Creating and Maintaining a Yocto Project in the Real World

Ming, Silicon Blade S.A.R.L.

Yocto Project Summit, 2024.12

Iain Menzies-Runciman

Known as 'Ming'

About Me

Silicon Blade Consultants

- Linux and Unix Consultant for over 30 years
- Embedded Linux Consultant and Trainer for over 15 years
- Spend my days building applications and systems for clients with the Yocto Project

www.siliconbladeconsultants.com

Typical Responsibilities

- **Support Multiple Boards and Configurations**
  - Looking after Board Bring-up and Kernel Configurations
- **Support Multiple Developers who may not know Linux well**
  - Likely not to know The Yocto Project!
- **Maintain and Update the System**
- **Update Devices in the Field**

A Yocto Project Developer/Maintainer

Needs to be:

- System Administrator
- Network Administrator
- Performance Specialist
- Board Bring-up Specialist
- Kernel Specialist

Creating Your Project

A First Project

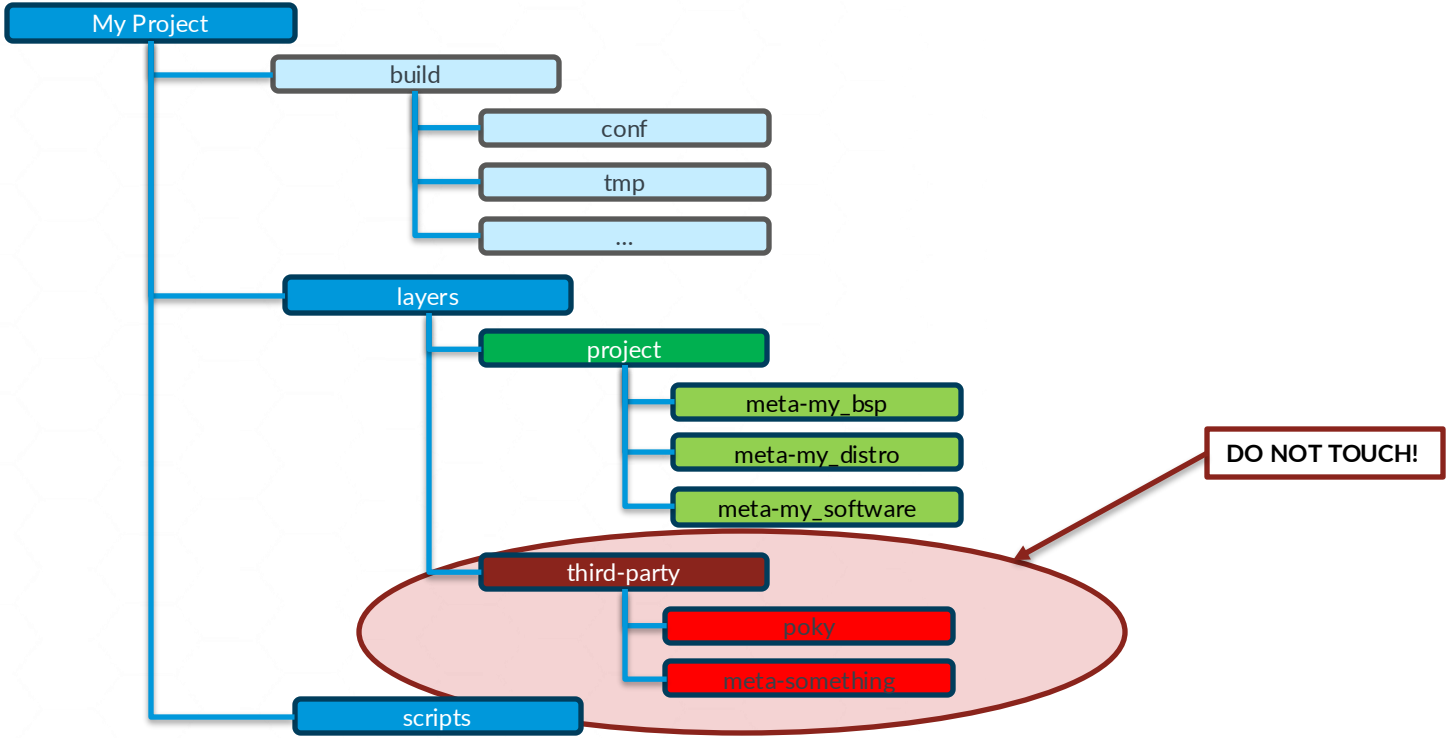
- **Your very first project should not be a Yocto Project!**
- **Create a Linux System From Scratch**
  - Learn how to put together a system by hand the hard way
  - Makes it easier to understand how it all slots together

https://www.linuxfromscratch.org/lfs/view/stable/

Guidelines

- **Keep a clean project layout**
  - This makes it easier to update and share
- **Do not keep your layers in your top-level directory**
  - It can get very messy very quickly
- **Separate the layers you download from the layers you create**
  - Makes it easier to maintain

My Project Layout



Project Layers

Suggested Minimum

meta-my_bsp

- **Anything to do with board bring up**
  - Machine Definitions
  - Kernel, u-boot, etc

meta-my_distro

- **Anything to do with Linux User Space**
  - Distro Configurations
    - Remember Poky is a reference not an end
  - Anything adding or modifying ‘standard’ Linux stuff,
    - e.g. Users, Configurations, etc
  - Mainly lots of bbappend’s

meta-my_software

- **Anything created in-house**
  - Recipes to install your own software

Other Layers

- **The whole point of having layers is for flexibility**
- **Create layers for anything that needs to be shared across multiple projects**
- **I prefer many small layers over one large layer**
  - Speeds up start-up time if you do not have to parse lots irrelevant recipes
- **Can still keep all the layers in one repository**
  - Allows the user to select which layers are actually needed

Working with the Project

Setting up and Running

Setting The Environment

- **Based on the three virtues of a great programmer***
  - I am too lazy to type the full path to oe-init-build-env

```
#!/bin/bash --init-file

# Ensure the shared directories exist for other projects
mkdir ~/yocto/shared > /dev/null 2>&1

source ./layers/third-party/poky/oe-init-build-env ./build
```

*By Larry Wall
1. Laziness
2. Impatience
3. Hubris

https://thethreevirtues.com

Using Templates

- **I always add the build directory to my .gitignore**
- **This means that the local.conf and bblayers.conf are not saved in the repository**
- **Fortunately, these do not change very often**
  - Can be saved in the project as a template
    - Useful for sharing (see next section)

Sharing Your Project

Working in a Team

Sharing Your Project

- **I often work with multiple remote developers**
- **Everyone wants their own local copy of the project**
- **No one reads the setup instructions**
- **Options:**
  A. Refuse to talk to them until they read them
  B. Design around them

local.conf

- **Given I have multiple projects on the go, I like to keep shared directories outside of the project**
  - ${HOME}/yocto/share/…
    - DL_DIR
    - SSTATE_DIR
- **This is also a good place to keep secrets**
  - But do not include the secrets in the template!

Secrets

In local.conf

```
SECRET_KEY1 = "ABC123"
```

In Source File

```
SECRET_KEY = "###SECRET_KEY1###"
```

In Recipe

```
do_install() {
  install -m 0644 ${WORKDIR}/foo.cfg ${D}${sysconfdir}/foo.cfg
  sed 's,###SECRET_KEY1###,${SECRET_KEY1},' -i ${D}${sysconfdir}/foo.cfg
…
}
```

bblayers.conf

- **I always keep paths relative**
  - Allows the project to be shared
    - So, works as a Template
  - Allows for the project to be moved
    - To a different location
    - To a different machine

Sample Snippet

```
YOCTOROOT = "${TOPDIR}/../layers"

BBLAYERS = " \
  ${YOCTOROOT}/third-party/poky/meta \
  ${YOCTOROOT}/third-party/poky/meta-poky \
  ${YOCTOROOT}/third-party/poky/meta-yocto-bsp \
  ${YOCTOROOT}/project/meta-my-bsp \
  ${YOCTOROOT}/project/meta-my-distro \
  ${YOCTOROOT}/project/meta-my-software \
"
```

Template Directory

- **Having created your local configuration files**
  - Create a template directory in your project
  - Copy the configuration files into it
    - Removing any secrets!
  - Set the `TEMPLATECONF` environment variable in the start-up script
    - `TEMPLATECONF=../path/to/templates`

Production and Development

The Same but Different

Working with Production and Development

- I normally ship production systems with a Read-Only Root Filesystem
- Good for Security & Delta Updates in the field
- Not so good for fiddling during development

Development Image Requirements

- **Has all of Production**
- **Is Read-Write**
- **Has extra tools for**
  - Debugging
  - Performance monitoring
  - Testing
  - Etc

Solution: Include and Modify

```
require images/my-production-image.bb

# Remove the requirement for delta updates to have read-only rootfs
IMAGE_FEATURES:remove = "read-only-rootfs"

DEV_EXTRA_TOOLS = " \
  vim \
  iperf3 \
  opkg \
"

IMAGE_INSTALL += " \
  ${DEV_EXTRA_TOOLS} \
"
```

Questions?

Thank you for Staying Awake!

Email: ming@siliconbladeconsultants.com

Web: www.siliconbladeconsultants.com





yocto PROJECT | THE LINUX FOUNDATION