Clickable Documentation

Release 5.9.1

Brian Douglass

Mar 11, 2019

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 GitLab. Clickable is developed by Brian Douglass and Jonatan Hatakeyama Zeidler with a huge thank you to all the contributors.

CHAPTER 1

Using Clickable

1.1 Install

1.1.1 Install Via Pip (Recommended)

- Install docker, adb, and pip3
- Run (may need sudo): `pip3 install git+https://gitlab.com/clickable/clickable.git`

1.1.2 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`

1.1.3 Install Via AUR (Arch Linux)

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

1.2 Getting Started

- Run `clickable create` 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 Run Automatic Review

Apps submitted to the OpenStore will undergo automatic review, to test your app before submitting it, run `clickable review` after you've compiled a click. This runs the `click-review` command against your click within the clickable container (no need to install it on your computer).

1.2.6 Publishing to the OpenStore

If this is your first time publishing to the OpenStore, you need to signup for an account. You can signup with your GitHub, GitLab, or Ubuntu account.

If your app is new to the OpenStore you must first create your app by entering the name from your manifest.json and the app's title on the OpenStore's submission page.

If your app already exists you can use the `clickable publish` command to upload your compiled click file to the OpenStore. In order to publish to the OpenStore you need to grab your api key from the OpenStore. After you have your api key you need to let Clickable know about it. You can either pass it as an argument every time: `clickable publish --apikey XYZ` Or you can set it as an environment variable: `export OPENSTORE_API_KEY=XYZ` (you can add this to your `~/.bashrc` to keep it set).

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 by running `clickable create`.

To run the default set of sub-commands, simply run `clickable` in the root directory of your app's code. Clickable will attempt to auto detect the *build template* and other configuration options.

Running the default sub-commands will:

1. Clean the build directory (by default the build directory is `./build/`)
2. Compile the app
3. Build the click package (can be found in the build directory)
4. Install the app on your phone (By default this uses adb, see below if you want to use ssh)
5. Kill the running app on the phone
6. Launch the app on your phone

1.3.2 Configuration

If you need more advanced usage options, you may specify a configuration file in the *clickable.json format* with `--config`. If not specified, clickable will look for an optional configuration file called `clickable.json` in the current directory.

1.3.3 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 or hostname on the command line (ex: `clickable logs --ssh 192.168.1.10`) or you can use the `CLICKABLE_SSH` env var.

1.3.4 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 `--serial-number` or `-s` for short. You can get the serial number by running `clickable 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. A dirty build without cleaning the build dir can be achieved by running `clickable --dirty`.

1.4.2 `clickable desktop`

Compile and run the app on the desktop.

Note: ArchLinux user might need to run `xhost +local:clickable` before using desktop mode.

1.4.3 clickable create

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

```
clickable create <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 clean-libs

Cleans out all library build dirs.

1.4.6 clickable build-libs

Builds the dependency libraries specified in the clickable.json.

1.4.7 clickable clean

Cleans out the build dir.

1.4.8 clickable build

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

1.4.9 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`.

1.4.10 clickable review

Takes the built click package and runs click-review against it. This allows you to review your click without installing click-review on your computer.

1.4.11 clickable install

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

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

Installs the specified click package on the device

1.4.12 clickable launch

Launches the app on a device.

`clickable launch <app name>`

Launches the specified app on a device.

1.4.13 clickable logs

Follow the apps log file on the device.

1.4.14 clickable log

Dumps the apps log file on the device.

1.4.15 clickable publish

Publish your click app to the OpenStore. Check the *Getting started doc* for more info.

`clickable publish "changelog message"`

Publish your click app to the OpenStore with a message to add to the changelog.

1.4.16 clickable run "some command"

Runs an arbitrary command in the clickable container.

1.4.17 clickable update

Update the docker container for use with clickable.

1.4.18 clickable no-lock

Turns off the device's display timeout.

1.4.19 clickable writable-image

Make your Ubuntu Touch device's rootfs writable. This replaces to old `phablet-config writable-image` command.

1.4.20 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.

1.4.21 clickable <custom command>

Runs a custom command specified in the "scripts" config

1.4.22 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.

1.4.23 clickable desktop --nvidia

Use clickable's desktop mode with proprietary Nvidia drivers. This requires nvidia-docker to be installed and setup. Please note, only version 1 of nvidia-docker is supported at this time (version 2 does not support OpenGL).

1.5 clickable.json Format

Example:

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

1.5.1 clickable_minimum_required

Optional, a minimum Clickable version number required to build the project. Ex: `"4"` or `"5.4.0"`

1.5.2 arch

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

1.5.3 template

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

1.5.4 prebuild

Optional, a custom command to run before a build.

1.5.5 build

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

1.5.6 postbuild

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

1.5.7 postmake

Optional, a custom command to execute after make (during build).

1.5.8 launch

Optional, a custom command to launch the app.

1.5.9 dir

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

1.5.10 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.

1.5.11 scripts

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

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

1.5.12 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.

1.5.13 default

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

Can be specified as a string or a list of strings.

1.5.14 dependencies_build

Optional, a list of dependencies that will be installed in the build container.

Can be specified as a string or a list of strings.

1.5.15 dependencies_target

Optional, a list of dependencies that will be installed in the build container. These will be assumed to be *dependency:arch*, unless an architecture specifier is already appended. In desktop mode `dependencies_target` is handled just like `dependencies_build`.

Can be specified as a string or a list of strings.

1.5.16 dependencies

This parameter is deprecated and will be removed in a future version. Use `dependencies_build` or `dependencies_target` instead!

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*.

Can be specified as a string or a list of strings.

1.5.17 dependencies_ppa

Optional, a list of PPAs, that will be enabled in the build container. This is only supported for doker mode. Ex:

```
"dependencies_ppa": [
    "ppa:bhdouglass/clickable"
]
```

Can be specified as a string or a list of strings.

1.5.18 docker_image

Optional, the name of a docker image to use. When building a custom docker image it's recommended to use one of the Clickable images as a base. You can find them on Docker Hub.

1.5.19 ignore

Optional, a list of files to ignore when building a *pure* template Example:

```
"ignore": [
    ".clickable",
    ".git",
    ".gitignore",
    ".gitmodules"
]
```

Can be specified as a string or a list of strings.

1.5.20 gopath

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

1.5.21 cargo_home

Optional, the Cargo home path (usually *~/.cargo*) on the host machine. If left blank, the `CARGO_HOME` env var will be used.

1.5.22 build_args

Optional, arguments to pass to qmake or cmake. When using *–debug-build*, `CONFIG+=debug` is additionally appended for qmake and `-DCMAKE_BUILD_TYPE=Debug` for cmake and cordova builds. Ex: `CONFIG+=ubuntu`

Can be specified as a string or a list of strings.

1.5.23 make_args

Optional, arguments to pass to make, e.g. a target name. To avoid configuration conflicts, the number of make jobs should not be specified here, but using `make_jobs` instead, so it can be overriden by the according environment variable.

Can be specified as a string or a list of strings.

1.5.24 make_jobs

Optional, the number of jobs to use when running make, equivalent to make's *-j* option. If left blank this defaults to *-j*, allowing make to execute many recipes simultaneously.

1.5.25 dirty

Optional, whether or not do a dirty build, avoiding to clean the build directory before building. You may also specify this as a cli arg (`--dirty`). The default is `false`.

1.5.26 libraries

Optional, libraries to be build in the docker container by calling `clickable build-libs`. It's a dictionary of dictionaries basically looking like the clickable.json itself. Example:

```
"libraries": {
    "opencv": {
        "template": "cmake",
        "make_jobs": "4",
        "build_args": [
            "-DCMAKE_BUILD_TYPE=Release",
            "-DBUILD_LIST=core,imgproc,highgui,imgcodecs",
            "-DBUILD_SHARED_LIBS=OFF"
        ]
        "prebuild": "git submodule update --init --recursive"
    }
}
```

The keywords `prebuild`, `build`, `postbuild`, `postmake`, `make_jobs`, `make_args`, `build_args`, `docker_image`, `dependencies_build`, `dependencies_target` and `dependencies_ppa`, can be used for a library the same way as described above for the app. The libraries are compiled for the same architecture as specified for the app itself.

A single library can be build by specifying its name as `clickable build-libs lib1` to build the library with name `lib1`.

template

Required, but only `cmake`, `qmake` and `custom` are allowed.

src_dir

Optional, library source directory. Must be relative to the project root. It defaults to `libs/<name>`

dir

Optional, library build directory. Must be relative to the project root. It defaults to `build/<name>`. The architecture triplet is appended, so that builds for different architectures can exist in parallel (`arm-linux-gnueabihf` for `armhf` and `x86_64-linux-gnu` for `amd64`).

Consider defining a custom build directory for the app itself (Ex: `build/app`). Otherwise cleaning the app would clean the library, too.

1.6 Environment Variables

Environment variables will override values in the clickable.json and can be overridden by command line arguments.

1.6.1 CLICKABLE_ARCH

Overrides the clickable.json's *arch*.

1.6.2 CLICKABLE_TEMPLATE

Overrides the clickable.json's *template*.

1.6.3 CLICKABLE_DIR

Overrides the clickable.json's *dir*.

1.6.4 CLICKABLE_LXD

Overrides the clickable.json's *lxd*.

1.6.5 CLICKABLE_DEFAULT

Overrides the clickable.json's *default*.

1.6.6 CLICKABLE_MAKE_JOBS

Overrides the clickable.json's *make_jobs*.

1.6.7 GOPATH

Overrides the clickable.json's *gopath*.

1.6.8 CARGO_HOME

Overrides the clickable.json's *cargo_home*.

1.6.9 CLICKABLE_DOCKER_IMAGE

Overrides the clickable.json's *docker_image*.

1.6.10 CLICKABLE_BUILD_ARGS

Overrides the clickable.json's *build_args*.

1.6.11 CLICKABLE_MAKE_ARGS

Overrides the clickable.json's *make_args*.

1.6.12 OPENSTORE_API_KEY

Your api key for *publishing to the OpenStore*.

1.6.13 CLICKABLE_CONTAINER_MODE

Same as *–container-mode*.

1.6.14 CLICKABLE_SERIAL_NUMBER

Same as *–serial-number*.

1.6.15 CLICKABLE_SSH

Same as *–ssh*.

1.6.16 CLICKABLE_OUTPUT

Override the output directory for the resulting click file

1.6.17 CLICKABLE_NVIDIA

Same as *–nvidia*.

1.6.18 CLICKABLE_VIVID

Same as `--vivid`

1.6.19 CLICKABLE_DIRTY

Overrides the clickable.json's *dirty*.

1.6.20 CLICKABLE_DEBUG_BUILD

Same as `--debug-build`

1.7 App Templates

1.7.1 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 GitLab.

1.7.2 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 GitLab.

1.7.3 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 GitLab.

1.7.4 HTML App

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

Check it out on GitLab.

1.7.5 Simple Webapp

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

Check it out on GitLab.

1.7.6 Go/QML App

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

Check it out on GitLab.

1.7.7 C++/QML App (built using CMake with a main.cpp)

An app template that is setup for a QML app with a C++ plugin. It includes a CMake setup to allow for easy translation. It also includes a main.cpp to build a custom binary rather than relying on qmlscene.

Check it out on GitLab.

1.7.8 Rust/QML App

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

Check it out on GitLab.

1.8 Build Templates

1.8.1 pure-qml-qmake

A purely qml qmake project.

1.8.2 qmake

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

1.8.3 pure-qml-cmake

A purely qml cmake project

1.8.4 cmake

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

1.8.5 custom

A custom build command will be used.

1.8.6 cordova

A project that builds using cordova

1.8.7 pure

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

1.8.8 python

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

1.8.9 go

A project that uses go version 1.6.

1.8.10 rust

A project that uses rust. Debug builds can be enabled by specifying `--debug-build`.

1.9 Continuous Integration

1.9.1 Clickable CI Docker Images

Two docker images are available for easily using Clickable with a continuous integration setup. They can be found on Docker hub: `clickable/ci-15.04-armhf` and `clickable/ci-16.04-armhf` for vivid/15.04 and xenial/16.04 respectively. The images come with Clickable pre installed and already setup in container mode.

1.9.2 GitLab Example

With GitLab's CI solution it is trivial to add Clickable building and publish to your click apps. For an example of this in action, check out the Clickable GitLab example app.

To implement this in your own repository, create a `.gitlab-ci.yml`:

```
build_vivid:
  image: clickable/ci-15.04-armhf
  script:
  - clickable --vivid clean build click-build review publish

build_xenial:
  image: clickable/ci-16.04-armhf
  script:
  - clickable clean build click-build review publish
```

After that's setup, the last step is to add the environment variable `OPENSTORE_API_KEY` to your GitLab project (You can find your OpenStore api key when you log into the OpenStore).

1.10 Changelog

1.10.1 Changes in v5.9.1

- Fixed missing schema file

1.10.2 Changes in v5.9.0

- New schema validation for clickable.json
- Publish to the OpenStore with a changelog message

1.10.3 Changes in v5.8.1

- Fixed a bug in `make_args`

1.10.4 Changes in v5.8.0

- New configuration option for automatically including ppas in the build environment: *dependencies_ppa*.
- Changed *libraries* format from a list to a dictionary (the old format is still supported for now)
- The default `cargo_home` is now set to `~/.cargo`

1.10.5 Changes in v5.7.0

- Introduced two new dependency options to separate *build* <*clickable-json-dependencies_build*> and *target* <*clickable-json-dependencies_target*> dependencies

1.10.6 Changes in v5.6.1

- Fixed build lib
- Made cordova build respect the –debug-build argument

1.10.7 Changes in v5.6.0

- Fixed Cordova build
- Added `--debug-build` support for QMake and CMake templates

1.10.8 Changes in v5.5.1

- New `--config` argument to specify a different path to the clickable.json file
- New configuration called `clickable_minimum_required` to specify a minimum version of Clickable
- New `make_args` configuration for passing arguments to make

1.10.9 Changes in v5.5.0

- build-libs now only uses the same arch as specified in clickable.json or in the cli args
- Added the option to build/clean only one lib
- Added support for GOPATH being a list of paths
- Exits with an error with an invalid command

1.10.10 Changes in v5.4.0

- Added support for Rust apps
- Added support for distros using SELinux

1.10.11 Changes in v5.3.3

- More fixes for building libraries
- Set the home directory to /home/phablet in desktop mode

1.10.12 Changes in v5.3.2

- Fixed issue building libraries
- Create arch specific directories in .clickable
- Fixed –dirty breaking when using a custom default set of commands

1.10.13 Changes in v5.3.1

- Fixed dependencies in library prebuild

1.10.14 Changes in v5.3.0

- *Added options for compiling libraries*

1.10.15 Changes in v5.2.0

- Fixed bug in build template auto detection
- Added new dirty build option

1.10.16 Changes in v5.1.1

- Fixed bug in “shell” command

1.10.17 Changes in v5.1.0

- Added app template for QML/C++ with a main.cpp

1.10.18 Changes in v5.0.2

- Fixed publish command not exiting with an error code when there is an error

1.10.19 Changes in v5.0.1

- Fixed typo in cache path
- Improved Cordova support

1.10.20 Changes in v5.0.0

- **New features**
  - Xenial by default (use `--vivid` to compile for 15.04)
  - Major code refactor
  - **More environment variables**
    - `CLICKABLE_ARCH` - Overrides the clickable.json's `arch`
    - `CLICKABLE_TEMPLATE` - Overrides the clickable.json's `template`
    - `CLICKABLE_DIR` - Overrides the clickable.json's `dir`
    - `CLICKABLE_LXD` - Overrides the clickable.json's `lxd`
    - `CLICKABLE_DEFAULT` - Overrides the clickable.json's `default`
    - `CLICKABLE_MAKE_JOBS` - Overrides the clickable.json's `make_jobs`
    - `GOPATH` - Overrides the clickable.json's `gopath`
    - `CLICKABLE_DOCKER_IMAGE` - Overrides the clickable.json's `docker_image`
    - `CLICKABLE_BUILD_ARGS` - Overrides the clickable.json's `build_args`
    - `OPENSTORE_API_KEY` - Your api key for publishing to the OpenStore
    - `CLICKABLE_CONTAINER_MODE` - Same as `--container-mode`
    - `CLICKABLE_SERIAL_NUMBER` - Same as `--serial-number`
    - `CLICKABLE_SSH` - Same as `--ssh`
    - `CLICKABLE_OUTPUT` - Override the output directory for the resulting click file
    - `CLICKABLE_NVIDIA` - Same as `--nvidia`
    - `CLICKABLE_VIVID` - Same as `--vivid`
- **Removed**
  - Chroot support has been removed, docker containers are recommended going forward
- **clickable.json**
  - **Removed**
    - `package` - automatically grabbed from the manifest.json
    - `app` - automatically grabbed from the manifest.json
    - `sdk` - Replaced by docker_image and the `--vivid` argument

        * `premake` - Use `prebuild`
        * `ssh` - Use the `--ssh` argument
- **Commands**
    - **New**
        * `log` - Dumps the full log file from the app
        * `desktop` - Replaces `--desktop` to run the app in desktop mode
    - **Changed**
        * `init` - Changed to `create` (`init` will still work)
        * `update-docker` - Changed to `update`
    - **Removed**
        * `kill` - Changed to be part of the `launch` command
        * `setup-docker` - Automatically detected and run when using docker
        * `display-on` - Not very useful
- **Command line arguments**
    - **New**
        * `--vivid` - Compile the app for 15.04
        * `--docker-image` - Compile the app using a specific docker image
    - **Changed**
        * `--serial-number` - Replaces `--device-serial-number`
        * `--ssh` - Replaces `--ip`
    - **Removed**
        * `--desktop` - Use the new `desktop` command
        * `--xenial` - Xenial is now the default
        * `--sdk` - Use `--vivid` or `--docker-image`
        * `--device` - Use `shell`
        * `--template` - Use the `CLICKABLE_TEMPLATE` env var
        * `--click` - Specify the path to the click after the `install` command: `clickable install /path/to/click`
        * `--app` - Specify the app name after the `launch` command: `clickable launch app.name`
        * `--name` - Specify the app template after the `create` command: `clickable create pure-qml-cmake`

CHAPTER 2

Install Via Pip (Recommended)

- Install docker, adb, and pip3
- Run (may need sudo): `pip3 install git+https://gitlab.com/clickable/clickable.git`

CHAPTER 3

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`

CHAPTER 4

Install Via AUR (Arch Linux)

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

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 GitLab.