ArchiTech
SILICA Design Tools

Yocto Linux Fundamentals :
targeting i.MX 6SoloX Applications Processors

Chris Young – Embedded S/W Specialist

AVNET SILICA Europe

SILICA™
An Avnet Company

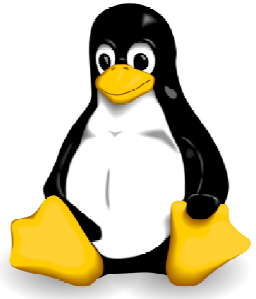
Agenda

- Embedded Linux development
- Introduction to Yocto
- Introducing the ArchiTech Yocto SDK
- Hands-on Lab targeting i.MX 6SoloX SABRE SD

Linux

The Open Source OS

- When consolidation came up in the “mainstream” computer market at the beginning of 1990, people started to look at an alternative to large commercial operating systems:
  - LINUX
- Linux was never a ready-to-use solution
  - People had a problem, searched for a solution, but didn’t find one
  - So people started to develop one themselves, often several people did this in parallel and huge community formed around developing the Linux Kernel
  - Linux Kernel is now one of the most reliable and best performing kernel options available today
  - Today it has shown its effectiveness in the Server, Mobile and Desktop markets



Linux

What is a 'Linux Distribution' ?

- A 'Linux distribution' is a combination of the components required to provide a working Linux environment for a particular platform

- Bootloader
  - Software to initialise memory subsystem and peripherals. Loads and executes Kernel
- Linux kernel port
  - Kernel and board specific device drivers
- Linux 'file system'
  - This contains shared libraries, initialisation routines and all user applications
- Development tools
  - Cross compiler, linker etc

Linux

Embedded Linux Distribution

- PC developers use standard Linux distributions like Ubuntu or RedHat.
- Things are different for embedded
  - Restricted hardware resources
    - Mainstream distributions cannot be installed under 100MB without loosing major functionality
    - Mainstream distributions execute tons of processes which can affect performance and security of an embedded device
  - Different hardware platforms
    - Embedded developers need to customize Linux for running on their specific hardware
  - This creates a need for a custom built distribution tailored for application specific requirements

Embedded Linux

Where do you start ?

- DIY / Roll-Your-Own or modified traditional distro:
  - Long Term Maintenance is difficult
  - Changes to mainline kernel are difficult to track
  - Not embedded friendly
  - Licensing issues
  - No commercial embedded support
- Commercial/Community Embedded Linux:
  - Too many competing systems
  - Incompatible distributions/build systems

- ***Developers spend lots of time porting or making build systems***
- ***Leaves less time/money to develop interesting software features***

Embedded Linux Build systems

Creating your target system image

- **Linux Build systems**
- Group of tools that allow you to automatically build a complete target embedded Linux system (bootloader, kernel and file system)
- They configure, compile and install all needed components :
  - Configuring and building Linux Kernel and Bootloader
  - Files system package architecture patching
  - Configuring, building and deployment of each file system package
  - Resolving file system package dependencies
  - Create file system architecture + configurations files + init scripts…
- Easy tools to develop a system : reproducible build (bug fix, change, experiment)

Embedded Linux Build systems

Embedded Linux Build systems

- LTIB (Linux Target Image Builder): complete and autonomous Linux auto-build system developed by Freescale and based on RPM, menuconfig and Perl
- Buildroot: complete and autonomous Linux auto-build system based on Makefiles and patches for generation of a cross- compiler tool chain as well as the creation of a complete file system.
- Ptxdist : standalone auto-build system based on Kconfig and a set of patches and Makefiles developed by Pengutronix
- OpenEmbedded : open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems.

Introduction to Yocto

Yocto Project

What is it?

- It's not an embedded Linux distribution - it creates a custom one for you
- The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture.
- Based on OpenEmbedded
- Who is behind it?
  - Linux Foundation
  - Sponsored by many key semiconductor manufacturers
  - **SILICA is a Yocto Project Participant**
- Provides a highly flexible, yet complex way to build a custom distribution
- Provides SDK tools (Eclipse plugin)
- Provides GUI tools to hide complexity to end user – HOB
- http://www.yoctoproject.org

What the Yocto Project Provides

- The industry needed a common build system and core technology
  - Bitbake and OpenEmbedded build system
- The benefit of doing this is:
  - Long term maintainability
  - Designed specifically for Embedded products
  - Simple integration of major kernel updates
  - Very active developer community

- ***Less time spent on things which don't make money (build system, core Linux components)***
- ***More time spent on things which do make money (app development, product development, etc)***

What is the Yocto Project™?

ArchiTech
SILICA Design Tools

www.yoctoproject.org

- Consists of several separate projects :
  - **Bitbake** : Build engine - parses metadata and runs tasks
  - **OpenEmbedded Core** : core metadata and build information to build baseline embedded systems
  - **Poky** : Yocto example distribution which integrates all the required pieces and makes an official release
  - **Hob** : GUI tool to select packages to build and easily create custom image

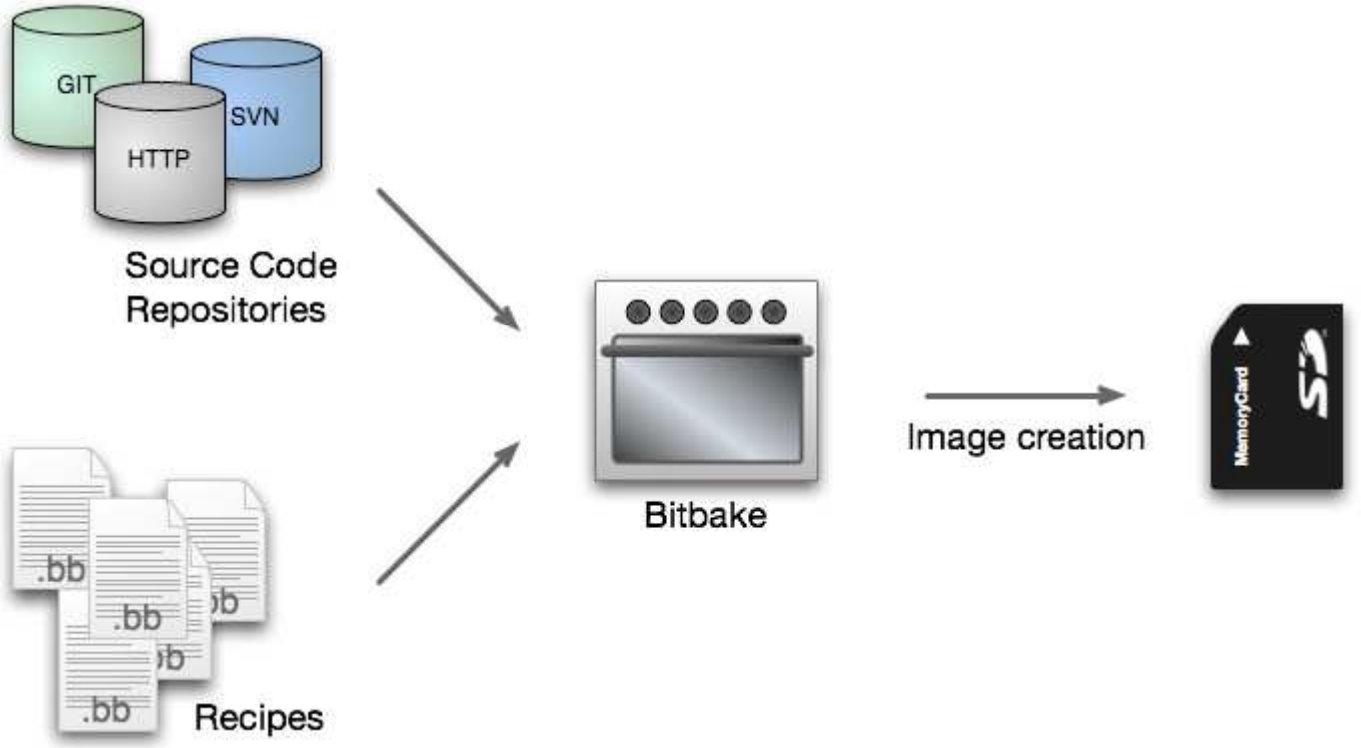
yocto PROJECT

PROJECTS

- Poky
- Cross-Prelink
- Eclipse IDE Plug-in
- Openembedded Core
- Pseudo
- Swabber
- AutoBuilder
- Application Development Toolkit (ADT)
- Hob
- EGLIBC
- Build Appliance

Yocto in a nutshell

Bitbake build process



Bitbake

Baking your recipes

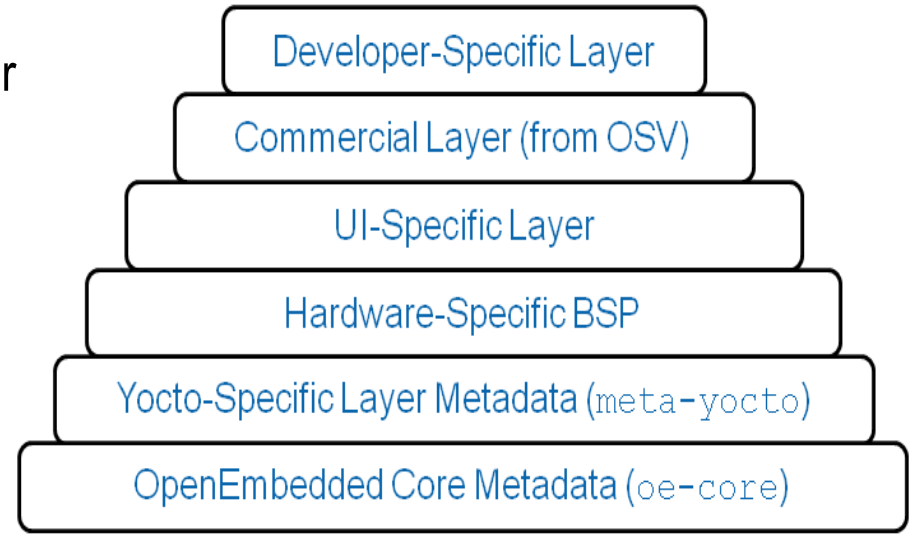
- A make like build tool
- Started within the OpenEmbedded project, then was separated into a standalone tool
- Processes build instructions (recipe files) to produce desired tool chain, package or final board image

Bitbake recipes / meta data

- Bitbake recipes specify how a particular package is built
- Bitbake recipe extension **.bb**
- Recipes can control the build of a simple user application right through to a complete Toolchain or Linux System image
- Includes all the package dependencies, source code locations, configuration, compilation, build and install instructions
- Some Recipes can inherit attributes/methods from Bitbake classes (**.bbclass**)
- All predefined classes are under
  - **meta/classes**
- Meta-data includes recipes, bitbake classes, configuration files

Yocto Layers

- Yocto meta-data is organized into **layers**, so that configurations can be kept isolated from each other
- Each layer contains ‘recipe files’ which control how a particular feature is built.
- Each SOC variant will have it’s own vendor layer to implement H/W specific features.
- Package recipes are organized in layer sub-folders.
- Layers can have dependencies on other layers.



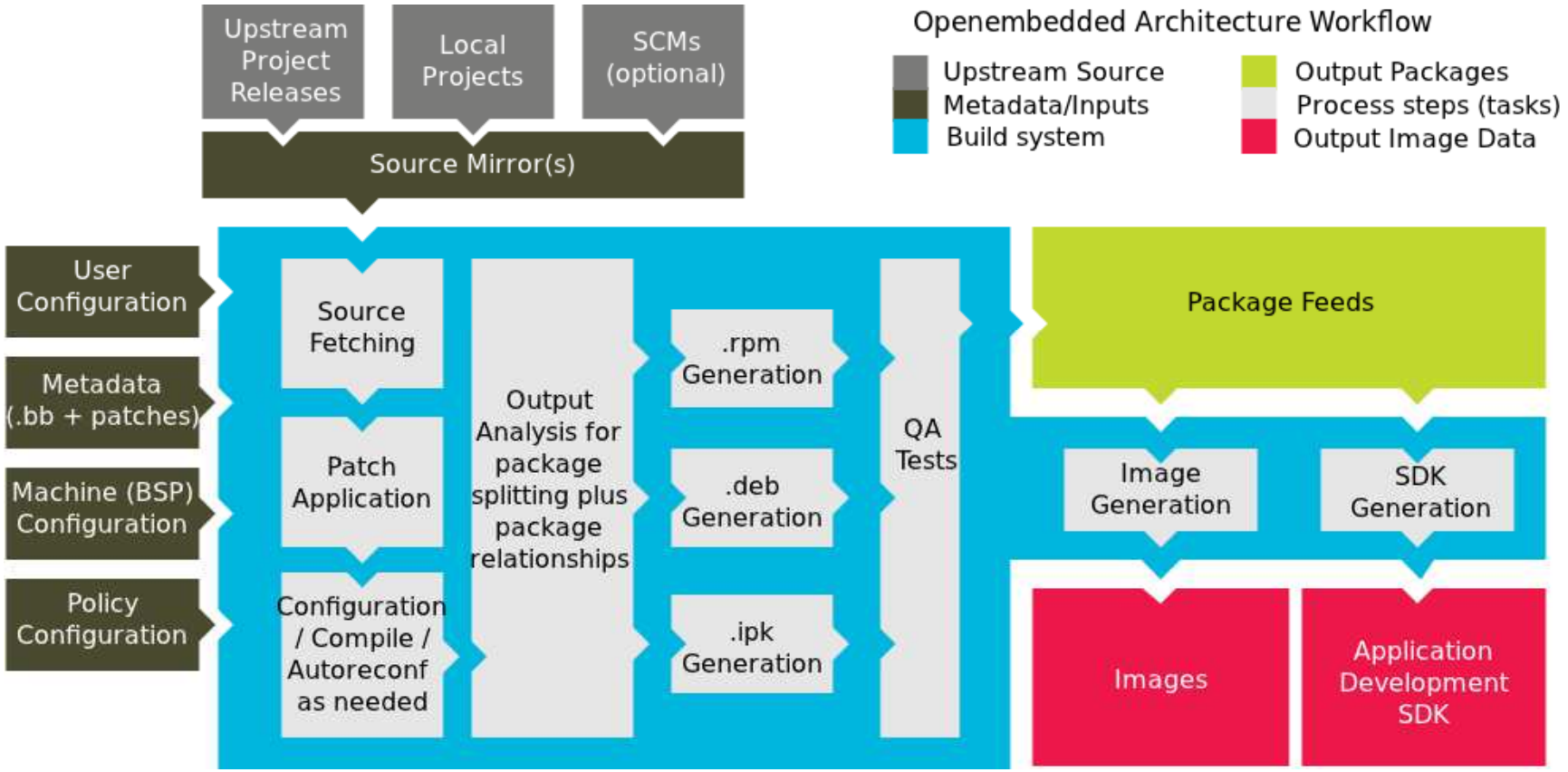
Yocto Layers

ArchiTech
SILICA Design Tools

- Layers are a way to manage extensions, and customizations to the system
  - Layers can extend, add, replace or modify recipes
  - Layers can add or replace bbclass files
  - Layers can add or modify configuration settings
  - Layers are added via BBLAYERS variable in build/conf/bblayers.conf
- Best Practice: Layers should be grouped by functionality
  - Custom Toolchains (compilers, debuggers, profiling tools)
  - Distribution specifications (i.e. meta-yocto)
  - BSP/Machine settings (i.e. meta-yocto-bsp)
  - Functional areas (selinux, networking, etc)
  - Project specific changes

Yocto Project Development Environment

www.yoctoproject.org/docs/current/yocto-project-qs/yocto-project-qs.html



Yocto build process

User Configuration

- build/conf/local.conf is where you override and define what you are building
  - BB_NUMBER_THREADS and PARALLEL_MAKE
  - MACHINE settings
  - DISTRO settings
  - INCOMPATIBLE_LICENSE = “GPLv3”
  - EXTRA_IMAGE_FEATURES
- build/conf/bblayers.conf is where you configure which layers to use
  - Add Yocto Project Compatible layers to the BBLAYERS
  - Default: meta (oe-core), meta-yocto and meta-yocto-bsp

Yocto build process

Metadata (Recipes)

- Metadata and patches:
- Recipes for building packages
  - Recipe:
    - meta/recipes-core/busybox_1.20.2.bb
  - Patches:
    - meta/recipes-core/busybox/busybox-1.20.2.patch
- Recipes inherit the system configuration and adjust it to describe how to build and package the software
- Can be extended and enhanced via layers

Yocto build process

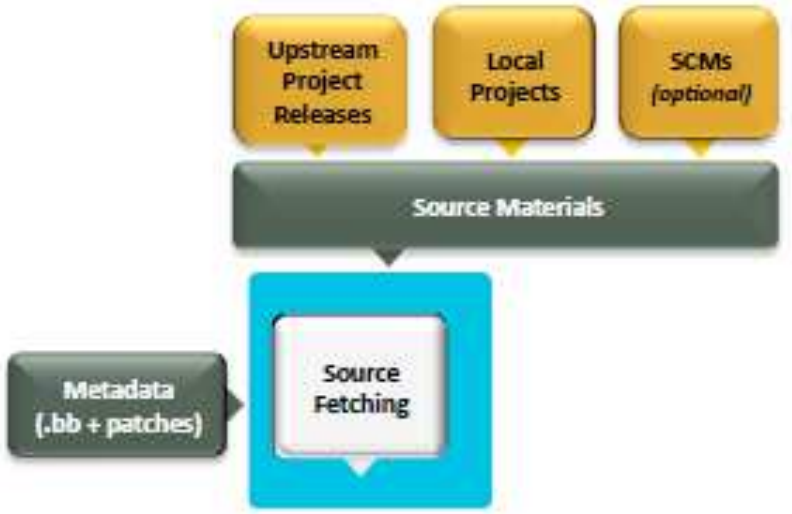
Machine (BSP) Configuration

- Configuration files that describe a machine
  - Define board specific kernel configuration
  - Form factor configurations
  - Processor/SOC tuning files
- Eg, meta-fsl-arm/conf/machine/imx6sxsabresd.conf
- Machine configuration refers to kernel sources and may influence some user space software

Yocto build process

Source Fetching

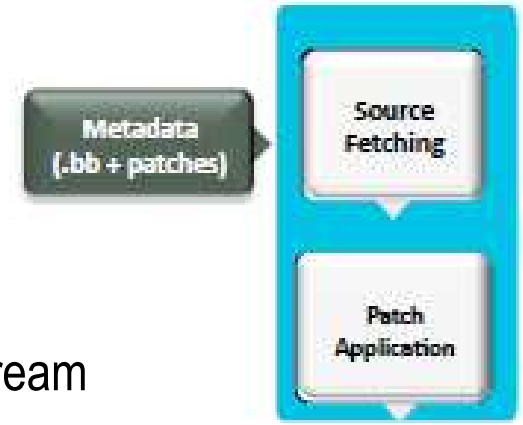
- Recipes call out the location of all sources and patches.
- These may exist on the internet or be local. (See SRC_URI in the *.bb files)
- Bitbake can get the sources from git, svn, bzr, tarballs, and many more
- Versions of packages can be fixed or updated automatically
  (Add SRCREV_pn-PN = "${AUTOREV}" to local.conf)
- The Yocto Project downloads all sources to a local repository to ensure source availability



Yocto build process

Patching

- Once sources are obtained, they are extracted
- Patches are applied in the order they appear in SRC_URI
  - quilt is used to apply patches
- This is where local integration patches are applied
- Patch authors are encouraged to contribute their patches upstream whenever possible
- Patches are documented according to the patch guidelines:
  - http://www.openembedded.org/wiki/Commit_Patch_Message_Guidelines

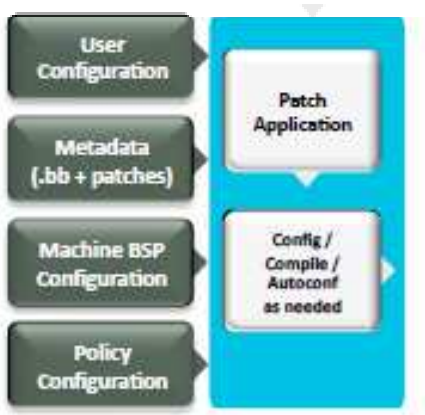


Yocto build process

Configure / Compile / Install

- Recipe specifies configuration and compilation rules
  - Various standard build rules are available, such as autotools and gettext
  - Standard ways to specify custom environment flags
  - Install step runs under ‘pseudo’, allows special files, permissions and owners/groups to be set
- Recipe example:

```
DESCRIPTION = "GNU Helloworld application“
SECTION = "examples"
LICENSE = "GPLv2+"
LIC_FILES_CHKSUM = "file://COPYING;md5=751419260aa954499f7abaabaa882bbe"
PR = "r0"
SRC_URI = "${GNU_MIRROR}/hello/hello-${PV}.tar.gz"
inherit autotools gettext
```

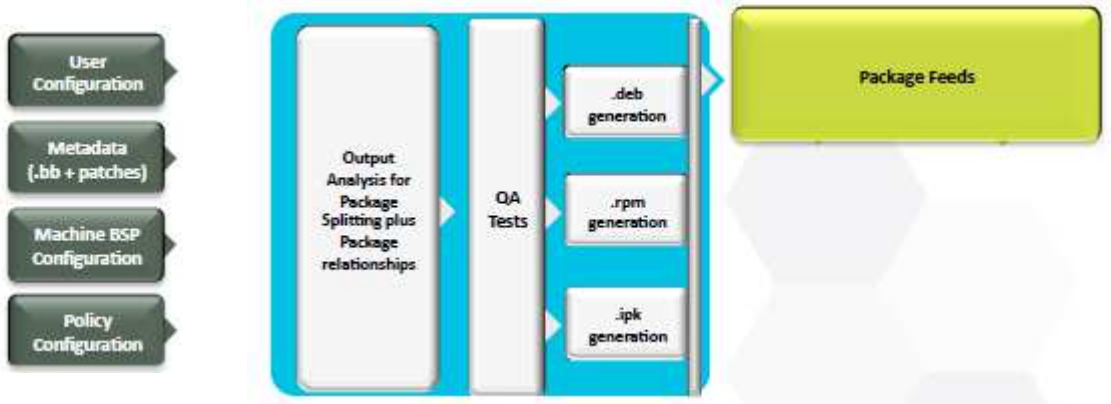


Yocto build process

ArchiTech
SILICA Design Tools

Output Analysis / Packaging

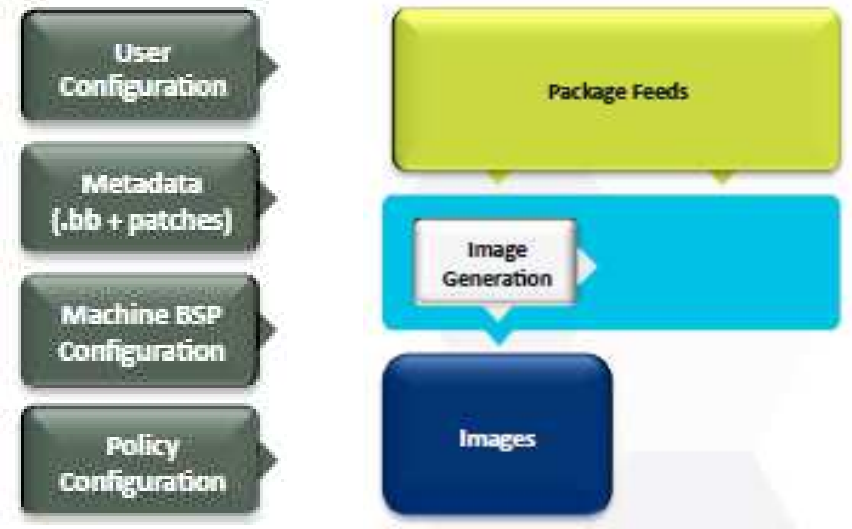
- Output Analysis:
  - Categorize generated software (debug, dev, docs, locales)
  - Split runtime and debug information
- Package Generation:
  - Support the popular formats, RPM, Debian, and ipk
  - Set preferred format using PACKAGE_CLASSES in local.conf
  - Package files can be manually defined to override automatic settings



Yocto build process

Image Generation

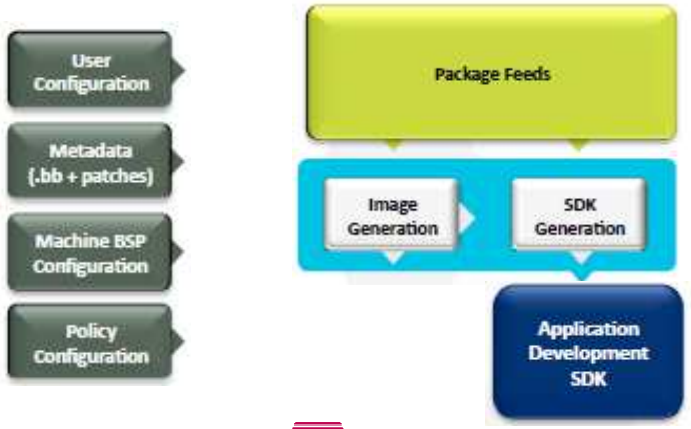
- Images are constructed using the packages built earlier and put into the Package Feeds
- Decisions of what to install on the image is based on the minimum defined set of required components in an image recipe. This minimum set is then expanded based on dependencies to produce a package solution.
- Images may be generated in a variety of formats (tar.bz2, ext2, ext3, jffs, etc…)



Yocto build process

Yocto SDK Generation

- Yocto can generate a cross compiler tool chain for your specific target board
- Specifies compiler, linker and build settings
- SDK may be based on the contents of the image generation
  - Reference specific libraries used in Root File System
  - Introduced in Danny release
- A plug in to Eclipse is available to set up the application development environment
- May contain a QEMU target emulation to assist app developers



Hob User Interface

Graphical tool to modify package configurations

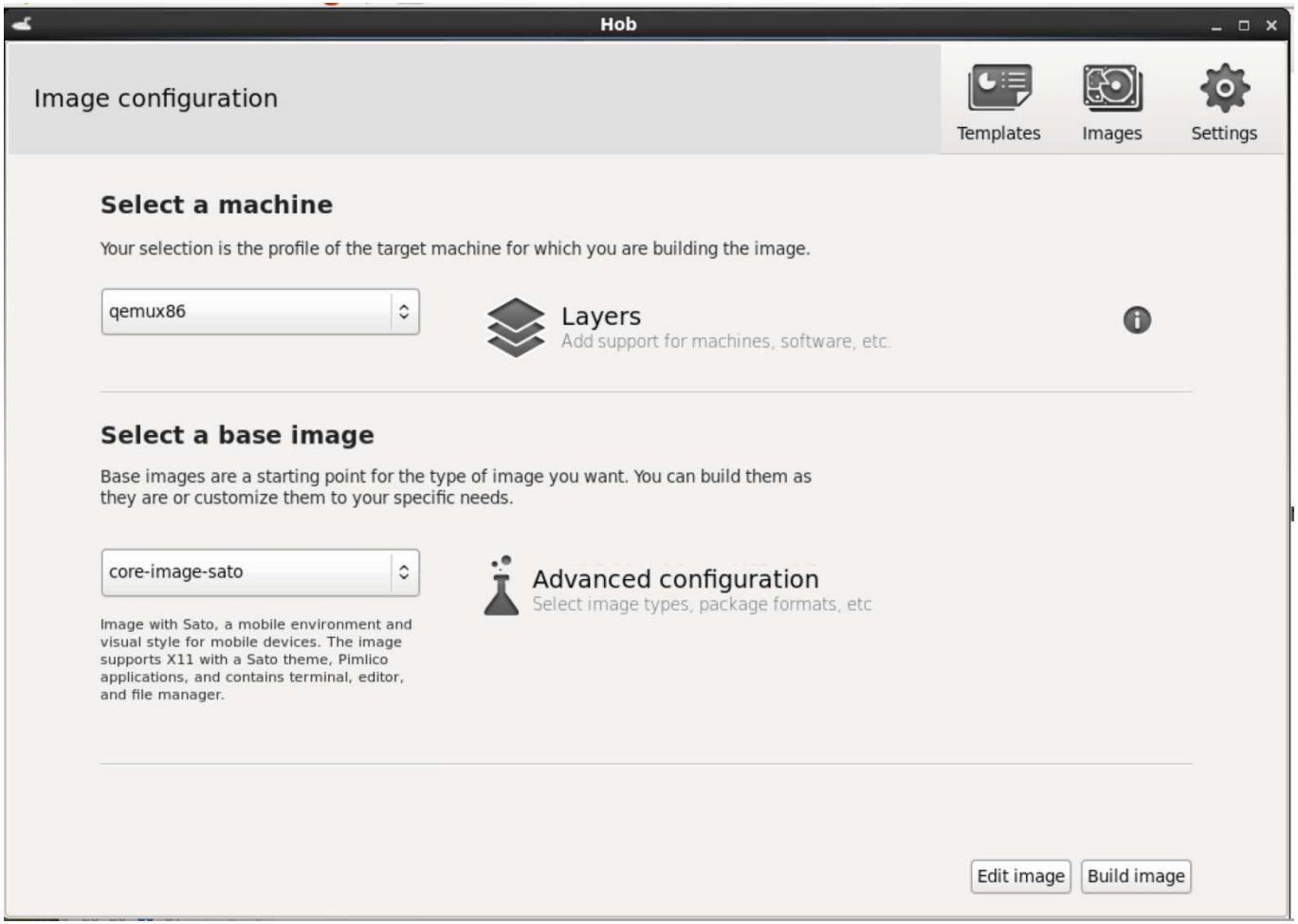
- There has to be an easier way than setting various configuration files…

- The Hob User Interface is that way.
  - Graphical interface
  - Local config can be edited
  - Package recipes can be included or excluded
  - Graphical feedback on build progress

  - Tool is still considered experimental
  - Will be replaced by another tool called Toaster

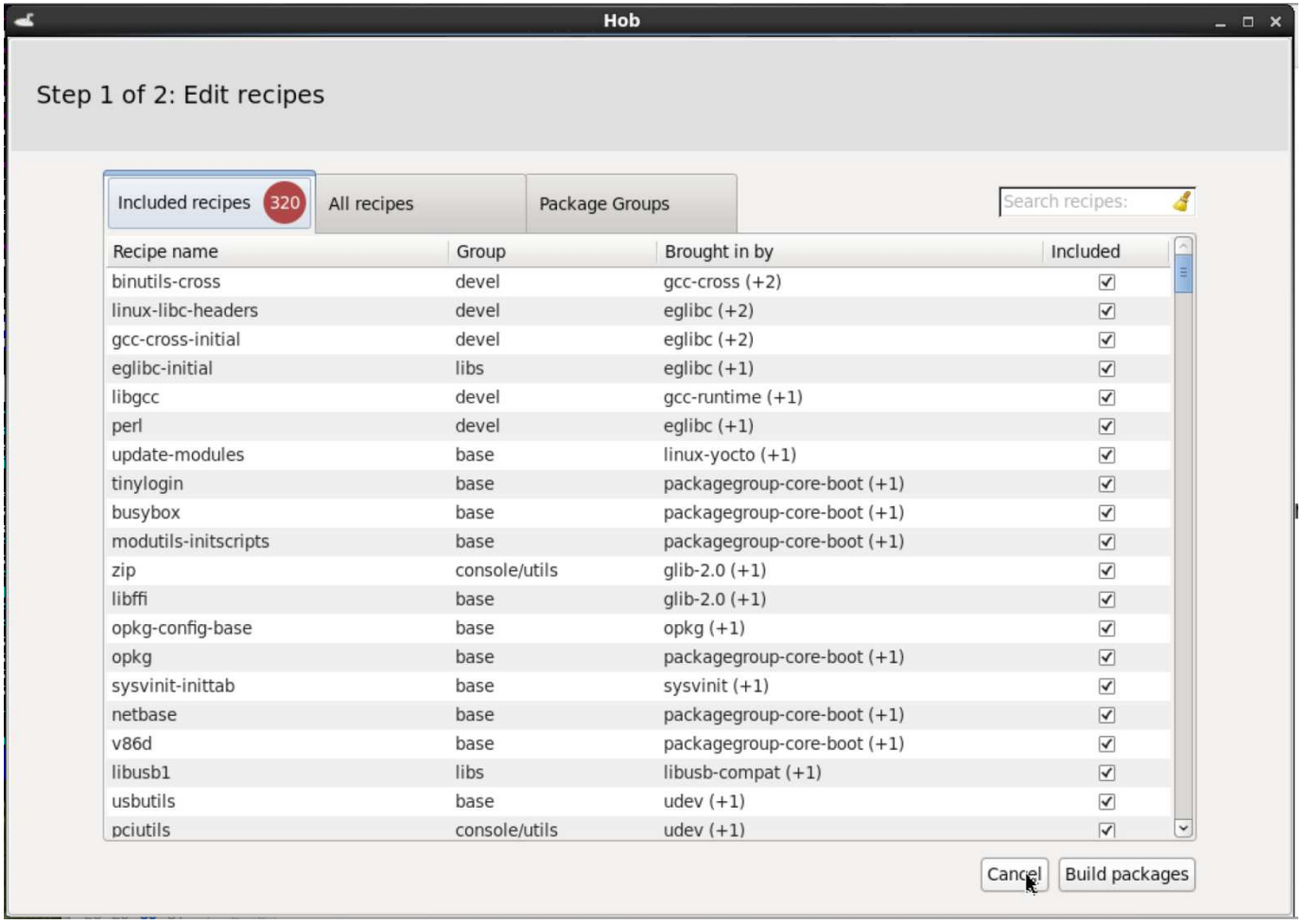
Hob User Interface

Graphical tool to modify package configurations



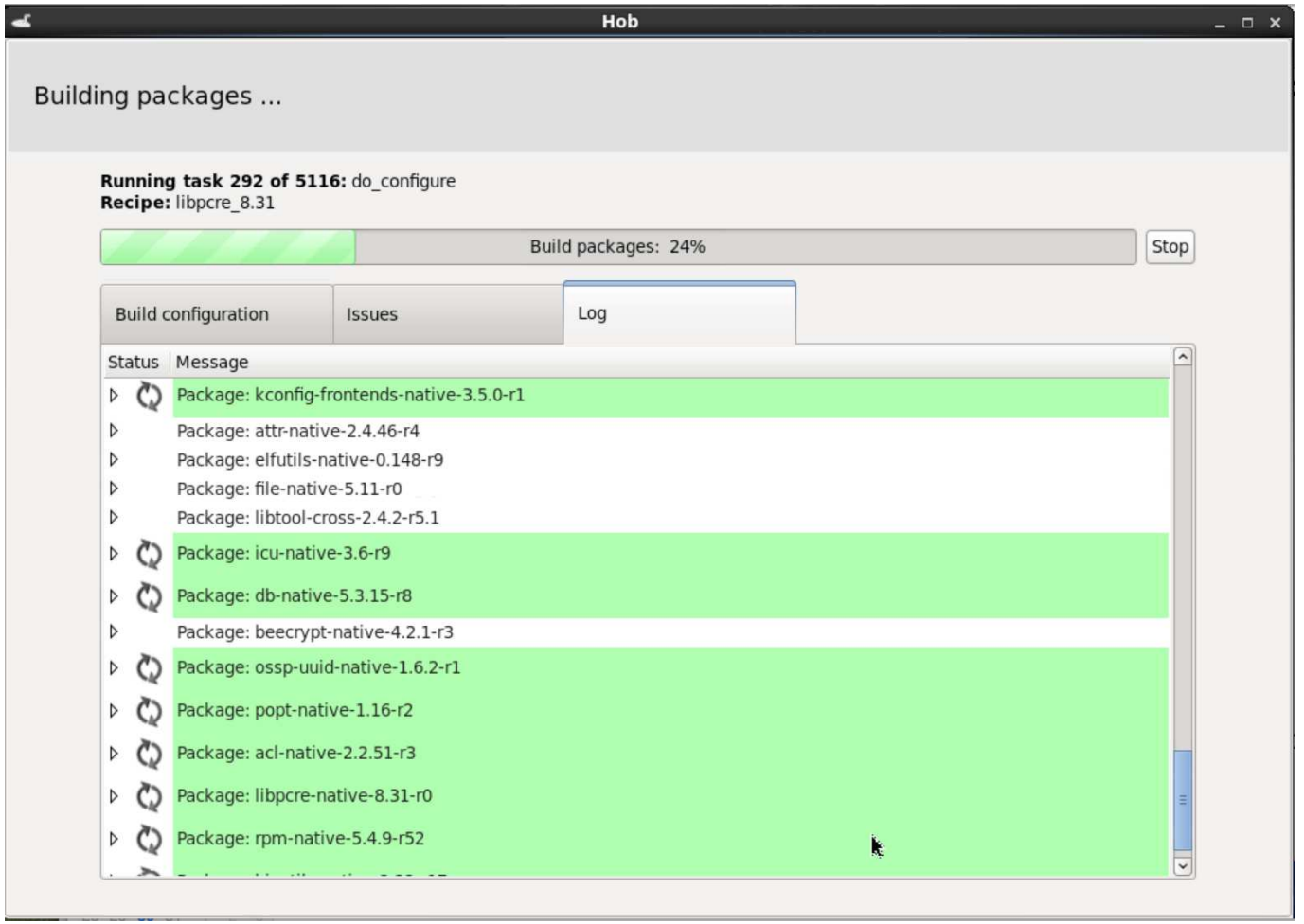
Hob User Interface

Graphical tool to modify package configurations



Hob User Interface

Graphical tool to modify package configurations



Yocto Summary

- The Yocto Project provides tools, templates and best practices for you to create your embedded Linux OS
- The Poky project provides a set of reference distribution components in one place to make it easy to get started
  - It helps set up the embedded app developer
  - Both device and app development models supported
  - Getting started is easy
- It's not an embedded Linux distribution – it creates a custom one for you

Introducing the ArchiTech Yocto SDK

ArchiTech

...delivering so much more than just a development board

- Choosing the right silicon device for a project – no longer all about the device itself
- Especially true for Embedded Processors
  - What software will the device will run?
  - What ecosystem is available to help you develop that S/W ?
  - What support can I expect from my supplier?
- SILICA Mission
  - Take you from dev board to production – shortest time with the minimum of hassle
- ArchiTech range of dev boards and software solutions
  - Partnered with leading semi vendors
  - Industrial quality development boards
  - Fully evaluate device feature set
  - Accelerate your design cycle

ArchiTech Yocto SDK

...delivering so much more than just a development board

- All boards supported by ArchiTech Linux Distribution – based on Yocto Project
- SILICA an active participant in Yocto Project – along with major semi vendors
- ArchiTech removes complexity – ArchiTech Yocto SDK
  - VirtualBox Virtual machine image – Linux development host
  - Pre-installed Yocto and SDK Tools
  - Pre-configured with ArchiTech BSP for each board
  - No requirement to set up a separate Linux Host
- Quickly and easily re-create boot loader, kernel and root file system
- Simple to reconfigure kernel and rootfs to your own requirements

ArchiTech Yocto SDK

...delivering so much more than just a development board

- Yocto HOB tool included
  - GUI tool to configure build and manage package selection
- Eclipse and Qt Creator S/W Development IDE's included
  - Pre-configured – eliminates complicated setup
  - Start application development immediately
- ArchiTech Linux distribution and BSPs 'Open Source' and freely available
  - Take our BSP layer and make simple modifications for your own custom board
- ArchiTech SDK – one common development flow across all boards
  - Learn only one tool – not vendor specific

ArchiTech Yocto SDK

...delivering so much more than just a development board

- Board BSPs developed in house at SILICA
- ArchiTech boards and Yocto SDK supported by SILICA
  - Extensive team of Field Apps Engineers
  - SILICA Software Specialists
  - ArchiTech Support Forum
  - Board Schematic and gerber files freely available
- Get up to speed even quicker
  - ArchiTech Yocto training classes all across Europe
- For more information go to www.architechboards.org

Hands-on Lab
targeting i.MX 6SoloX SABRE SD

Goal of Training

- Create and run an embedded Linux system on a real board
  - System created with Yocto
- All steps are covered
  - Download and setup of Yocto build framework
  - Configuring Yocto to build a Linux board image
  - Flashing and booting the image to the board
  - Creating a “Hello World” recipe
  - Package management
  - Customising the Kernel
- You may not complete all of the lab material today

Yocto under the hood

- This training gives you a good insight into how Yocto is used
- You need an understanding of how Yocto is installed and configured
- You also need to understand how a board image is created
- ArchiTech have released a Virtual Machine with pre-configured Yocto tools and a simple menu driven interface
  http://downloads.architechboards.com/sdk/virtual_machine/download.html
- This remove much of the pain for developers
- However, if something breaks, this training will stand you in good stead

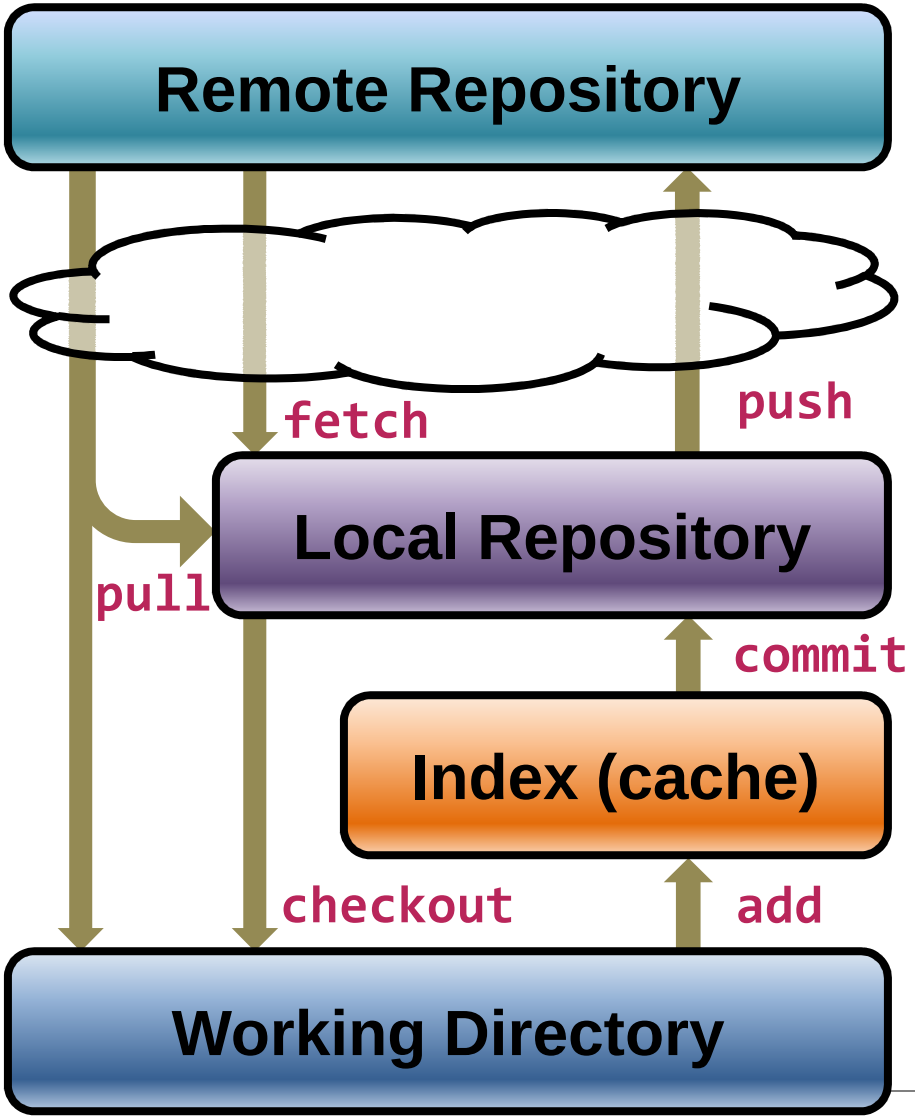
Linux Virtual Machine

- It is a convention that development for Embedded Linux is performed on a Linux machine (or Host)
- For these labs we are supplying a pre built Virtual machine
  - Runs under VirtualBox
  - Implements a Linux Host running the 32 bit version of Ubuntu 14.04 LTS
- Yocto tools have been pre installed into the Virtual Machine
- Instructions on how to start the Virtual Machine are given in chapter 1 of the lab doc.

All tools are pre-installed

- Certain steps have already been performed for you.
- Internet access would normally be required to set up the Yocto build environment
- In the interests of saving time, we have done this for you, BUT...
  - Full instructions on how to do this yourself from scratch have been included
  - Please read chapters 2, 3 and 4 to get a feel for what is required
  - However... DO NOT perform these steps (this is highlighted in the Lab instructions)
- Once you have started the Virtual Machine and READ through chapters 2-4, please start the lab proper at chapter 5

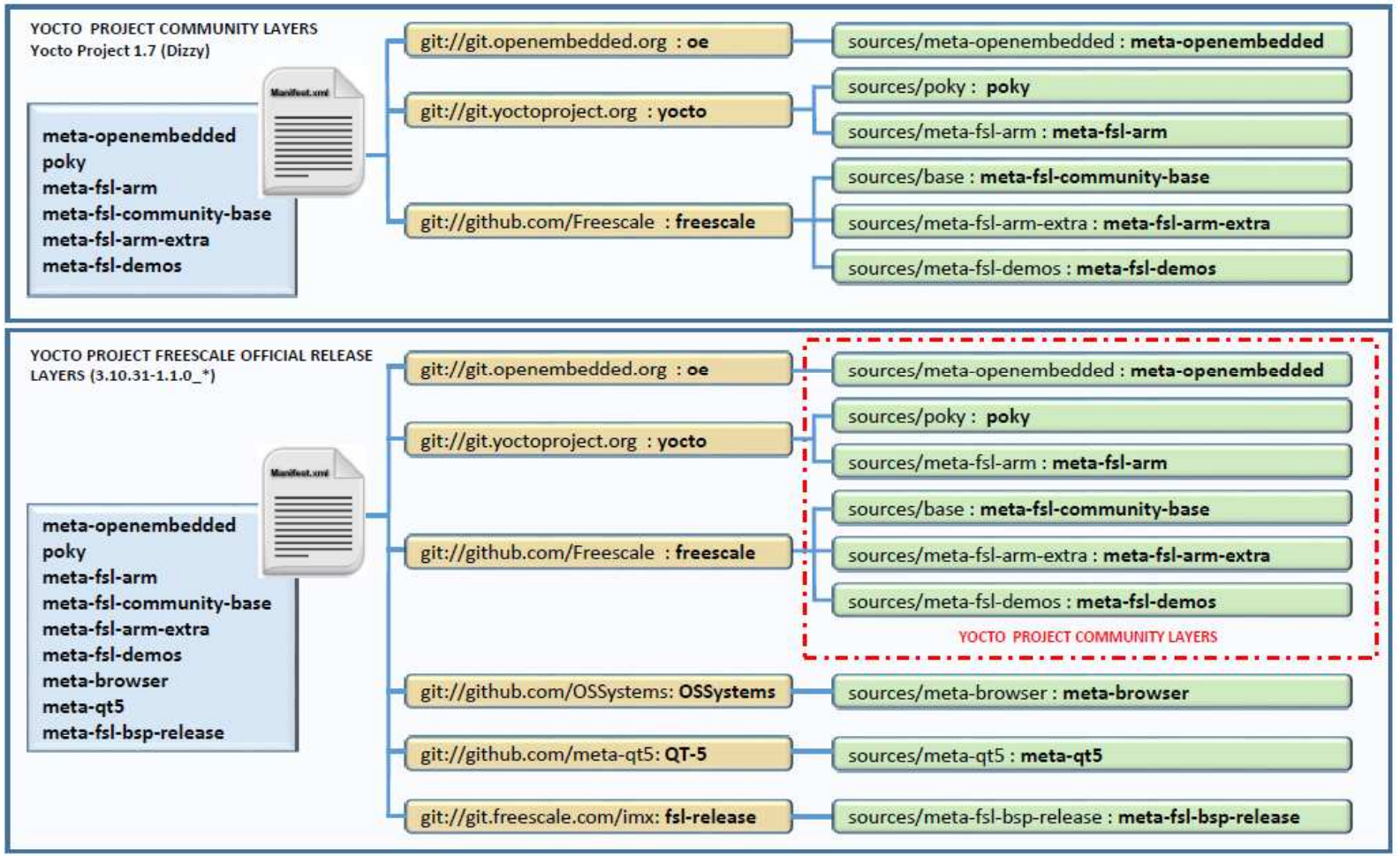
What is Git?



- Source code management system
- Distributed revision control system
- Free software distributed under the terms of the GPL
- Intended to replace CVS and SVN
- For more info visit
  http://git-scm.com/

Freescale Yocto support

Community versus Official FSL BSP



Community versus Official FSL BSP

- Community release checks out the head of a release branch
  - Each checkout will give the most up to date files
- Official FSL release uses specific checkout hashes
  - Guarantees the same source files are used every time
- Official BSP includes extra layers
  - meta-browser
  - meta-qt5
  - meta-fsl-bsp-release (Freescale modifications to community BSP)
- Both include setup scripts to automate environment setting
  - Machine selection and EULA set in local.conf
  - Layers added to bblayers.conf
- Both use Google Repo tool to automate git cloning of layers

Common Terminology

- You will often see reference to $HOME
  - This points to the users 'HOME' directory
  - The symbol ~ can also be used instead of $HOME
- sudo command (Super User Do)
  - This gives an ordinary user admin privileges
  - Needed to run certain tasks or modify certain protected files
- File permissions and ownership
  - Linux extends the concept of file permissions with the concept of file ownership
  - Three groups are defined
    - owner, group and other
  - Each one can have separate read, write and execute rights

Lab chapter 5

Steps to configure Yocto to build a SoloX Sabre SD Image

- *source poky/oe-init-build-env*
  - This runs the 'oe-init-build-env' script to set up the correct environment variables for your build. It also creates the 'build' directory along with others
- Edit the *~/yocto/build/conf/local.conf* file
  - *BB_NUMBER_THREADS* and *PARALLEL_MAKE* specify the number of threads/CPUs to be used for multiprocessing. This will drastically speed up the build.
  - *MACHINE = "imx6sxsabresd"* specifies we are targeting a SoloX Sabre SD board
- Edit the *~/yocto/build/conf/bblayers.conf* file
  - Here you will add the extra meta-layers so that Yocto can find them
- Build the actual Linux image with the command
  - *bitbake core-image-minimal*

Lab chapters 9+10

ArchiTech
SILICA Design Tools

Package Management

- 'Packages' are archives containing pre-compiled applications or utilities
- A package manager tool can be used to extract and install these packages directly on the hardware system
- A remote package repository can be created which the board can access to 'update itself'
- There are several different package formats supported by Yocto
  - RPM, IPK, DEB and more
- *bitbake package-index*
  - Re-builds the package index containing all available packages