Several of our dependencies are automatically rolled (updated) by bots.

Clang to Engine

We use an auto-roller for Clang on Linux and MacOS (Windows is pending availability of a Windows Clang package from the Fuchsia infra team). In case of build failures or other errors, ping the #hackers-engine channel on Discord.

These rollers may fail if Clang catches a new compilation warning or error that it previously did not, or if a test relies on undefined behavior that has now changed in the new revision of Clang. It is best to resolve such issues ASAP to let the rollers continue and avoid a pile up of issues to resolve.

The rollers work by updating a CIPD package version in the DEPS file. You can map from a CIPD version to a git revision by checking in CIPD.

Fuchsia SDK to Engine

We use an auto-roller for the Fuchsia SDK on Linux and MacOS (Windows is pending availability of a Windows Fuchsia SDK package from the Fuchsia infra team). In case of build failures or other errors, ping the #hackers-engine channel on Discord.

These rollers may fail if the Fuchsia SDK contains a breaking change. It is best to resolve such issues ASAP to let the rollers continue and avoid a pile up of issues to resolve.

The rollers work by updating a CIPD package version in the DEPS file. You can map from a CIPD version to a JIRI snapshot or a git revision by checking in CIPD.

Skia to Engine

We use an auto-roller for Skia rolls. It's status can be viewed at https://skia-flutter-roll.skia.org/. In case of build failures or other errors, ping the Flutter-Skia chat channel. In case you get no response, you can login with an @google.com account and pause the roller (or ask someone with an @google.com account to do so). Please specify a descriptive reason and file a bug to re-enable the rollers as soon as possible.

The bot updates the `skia_revision` line of https://github.com/flutter/engine/blob/master/DEPS.

Skia also uses an auto-roller for Fuchsia; see https://autoroll-internal.skia.org/r/fuchsia-autoroll.

Engine to Framework

The engine is automatically rolled to the framework. It is configured by https://skia.googlesource.com/buildbot/+/refs/heads/master/autoroll/config/flutter-engine-flutter.json.

The bot updates https://github.com/flutter/flutter/blob/master/bin/internal/engine.version to point to the latest revision of the engine *whose artifacts built successfully*, as determined by looking at the Engine Console.

Making a breaking change

Our breaking change policy disallows making changes to the engine that require changes to the framework. If you find the need to do this, you should instead make a soft-breaking change which you can land in multiple phases, as described in that process.

If for some reason you need an exemption to this policy, contact @Hixie.

Doing a manual roll

To roll the engine manually in the case you have a breaking change exemption, you'll need to land the change to `engine.version` manually in the same PR to the framework as the one where you fix the framework to work with the new API.

When you change the `engine.version` file locally, you should delete `$FLUTTER_ROOT/bin/cache` and then run `flutter precache` to ensure that all your local artifacts and snapshots are updated. You can then run tests and be sure that they are running against the latest version of the assets you need.

You may find it helpful to use the `$ENGINE_ROOT/src/flutter/tools/engine_roll_pr_desc.sh` to create a PR description. Doing this helps us track down what commits have rolled in more quickly, and properly link to other commits and pull requests for commenting and tracking.

For example, to generate a description from hash deadbeef to beefdead:

```
$ ./tools/engine_roll_pr_desc.sh deadbeef..beefdead
```

See also: [[Debugging the engine]], which includes instructions on bisecting a roll failure.

Dart to Engine

The Dart SDK is automatically rolled into the engine on a regular basis, following the steps laid out at the Rolling Dart page. Since this process is a bit more involved, this autoroller does not use the Skia infrastructure and has a custom dashboard hosted at go/dart-sdk-roller-dashboard (**note: this is likely only accessible from a machine on the Google network**). Using the dashboard,

the autoroller can be paused, rolls can be triggered and cancelled, and rolls to a particular revision can be done.

If there are any issues with this process or the autoroller dashboard, contact bkonyi@ or a member of the Dart VM team.