THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

DEBUG EVERYTHING: Building a Debuginfod Backbone for Embedded Linux at Scale

Colin Pinnell McAllister, Garmin
Joshua Pevehouse, Garmin

Colin Pinnell McAllister

- Software Engineer Team Lead at Garmin
- Involved with Embedded Linux for 8 years
- Open-source contributor
  - Meta-lts-collab maintainer
  - Contributes to Yocto & OE, swupdate, U-Boot and others
- When not in front of a keyboard, you can find me on a bike

Joshua Pevehouse

- Senior Software Engineer at Garmin
- Working in DevOps for 8 years
- Avid homelabber
  - Contributes to Immich and various FoundryVTT modules
- Usually making noise or playing tabletop games

Disclaimer

The views expressed within this presentation are those of the presenters; they do not necessarily reflect the views of Garmin

The Developer Holy Trinity

Executable	The compiled machine-readable code that runs on your product
Source Code	The human-readable code that the executable is built from
Symbols	The bridge linking human-readable source to compiled machine code

Debugging a live process or core dump demands all three

Symbols and source are essential

```
(gdb) bt
#0  0x00005618b5b3749a in ?? ()
#1  0x00005618b5b374f1 in ?? ()
#2  0x00005618b5b3757d in ?? ()
#3  0x00005618b5b37635 in ?? ()
#4  0x00005618b5b376aa in ?? ()
#5  0x00005618b5b37710 in ?? ()
#6  0x00005618b5b37767 in ?? ()
#7  0x00005618b5b377dc in ?? ()
#8  0x00005618b5b3787c in ?? ()
#9  0x00005618b5b3792e in ?? ()
#10 0x00005618b5b379c0 in ?? ()
#11 0x00007fe8cd0ecd90 in __libc_s
#12 0x00007fe8cd0ece40 in __libc_s
    at ../csu/libc-start.c:392
#13 0x00005618b5b37385 in ?? ()
(gdb)
```

Symbols and source are essential

```
(gdb) bt
#0  0x00005618b5b3749a in level_ten (ptr=0x0) at segfault_demo.cpp:20
#1  0x00005618b5b374f1 in level_nine (ptr=0x0) at segfault_demo.cpp:27
#2  0x00005618b5b3757d in level_eight (ptr=0x0) at segfault_demo.cpp:34
#3  0x00005618b5b37635 in level_seven (ptr=0x0) at segfault_demo.cpp:41
#4  0x00005618b5b376aa in level_six (ptr=0x0) at segfault_demo.cpp:48
#5  0x00005618b5b37710 in level_five (ptr=0x0) at segfault_demo.cpp:55
#6  0x00005618b5b37767 in level_four (ptr=0x0) at segfault_demo.cpp:62
#7  0x00005618b5b377dc in level_three (ptr=0x0) at segfault_demo.cpp:69
#8  0x00005618b5b3787c in level_two (ptr=0x0) at segfault_demo.cpp:76
#9  0x00005618b5b3792e in level_one (ptr=0x0) at segfault_demo.cpp:83
#10 0x00005618b5b379c0 in main () at segfault_demo.cpp:94
```

Symbols are large, source is even larger

Fits in 512MB A/B layout

Core-image-full-cmdline	Size	Increase
Stripped	165M	
With Symbols	773M	371%
With Symbols and Source	1.2G	603%

Requires 4GB for A/B layout

qemux86-64, Poky Scarthgap (b301218f4d)

The Pragmatic Compromise

- What if we just include the symbols we care about?
  - Include "-dbg" packages for in-house code on development builds
  - Exclude symbols in for stable upstream software packages
- Lower filesystem size versus including everything

The Self-contained Debug Archive

- When a service core dumps, the following are packaged and archived together...
  - Core file
  - Executable
  - Symbols
- Provides developers all the resources they need to debug a crash
- Easy for developers to use with an unlimited shelf life

Then the cracks started showing...

- Including limited symbols still made updates larger and slower to apply
- The self-contained debug archive...
  - Was slow to create and transfer
  - Hogged valuable non-vol space
- More services meant more trouble deciding which should have symbols included

What about keeping symbols off-target?

- Removed symbols on target and started using the Yocto Debugfs
  - Root filesystem with symbols for another root filesystem
  - Enabled by setting IMAGE_GEN_DEBUGFS = "1"
- CI configured to build and published Debugfs archives to a central artifact repository

Developers had a very tough time finding the right archive

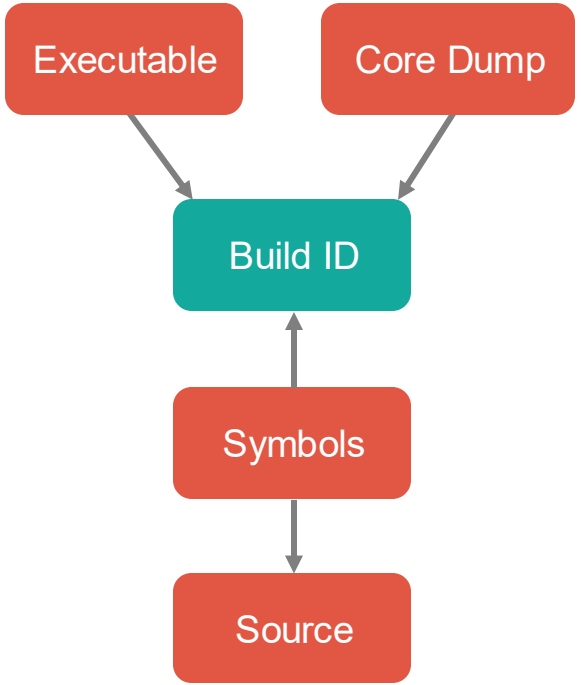
The Perceived Tradeoff

A Savior Arrives: Dorinda & Debuginfod

https://youtu.be/S3QLr113mx8

The SHA-1 Build ID

- Embedded into every binary at compile time
- Hash of the code and data sections of the binary
- All the following affects the build ID
  - Source code
  - Compiler
  - Compiler flags
  - Architecture
  - Compile-time dependencies
- Build ID survives stripping
  - Present in both the executable and symbol file after stripping



Debuginfod

- REST API Webserver that serves executables, symbols, and source code
- Server indexes artifacts within package archives
  - e.g., RPM, DEB, IPK packages
- Build ID is used as look-up key for artifacts
- Client tooling automatically handles fetching artifacts from server
  - User only needs DEBUGINFOD_URLS environment variable set to server URL(s)
  - Libdebuginfod client library exists for tools to integrate with service
  - Tools with built-in support: GDB, Valgrind, Perf, Systemtap, and more!

Comparing image size versus package size

Core-image-full-cmdline	Size
Stripped	165M
With Symbols	773M
With Symbols and Source	1.2G

Stored in flash on every device

Core-image-full-cmdline	Size
All RPMs	2G
"-doc" RPMs	18M

Stored on single volume

qemux86-64, Poky Scarthgap (b301218f4d)

Integration with Yocto

- Debuginfo distro feature already exists
  - Enables debuginfo support in Elfutils
  - Enables oe-debuginfod command to run a local debuginfod server
    - Serves local build artifacts
- Oe-debuginfod works well for a single build machine, but what about hosting artifacts for a CI system with...
  - Different build types?
  - Tens or even hundreds of products?
  - Nightly builds, feature branch builds, release candidate builds, release builds?

Can we scale Debuginfod to work on an enterprise level at an embedded software company?

Scaling Debuginfod Considerations

Expandable Storage

Time to Index

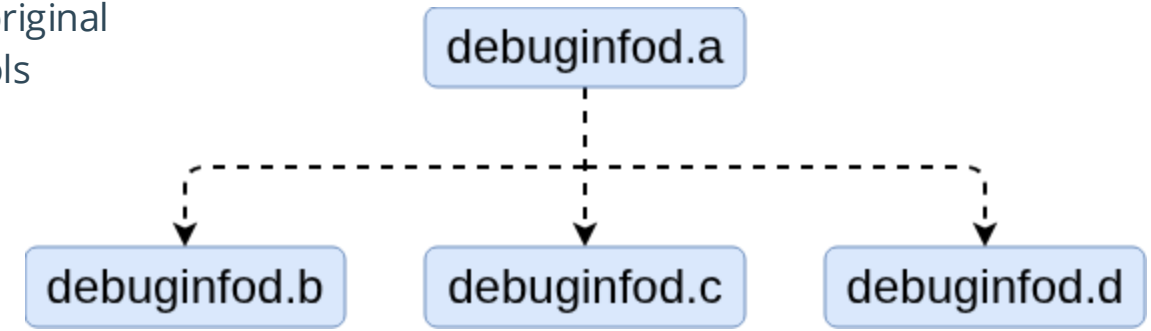
Developer Experience

Service Setup

Environment Variable	Value
DEBUGINFOD_PORT	8002
DEBUGINFOD_EXTRA_ARGS	-d /path/to/file.sqlite
DEBUGINFOD_PATHS	-U /path/to/symbols
DEBUGINFOD_CACHE_DIR	/path/to/cache
DEBUGINFOD_URLS	"http://debuginfod.feature.site.com http://debuginfod.release.site.com"

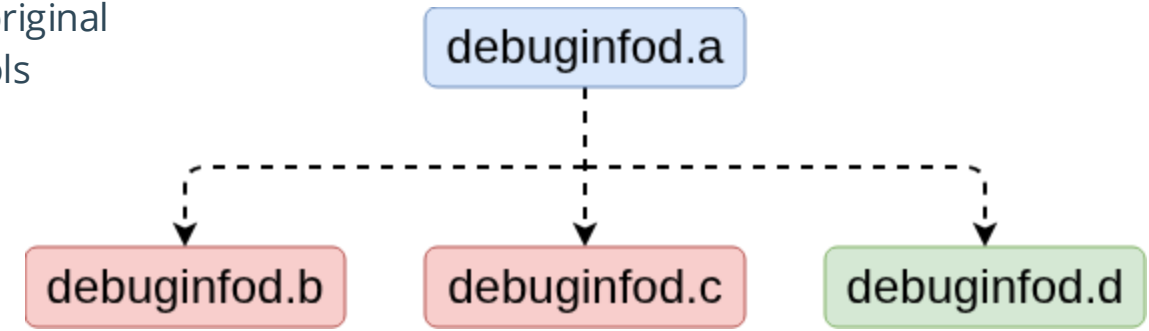
Federated Servers

- DEBUGINFOD_URLS designates upstream servers
- Upstream servers checked if original server lacks requested symbols

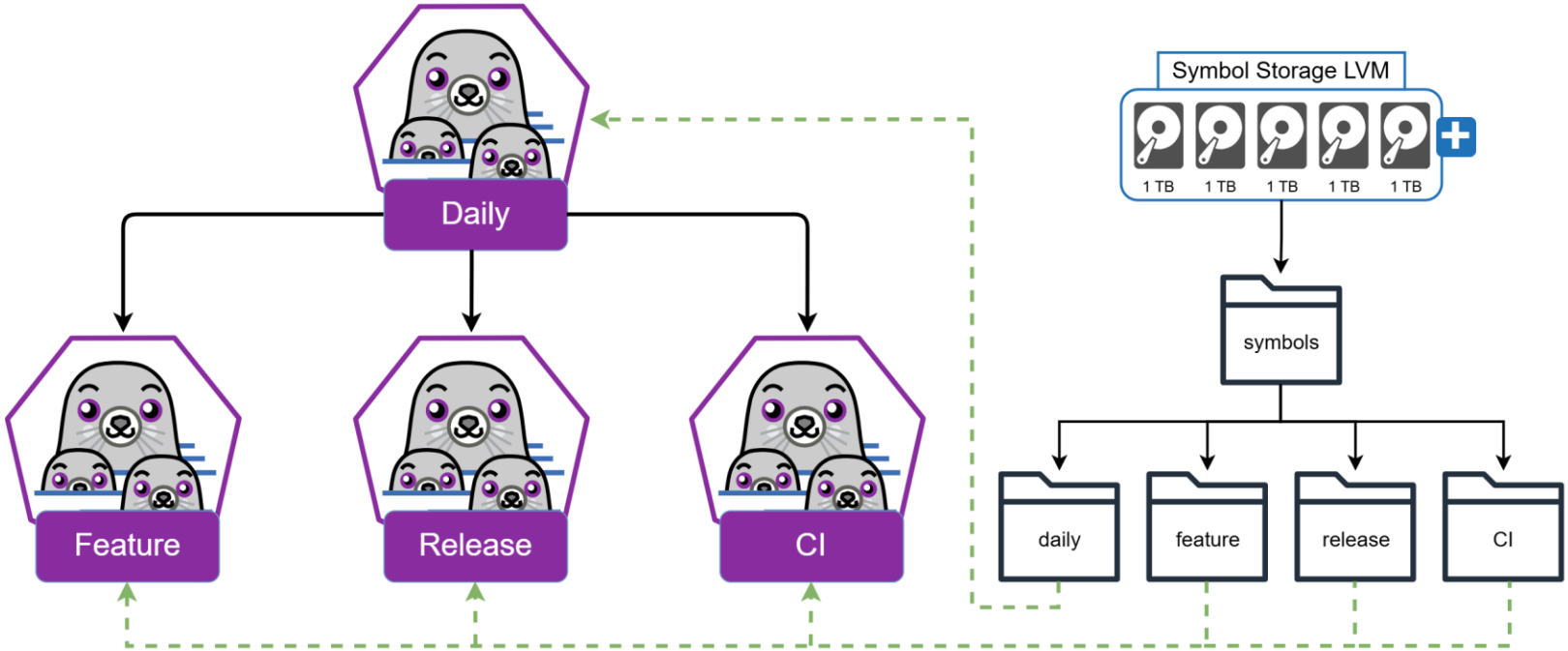


Federated Servers

- DEBUGINFOD_URLS designates upstream servers
- Upstream servers checked if original server lacks requested symbols
- Positive search results cached

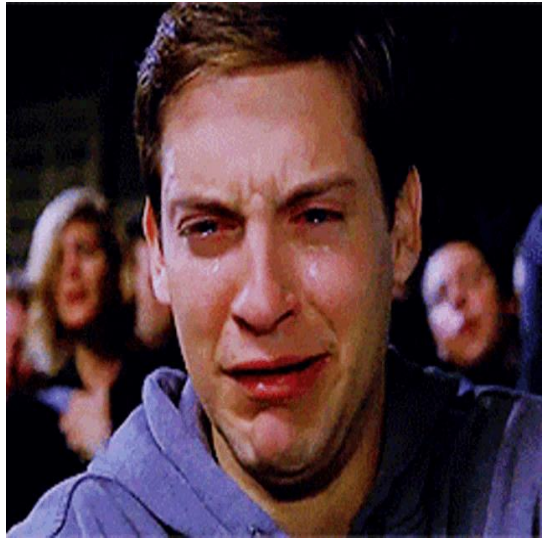


Containerize everything!



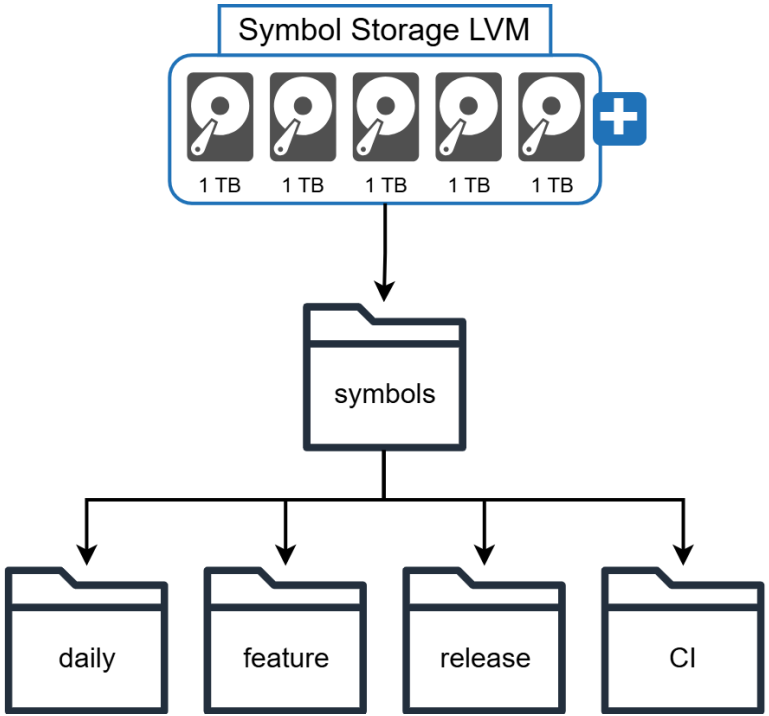
A Swing and a Miss

- Memory leaks
- Manual hardware allocation
- Cloud provider storage limitations



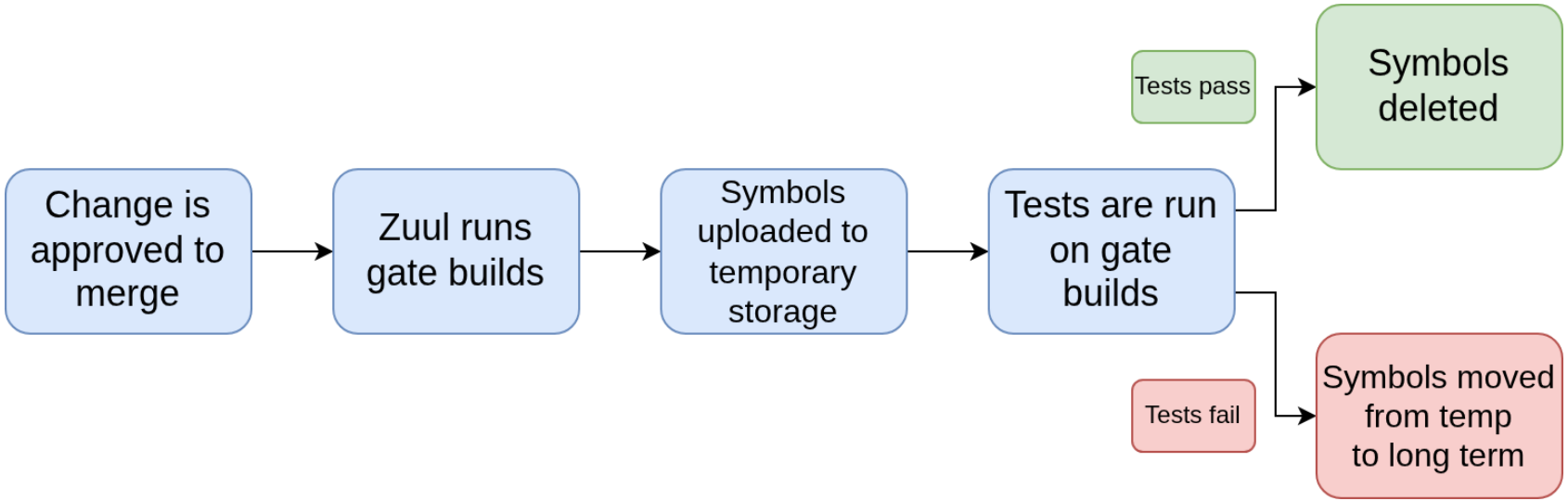
Something Worth Saving

- Using an LVM means easy expansion
  - Limited to 1TB HDDs
- Cron jobs enforce our retention policy
  - Daily builds – 3 months
  - Feature builds – 1 year
  - Release builds – forever
  - CI builds – 5 days*



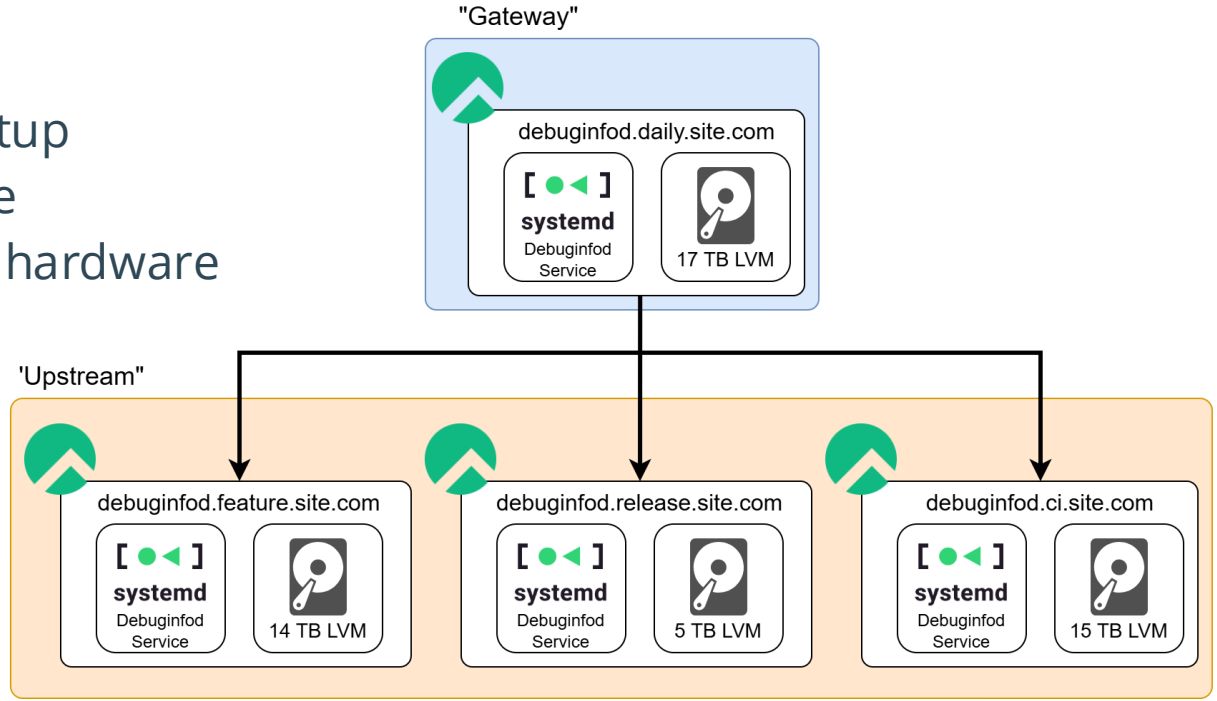
* - promotions workflow

Symbol Promotions Workflow

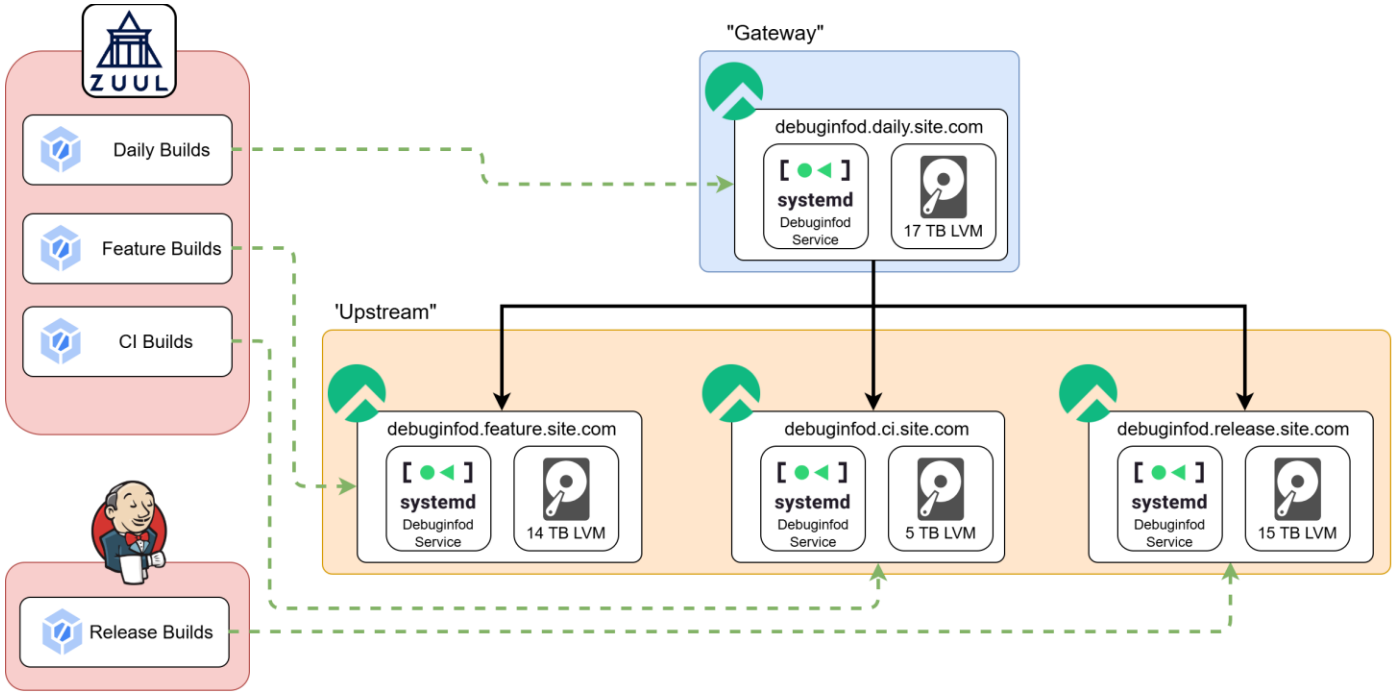


Virtual Machine Implementation

- Leverages federated setup
- LVM per virtual machine
- Service has full reign of hardware



Continuous Delivery



Simplified Developer Experience

1. Grab a core dump
2. Set the DEBUGINFOD_URLS environment variable
3. Profit

Yocto SDK Support for DEBUGINFOD_URLS

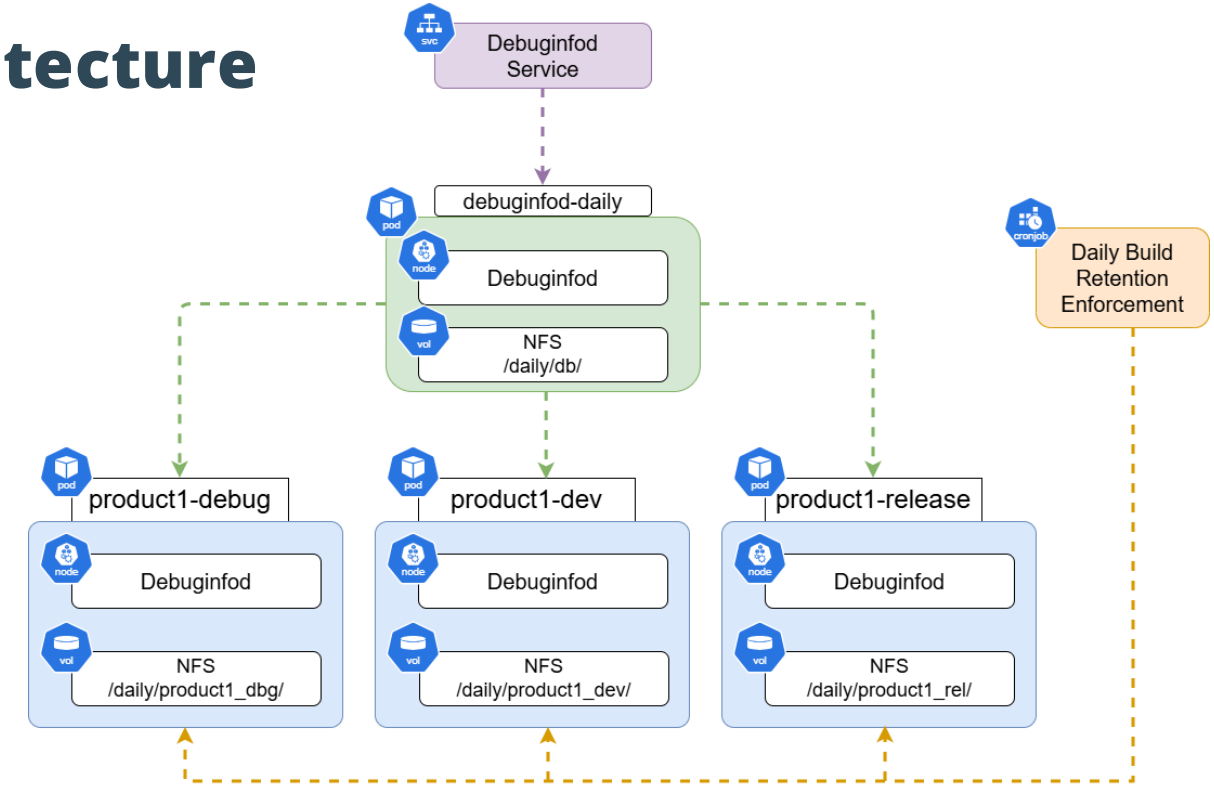
- Yocto SDKs configured with additional SDK_DEBUGINFOD_URLS variable
- SDK_DEBUGINFOD_URLS adds URLs to DEBUGINFOD_URLS in developer's environment when using the SDK
- Developers that source the SDK's activation script will be automatically configured to use the specified Debuginfod server(s)
- Currently exploring upstreaming this

Deduplicating artifacts

- Lots of duplicate packages currently exist on our server
- Difficult to distinguish which packages are identical
  - Yocto package naming convention is not as unique as build IDs
  - Changes can affect build ID, but not package name
    - Appling CVE patches, modifying packageconfig options or other compile flags, etc.
- Currently developing tooling to find and prune duplicate packages

Kubernetes Architecture

- Pod with no symbols as entry point for build type
- Pod per mode per product for indexing
- Cron jobs for retention enforcement
- NFS file storage



Questions?

@colin-pm

@colinpmcallister

@joshpevehouse