Embedded Linux Conference

Binary Artifacts and the Ease of Use Onramp for the Yocto Project

Bruce Ashfield, Xilinx

#lfelc @zeddii

THE LINUX FOUNDATION

Agenda & Goals

- Agenda
  - Level Set
  - Use Case Summary
  - Technology explanation
  - Examples and Future work

- Goals:
  - Introduce the various binary artifacts produced by the Yocto Project
  - Explore the binary plumbing / infrastructure
  - Show how "ease of use" and the Yocto project are not mutually exclusive

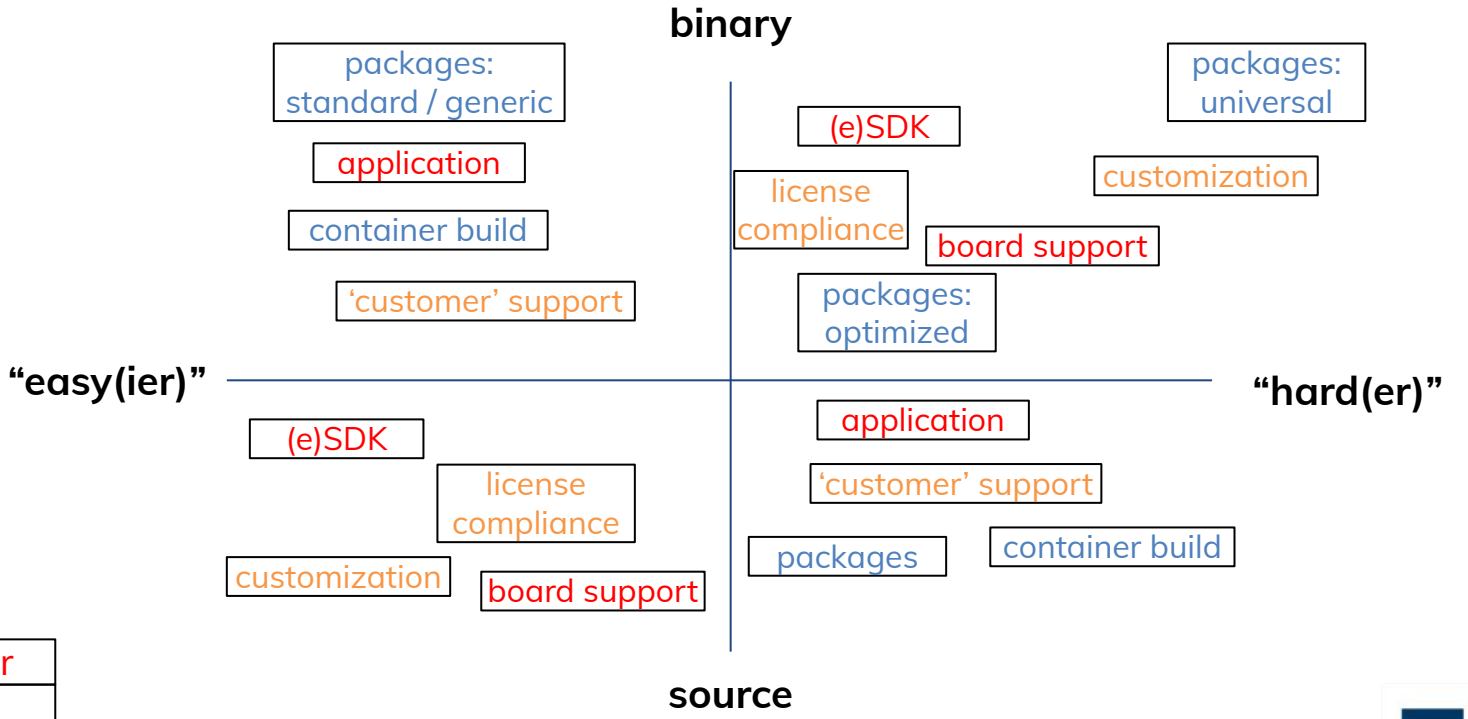
Terminology / Concepts

- What do we mean by "binary artifacts" and "ease of use" ?
- Binary artifacts:
  - Outputs from a defined build that can be used / installed on a running target, or to construct a target image. The architecture and optimization are defined by the build parameters, and can impact the level of reusability
  - ARM platforms have unique challenges
    - Instruction and optimization techniques vary between platforms
    - Conflicts with the desire to run common/generic binaries
- Ease of use:
  - It is obvious / clear how to complete (initial) steps towards a goal
    - Details vary by use case
  - Transitioning between use cases is supported and documented

Common Questions / Comments

- Are binary packages supported ?
- Do I have to start building from source ?
- Are the binary outputs:
  - Compatible with 3rd party packages ?
  - Fully optimized for platform ‘x’ or software stack ‘y’ ?
- What is behind the binary artifacts ?
  - OE core and the ecosystem metadata (recipes + configuration)
  - Not the sources of other distributions or base/binary packages of other distros
- Can I apt/dnf update my target ?
- ‘docker’ build ?
- Why would I use the Yocto Project binary artifacts versus distro ‘x’ ?
- ….

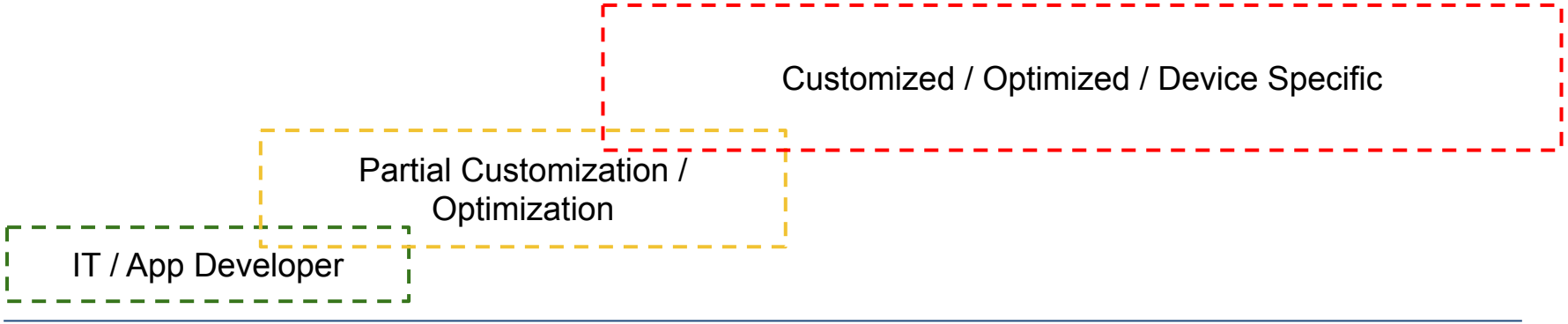
Ease of Use / Complexity



category:

developer
user
production

Is a Binary Distribution appropriate? Reuse?



Yocto Project: Binary Artifacts

- OE / Yocto Project history with Binary Artifacts
  - Build Appliance (containers)
  - Buildtools (SDK to augment older hosts)
  - Toolchains
  - BSP/Machine artifacts (DTB, bootloader, kernel, images) for testing
- Designed to support total re-use, some customization, or total customization

WIP: a reference binary feed for those that do not need to customize base / standard packages or for those that wish to embrace & extend

Binary Distribution Artifacts

- Inputs
  - Build configuration
  - Layers
  - Site and Local configuration (minimize this for sharing)

- Outputs / Artifacts
  - Some binary artifacts are internal others are user visible
  - Shared-state (build cache)
  - Hash Equivalency (cache re-use)
  - PR Service (manage package upgrades)
  - Package feeds
    - package manager of choice (deb, rpm, ipk)
  - Non-OS components
    - Bootloaders, dtbs, firmware, etc.
  - Pre-built images
    - Starting points -- download and run
  - OCI Images (containers)

Example Binary Distribution Artifacts: Xilinx

- System Configuration
  - Build scripts, layers, etc.
- Intermediate Artifacts
  - Shared State, etc.
  - DTBs, WIC (image) generation, etc.
- Getting started
  - SD Card Image(s)
  - eSDK (Used to build/customize packages and images)
  - SDK (Used to build applications)

Use Case Evolution

- Hide the learning / complexity curve until it is needed
  - Solving problems you don't know you have
- Heterogeneous systems
  - Firmware
  - MCUs
- Not just images for flashing
  - containers, binary deltas
  - deep software stacks: k*s
  - microservices
- Blended embedded/edge and enterprise features
  - more than just a package installer
  - accelerated containers, safe/secure containers (i.e. runx)
  - low footprint runtimes
  - maintenance and in-service upgrades
- Use case ‘mobility’ is key
  - Is there a defined / structured way to change use cases

Beyond Packages: Things to Consider

- Reproducibility
  - core Yocto Project capability (see reproducible-builds.org)
- Licensing / SBOM
  - core capability
  - multiple ways to consume it and accompany binary deliveries
- Customization
- Support from the ecosystem (if modified/extended, or not)
- Platform Extension
- Application **AND** system developers
- Support, maintenance, and updates

When adding the above to many traditional / enterprise distros, it can be ad-hoc and potentially less structured than the Yocto Project (at that point, the complexity and learning curves are similar)

Sample / Demo: Image with package feed

- config:
  - Minimal image
    - configuration for package management, systemd and base image
- package management (dnf) commands on target
- build:
  - add packages to the feed
  - install on target
- containers:
  - add docker + dependencies to the image
  - bitbake base container image
  - push to registry, pull/run on target

Demo: abbreviated setup and initial build

- OE/poky's core-image-minimal

```
$ git clone git://git.yoctoproject.org/poky poky-elc
$ cd poky-elc; source oe-init-build-env
$ cat << EOF >> conf/local.conf
IMAGE_ROOTFS_EXTRA_SPACE = "2097152"
IMAGE_FEATURES += "ssh-server-dropbear package-management"
DISTRO_FEATURES:append = " systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
PACKAGE_FEED_URIS="http://10.10.10.182/"
PACKAGE_FEED_BASE_PATHS = "rpm"
PRSERV_HOST = "localhost:0"
EOF
```

- Note: 10.10.10.182 is the example build machine (but can be whatever machine is serving your packages)

```
$ bitbake core-image-minimal
$ bitbake package-index

# ---- on the build host:
$ cd tmp/deploy
$ sudo python3 -m http.server 80

$ runqemu qemux86-64 nographic kvm slirp
```

Demo: package management on target

```
root@qemux86-64:~# uname -a
Linux qemux86-64 5.13.12-yocto-standard #1 SMP PREEMPT Mon Aug 23 03:00:24 UTC 2021 x86_64 GNU/Linux

root@qemux86-64:~# dnf makecache
OE Remote Repo: rpm
9.6 MB/s | 1.4 MB     00:00
Last metadata expiration check: 0:00:01 ago on Fri Sep  3 02:46:42 2021.
Metadata cache created.

root@qemux86-64:~# dnf search busybox
Last metadata expiration check: 0:02:22 ago on Fri Sep  3 02:46:42 2021.
================================================================================================================
Name Exactly Matched: busybox
================================================================================================================
busybox.core2_64 : Tiny versions of many common UNIX utilities in a single small executable
================================================================================================================
=== Name Matched: busybox
================================================================================================================
===
busybox-dbg.core2_64 : Tiny versions of many common UNIX utilities in a single small executable - Debugging files
busybox-dev.core2_64 : Tiny versions of many common UNIX utilities in a single small executable - Development
files
busybox-ptest.core2_64 : Tiny versions of many common UNIX utilities in a single small executable - Package test
files
busybox-src.core2_64 : Tiny versions of many common UNIX utilities in a single small executable - Source files
busybox-syslog.core2_64 : Tiny versions of many common UNIX utilities in a single small executable
busybox-udhcpc.core2_64 : Tiny versions of many common UNIX utilities in a single small executable

root@qemux86-64:~# dnf search docker
Last metadata expiration check: 0:03:18 ago on Fri Sep  3 02:46:42 2021.
No matches found.
```

Demo: build a new package, add it to the running target

```
# on the build host:
$ bitbake vim
$ bitbake package-index

# on the target
root@qemux86-64:~# dnf clean all
5 files removed
root@qemux86-64:~# dnf makecache
OE Remote Repo: rpm
2.8 MB/s | 1.6 MB     00:00
Metadata cache created.
root@qemux86-64:~# dnf search vim
Last metadata expiration check: 3:59:22 ago on Fri Sep  3 20:21:05 2021.
================================== Name Exactly Matched: vim ==================================
vim.core2_64 : Vi IMproved - enhanced vi editor
====================================== Name Matched: vim ======================================
vim-common.core2_64 : Vi IMproved - enhanced vi editor
root@qemux86-64:~# dnf install vim
Last metadata expiration check: 4:01:41 ago on Fri Sep  3 20:21:05 2021.
Dependencies resolved.
================================================================================================
 Package                        Architecture  Version              Repository               Size
================================================================================================
Installing:
 vim                            core2_64      8.2-r0               oe-remote-repo-rpm      1.3 M
Transaction Summary
================================================================================================
Install  68 Packages

Total download size: 23 M
Installed size: 90 M
root@qemux86-64:~# vim --version
VIM - Vi IMproved 8.2 (2019 Dec 12, compiled Dec 12 2019 13:18:35)
```

Demo: upgrade a package

```
root@qemux86-64:~# expand
-sh: expand: command not found
```

- On the build host, add CONFIG_EXPAND to the busybox config

```
$ bitbake busybox
$ bitbake package-index
$ ls -alF tmp/deploy/rpm/core2_64/busybox-1.34.0-r0.3.core2_64.rpm
-rw-r--r-- 2 bruce bruce 386661 Sep  3 15:02 ./core2_64/busybox-1.34.0-r0.3.core2_64.rpm

root@qemux86-64:~# dnf makecache
OE Remote Repo: rpm
1.2 MB/s | 3.0 kB     00:00
OE Remote Repo: rpm
8.6 MB/s | 1.7 MB     00:00
Metadata cache created.
root@qemux86-64:~# dnf upgrade busybox
Last metadata expiration check: 0:00:02 ago on Fri Sep  3 19:04:23 2021.
Dependencies resolved.
Upgrading:
 busybox            core2_64         1.34.0-r0.3      oe-remote-repo-rpm     378 k

root@qemux86-64:~# expand --help
BusyBox v1.34.0 () multi-call binary.

Usage: expand [-i] [-t N] [FILE]...
```

Demo: build docker

```
$ git clone git://git.yoctoproject.org/meta-virtualization
$ git clone git://git.yoctoproject.org/meta-security
$ git clone git://git.openembedded.org/meta-openembedded
$ cat << EOF >> conf/local.conf
DISTRO_FEATURES:append = " virtualization"
DISTRO_FEATURES:append = " seccomp"
EOF

$ bitbake-layers add-layer ../meta-virtualization
$ bitbake-layers add-layer ../meta-openembedded/meta-oe/
$ bitbake-layers add-layer ../meta-openembedded/meta-filesystems
$ bitbake-layers add-layer ../meta-openembedded/meta-python
$ bitbake-layers add-layer ../meta-openembedded/meta-networking
$ bitbake-layers add-layer ../meta-openembedded/meta-perl
$ bitbake-layers add-layer ../meta-openembedded/meta-security
$ bitbake-layers add-layer ../meta-security
$ bitbake-layers add-layer ../meta-virtualization

$ bitbake docker-moby
# note: this rebuilds the kernel, so we either need to do a dnf kernel update (which may have issues), or rebuild the image
$ bitbake package-index
# option a)
$ bitbake core-image-minimal
# option b):
root@qemux86-64:~# dnf upgrade kernel-5.13.12-yocto-standard.qemux86_64
root@qemux86-64:~# halt

$ runqemu qemux86-64 nographic kvm slirp
```

Demo: install docker

```
root@qemux86-64:~# dnf install docker-moby
Last metadata expiration check: 0:01:13 ago on Fri Sep  3 12:54:14 2021.
Dependencies resolved.
<snip>
Total download size: 51 M
Installed size: 246 M

root@qemux86-64:~# dnf list installed |grep docker
docker-moby.core2_64
20.10.8+gitd24c6dc5cf5e68dfb30027b2db454099566a9b9e-r0 @oe-remote-repo-rpm
docker-moby-cli.core2_64
20.10.8+gitd24c6dc5cf5e68dfb30027b2db454099566a9b9e-r0 @oe-remote-repo-rpm
 root@qemux86-64:~# dnf list installed |grep runc
runc-opencontainers.core2_64         1.0.2+git0+86d83333d7-r0      @oe-remote-repo-rpm

root@qemux86-64:~# dnf list installed |grep kernel-module | wc -l
31
root@qemux86-64:~# dnf list installed |grep iptable | wc -l
104
```

Demo: run a sample container

```
root@qemux86-64:~# systemctl start docker
root@qemux86-64:~# docker --version
 Docker version 20.10.8, build 62eae52c2a
root@qemux86-64:~# docker pull busybox
 Using default tag: latest
 latest: Pulling from library/busybox
 8ec32b265e94: Pull complete
 Digest: sha256:b37dd066f59a4961024cf4bed74cae5e68ac26b48807292bd12198afa3ecb778
 Status: Downloaded newer image for busybox:latest
 docker.io/library/busybox:latest
root@qemux86-64:~# docker run -it busybox /bin/sh
 [  829.164819] docker0: port 1(vethb404865) entered blocking state
 [  829.165495] docker0: port 1(vethb404865) entered disabled state
 [  829.166965] device vethb404865 entered promiscuous mode
 [  830.891459] cgroup: cgroup: disabling cgroup2 socket matching due to net_prio or net_cls activation
 [  831.034937] eth0: renamed from veth0ae59d2
 [  831.036209] IPv6: ADDRCONF(NETDEV_CHANGE): vethb404865: link becomes ready
 [  831.037169] docker0: port 1(vethb404865) entered blocking state
 [  831.037678] docker0: port 1(vethb404865) entered forwarding state
 [  831.038189] IPv6: ADDRCONF(NETDEV_CHANGE): docker0: link becomes ready
 / #
```

Demo: build a container, pull and run it on the target

```
$ bitbake container-base
$ skopeo copy --dest-creds zeddii:"********"
oci:tmp/deploy/images/qemux86-64/container-base-qemux86-64-20210903130751.rootfs-oci:latest docker://zeddii/container-base

root@qemux86-64:~# docker pull zeddii/container-base
Using default tag: latest
latest: Pulling from zeddii/container-base
Digest: sha256:c7a66d2610675ebf22dd4905bd9309fdade3b1c93a35835ae968ccc26db360c1
Status: Image is up to date for zeddii/container-base:latest
docker.io/zeddii/container-base:latest

root@qemux86-64:~# docker run -it zeddii/container-base
[ 6886.026809] docker0: port 1(veth77425be) entered blocking state
[ 6886.027509] docker0: port 1(veth77425be) entered disabled state
[ 6886.028245] device veth77425be entered promiscuous mode
[ 6886.729500] eth0: renamed from vethf51e0d7
[ 6886.730863] IPv6: ADDRCONF(NETDEV_CHANGE): veth77425be: link becomes ready
[ 6886.731831] docker0: port 1(veth77425be) entered blocking state
[ 6886.732316] docker0: port 1(veth77425be) entered forwarding state
/ #
```

- You can now use this as a base container for other docker builds, etc.

Binary artifacts working group

- Part of medium -> long term planning
  - ~monthly sync
  - focus on code and tangible outputs (not just planning)
- Goals
  - Documentation, QA and structure around binary infrastructure
    - consistent, reliable operation
  - Creating a place for collaboration and technology sharing
    - Embrace and extend
  - Address the ease of use and onramp questions
  - Tested reference binary feed
    - Attention to upgrade paths, package management issues
  - Reference containers

Binary artifacts working group

- **NON**-goals:
  - Creating a commercial / supported binary distribution
  - Replacing existing OE based binary distributions

Interested ?

Join OE / The Yocto Project and help unlock the potential of binary artifacts

Future Work

- Reference distribution(s)
  - core package binary feeds
  - prebuilt images
  - installer
  - etc
- Extending reference feeds outside of OEcore
- Reference base containers
  - prebuilt
  - recipes ...
- ....

Embedded Linux Conference