Clickable Documentation

Release 4.0.0

Brian Douglass

May 15, 2018

Contents

Build and compile Ubuntu Touch apps easily from the command line. Deploy your apps to your Ubuntu Touch device for testing or test them on any desktop Linux distribution. Get logs for debugging and directly access a terminal on your device.

Clickable is fully Open Source and can be found on GitHub. Clickable is developed by Brian Douglass with a huge thank you to all the contributors.

CHAPTER 1

Using Clickable

1.1 Install

1.1.1 Install Via PPA (Ubuntu)

- Add the PPA to your system: `sudo add-apt-repository ppa:bhdouglass/clickable`
- Update your package list: `sudo apt-get update`
- Install clickable: `sudo apt-get install clickable`
- Configure docker for clickable: `clickable setup-docker`

1.1.2 Install Via AUR (Arch Linux)

- Using your favorite AUR helper, install the clickable package
- Example: `pacaur -S clickable`

1.1.3 Install Via Git

- Install Docker
- Install Cookiecutter
- Clone this repo: `git clone https://github.com/bhdouglass/clickable.git`
- Add clickable to your PATH: `echo "export PATH=\$PATH:\$HOME/clickable" >> ~/.bashrc`
- Read the new `.bashrc` file: `source ~/.bashrc`
- Configure docker for clickable: `clickable setup-docker`

1.2 Getting Started

- Run `clickable init` to get started with a new app.
- Choose from the list of *app templates*.
- Provide all the needed information about your new app.
- When the app has finished generating, enter the newly created directory containing your app.
- Run `clickable` to compile your app and install it on your phone.

1.2.1 Getting Logs

To get logs from you app simply run *clickable logs*. This will give you output from C++ (`QDebug() << "message"`) or from QML (`console.log("message")`) in addition to any errors or warnings.

1.2.2 Running on the Desktop

Running the app on the desktop just requires you to run `clickable --desktop`. This is not as complete as running the app on your phone, but it can help speed up development.

1.2.3 Accessing Your Device

If you need to access a terminal on your Ubuntu Touch device you can use `clickable shell` to open up a terminal to your device from your computer. This is a replacement for the old `phablet-shell` command.

1.2.4 Ubuntu Touch SDK Api Docs

For more information about the Ubuntu Touch QML or HTML SDK check out the docs over at UBports.

1.2.5 Submitting to the OpenStore

When you are ready to publish your app, head to the OpenStore's submission page.

1.3 Usage

1.3.1 Getting Started

You can get started with using clickable with an existing Ubuntu Touch app. You can use clickable with apps generated from the old Ubuntu Touch SDK IDE or you can start fresh with this app template.

To run the default set of sub-commands, simply run `clickable` in the root directory of your app's code. Clickable with attempt to auto detect the *build template* and other configuration options, if you need more advanced usage options read the *clickable.json format guide*.

Running the default sub-commands will:

1. Kill the running app on the phone
2. Clean the build directory (by default the build directory is `./build/`)

3. Compile the app (if needed)
4. Build the click package (can be found in the build directory)
5. Install the app on your phone (By default this uses adb, see below if you want to use ssh)
6. Launch the app on your phone

1.3.2 Connecting to a device over ssh

By default the device is connected to via adb. If you want to access a device over ssh you need to either specify the device IP address on the command line (ex: `clickable logs --ip 192.168.1.10`) or you

1.3.3 Multiple connected devices

By default clickable assumes that there is only one device connected to your computer via adb. If you have multiple devices attached to your computer you can specify which device to install/launch/etc on by using the flag `--device-serial-number` or `-s` for short. You can get the serial number by running `adb devices`.

1.4 Commands

From the root directory of your project you have the following sub-commands available:

You can combine the commands together like `clickable build click_build install launch`

1.4.1 `clickable`

Runs the default sub-commands specified in the "default" config

1.4.2 `clickable --desktop`

Compile and run the app onthe desktop.

1.4.3 `clickable init`

Generate a new app from a list of *app template options*.

`clickable init -n <app template name>`

Generate a new app from an *app template* by name.

1.4.4 `clickable shell`

Opens a shell on the device via ssh. This is similar to the `phablet-shell` command.

1.4.5 `clickable kill`

Kills a running process (specified by the config). Using this you can relaunch your app.

1.4.6 clickable clean

Cleans out the build dir.

1.4.7 clickable build

Builds the project using the specified template, build dir, and build commands.

1.4.8 clickable click-build

Takes the built files and compiles them into a click package (you can find it in the build dir).

```
clickable click-build --output=/path/to/some/diretory
```

Takes the built files and compiles them into a click package, outputting the compiled click to the directory specified by `--output`.

1.4.9 clickable install

Takes a built click package and installs it on a device.

```
clickable install --click ./path/to/click/app.click
```

Installs the specified click package on the device

1.4.10 clickable launch

Launches the app on a device.

```
clickable launch --app <app name>
```

Launches the specified app on a device.

1.4.11 clickable logs

Follow the apps log file on the device.

1.4.12 clickable run "some command"

Runs an arbitrary command in the clickable container.

1.4.13 clickable setup-docker

Configure docker for use with clickable.

1.4.14 clickable display-on

Turns on the device's display and keeps it on until you hit CTRL+C.

1.4.15 clickable no-lock

Turns off the device's display timeout.

1.4.16 clickable devices

Lists the serial numbers and model names for attached devices. Useful when multiple devices are attached and you need to know what to use for the `-s` argument.

1.4.17 clickable <custom command>

Runs a custom command specified in the "scripts" config

`clickable <custom command> --device`

Runs a custom command specified in the "scripts" config on the device.

`clickable <any command> --container-mode`

Runs all builds commands on the current machine and not in a container. This is useful from running clickable from within a container.

1.5 clickable.json Format

Example:

```
{
    "template": "cmake",
    "scripts": {
        "test": "make test"
    },
    "dependencies": [
        "libpoppler-qt5-dev"
    ]
}
```

1.5.1 package

The full package name (appname.developer). This is optional and will be read from manifest.json if left blank.

1.5.2 app

The app name (appname.developer). This is optional and will be read from manifest.json if left blank.

1.5.3 sdk

Optional, Defaults to *ubuntu-sdk-15.04*

1.5.4 arch

Optional, the default is armhf. You may also specify this as a cli arg (ex: `--arch="armhf"`)

1.5.5 prebuild

Optional, a custom command to run before a build.

1.5.6 template

Optional, see *build template* for the full list of options. If left blank the template will be auto detected.

1.5.7 premake

Optional, a custom command to execute before make is run.

1.5.8 build

Optional, a custom command to run instead of the default build. If using the *custom* template this is required.

1.5.9 postbuild

Optional, a custom command to execute after build and before click build.

1.5.10 launch

Optional, a custom command to launch the app.

1.5.11 ssh

Optional, the IP of the device you wish to install and launch the app on.

1.5.12 dir

Optional, a custom build directory. Defaults to `./build/`

1.5.13 kill

Optional, a custom process name to kill (useful for killing the running app, then relaunching it). If left blank the process name will be assumed.

1.5.14 scripts

Optional, an object detailing custom commands to run. For example:

```
{
    "test": "make test",
    "echo": "echo Hello World"
}
```

To run the command on the device use the `--device` argument (ex: `clickable test --device`).

1.5.15 chroot

Optional, whether or not to use a chroot to build the app. Default is to use docker to build the app. Chroots are deprecated and their support will be removed in a future version of clickable.

1.5.16 lxd

Optional, whether or not to use a lxd container to build the app. Default is to use docker to build the app. LXD is deprecated and its support will be removed in a future version of clickable.

1.5.17 default

Optional, a list of space separated sub-commands to run when no sub-commands are specified. Defaults to `kill clean build click-build install launch`.

1.5.18 dependencies

Optional, a list of dependencies that will be installed in the build container. These will be assumed to be *dependencie:arch* unless *specificDependencies* is set to *true*.

1.5.19 ignore

Optional, a list of files to ignore when building a *pure* template Example: ‘‘ “ignore”:[

“.clickable”, “.git”, “.gitignore”, “.gitmodules”

]

“

1.5.20 make_jobs

Optional, the number of jobs to use when running make, equivalent to make’s *-j* option. If left blank this defaults to the number of cpus your computer has.

1.5.21 gopath

Optional, the gopath on the host machine. If left blank, the `GOPATH` env var will be used.

1.6 App Templates

1.6.1 Pure QML App (built using CMake)

An app template that is setup for a purely QML app. It includes a CMake setup to allow for easy translations.

Check it out on GitHub.

1.6.2 C++/QML App (built using CMake)

An app template that is setup for a QML app with a C++ plugin. It includes a CMake setup to allow for easy translation.

Check it out on GitHub.

1.6.3 Python/QML App (built using CMake)

An app template that is setup for an app using Python with QML. It includes a CMake setup to allow for easy translation.

Check it out on GitHub.

1.6.4 HTML App

An app template that is setup for a local HTML app.

Check it out on GitHub.

1.6.5 Simple Webapp

An app template that is setup as a thin wrapper around an existing website.

Check it out on GitHub.

1.6.6 Go/QML App

An app template that is setup for a QML app with a Go backend.

Check it out on GitHub.

1.7 Build Templates

1.7.1 pure-qml-qmake

A purely qml qmake project.

1.7.2 qmake

A project that builds using qmake (has more than just QML).

1.7.3 pure-qml-cmake

A purely qml cmake project

1.7.4 cmake

A project that builds using cmake (has more than just QML)

1.7.5 custom

A custom build command will be used.

1.7.6 cordova

A project that builds using cordova

1.7.7 pure

A project that does not need to be compiled. All files in the project root will be copied into the click.

1.7.8 python

A project that uses python and does not need to be compiled.

1.7.9 go

A project that uses go version 1.6.

CHAPTER 2

Install Via PPA (Ubuntu)

- Add the PPA to your system: `sudo add-apt-repository ppa:bhdouglass/clickable`
- Update your package list: `sudo apt-get update`
- Install clickable: `sudo apt-get install clickable`
- Configure docker for clickable: `clickable setup-docker`

CHAPTER 3

Install Via AUR (Arch Linux)

- Using your favorite AUR helper, install the clickable package
- Example: `pacaur -S clickable`

CHAPTER 4

Install Via Git

- Install Docker
- Install Cookiecutter
- Clone this repo: `git clone https://github.com/bhdouglass/clickable.git`
- Add clickable to your PATH: `echo "export PATH=\$PATH:\$HOME/clickable" >> ~/.bashrc`
- Read the new `.bashrc` file: `source ~/.bashrc`
- Configure docker for clickable: `clickable setup-docker`

CHAPTER 5

Getting Started

Read the getting started guide to get started developing with clickable.

CHAPTER 6

Code Editor Integrations

Use clickable with the Atom Editor by installing atom-build-clickable made by Stefano.

CHAPTER 7

Issues and Feature Requests

If you run into any problems using clickable or have any feature requests you can find clickable on GitHub.