Chrome OS install

Contents

System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Linux (64-bit)
- **Disk Space**: 600 MB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these command-line tools being available in your environment.
  - `bash`
  - `curl`
  - `git` 2.x
  - `mkdir`
  - `rm`
  - `unzip`
  - `which`
  - `xz-utils`
- **Shared libraries**: Flutter `test` command depends on this library being available in your environment.
  - `libGLU.so.1` - provided by mesa packages such as `libglu1-mesa` on Ubuntu/Debian

For the best experience right now, you should put your Chrome OS Device into developer mode (this is necessary to push apps on the Chrome OS Device). For more information, see how to enable developer mode on your Chromebook.

Get the Flutter SDK

1. Download the following installation bundle to get the latest stable release of the Flutter SDK:

   flutter_linux_v1.12.13+hotfix.9-stable.tar.xz

   For other release channels, and older builds, see the SDK archive page.

2. Extract the file in the desired location, for example:

   ```
   $ cd ~/development
   $ tar xf
   ~/Downloads/flutter_linux_v1.12.13+h
   otfix.9-stable.tar.xz
   ```

   content_copy

If you don’t want to install a fixed version of the installation bundle, you can skip steps 1 and 2. Instead, get the source code from the Flutter repo on GitHub, and change branches or tags as needed. For example:

```
$ git clone https://github.com/flutter/flutter.git -b stable
```

3. Add the `flutter` tool to your path:

```
$ export PATH="$PATH:`pwd`/flutter/bin"
```

   This command sets your `PATH` variable for the *current* terminal window only. To permanently add Flutter to your path, see Update your path.

4. Optionally, pre-download development binaries:

   The `flutter` tool downloads platform-specific development binaries as needed. For scenarios where pre-downloading these artifacts is preferable (for example, in hermetic build environments, or with intermittent network availability), iOS and Android binaries can be downloaded ahead of time by running:

```
$ flutter precache
```
content_copy

For additional download options, see `flutter help precache`.

You are now ready to run Flutter commands!

> **Note:** To update an existing version of Flutter, see Upgrading Flutter.

Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the `-v` flag):

```
$ flutter doctor
```
content_copy

This command checks your environment and displays a report to the terminal window. The Dart SDK is bundled with Flutter; it is not necessary to install Dart separately. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

```
[-] Android toolchain - develop for
Android devices
      • Android SDK at
/Users/obiwan/Library/Android/sdk
      ✗ Android SDK is missing command
line tools; download from
https://goo.gl/XxQghQ
      • Try re-installing or updating your
Android SDK,
        visit
https://flutter.dev/setup/#android-setup
for detailed instructions.
```

The following sections describe how to perform these tasks and finish the setup process.

Once you have installed any missing dependencies, run the `flutter doctor` command again to verify that you've set everything up correctly.

Warning: The `flutter` tool uses Google Analytics to anonymously report feature usage statistics and basic crash reports. This data is used to help improve Flutter tools over time.

Flutter tool analytics are not sent on the very first run. To disable reporting, type `flutter config --no-analytics`. To display the current setting, type `flutter config`. If you opt out of analytics, an opt-out event will be sent, and then no further information will be sent by the Flutter tool.

By downloading the Flutter SDK, you agree to the Google Terms of Service. Note: The Google Privacy Policy describes how data is handled in this service.

Moreover, Flutter includes the Dart SDK, which may send usage metrics and crash reports to Google.

Update your path

You can update your PATH variable for the current session at the command line, as shown in Get the Flutter SDK. You'll probably want to update this variable permanently, so you can run `flutter` commands in any terminal session.

The steps for modifying this variable permanently for all terminal sessions are machine-specific. Typically you add a line to a file that is executed whenever you open a new window. For example:

1. Determine the directory where you placed the Flutter SDK. You need this in Step 3.
2. Open (or create) the `rc` file for your shell. For example, Linux uses the Bash shell by default, so edit `$HOME/.bashrc`. If you are using a different shell, the file path and filename will be different on your machine.
3. Add the following line and change `[PATH_TO_FLUTTER_GIT_DIRECTORY]` to be the path where you cloned Flutter's git repo:

   ```
   $ export PATH="$PATH:[PATH_TO_FLUTTER_GIT_DIRECTORY]/flutter/bin"
   ```
   content_copy

4. Run `source $HOME/.<rc file>` to refresh the current window, or open a new terminal window to automatically source the file.
5. Verify that the `flutter/bin` directory is now in your PATH by running:

   ```
   $ echo $PATH
   ```
   content_copy

   Verify that the `flutter` command is available by running:

```
$ which flutter
```
content_copy

Update path directly

In some cases, your distribution may not permanently acquire the path when using the above directions. When this occurs, you can change the environment variables file directly. These instructions require administrator privileges:

1. Determine the directory where you placed the Flutter SDK.

2. Locate the `etc` directory at the root of the system, and open the `profile` file with root privileges.

   ```
   $ sudo nano /etc/profile
   ```
   content_copy

3. Update the PATH string with the location of your Flutter SDK directory.

```
if [ "`id -u`" -eq 0 ]; then
   PATH="..."
else
   PATH="/usr/local/bin:...:
[PATH_TO_FLUTTER_GIT_DIRECTORY]/flut
ter/bin"
fi
export PATH
```

4. End the current session or reboot your system.
5. Once you start a new session, verify that the `flutter` command is available by running:

```
$ which flutter
```

For more details on setting the path in Bash, see [this StackExchange question](). For information on setting the path in Z shell, see [this StackOverflow question]().

Android setup (without Android Studio)

Install Java

```
$ sudo apt update                        content_copy
$ sudo apt install default-jre
$ sudo apt install default-jdk
```

Install the Android SDK’s

Download the Android SDK tools and select the “Command Line Tools only” option.

Drag and drop the downloaded zip into your Linux Files folder through the Chrome OS Files app. This moves it to the home directory, notated as $TOOLS_PATH going forward (`~/`).

Unzip the tools and then add it to your path.

```
$ unzip ~/sdk-tools-linux*               content_copy
$ export
PATH="$PATH:$TOOLS_PATH/tools/bin"
```

Navigate to where you’d like to keep the SDK packages ($PLATFORM_PATH in these snippets) and download the SDK packages using the sdkmanager tool (version numbers here are the latest at time of publishing):

```
$ sdkmanager "build-tools;28.0.3" "emulator" "tools" "platform-tools" "platforms;android-28" "extras;google;google_play_services" "extras;google;webdriver" "system-images;android-28;google_apis_playstore;x86_64"
```

Add the Android platform tools to your path (you should find this where you ran the sdkmanager command: $PLATFORM_PATH):

```
$ export PATH="$PATH:$PLATFORM_PATH/platform-tools"
```

Set the ANDROID_HOME variable to where you unzipped sdk-tools before (aka your $TOOLS_PATH):

```
$ export ANDROID_HOME="$TOOLS_PATH"
```

Now, run flutter doctor to accept the android-licenses:

```
$ flutter doctor --android-licenses
```

content_copy

Next step

Set up your preferred editor.

Flutter & Chrome OS tips & tricks

Wondering how to run your app? On Chrome OS, you can either connect your phone or push directly to the Android container on device. To do that you must enable Developer mode on your machine, and then connect to the local container with ADB:

```
$ adb connect 100.115.92.2:5555
```

content_copy

Want to build your first app optimized for Chrome OS? Clone the flutter-samples repo and build our specific Chrome OS Best Practices example:

```
$ git clone https://github.com/flutter/samples
$ cd samples/chrome-os-best-practices
$ flutter run
```

Wondering how to access your favorite F-Key shortcuts on the Chrome OS keyboard?

- Press the search key along with 1 through = to access F1–F12.

For the current versions of Chrome OS, only certain ports from Crostini are exposed to the rest of the environments. Here's an example of how to launch Flutter DevTools for an Android app with ports that will work:

```
$ flutter pub global run devtools -p 8000
$ cd path/to/your/app
$ flutter run --observatory-port=8080
```

Then, navigate to http://localhost:8000/?port=8080 in your Chrome browser.

Flutter Chrome OS lint analysis

The Flutter team is adding Chrome OS specific Lint Analysis checks that are available to make sure that the app that you’re building is going to work well on Chrome OS. It looks for things like required hardware in your Android Manifest that aren’t available on Chrome OS devices, permissions that imply requests for unsupported hardware, as well as other properties or code that would bring a lesser experience on these devices.

To activate these, you need to create a new analysis_options.yaml file in your project folder to include these options. (If you have an existing analysis_options.yaml file, you can update it)

content_copy

```yaml
include: package:flutter/analysis_options_user.yaml
analyzer:
 optional-checks:
    chrome-os-manifest-checks
```

To run these from the command line, use the following command:

content_copy

```
$ flutter analyze
```

Sample output for this command might look like:

```
Analyzing ...                                content_copy
warning • This hardware feature is not
supported on Chrome OS •
android/app/src/main/AndroidManifest.xml
:4:33 • unsupported_chrome_os_hardware
```

This functionality is still under development, but check back for instructions on how you can make this functionality work with your Chrome OS targeted Flutter app.