FLUTTER FOR DESKTOP: GET STARTED WITH CROSS-PLATFORM DEVELOPMENT



Flutter is a software development kit (SDK) released by Google in 2018. Today, it is gaining lots of traction. Why? Its major selling points are:

- Simple coding language in Dart
- Simple design of widgets on widgets
- Cross-platform development abilities

In fact, Flutter just improved on that last one—it's now available for desktop applications as an Alpha release. Let's look at why Flutter went this route. Then, I'll share tips on getting started on desktop.

Why desktop?

"At the core of Flutter is the engine..." - Chris Sells, Flutter Product Manager

Desktops still allow users to interact with their applications in good ways that mobile just can't. With the desktop, users spend more *quality* time at the computer, not just mindless scrolling or basic information retrieval. Here, on the desktop, users can:

- See more information

- Process more information (i.e., switching between screens or browser tabs).
- Sit and spent time with the application

As we know, desktop computers have larger screens that allow for more information to be displayed. Hard drives are generally open and easy for the user to read and write from (though iOS has always metered this).

The desktop has an experience already different from mobile. Flutter's expansion to it opens the door for Flutter developers to *also* create new experiences.

Rise in cross-platform development

Overall interest in native app development is declining, mostly for its drawbacks. People want to be able to write one body of code and use it on multiple platforms—something that is complex in native development.

Felgo was the original cross-platform development software, then React Native. Compared to React Native, Flutter is bundled with UI rendering components, navigation, testing, and a huge number of libraries. Its Flutter engine is unique and provides everything a developer needs to start building their app.

In fact, because of these changes, many developers think Flutter is on its way to replacing Electron, a leading option, as the SDK for Desktop Development.

The Flutter engine

The Flutter team's goal is to create a cross-platform UI toolkit that allows code to be reused. Essential to that mission is the Flutter engine. Simply put, the Flutter engine is responsible for putting the pixels on the screen when they are needed. The Flutter engine is crucial to Flutter's speedy yet high-quality output.

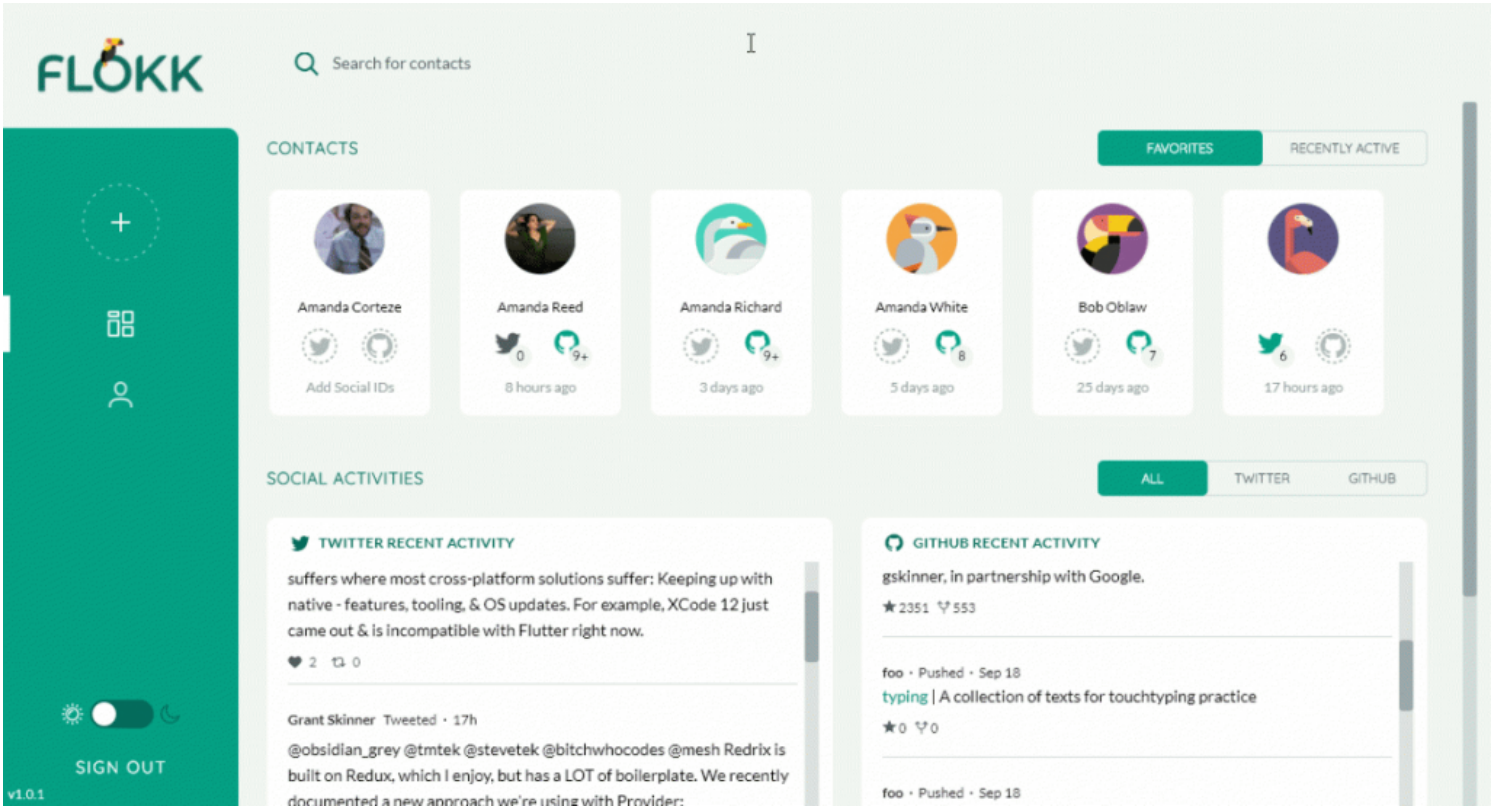
Flutter was first made available for Android and iOS. Whenever a platform is added or upgraded, it must tweak its Flutter engine. Different from mobile platforms, the new Flutter Desktop Alpha release had to account for

- More keyboard inputs
- Mouse controls (expanding from touchscreen interfaces)
- Larger screen displays

At each step, it must add to the Flutter engine. Like the other platforms, the project gets its own shell application that compiles at runtime. From here, developers can add their own native code.

Desktop app examples

Flokk shows what you can do with Flutter on desktop. They set out to make something beautiful for the desktop—and I think they succeeded:



Or, watch a developer create a simple app with Flutter Desktop Alpha:

Starter desktop packages for Flutter

There are already a lot of Flutter packages that work on all desktop, web, Android, and iOS platforms. Many support web and mobile and may need just a little reconfiguring to get working appropriately for mobile.

Here are some I like:

- Provider is a wrapper around InheritedWidget to make them easier to use and more reusable.
- Url_launcher is a Flutter plugin for launching a URL.
- SimpleAnimations is a powerful framework to create beautiful custom animations in no time.

Check out all the Flutter packages.

Setting up Flutter for desktop

The Flutter channels are mirror images of the GitHub channels. To switch, and enable, the desktop development experience, follow:

```
flutter channel dev
flutter upgrade
flutter config --enable--desktop
```

<*platform*> can be one of *windows*, *macos*, *linux*.

You should see the platform listed on your device when you run the command:

```
flutter devices
```

All that's left is to create your app and run it:

```
flutter create myapp
cd myapp
flutter run -d windows
flutter run -d macos
flutter run -d linux
```

Have fun!

Additional resources

For related reading and tutorials, explore these resources:

- BMC DevOps Blog
- Hello World: Creating Your First Flutter App
- API/Developer Portals: How To Create Great API Portals
- Kubernetes Guide, a series of articles and tutorials
- What Is Behavior-Driven Development (BDD)?