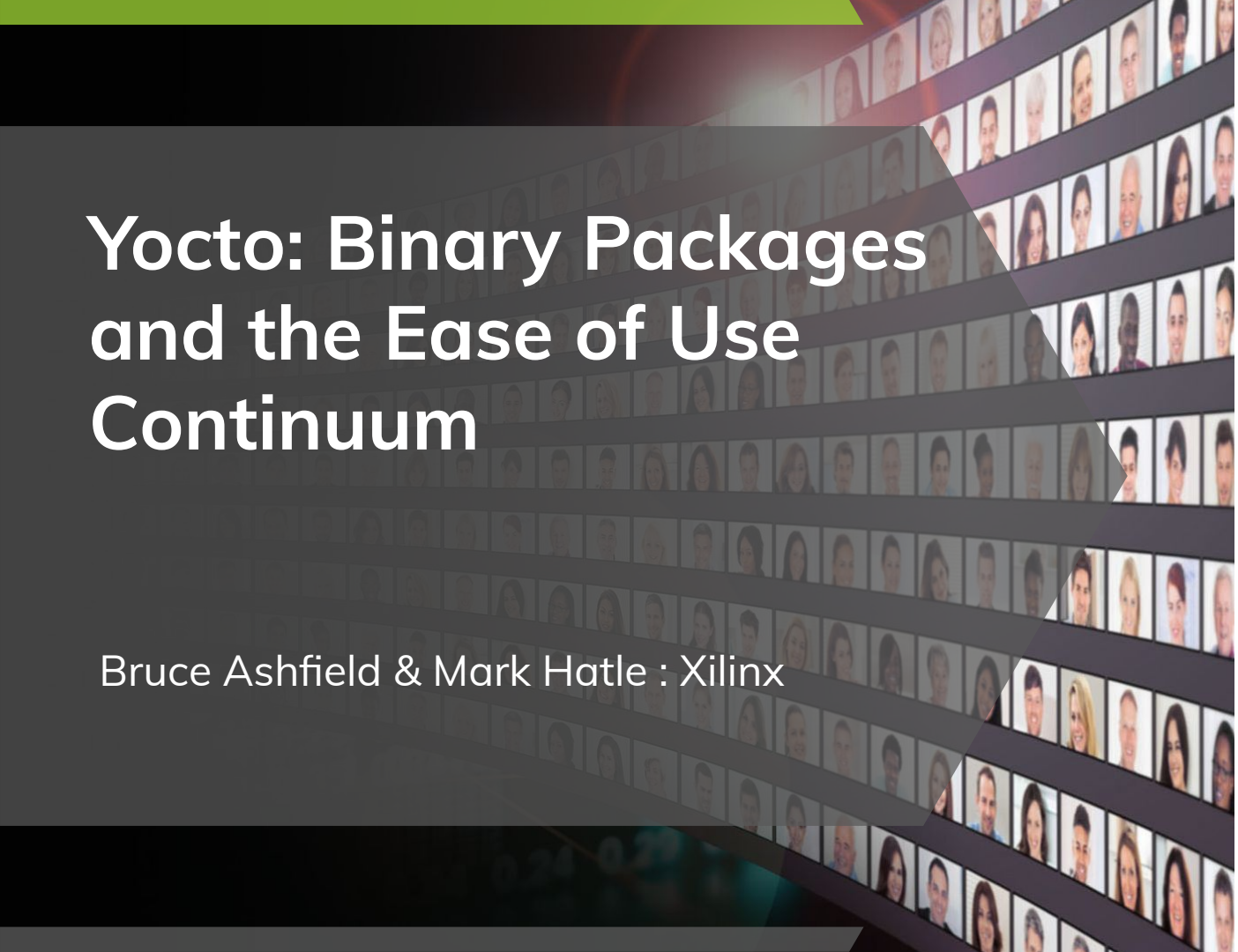


Yocto: Binary Packages and the Ease of Use Continuum

Bruce Ashfield & Mark Hatle : Xilinx

Linaro Connect
VIRTUAL • SPRING 2021

Overview / Goals

- Overview
  - 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

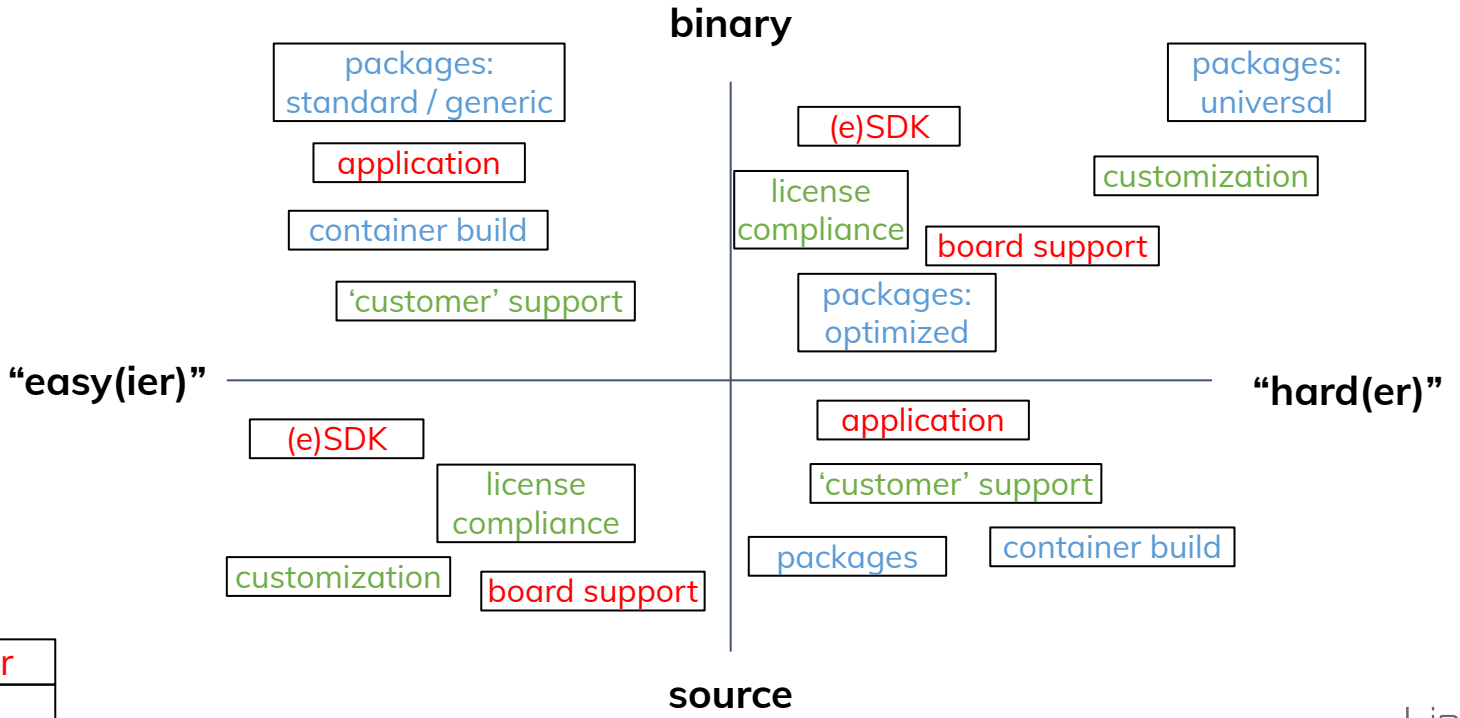
Terminology / Concepts

- 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 greatly 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

Common Questions / Comments

- Are binary packages supported ? or 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 meta data (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’ ?
- ....

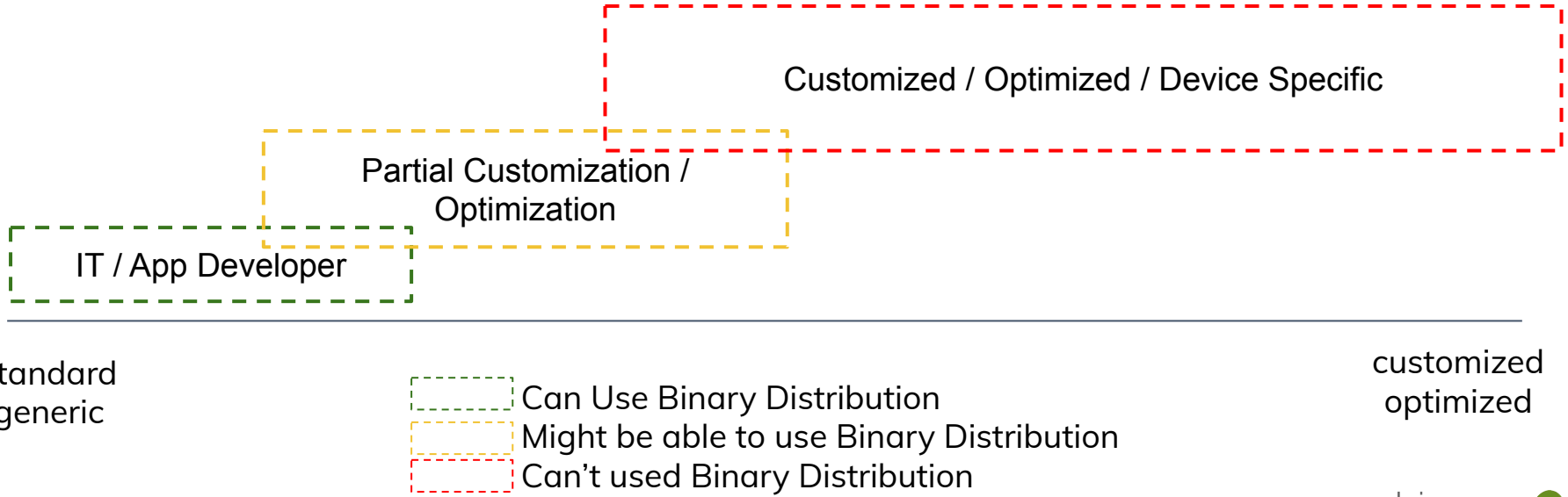
Ease of Use / Complexity



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

Missing: 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

- Build Configuration
  - Which layers?
  - Site and Local (minimize this for sharing)
- Build 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

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

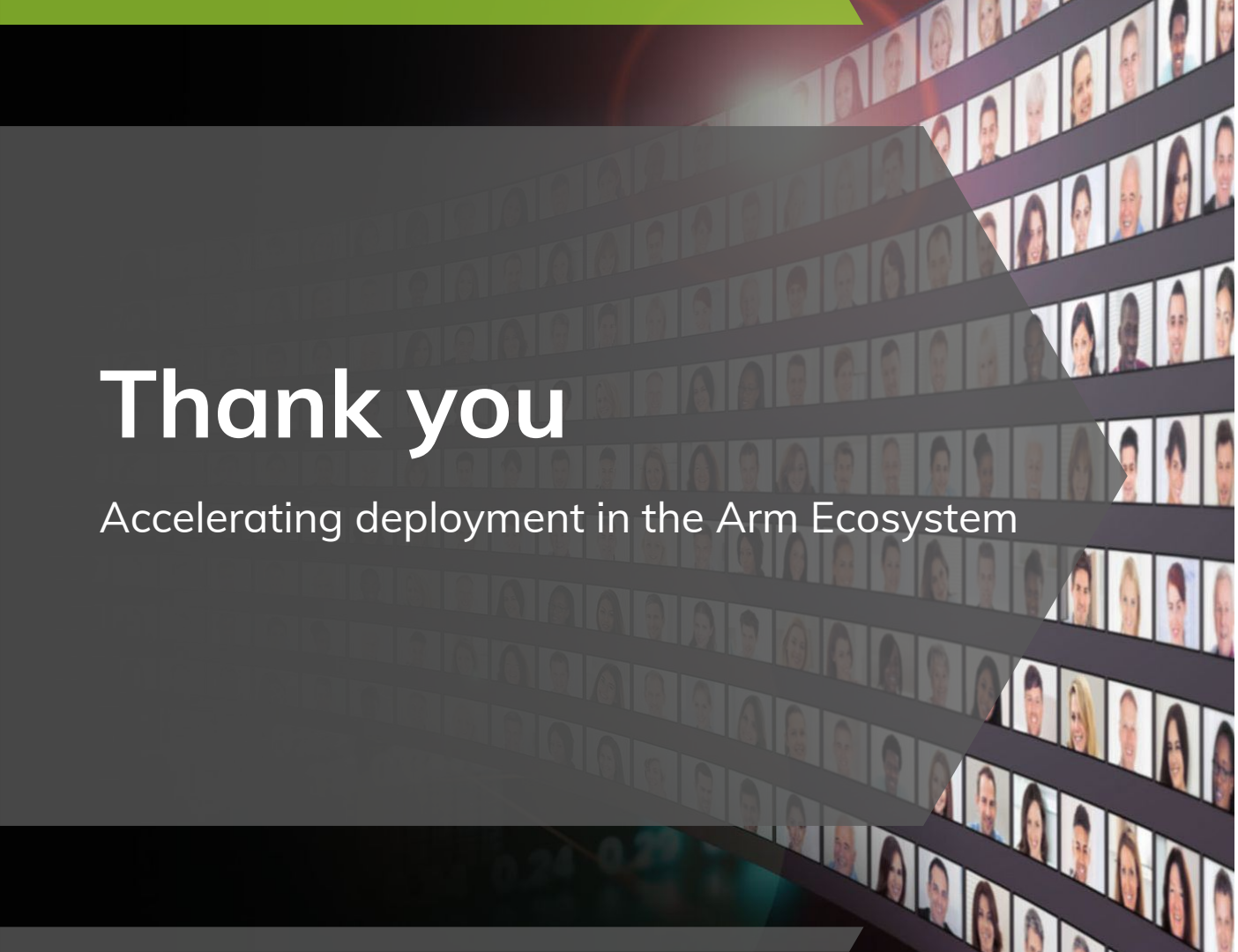
Beyond Packages: Things to Consider

- Reproducibility
  - core Yocto Project capability (see reproducible-builds.org)
- Licensing
  - 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 is ad-hoc and can be less structured than the Yocto Project (at that point, the complexity and learning curves are similar)

Future Work

- Reference distribution(s)
  - Core package binary feeds
  - Prebuilt images
  - etc
- "Docker Hub" reference base containers



Thank you

Accelerating deployment in the Arm Ecosystem

Linaro Connect
VIRTUAL • SPRING 2021