Lessons Learned: Migrating a Production Platform to Yocto

Mitch Gaines – Embue / Farmblox
About Mitch Gaines

• Embedded Systems Engineer at Farmblop & Embue
• Yocto Developer
• Contributor to meta-rauc-community
• Previously in Customer Success

https://www.linkedin.com/in/mitchgaines/
Agenda

- Why should I do this?
  - Debian vs. Yocto Build Processes
  - The Debian Blueprint
- Managing High Configurability
  - Exercising the Layer-based Architecture
  - Integrating Yocto into App Development
Debian vs Yocto Build Process

- Yocto can still build Debian packages (deb, ipk, & rpm)
- Output of a Yocto build is a system image
- Debian steps are disjointed from one another
- These principles can be applied beyond migrating from Debian
Debian Build Process

- Pull Source: Either from git or local on developer machine
- Cross Compile: crosstool-ng, containers, etc.
- Build .deb: Aggregate files and run dpkg-deb
- Upload to Server: Put your .deb onto a package management server and update tooling to reflect change
- Edge Devices Update: Client-side agent (Ansible, Puppet, etc.) updates system

Edge Devices

Update

Client-side agent (Ansible, Puppet, etc.) updates system
Yocto Build Process

Openembedded Architecture Workflow

Upstream Source
Metadata/Inputs
Process steps (tasks)
Output Packages
Build system
Output Image Data

Upstream Project Releases
Local Projects
SCMs (optional)

Source Mirror(s)

User Configuration
Metadata (bh + patches)
Machine (BSP) Configuration
Policy Configuration

Source Fetching
Patch Application
Configuration / Compile / Autoreconf as needed
Output Analysis for package splitting plus package relationships
rpm Generation
cpio Generation
.ipk Generation
QA Tests

Package Feeds

Image Generation
SDK Generation

Images
Application Development SDK
Using the Blueprint

- **do_configure**
  - Managing patch files

- **do_compile**
  - Use your app developers
  - Understand the environment variables

- **do_install**
  - Utilize your Debian install file
Highly Configurable System

- Flexibility to adapt the system to your needs
- Simpler provisioning process

Yocto Layers (meta, meta-yocto-bsp, meta-poky, meta-oe)
BSP Layer (meta-raspberrypi, meta-sunxi, etc.)
Third Party Application Layers
Custom Application Layer
The Migration Framework

1. Make it Boot
   - Determine kernel and u-boot requirements
     - Choose Yocto release to build from
     - Consider the update mechanism and integrate

2. Integrate Your Application Layer
   - Create recipes for each of your applications
     - Use Debian as your blueprint

3. Migrate Application Developers
   - Incorporate them into the migration process
     - Utilize devtool & AUTOREV
Thank You

• Connect with me for questions
• Thank you to everyone in the Yocto community

Additional Resources
• Chris Simmonds Debian or Yocto Project?
  https://youtu.be/iDIlXa8SzUg
• Yocto Project Mega-Manual

https://www.linkedin.com/in/mitchgaines/
gainesmith2@gmail.com