Flutter Linux Apps from Scratch

Maksim Lin
Developer Relations Engineer
Codemagic

A bit about me...


Long time Android developer now doing Flutter

Developer Relations Engineer @ Codemagic

Flutter & Dart GDE,
Melbourne GDG co-organiser

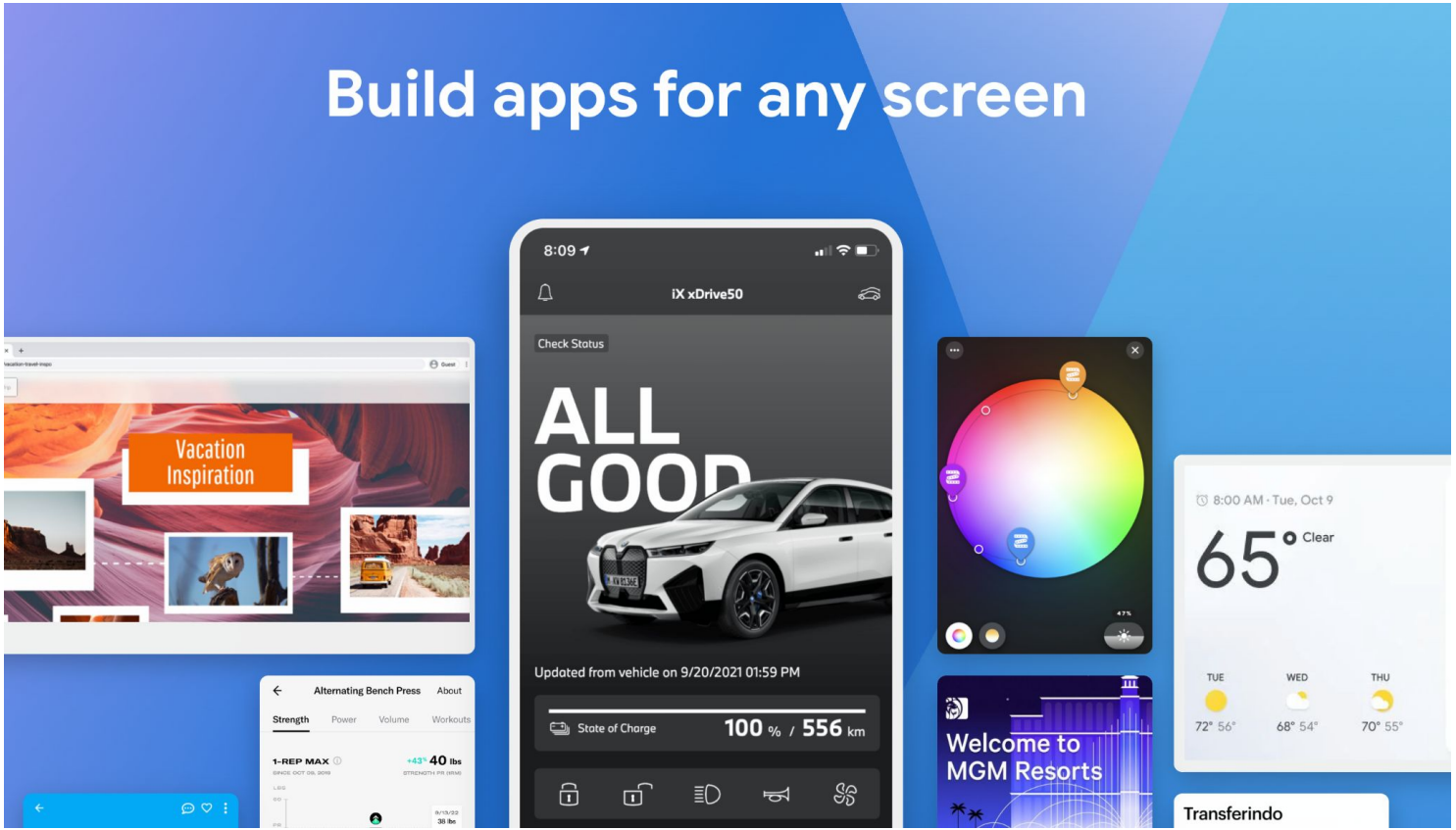
- Heard of Flutter?
- Tried using Flutter?
- Built an app with Flutter?

The Plan

1. What is Flutter?
2. Intro to Flutter development
3. Setup a development environment for Flutter
4. How to use cross platform plugins
5. Linux functionality using plugins
6. Writing Unit & Widget tests
7. Run tests, build & package using CI/CD service
8. Distribute the app to users!

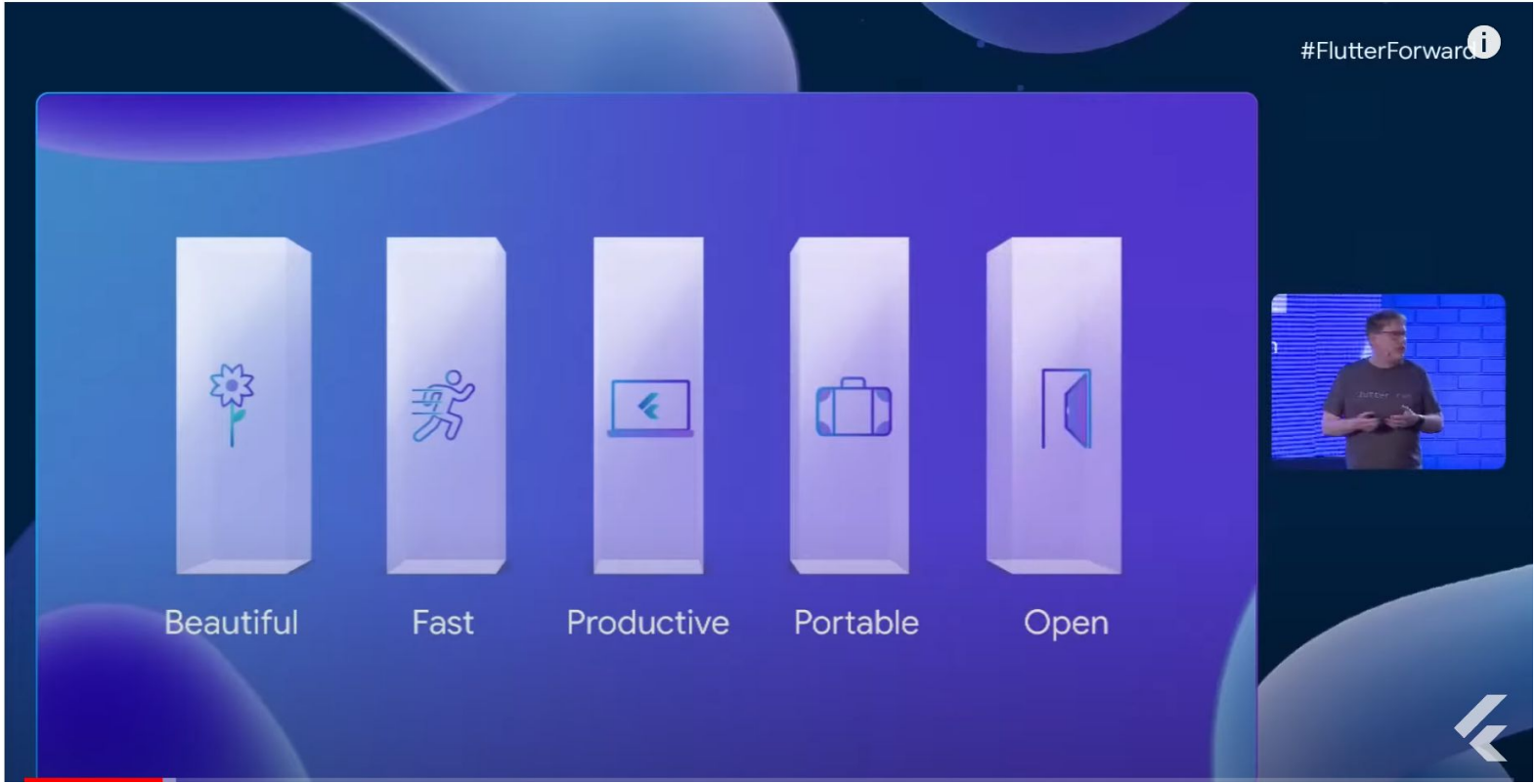
What is Flutter?

Build apps for any screen



“Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.”

Pillars



Open

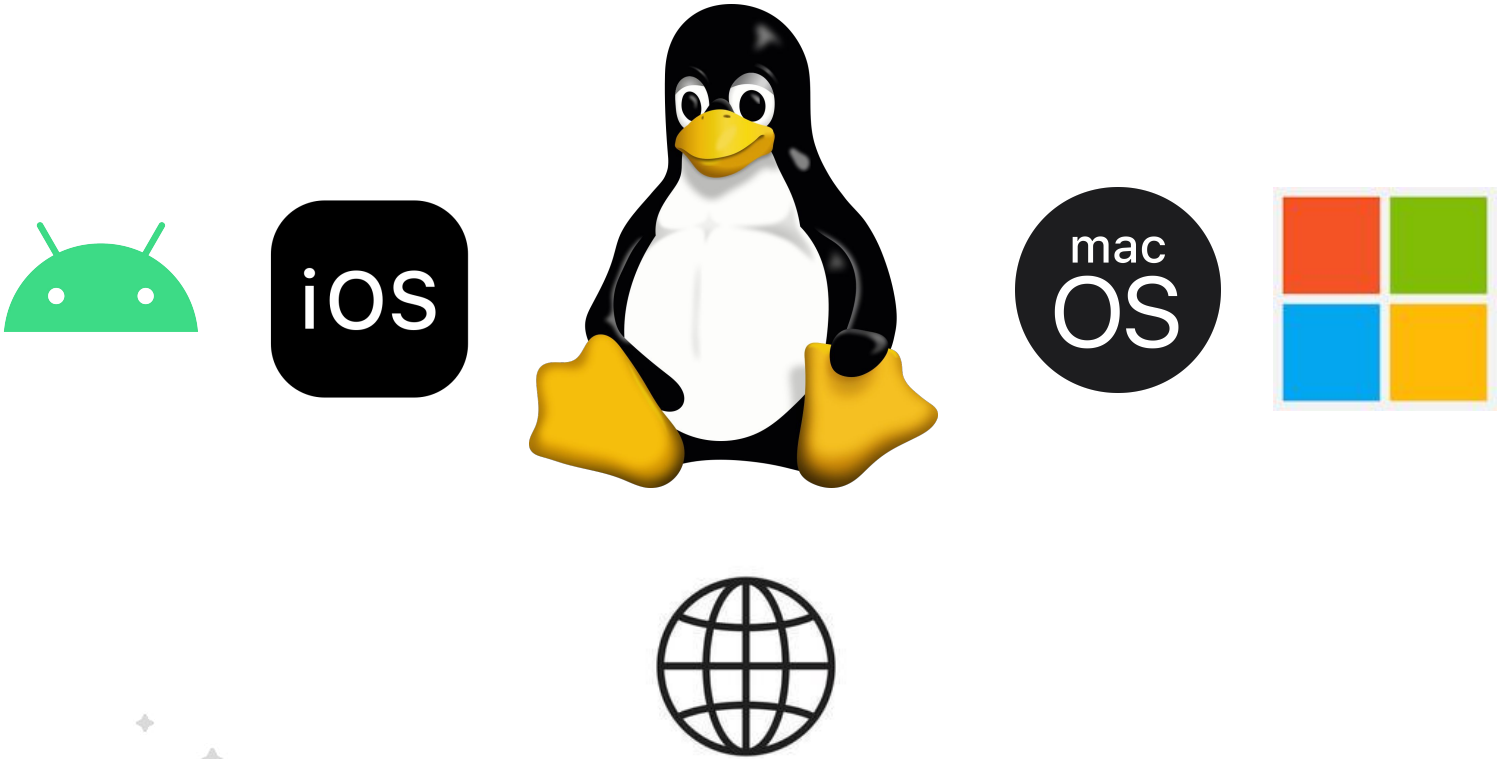
Open Source & **in the open development!**

- All git repos on Github
- All issues on Github (no private bugtrackers!)
- Most (all?) design documents are public Google Docs
- Flutter team (google & external) public discussions on Discord
- Public yearly Roadmaps on Github wiki
- Third most popular project on Github ***by contributors*** (**12.4k** !!)

Productive

- ⚡⚡⚡ ***Stateful*** **Hot-Reload** ⚡⚡⚡
  👆👆👆👆👆👆👆👆👆👆👆👆👆👆
- Especially useful when rapidly iterating on UI
- Not like web live-reload, more like Browser Devtools BUT with code saving
- Flutter uses Dart:
  - a OOP + FP lang
  - sound null-safe
  - easy for lang émigrés
- Everything in Dart, no extra UI language to learn (xml, html, css, etc)

Portable



Fast

- Release mode:
  Compiles **AOT** to native, *optimised* MACHINE INSTRUCTIONS executable code
- Uses JIT for *development* mode
- FFI for direct calling of C compatible libraries
- Multi-threading (with **non**-shared memory model)
- No “bridge” to native UI toolkit

Beautiful

- Uses own UI toolkit
- “Owns every pixel”
- Makes completely custom UI elements very easy
- Built in support for animations

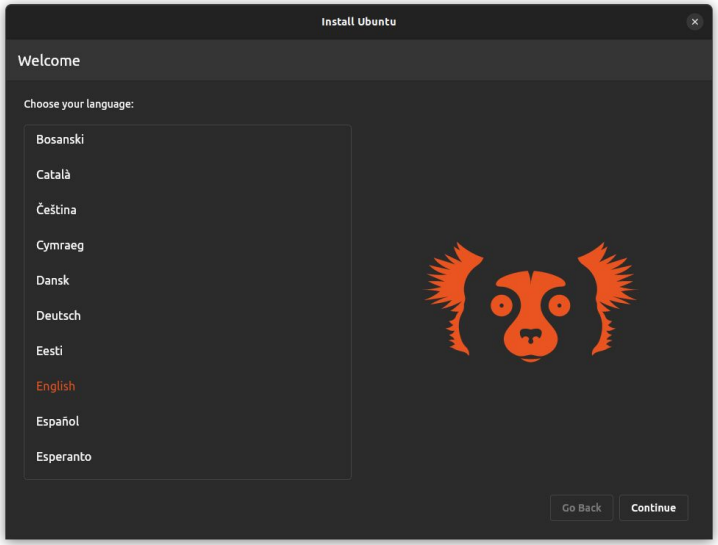
Production Ready?

>700 000

Flutter Apps

Production Ready *on Linux*?

Canonical Ubuntu Installer



https://github.com/canonical/ubuntu-desktop-installer

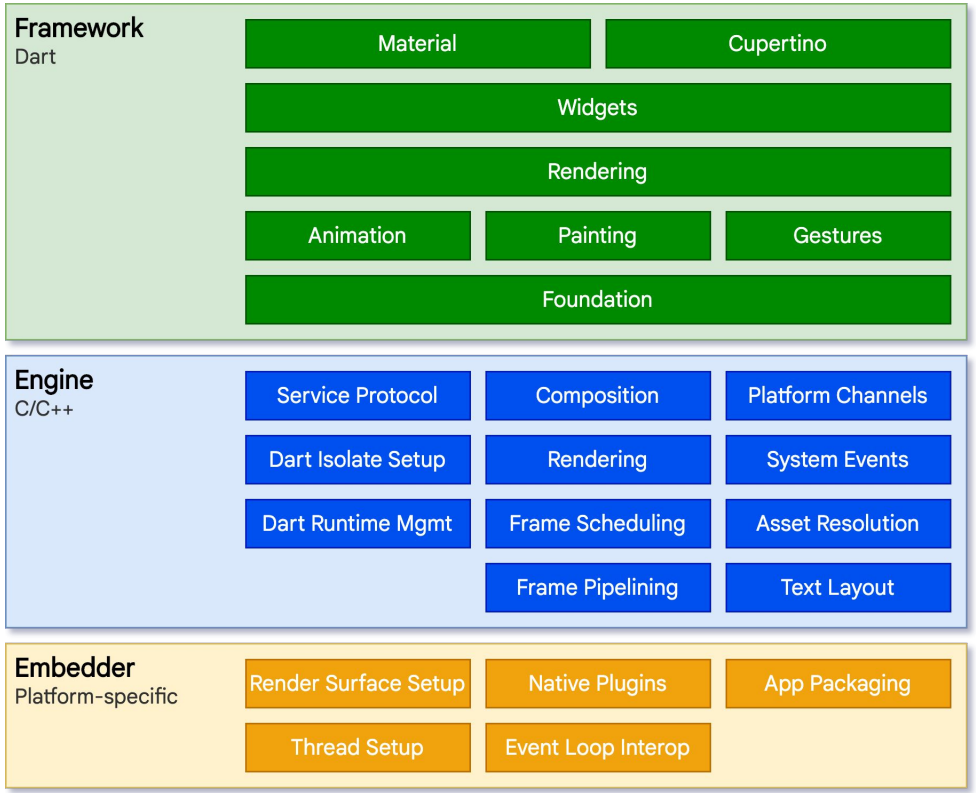
Expected to ship as default in 23.03: https://www.omgubuntu.co.uk/2023/01/ubuntu-new-flutter-installer-first-look

Dash! (MCM*)

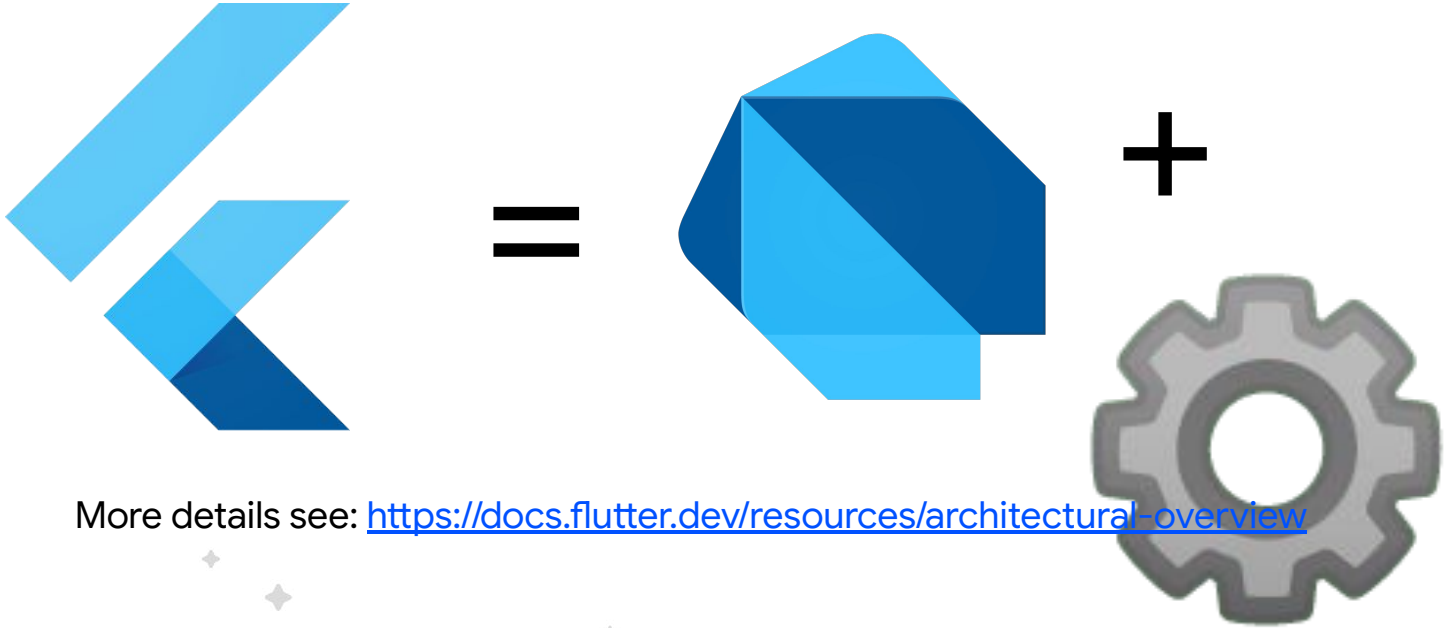
* Mandatory Cute Mascot

What is Flutter?
The details...

Flutter Architecture



Flutter = Dart + Engine



More details see: https://docs.flutter.dev/resources/architectural-overview

Framework?

“On the surface, Flutter is a reactive, pseudo-declarative UI framework, in which the developer provides a mapping from application state to interface state, and the framework takes on the task of updating the interface at runtime when the application state changes. This model is inspired by work that came from Facebook for their own React framework...”

UI = f(state)

90mins Today

Today:
90mins

Setup a development environment for Flutter

Install:

1. Git
2. Flutter SDK
3. VSCode
4. DartCode Extension
5. Linux Build prerequisites

Install: Git

```
> sudo apt install git
```

Install: Flutter SDK

Use Snapcraft

Or:

direct from:

https://docs.flutter.dev/get-started/install/linux

Install: VSCode

Use Snapscraft
Or:
direct from:
https://code.visualstudio.com/download

Install: DartCode Extension

Click on button from https://dartcode.org/

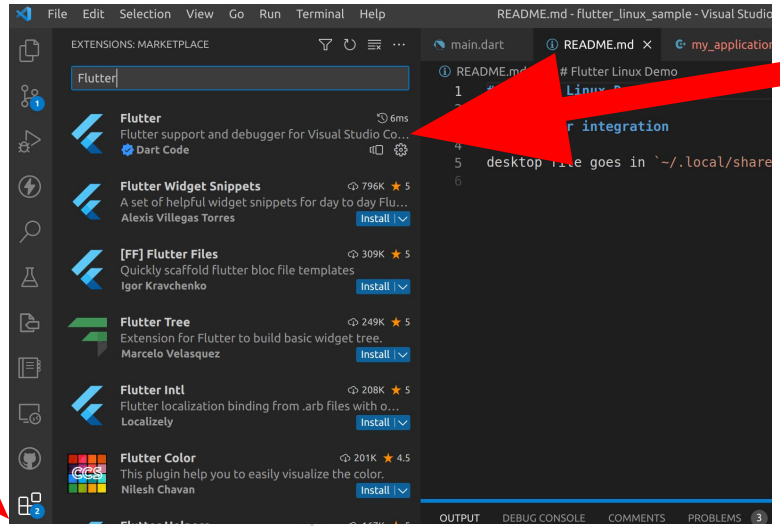
Or:

Ctrl+P then: `ext install Dart-Code.dart-code`

Or:

(1)

(2)



Install: Linux Build Prerequisites

```
sudo apt-get install clang cmake
ninja-build pkg-config libgtk-3-dev
liblzma-dev
```

Basics of Flutter app development

1. Get started

- Create a scaffold app:

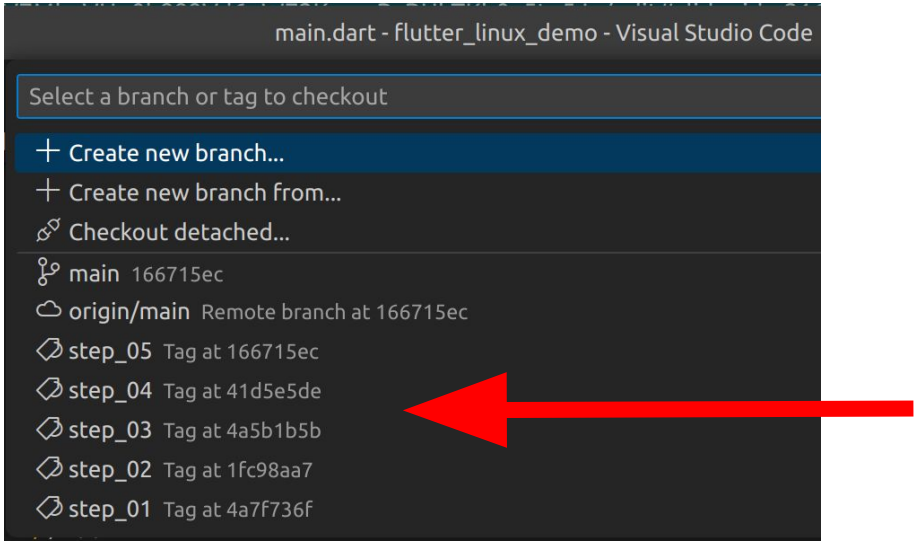
```
> flutter create fl_from_scratch
```

- Open in VSCode:

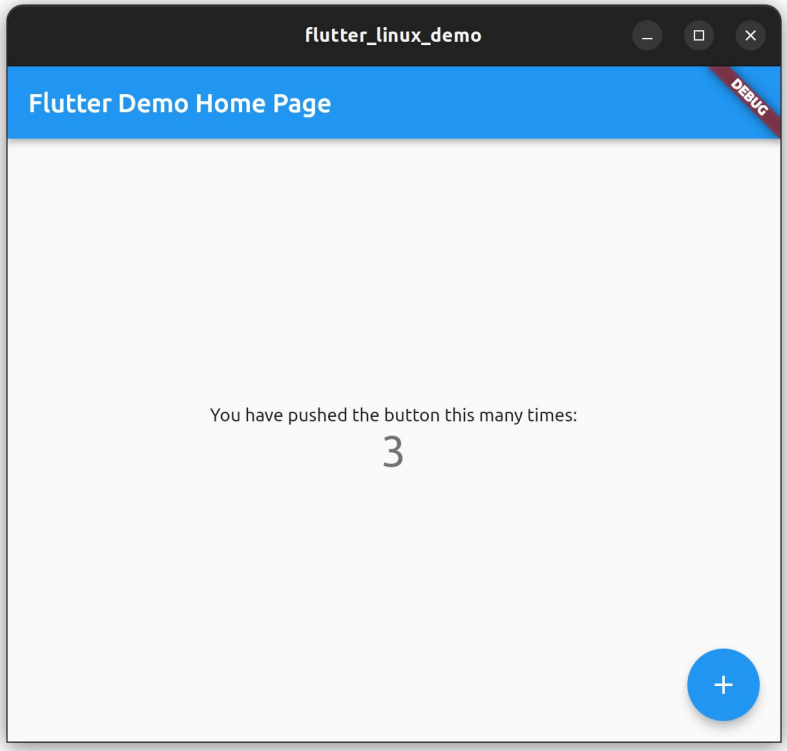
```
> code fl_from_scratch
```

Get finished! 🙂

```
git clone https://github.com/maks/flutter_linux_demo
Checkout git tags for each stage:
> git checkout step_01
```



Counter app: aka Flutter's Helloworld

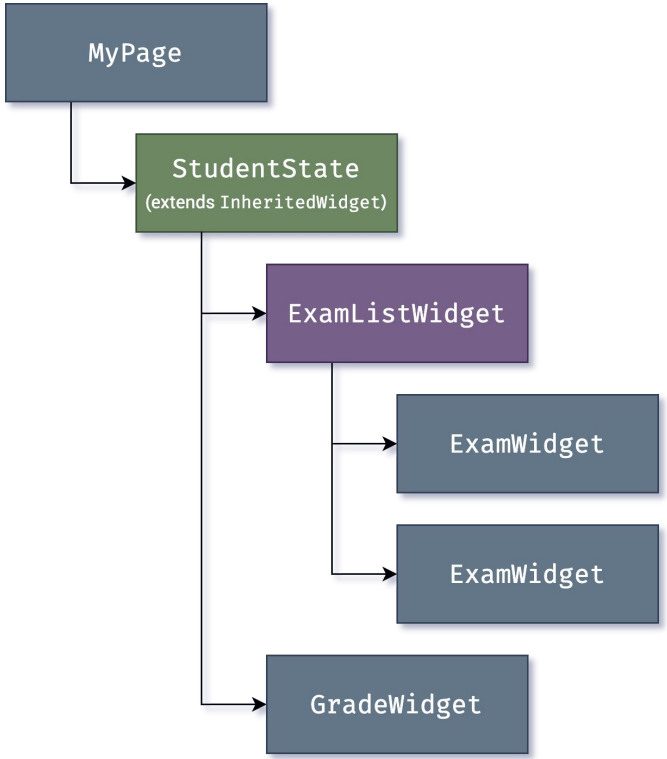


Widgets!!!

Widgets: Stateless & Stateful

- *Stateless* widgets are **immutable**
  eg. Text(“Hello world!”);
- Stateful widgets are Stateless widgets + State class
  eg. TextField(...);

Trees of widgets



DEEP!! trees of widgets

More details see:
https://docs.flutter.dev/development/tools/devtools/inspector

UI = f(state)

State Management: The final frontier!

- State management is the most controversial topic in Flutter development
- Lots of choices: ***Provider*** package is the recommended starting point
- My personal preference: ***Riverpod*** (from the author of Provider)
- Alternatives: **Bloc**, ***Redux***, etc.
- KISS: we will start with just Stateful widgets + globals 😱😉

Let’s getting building!

apod.nasa.gov/apod/ap230301.html

Astronomy Picture of the Day

Discover the cosmos! Each day a different image or photograph of our fascinating universe is featured, along with a brief explanation written by a professional astronomer.

2023 March 1

The Flaming Star Nebula
Image Credit & Copyright: Thomas Röell

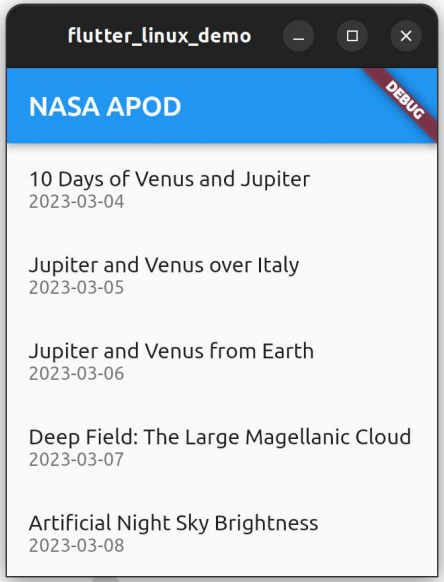
Explanation: Is star AE Aurigae on fire? No. Even though AE Aurigae is named the Flaming Star and the surrounding nebula IC 405 is named the Flaming Star Nebula, and even though the nebula appears to some like a swirling flame, there is no fire. Fire, typically defined as the rapid molecular acquisition of oxygen, happens only when sufficient oxygen is present and is not important in such high-energy, low-oxygen environments such as stars. The bright star AE Aurigae occurs near the center of the Flaming Star

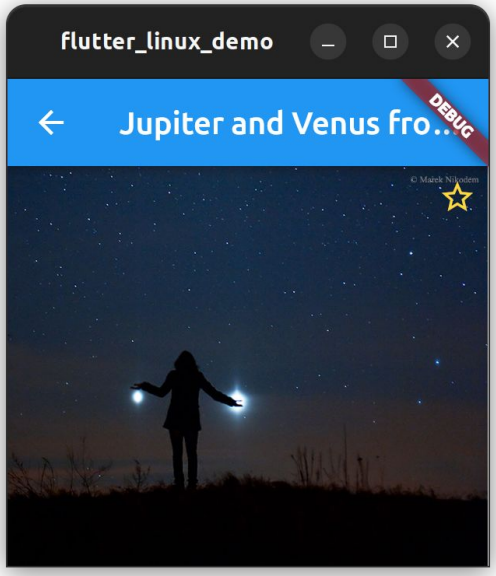
Keys...🔑🔑🔑

- Need an API key to use the NASA API
- Can request your own easily at: https://api.nasa.gov/
- Or use: “DEMO_KEY”

Using network APIs

Use NASA API to fetch the astronomy image of the day:





Favourites!

Using cross-platform plugins

Notifications & User Preferences

```
> flutter pub add desktop_notifications
> flutter pub add shared_preferences
```

Linux functionality using Flutter plugins

Showing a Icon badge

Unity Desktop API: https://wiki.ubuntu.com/Unity/LauncherAPI
(uses DBus - this will be important later on)

Need a .desktop file

Then must use this .desktop file name with launcher entry package.

Add launcher to: ~/.local/share/applications

Desktop file name MUST match appid,
eg. com.manichord.flutter_linux_demo

Linux functionality: Adding some polish...

Set app bar title, set initial window size:

6 linux/my_application.cc

```
@@ -40,14 +40,14 @@ static void my_application_activate(GApplication* application) {
  if (use_header_bar) {
    GtkHeaderBar* header_bar = GTK_HEADER_BAR(gtk_header_bar_new());
    gtk_widget_show(GTK_WIDGET(header_bar));
-   gtk_header_bar_set_title(header_bar, "flutter_linux_demo");
+   gtk_header_bar_set_title(header_bar, "NASA APOD");
    gtk_header_bar_set_show_close_button(header_bar, TRUE);
    gtk_window_set_titlebar(window, GTK_WIDGET(header_bar));
  } else {
-   gtk_window_set_title(window, "flutter_linux_demo");
+   gtk_window_set_title(window, "NASA APOD");
  }

- gtk_window_set_default_size(window, 1280, 720);
+ gtk_window_set_default_size(window, 640, 480);
  gtk_widget_show(GTK_WIDGET(window));
```

Unit & Widget tests

But first: Architecture

- Testable code drives good architecture, it’s very difficult to test badly architected code
- Good architecture? Clean, SOLID
- eg.
  - Inversion of Control
  - Single responsibility
  - Interfaces & Implementations
  - etc

Refactoring

- Instead of accessing `SharedPreferences` system api directly in `NasaAPODService` (*interfaces/implementations*), make it a separate service that is provided to `NasaAPODService` (*inversion of control*)
- Instead of doing network requests directly inside `NasaAPODService` do them in a separate service (*single responsibility*) again provided to `NasaAPODService` (*inversion of control*)

Unit Tests

- *“A unit test tests a single function, method, or class.”*
- *“The goal of a unit test is to verify the correctness of a unit of logic under a variety of conditions.”*
- Foundation of testing
- Cheaper to write, maintain
- Good for testing functionality
- Don’t give confidence for inter-unit interoperability

Widget Tests

- *“A widget test (in other UI frameworks referred to as component test) tests a single widget.”*
- *“The goal of a widget test is to verify that the widget’s UI looks and interacts as expected.”*
- Test widgets in a stand-alone environment with “enough Flutter” to be able to test widgets lifecycles

Integration Tests

There are also integration tests which test are actual app running in real environment, but we won’t cover details of writing and running those today.

Build, Test & Package on CI/CD

Automation

- Create a shell script that will do all the steps for us:

```
git clone https://github.com/maks/fl_from_scratch
cd fl_from_scratch
flutter pub get
flutter test
flutter build linux
snapcraft snap --output flutter_linux_sample.snap
```

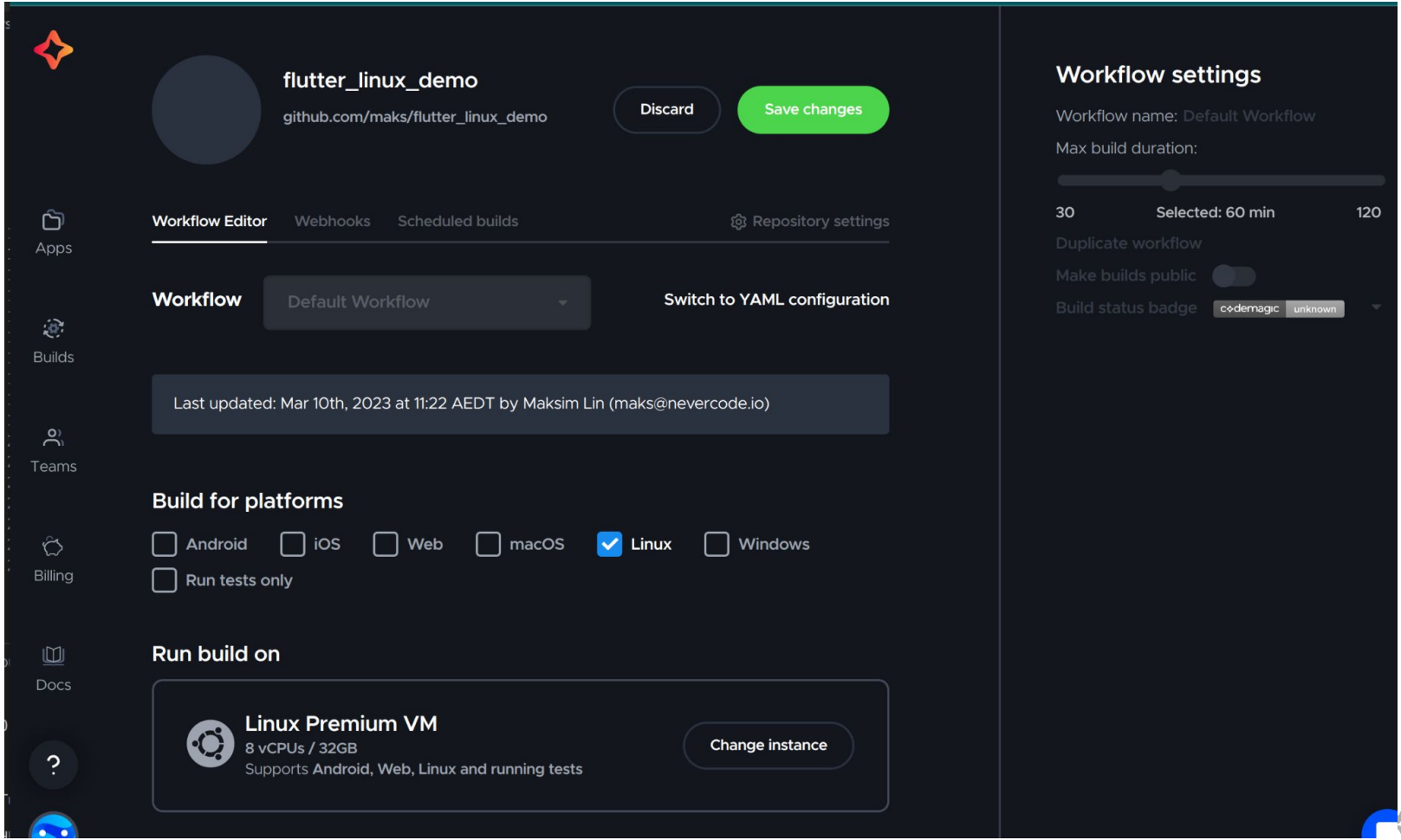
CI: Continuous Integration

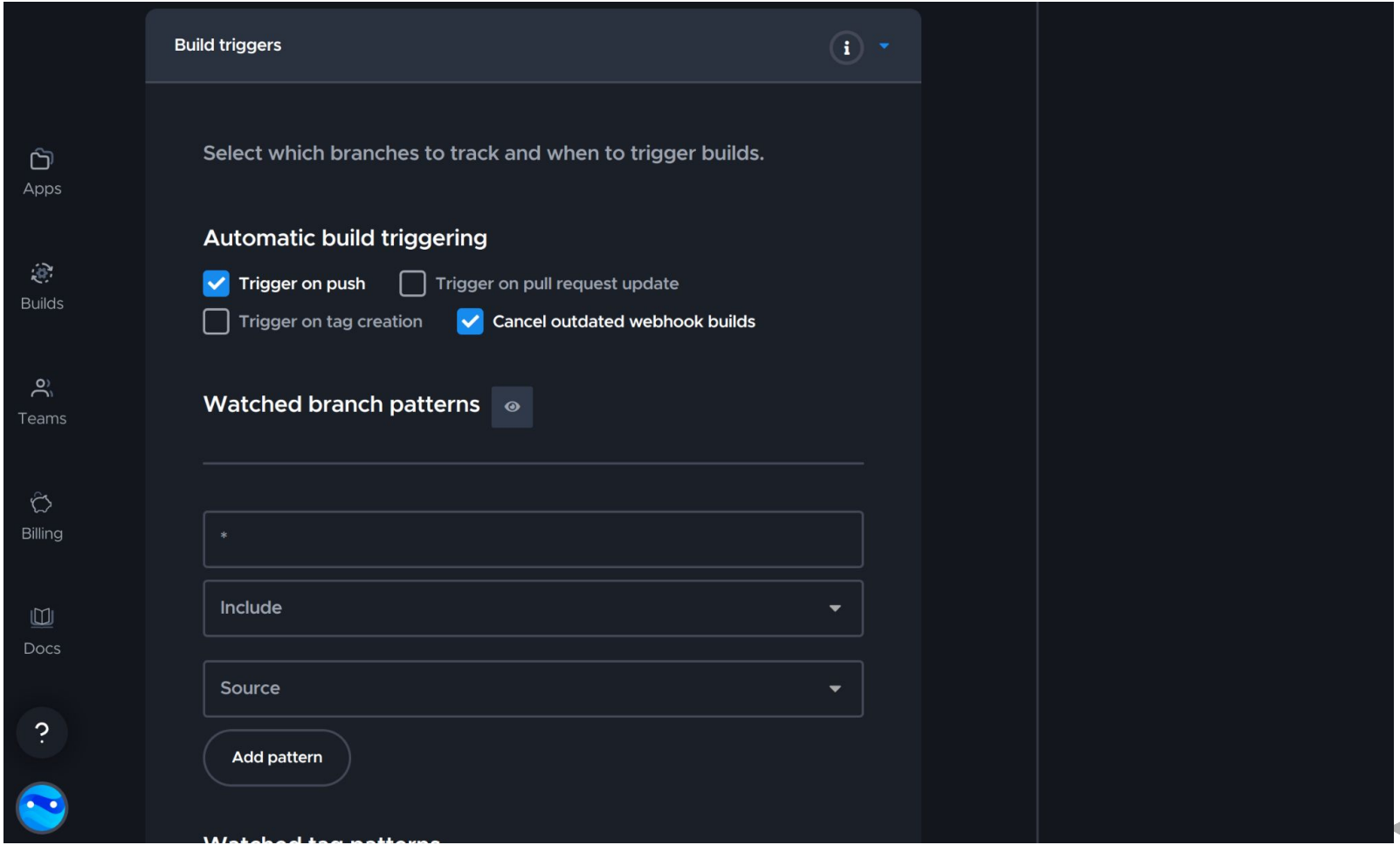
- Runs the scripts for us, plus sets up the environment for us
- Run builds in a consistent, “clean” environment
  Avoids the *“it built on my machine”* problem
- Trigger builds manually or based on webhooks (eg. PRs, git pushes)
- Run tests, gather test coverage
- Use above as pre-conditions for PR approval
- Build for targets you don’t have locally (eg. iOS builds require MacOS)

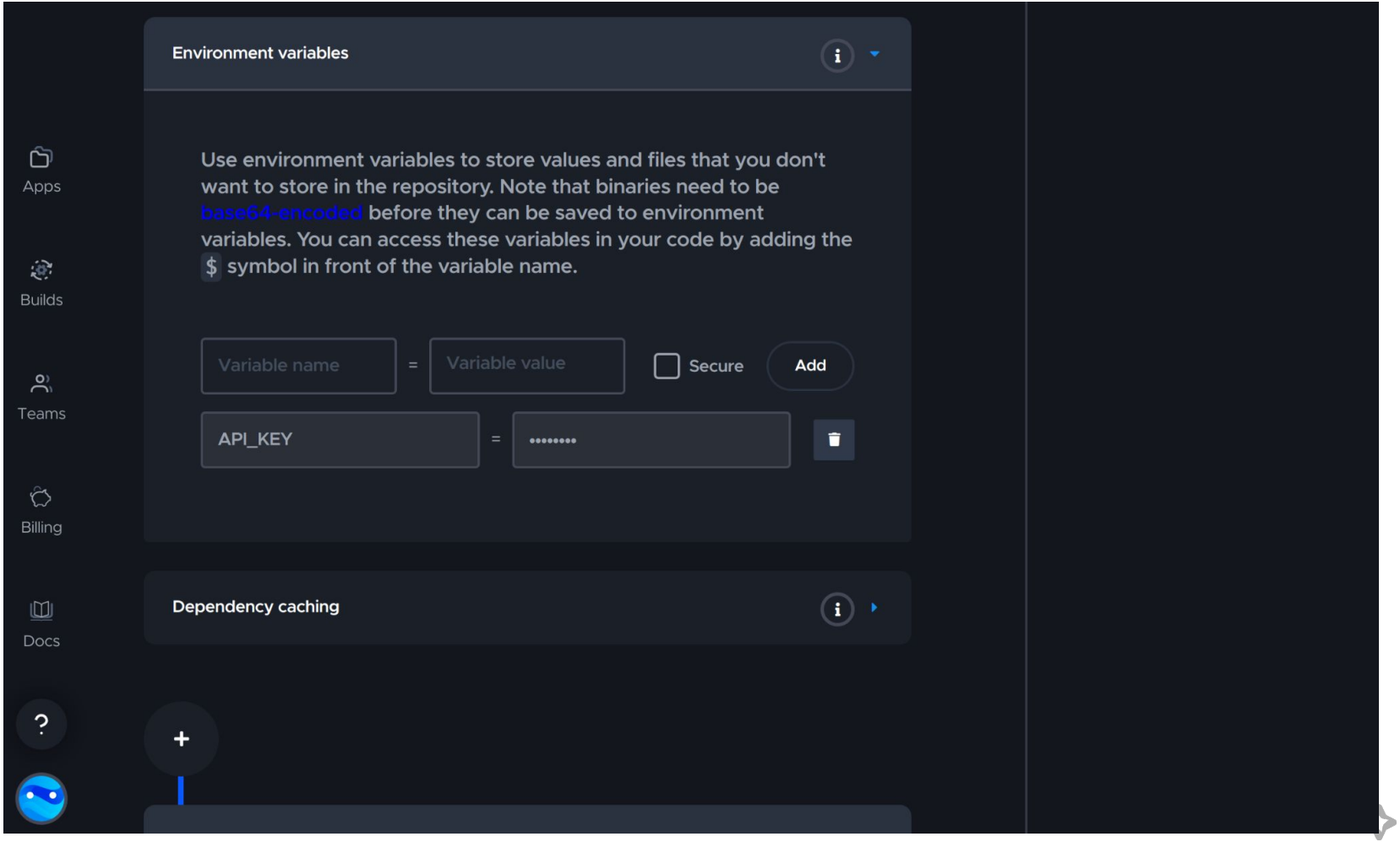

codemagic

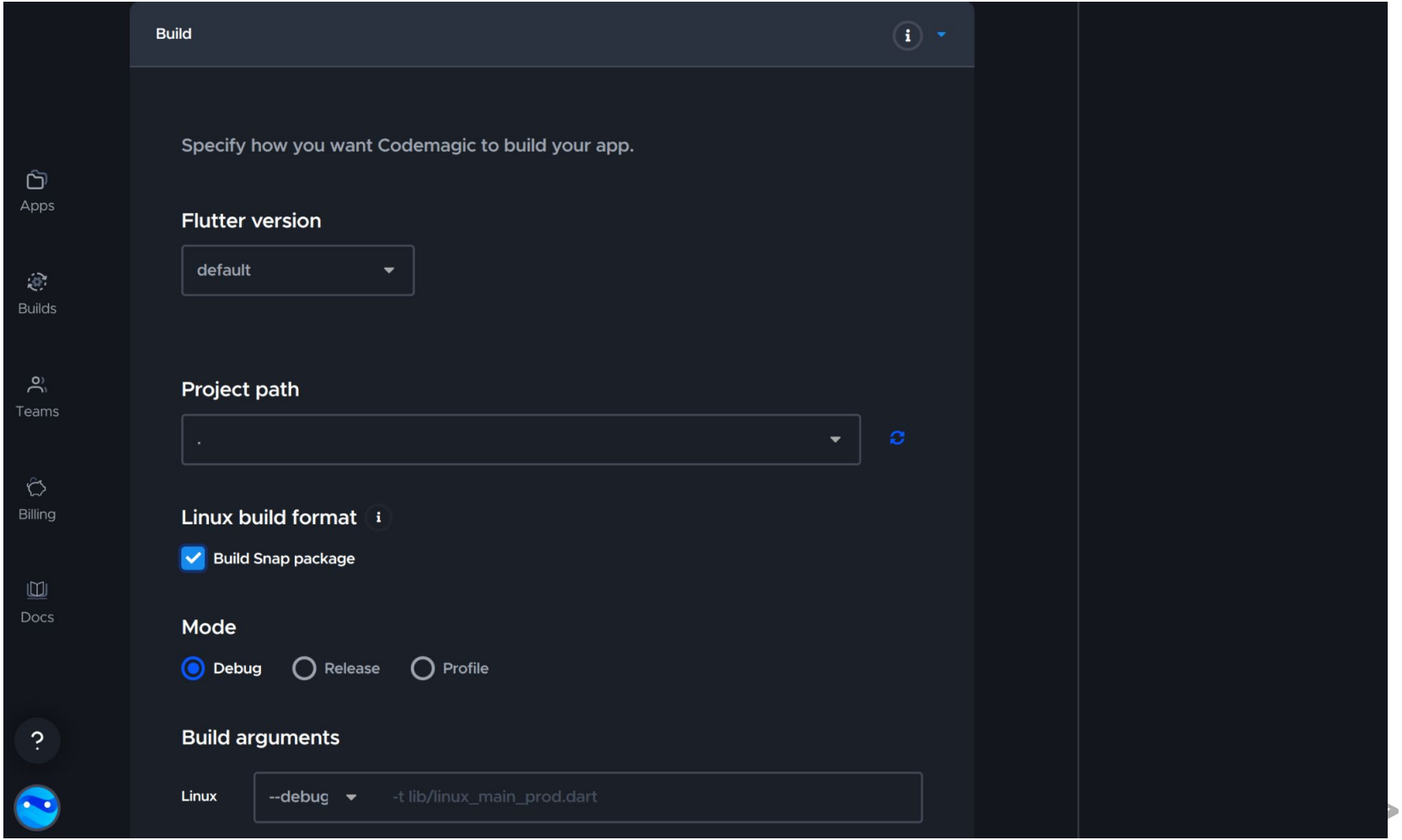
Codemagic: Workflow Editor

- Workflow Editor provides web based GUI to setup for Flutter Apps
- Easiest way to get started (in mins!)
- Not ideal long term as no way to do change management
- Not flexible enough to handle all use cases


flutter_linux_demo
github.com/maks/flutter_linux_demo
Discard
Save changes
Apps
Builds
Teams
Billing
Docs
Workflow Editor
Webhooks
Scheduled builds
Repository settings
Workflow
Default Workflow
Switch to YAML configuration
Last updated: Mar 10th, 2023 at 11:22 AEDT by Maksim Lin (maks@nevercode.io)
Build for platforms
Android
iOS
Web
macOS
Linux
Windows
Run tests only
Run build on
Linux Premium VM
8 vCPUs / 32GB
Supports Android, Web, Linux and running tests
Change instance
Workflow settings
Workflow name: Default Workflow
Max build duration:
30
Selected: 60 min
120
Duplicate workflow
Make builds public
Build status badge
codemagic
unknown

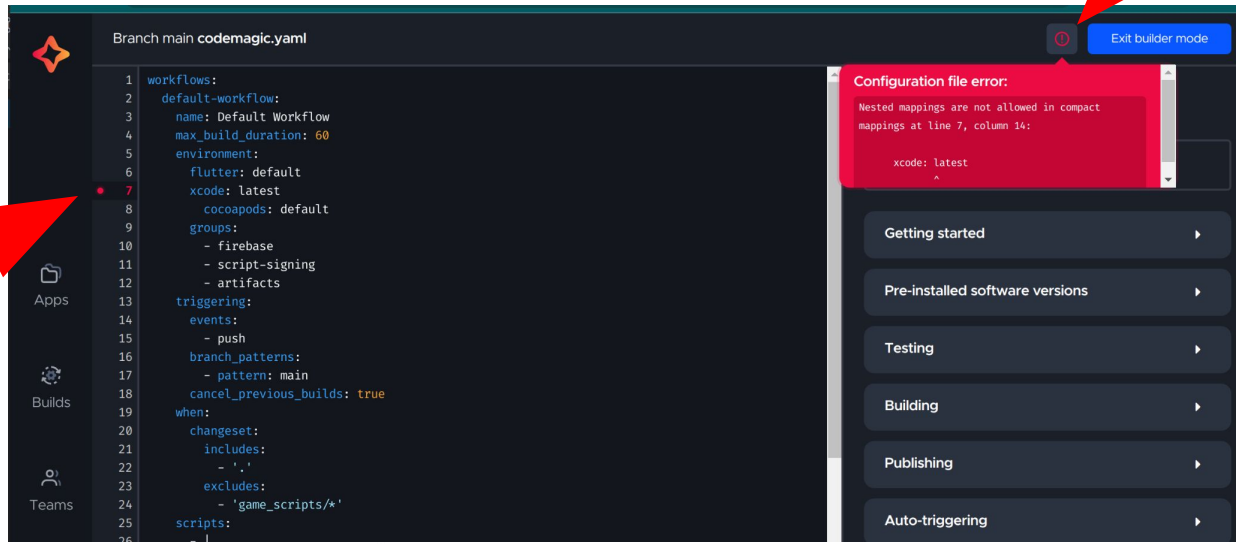

Build triggers
Apps
Builds
Teams
Billing
Docs
Select which branches to track and when to trigger builds.
Automatic build triggering
Trigger on push
Trigger on pull request update
Trigger on tag creation
Cancel outdated webhook builds
Watched branch patterns
*
Include
Source
Add pattern


Environment variables
Use environment variables to store values and files that you don't want to store in the repository. Note that binaries need to be base64-encoded before they can be saved to environment variables. You can access these variables in your code by adding the $ symbol in front of the variable name.
Variable name
=
Variable value
Secure
Add
API_KEY
=
••••••••
Dependency caching
Apps
Builds
Teams
Billing
Docs


Build
Specify how you want Codemagic to build your app.
Apps
Builds
Teams
Billing
Docs
Flutter version
default
Project path
.
Linux build format
Build Snap package
Mode
Debug
Release
Profile
Build arguments
Linux
--debug
-t lib/linux_main_prod.dart

Codemagic: YAML

- Use YAML to allow us to change manage our CI configuration
- Can easily re-use previous configurations
- Validate using Builder Mode:



App Distribution

CD: Continuous Deployment

- First need to get Snapcraft credentials for CI:

```
snapcraft export-login snapcraft-login-credentials
cat  snapcraft-login-credentials | base64
```

- Using Snap Store for Linux app distribution:

```
snapcraft snap upload
flutter_linux_sample.snap --release stable
```

Snap! and it’s broken...😓

- Snap imposes a security sandbox using AppArmor
- Restrictions include access to network - that app uses!
- Restrictions on DBus - that apps uses! (for Unity Launcher API)
- Snap also changes name of our .desktop file!

Alternatives to Snap: AppImage & Flatpak

Some helpful resources:

https://github.com/AppImage/AppImageKit/issues/1122

https://github.com/AppImageCrafters/appimage-builder-flutter-example/

https://github.com/Merrit/flutter_flatpak_example

Credits

Artemii Yanushevskyi:

https://github.com/codemagic-ci-cd/flutter-snapcraft-example/

Thank You!
Questions?

fluttercommunity.social/@maks

@mklin

maks