Managing build infrastructure of a Debian derivative

Andrej Shadura
4 February 2018
Presentation Outline

Who am I

Enter Apertis

Build infrastructure

Packaging workflows

Image builds

Open First
Andrej Shadura

- contributing to Debian since 2007
- Debian Developer since 2013
- working for Collabora since 2015
Andrej Shadura

- contributing to Debian since 2007
- Debian Developer since 2013
- working for Collabora since 2015
- doing packaging since 2010
- never ran any ‘real’ Debian infrastructure
- only used mini-dinstall to publish packages
Presentation Outline

Who am I

Enter Apertis

Build infrastructure

Packaging workflows

Image builds

Open First
Apertis

- Debian derivative tailored for automotive needs
- Originally developed for infotainment systems
- Fit for a wide variety of electronic devices
Apertis

- code hosting
- code review tools
- package build services
- image generation services
- automated testing infrastructure
Apertis

- based mostly on Ubuntu
- takes several packages directly from Debian
- provides its own software packages, frameworks, APIs
Apertis

- systemd for application lifecycle tracking
- AppArmor for policy enforcement
- Wayland for graphics
- Gstreamer for multimedia playback
Apertis + OSTree + Flatpak = ♥

- Apertis app bundle format based on Flatpak
- original system updater used btrfs to ensure atomicity and enable recovery mode
- OSTree-based solution replaces an older btrfs-based updater
Apertis + OSTree + Flatpak = ❤️

Drawbacks of the btrfs updater:

- significant maintenance effort
- bootloader doesn’t support btrfs
  - can’t use on /boot
- requires initramfs
- directly manipulates btrfs volumes
  - unsuitable for e.g. unprivileged containers
- testing is difficult
- btrfs itself has a lot of issues
Apertis + OSTree + Flatpak = ♥

OSTree:
- works with any filesystem
- stores multiple trees in the same repo
- no need for extra partitions for safe upgrades — saves space
- less custom code
- works better with containers
- a full solution on its own, not just a building block
Why Ubuntu+Debian, not just Debian?

- the Universal operating system
- composed entirely of free software
- developed by a community of individuals
Why Ubuntu+Debian, not just Debian?

- stable moves too slowly, changes between releases are quite significant
- unstable breaks things a bit way too often
- until recently, releases had unpredictable timing, there wasn’t an LTS release
Why Ubuntu+Debian, not just Debian?

Advantages:

▶ large install base, so despite more frequent releases there’s still a lot of testing
▶ Ubuntu is an upstream for AppArmor, on which we rely
▶ regularly timed releases — and LTS ones too
Why Ubuntu+Debian, not just Debian?

Downsides:
- being derivative of a derivative complicates upgrades/rebases
- we don’t *really* need all patches Ubuntu applies (Mir, anyone?)
Presentation Outline

Who am I

Enter Apertis

Build infrastructure

Packaging workflows

Image builds

Open First
Overview of the Apertis infrastructure

**SOURCES**
- Debian maintained software (Debian source packages)
- Frequently updated FOSS software (Git)
- Customer specific software (Git)
- PSD vendor software (binary only)

**Code Hosting/Review**
- Automatic import

**Open Build Service (OBS)**
- Centralized handling of software compilation
- Granular Access Control
- High-perf compilation option
- Manages software dependencies

**BINARIES**
- Debian Package repositories (FOSS + PSD + Project specific)
- Image Creator Apertis optimized YAML based

**IMAGES/RFS**
- System images (VMs, SD cards, eMMC, …)
- RFS only (ready for custom flashing)

**TARGETS**
- Evaluation boards
- Project boards
- Virtual Machines (VirtualBox)

**UPDATES/OTA**
- New image / RFS
- Package-based during dev.

**Phabricator**
Project Management
Open Build Service

- builds packages in a fresh chroot every time
- all source packages are revision controlled
- provides fine-grained access control
- provides Subversion-style branching and merge review system

For more information on OBS, please watch a talk by Andrew Lee at FOSDEM 2018: https://col.la/fosdem18obs
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apertis:18.03:target</td>
<td>Apertis target</td>
</tr>
<tr>
<td>apertis:18.03:development</td>
<td>Apertis development</td>
</tr>
<tr>
<td>apertis:18.03:helper-libs</td>
<td>Apertis helper libraries</td>
</tr>
<tr>
<td>apertis:18.03:hmi</td>
<td>Apertis human machine interface packages</td>
</tr>
<tr>
<td>apertis:18.03:/sdk</td>
<td>Apertis SDK</td>
</tr>
<tr>
<td>apertis:18.03:snapshots</td>
<td>Apertis git snapshots</td>
</tr>
</tbody>
</table>

Supported architectures: amd64, armhf, arm64.
## OBS: components split

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apertis:18.03:target</td>
<td>Apertis target</td>
</tr>
<tr>
<td>apertis:18.03:development</td>
<td>Apertis development</td>
</tr>
<tr>
<td>apertis:18.03:helper-libs</td>
<td>Apertis helper libraries</td>
</tr>
<tr>
<td>apertis:18.03:hmi</td>
<td>Apertis human machine interface packages</td>
</tr>
<tr>
<td>apertis:18.03:sdk</td>
<td>Apertis SDK</td>
</tr>
<tr>
<td>apertis:18.03:Snapshots</td>
<td>Apertis git snapshots</td>
</tr>
</tbody>
</table>

Supported architectures: amd64, armhf, arm64.
OBS: how the packages are published

- OBS maintains internal package repositories, usually one per project
- Internal repositories aren’t in APT format, so `reprepro` is used to make packages available to APT
- It’s easy to do full rebuilds using more than one repository per project:
  - Add one more repository per project, rebuild
  - Depends on the main repository, non-publishing
OBS: how the packages are published

Repositories of `apertis:18.03:target`

You can configure individual flags for this project here.

```
18.03 (armv7hl, aarch64, x86_64)
apertis:18.03:development/18.03
```

**Edit 18.03**

**Architectures:**
- □ armv7l
- □ i586
- ✅ x86_64
- ✅ armv7hl
- □ armv6hl
- ✅ aarch64

**Additional package repositories (searched in descending order):**
```
apertis:18.03:development/18.03
```

- Add additional path to this repository
- Update 18.03
- Delete repository

**rebuild**
```
(armv7hl, aarch64, x86_64)
apertis:18.03:development/18.03
```

- Edit repository
- Delete repository
- Add repositories
Packaging workflows: Ubuntu/Debian packages

- packages with no changes are imported unchanged directly to OBS
- packages with minimal changes are maintained in OBS
  - mostly custom DEP-3 patches applied on top of the sources from Ubuntu
  - local changes get co1 version suffixes
- a fork of Ubuntu’s Merge-o-Matic called Merge-our-Misc, is used to pull new updates from Ubuntu LTS
- select packages are kept in Git
Packaging workflows: Git

Non-Apertis packages:

- **DEP-14:**
  - upstream/{latest,$version} branches with the upstream project code
  - apertis/{master,$distro} branches for Apertis packaging
  - apertis/$version tags (eg. apertis/2.48.0-1ubuntu4co1)

- **git-buildpackage and gbp-pq** to manage patches.
Native Apertis packages:

- **master** branch contains the sources for the current development release
- packaging metadata is also kept in master, not on a separate branch
- **$distro** branches (for instance, 17.12) contain the sources for past distro releases
  - $version tags for ‘upstream’ source releases (eg. 0.1803.1)
  - apertis/$version tags for packaging releases (eg. apertis/0.1803.1co1)
Packaging workflows: Git

Apertis Jenkins instance:

- builds every new commit in a controlled environment
- if the build succeeds, the source package is submitted to OBS for a clean rebuild in the :snapshots component
- for commits tagged as releases, Jenkins creates merge requests for the main component (:target, :development etc).
Packaging workflows: Git

- The build-snapshot script by Simon McVittie is used by Jenkins to build packages and create source packages to be uploaded to OBS.
- New patches submitted for review at Phabricator get built on top of the branch they apply to.
Image builds

- Jenkins
- Linaro image tools
- multi-stage process
- separation of hardware-dependent and hardware-independent components

Open First
Image builds

- Jenkins
- Linaro image tools: debos!
- Multi-stage process
- Separation of hardware-dependent and hardware-independent components
<table>
<thead>
<tr>
<th>ospack</th>
<th>hwpack</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>target-armhf</td>
<td>mx6q</td>
<td>target-mx6q</td>
</tr>
<tr>
<td>development-armhf</td>
<td>rpi</td>
<td>development-mx6q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>target-rpi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>development-rpi</td>
</tr>
</tbody>
</table>
Image builds

<table>
<thead>
<tr>
<th>ospack</th>
<th>hwpack</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>target-armhf</td>
<td>mx6q</td>
<td>target-mx6q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>target-rpi</td>
</tr>
<tr>
<td>development-armhf</td>
<td>rpi</td>
<td>development-mx6q</td>
</tr>
<tr>
<td></td>
<td></td>
<td>development-rpi</td>
</tr>
</tbody>
</table>
debos: Debian OS builder

- action: debootstrap
  suite: "stretch"
  components:
    - main
  mirror: https://deb.debian.org/debian
  variant: minbase
  merged-usr: true
debos: Debian OS builder

{{- if eq $type "target" "development" "sdk" }}
  - action: apt
    description: "Target packages"
    packages:
      - apertis-target
      - apertis-hmi
      - libegl1-mesa-drivers
{{- end -}}
debos: Debian OS builder

# add firmware
- action: overlay
  origin: firmware
  source: firmware-{{ $firmware_version }}/boot
  destination: /boot/firmware

https://github.com/go-debos/debos/
Sjoerd Simons <sjoerd@collabora.com>
Denis Pynkin <d4s@collabora.com>
Image builds

- build ospacks
- combine ospacks and h/w-specific packages and data into images
- generate sysroots for the SDK
- trigger tests on LAVA
- scan the package changelog and close bugs fixed in the packages
Image builds

Amd64 development - 13s

- General SCM
- docker pull docker-registry.apertis.org/apertis/apertis-18.03-image-builder
  - Shell Script
  - ${BUILD_DATE_FORMATTED."yyyMMdd"} ${BUILDS_TODAY_Z} - Determine
- Shell Script
- Shell Script
- echo docker:x${id -u}:${id -g}:docker gcos:/tmp/bin/false > ${NSS_WRAPPER}
  - Shell Script
- Shell Script
- sysroot/18.03/sysroot-apertis-18.03-amd64 - Write file to workspace
- echo docker:x${id -u}:${id -g}:docker gcos:/tmp/bin/false > ${NSS_WRAPPER}
  - Shell Script
- LD_PRELOAD=libnss_wrappers.so rsync -e "ssh -oStrictHostKeyChecking=no"
Challenges

- MoM can handle simple package merges, fails on conflicts
- Git workflows help, but you can’t put a whole distro in Git
- Removing old and obsolete packages
- OBS isn’t sbuild, have to deal with occasional FTBFS
- OBS ignores Essential, needs manual overrides
Future plans

- use GitLab for Git hosting
- focus shift to become a common platform for automotive systems, not just infotainment
Interested?

See also

https://apertis.org
https://collabora.co.uk
Managing build infrastructure of a Debian derivative

Thanks!

Q & A