Flutter has installation bundles which you can download and install for the `beta` channel. (They were previously also available for the `dev` channel, but the dev channel has been retired.)

The installation bundles were designed to allow you to have a completely populated environment without having to first download the Git repository, then compile the flutter tool, etc.

Creating an installation bundle consists of running the following steps. These steps will pre-cache all of the necessary components for working offline. One could follow these steps e.g. to create an installation bundle for the `master` branch.

1. *OPTIONAL*: Set and export the `PUB_CACHE` environment variable to point to the location where you would like your pub cache to be stored. If you don't set this, it will be stored in pub's default location in your home directory (varies based on your OS). Alternatively, if you create the `flutter/.pub-cache` directory before any commands that invoke `pub` (e.g. the `flutter packages` command), then the flutter tool will automatically use it, and you won't have to set PUB_CACHE in your environment.
2. *WINDOWS ONLY*: Download and install a command line version of Git. Windows doesn't come with git pre-installed, so you'll have to download it from one of the many sources on the Internet.
3. `git clone -b master https://github.com/flutter/flutter.git` to clone the Flutter Git repo into the directory "flutter" in the current directory.
4. Change directories into the newly created "flutter" directory.
5. Add the Flutter `bin` directory to your path:
   - Windows: `set PATH="%PATH%;%CD%\bin\flutter"`
   - Other platforms: `export PATH="$PATH:$PWD/flutter"`
6. `flutter doctor` to check your installation and build the initial snapshot of the `flutter` tool. Install any missing software it tells you about, such as the Android SDK, or XCode tools.
7. `flutter update-packages` will download all of the pub package dependencies needed to build any of the packages in the Flutter main distribution.
8. `flutter precache` will make sure that the `flutter` tool's cache of binary artifacts is up-to-date.
9. *OPTIONAL*: `flutter ide-config --overwrite` will generate and update IDE configuration files for the Flutter repo. You only need to do this if you are using an IDE like IntelliJ or VS Code.
10. In a temporary directory, run `flutter create --template=app app_sample`, `flutter create --template=package package_sample`, and `flutter create --template=plugin plugin_sample`. You may then remove the app_sample, package_sample, and plugin_sample directories. This will populate the pub cache with any additional packages needed for creating new flutter projects using each of those templates.

Another alternative to the above steps is to run the prepare_package.dart script directly (which is what we use to create the installation bundles in the first place). You would invoke that script like this:

1. Go as far as the `flutter doctor` step in the above steps (or if you have a working flutter repo already, you can skip that: it can be from another channel, but use a fairly current one for best results).
2. Invoke `./bin/cache/dart-sdk/bin/dart ./dev/bots/prepare_package.dart --temp_dir $TMPDIR --revision $REVISION --branch master --output $PWD`, from the `flutter` root, where `TMPDIR` is set to a directory where there are several gigabytes of free space, and `REVISION` is set to the 40-character git hash of the revision you wish to be working on.

This will build an archive bundle in the current directory that has done all of the above steps, and contains the `master` branch.