Yocto Project®: Building and deploying containers with meta-virtualization: now & in the future

Bruce Ashfield, Xilinx
Agenda

● Goals
● (Brief) History of meta-virtualization
● Technology timeline
● Why use the Yocto project for containers?
● Components of container build (and deployment)
● Past and current container build / deployment
● Future container build / deployment efforts
Presentation Goals

● Introduce the concepts driving container build and deploy
  ○ Not a how-to, not a survey
● Level set on the technology timeline
● Background / understanding of current capabilities
● What is where, and how (why) it works ..
● Insight into direction and upcoming features
meta-virtualization: a brief history

- Started June 2012
  - 1306+ commits made by 155+ contributors
- Point of integration for ‘virtualization’ technologies
  - VMs and containers
  - Core technology + support software
  - Many audiences: Bleeding edge and established tech
  - Tested (improving) and stable: needs CI
  - Baseline for creating OE derived virtualization solutions
  - Recipes migrate over time (into and out of meta-virtualization)
meta-virtualization: technology timeline

- **Complexity**
  - LXC
  - XEN
  - KVM

- **Container Image Type**
  - container
  - umoci

- **Tools**
  - docker
  - runc
  - kvmtool
  - cri-o
  - runc
  - containerd
  - k8s
  - k3s
  - runv
  - podman
  - skopeo
  - crun
  - xvisor
  - runx
  - nerdctl
  - umoci
  - sloci

- **VM Based**
  - oecore

- **Framework**
  - oe-core

- **Broken (patches accepted)**
  - kata

- **Timeline**
  - 2012
  - 2013
  - 2014
  - 2015
  - 2016
  - 2017
  - 2018
  - 2019
  - 2020
  - 2021
meta-virtualization: timeline summary

- Started with mainly VM solutions
  - now more container focused
- More solutions / choices than ever before
  - Very little has been removed / depreciated
- ‘Complexity’ has increased
  - Hence more complete solutions are possible
  - Need more system level configuration and flexibility
- Plumbing, tools and image creation assists
  - More are needed
Why the Yocto Project and containers

- Isn’t it all about the application?
  - Why care about building from source?
- Solves problems you don’t (yet) know you have
  - Evolving (think SBOM)
- Standards compliant and compatible
  - Choice: don’t pick winners and don’t lock in
  - Flexibility: Elements of the solution are spread through the ecosystem
- Configurability and tunability
What are container “build” and “deployment”? 

- For our purposes:
  - **build**: compilation / construction from source of a container (or fetching of OE built artifacts)
  - **deployment**: installing a container on a target (or image)
Container Build

- OE core has base support
- Techniques have evolved over time
  - Iterative / multiple builds and external assembly
  - Multiconfig
  - OCI Image type
- Leverage Yocto Project core values
  - Embrace, not replace or duplicate
Container Build: Challenges

- Clunky / Confusing
  - Regardless of which method you use
  - Not ‘end user’ friendly
- Path to binary container construction / reuse
  - Commonly cited guides don’t apply
- Streamlining work is in progress
Container Deployment

- Not standardized (and shouldn’t be)
  - Varies based on container runtime
- Some options:
  - Direct image install (for startup)
  - Registry / artifact repository push → pull
  - Management framework: k8s, k3s, etc
  - Custom ‘hacks’
Container Deployment: Challenges

- Not cross friendly
  - Daemons, root requirements, host requirements, licensing, reproducibility, etc
- Requirement creep
- Large set of runtimes and frameworks
Vision

- The Yocto Project as a 1st class platform for building CNCF technologies
  - Leveraging OE advantages and technologies, feeding into external solutions
    - baked directly into the outputs
- Simple inherit to generate containerized recipe output
- Direct deploy to images, or management framework
- Binary artifacts / reusable base containers
In progress / Upcoming (1/2)

- Streamlined build via bbclass
  - kernel-module-split style / dynamic packaging?
- Direct image install
  - via -native tools
  - Autostart (depending on runtime)
- Multi-layer OCI container image build
  - Pluggable / flexible back end
In progress / Upcoming (2/2)

- Reference container host and app / system container images
- Target container (on host) rapid test
- Framework test / deployment streamlining
  - k8s, k3s, runX ..
Thanks for your time
What is the Yocto Project®?

It's not an embedded Linux distribution, it creates a custom one for you.

The Yocto Project (YP) is an open source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture.

The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IoT devices, or anywhere a customized Linux OS is needed.