version directory to the build file. Additionally, you can quickly navigate from the dependency reference in the build.gradle file to where it is declared in the build directory by clicking on it and pressing Ctrl (Command on macOS). Integration with the project structure dialog If your project uses a version directory defined in the TOML file format, you can edit the variables defined in it from the Variables view of the project structure dialog (File > Project Structure > Variables) in Android Studio. Each version directory has a drop-down list with a list of variables from that directory. To edit a variable, click and overwrite its value. After saving these changes, the TOML file must be updated accordingly. Constraints can also be updated in the Project Structure Constraints View dialog box (File > Project Structure > Constraints). To update versions, navigate to the module and dependencies you want to edit, and then update the Requested Version field. After saving these changes, the TOML file must be updated accordingly. Note that if the dependency version was defined using a variable, updating the version directly in this way replaces the variable with the hard-coded value. Known issues and limitations The following are known issues or limitations related to Gradle version catalog support in Android Studio. Android Studio support for TOML version catalogs only Currently, code completion, navigation and support for project structure dialogs in Android Studio are only available for version catalogs defined in TOML file format. However, you can still add the version directory directly to the settings.gradle file and use its dependenciesProject. Dependencies cannot be added to the version directory from the Project Structure dialog. If your project uses version directories and you add dependencies from the Project Structure dialog, they are added to the build.gradle file instead of the directory. This issue will be fixed in a future release. Command+click navigation in the release gallery is not yet supported for build files written with a Kotlin script. Creating a trace The System Trace Utility is an Android tool that saves device activity in a trace file that provides an overview of your application's system processes over a period of time. Starting with Android Studio Flamingo, you can view your compositing features in the System Trace profiler using the Compose Tracing feature. Trace composition provides an unobtrusive system trace with the levels of detail of methods in compositing, helping you understand which compositing features are actually being recomposed. To start tracing a recomposition, you must update to at least the following versions: * Android Studio Flamingo Canary 5 * Recording UI: 1.3.0-beta01 * Build Compiler: 1.3.0 * The device or emulator running the tracing must be at least API level 30. You must also add the following Compose Runtime Tracing dependency: implementation("androidx.compose.runtime:runtime-tracing:1.0.0-alpha01") To view the recomposition trace, open Android Studio Profiler and select CPU Profiler. . In your app, navigate to the user interface you want to monitor and select Monitoring and System Recording. Use the app while recording to start recomposing. After you've completed the recording and tracing processes, you can see the compositing features in the recomposite trace directly on the stream's timeline. You can also see the Compose features on the Flame Chart, Top-Down, Bottom-Up, and Events tabs in the analysis panel. Changed the default value for the BuildConfig flag Starting with AGP 8.0.0-alpha09, the default value forbuildFeatures.buildConfig flag changed from true to false. If you need to enable this feature, make sure it is explicitly set to true. This build feature should only be enabled in modules that use it. Use the AGP Upgrade Assistant (Tools > AGP Upgrade Assistant) to set up the code to make these changes. New Settings Plugin AGP 8.0.0-alpha09 introduces a new settings plugin. The settings plugin allows you to centralize global configurations—configurations that apply to all modules—in one place, so you don't have to copy and paste configurations across multiple modules. In addition, the settings plugin can be used to create tool execution profiles or various instructions for running a tool and switching between them. Record. The settings plugin currently only works in Groovy. To use the settings plugin, use the plugin in settings.gradle: use the "com.android.settings" plugin Centralized Global Configurations Use the new Android block in settings.gradle to set up global configurations. Here's an example: android {compileSdk 31 minSdk 28 ... } Tool execution profiles The settings plugin also allows you to create execution profiles for some tools. The performance profile defines how the tool works; you can choose different execution profiles depending on your environment. In the execution profile, you can set the tool's JVM arguments and configure it to run in a separate process. Currently only R8 is supported. Create run profiles and set the default run profile in settings.gradle as shown in the following example: Error "] runInSeparateProcess true } } low { r8 { jvmOptions += ["-Xms256m", "-Xmx2048m", "-XX:+ HeapDumpOnOutOfMemoryError"] runInSeparateProcess true } } ci { r8.runInSeparate}} To replace the default profile, select a different profileandroid.experimental.settings.executionProfile property in gradle.properties file: android.experimental.settings.executionProfile=high You can also set this property on the command line, which allows you to set up different workflows. For example, if you have a continuous integration workflow, you can change the execution profile from the command line without changing the settings.gradle file: ./gradlew AssembleRelease \ -Pandroid.experimental.settings.executionProfile=ci -Pandroid.experimental. settings.executionProfile=ci