This page describes common workflows for working with the Flutter Engine on Fuchsia.

Prerequisites

This page assumes that:

- You have set up your Engine development environment.
- You have set up a Fuchsia development environment.
- You have set the `$ENGINE_DIR` shell environment variable to the `src/` directory of your Engine checkout.
- You have set the `$FUCHSIA_DIR` shell environment variable to your Fuchsia source checkout.

Building and deploying Flutter Engine on Fuchsia

See the main Compiling for Fuchsia wiki page. You will want to follow those instructions at least once and familiarize yourself with the different options and details described there. This page serves as a refresher on common workflows once you've gone through all the setup that page describes.

Creating and testing a feature branch

The requirement to sync Dart versions requires basing your work off of Fuchsia's version of the Flutter Engine. A utility script is provided to simplify starting new feature branches. A common workflow for creating and iterating on a feature branch for Flutter on Fuchsia in debug mode is:

```
# Create a new feature branch from your $FUCHSIA_DIR's checkout of Flutter Engine.
$ENGINE_DIR/flutter/tools/fuchsia/devshell/branch_from_fuchsia.sh "<my-new-feature-
branch>"

# Sync dependencies. This is only necessary if your previous 'gclient sync'
# was done on a checkout ahead of $FUCHSIA_DIR's Flutter Engine.
cd $ENGINE_DIR && gclient sync

# Do some work and stage some changes...

# Commit the work. This is useful for verifying that your build of the Flutter
Runner is
# being used. See 'Verifying that your Flutter Runner is being used' below.
git -C $ENGINE_DIR/flutter commit -m "[fuchsia] My new feature."
```

You can then build and deploy your changes to Fuchsia.

```
# Generate Ninja files for a debug JIT build of Engine without goma.
# More options are described at https://github.com/flutter/flutter/wiki/Compiling-
the-engine#build-the-engine.
$ENGINE_DIR/flutter/tools/gn --fuchsia --no-lto

# Build the Flutter JIT runner.
ninja -C $ENGINE_DIR/out/fuchsia_debug_x64

# Copy the Flutter JIT runner over to your Fuchsia prebuilts.
cp $ENGINE_DIR/out/fuchsia_debug_x64/flutter_jit_runner-0.far
```

```
$FUCHSIA_DIR/prebuilt/third_party/flutter/x64/debug/jit/flutter_jit_runner-0.far

# Register debug symbols.
$ENGINE_DIR/fuchsia/sdk/linux/tools/symbol-index add
$ENGINE_DIR/out/fuchsia_debug_x64/.build-id $ENGINE_DIR/out/fuchsia_debug_x64

# Build your configuration of Fuchsia with your Flutter runner.
cd $FUCHSIA_DIR && fx build

# Run an emulator and test your changes.
# https://fuchsia.dev/fuchsia-src/get-started/set_up_femu
fx vdl start -N

# Iterate on your change.
# '--amend' helps to reduce the number of commits that you need to rebase
# on top of Fuchsia's checkout of Flutter, simplifying your workflow.
git -C $ENGINE_DIR/flutter commit --amend

# Push and make a pull request once you've verified your change works.
git -C $ENGINE_DIR/flutter push origin my-feature-branch
```

Testing a commit from master on Fuchsia

When you have commits that you need to test on a branch that is synced to `upstream/master` (instead of Fuchsia's checkout of Flutter), you will need to sync back to Fuchsia's checkout of Flutter and then rebase your commits. This ensures that the Dart versions are synced between Fuchsia and your Engine build.

One way is to cherry-pick each commit you want to test on top of Fuchsia's checkout of Flutter.

```
# Checkout your feature branch that is synced to `upstream/master`.
git -C $ENGINE_DIR/flutter checkout "<my-existing-feature-branch>"

# Find the commits that you're interested in testing on Fuchsia.
git -C $ENGINE_DIR/flutter log

# Checkout Fuchsia's version of the Flutter Engine.
$ENGINE_DIR/flutter/tools/fuchsia/devshell/sync_to_fuchsia.sh

# Sync dependencies.
cd $ENGINE_DIR && gclient sync

# Cherrypick commits on top.
git -C $ENGINE_DIR/flutter cherry-pick "<commit1>" "<commit2>" ...

# Follow the regular flow to GN, build, and deploy the runners, as described above.
...
```

Verifying that your Flutter runner is being used

The Flutter runner is a Fuchsia package that runs Flutter apps on Fuchsia using the Flutter Engine. You can use `ffx inspect` to verify that your build of the Flutter runner is being used.

These instructions assume that you already have Fuchsia running Flutter on a connected device.

```
# Prerequisite: 'ffx' commands must be run from '$FUCHSIA_DIR'.
cd $FUCHSIA_DIR

# List running Flutter components that can be inspected.
ffx inspect list | grep flutter

# Inspect the Flutter runner that was listed. For example:
ffx inspect show flutter_jit_runner.cmx
```

The `build_info` from `ffx inspect show` will tell you what Git revision is being used for the Engine in your runner. You can compare this against the commit of the runner you deployed to make sure it matches.

```
      build_info:
        dart_sdk_git_revision = 8ed0a66c8802bd4e711105fe01d0dc581a95fdaf
        dart_sdk_semantic_version = 2.15.0-91.0.dev
        # Should match your latest commit from 'git -C $ENGINE_DIR/flutter log'.
        flutter_engine_git_revision = fdb7421adbbb4bd814107fdb986b85195dd34a66
        fuchsia_sdk_version = 6.20210910.1.1
```

Collecting memory usage

You can run the following commands to get a Dart heap snapshot from a Flutter app running on Fuchsia. Note that this will tell you about memory usage from the Dart VM but not from the Flutter runner itself.

1. Run a profile build of the runner by passing `--runtime-mode=profile` when compiling for Fuchsia.

```
$ENGINE_DIR/flutter/tools/gn --fuchsia --no-lto --runtime-mode=profile
```

2. Run a Flutter component using your runner.
3. Connect to your component's Dart VM.

```
cd $FUCHSIA_DIR
fx dart-tunnel
```

4. Connect to the website that the tool tells you.
5. Collect a heap snapshot.