Survey of 3rd Party Yocto Tools

Joshua Watt
Yocto Project Summit
About Me

- Worked at Garmin since 2009
- Using OpenEmbedded & Yocto Project since 2016
- Member of the OpenEmbedded Technical Steering Committee (TSC)
- Joshua.Watt@garmin.com
- JPEWhacker@gmail.com
- IRC (OFTC or libera): JPEW
- Twitter: @JPEW_dev
- LinkedIn: joshua-watt-dev
Outline

● Layer Source Code Management
  ○ Combo-layer
  ○ Git submodules
  ○ Git subtrees
  ○ Repo

● Build Environment
  ○ Buildtools-tarball & uninative
  ○ CROPS
  ○ Pyrex

● Configuration Management
  ○ TEMPLATECONF
  ○ Whisk
  ○ Kas

● Conclusion
Why Cover This?

- Education on available options
- Broadening of perspective
  - Important when discussing layer setup options to be added to the project

Specifically, this presentation is not intended to say one method is "better" than another
Nothing up my sleeves!

This is intended to be unbiased; however:

- I wrote Pyrex and Whisk
- I use Git submodules
Layer Source Code Management
Layer Source Code Management

- How are other layers pulled into our builds?
- How are new changes pulled from upstream?
- How are changes pushed back to upstream?
combo-layer
combo-layer

- Python script in OE-core
- Pulls multiple repositories together into a single git tree
- Used to manage Poky
Git submodules
Git submodules

- Integrated into Git
- Sub-repositories just another git repo in a subdirectory of the parent
- Parent repo references children by SHA-1
- Sub-repository has its own distinct git tree from parent
  - Although it lives in the parents .git directory
- Supports recursion (sub repositories can have sub repositories)
- Extra commands required to update, etc.
Git subtrees
Git subtrees

- [https://www.atlassian.com/git/tutorials/git-subtree](https://www.atlassian.com/git/tutorials/git-subtree)
- Pull subprojects into a single repository as additional remotes
- `git subtree` merge subproject in with a combination merge & rename commit
- Tool can also split commits from parent remote back to subprojects for upstreaming
- Single git repository is easy for users to consume (no tool or extra commands required)
Repo
Repo

- [https://gerrit.googlesource.com/git-repo/](https://gerrit.googlesource.com/git-repo/)
- Maintained by Google
- Used extensively in Android
- Git repositories in a project are described in a separately fetched XML manifest (usually kept in a separate git repo)
- Some code review integrations, usually for use with Gerrit
- Can track either branch heads or commit SHA-1
Build Environment
Build Environment

- Managing build dependencies
- Building across heterogeneous environment (e.g. different developers using different distros)
- Continuous Integration builds

In all cases, consistent results (reproducible!) are desired
Buildtools-tarball & uninnative
Uninative

- [link](https://docs.yoctoproject.org/singleindex.html#uninative-bbclass)  
- Part of OE core  
- Replaces host C library for consistent build behavior across hosts; in particular so sstate can be shared  
- Enabled by default
Buildtools-tarball

- Provides hosts tools required to build (e.g. GCC, binutils etc.)
- Built as an SDK out of OE-core
- Simple to enable and use
- Possible to build and publish your own if you need custom tools
CROPS

- https://github.com/crops
- Docker containers for building Yocto projects
- Several distros images are maintained: Debian, Ubuntu Fedora, OpenSuse
- You will usually want to use poky-container as it can support creating a user in the container that matches your user
- Simple setup and usage

```bash
docker run --rm -it -v /home/myuser/mystuff:/workdir crops/poky --workdir=/workdir
```
Pyrex

- [https://github.com/garmin/pyrex](https://github.com/garmin/pyrex)
- Maintained by Garmin (yours truly)
- "Transparencly" redirects commands to run in a Ubuntu container
  - E.g. users still invoke `bitbake` et al.
- Multiple LTS Ubuntu version supported for running older Yocto releases on newer hosts
- Comprehensive configuration options
Configuration Management
Configuration Management

- Need to build \( N \) different configurations (e.g. different products) out of the same codebase
- Ensure that users can easily and reproducibly select a specific configuration to build
  - Preferably, the configurations live with the code
- Local configuration changes for experimentation?
TEMPLATECONF

- Part of OE-core `oe-init-build-env` script
- Setting the environment variable `TEMPLATECONF` to a directory will cause OE-core to populate `conf/bblayers.conf` and `conf/local.conf` files from templates
  - Only if they don't already exist

```
$ ls conf/templates/foo
bblayers.conf.sample local.conf.sample
$ export TEMPLATECONF="$(pwd)/conf/templates/foo/"
$ . oe-init-build-env
```
Whisk
Whisk

- [https://github.com/garmin/whisk](https://github.com/garmin/whisk)
- Maintained by Garmin (yours truly)
- YAML configuration
- Optional Pyrex integration for build environment setup
- 3 axes of configuration - "product" "mode" and "site"
  - "Product" - The thing you want to build
  - "Mode" - How you want to build it (e.g. "Release", "Debug")
  - "Site" - Where you are building it from (e.g. "Jenkins", "Olathe")
- Makes heavy use of Multiconfig (e.g. everything is a multiconfig)
- Per-product BBLAYERS
- Limited layer customization support

  $ . init-build-env --product=eagle --mode=release --site=olathe
  $ bitbake all-targets

---

Olathe (oh-LAY-tha) is a suburb of Kansas City, Kansas, U.S.A where Garmin World Headquarters is located.
Kas
kas

- [https://github.com/siemens/kas](https://github.com/siemens/kas)
- Maintained by Siemens
- One tool that does everything (layer management, build environment, & configuration management) with a single command
- YAML based config files
- Project YAML files may be built up hierarchically, and even pulled from other layers for complex configurations

$ kas build project.yaml
Conclusion
Questions?