Creating Debian-Based Embedded Systems in the Cloud using Debos

Christopher Obbard

email: chris.obbard@collabora.com

twitter: @obbardc

About me

- Engineer at Collabora
- Working on...
  - Custom distros for cloud, embedded and PC
  - Continuous integration
  - Packaging
  - OTA upgrades
  - Tooling to make life easier
  - Learning Rust!

Overview

- Why use Debian as a base?
- Internal design decisions
- How to use Debos (warning: YAML)
- Future plans
- Q&A

What is a GNU/Linux distro?

- A collection of software packages
- But also a collection of like-minded developers
- Each distribution has different common goals
- Some goals may be financial, others social
- Debian/Ubuntu uses dpkg/apt
- Red Hat/Fedora uses rpm/yum
- …everyone has their own preference

Why create your own distro?

- Hardware dev kits are supplied with general-purpose distro for evaluatation
- Cloud images
- Lots of bloat, outdated/insecure packages, incompatibilities
- Your own distro would be nice
- A distro is a lot of work to maintain!
- No need to reinvent the wheel: base on a proven technology...

Yocto & Buildroot

- Usually for only embedded platforms
- Creates totally custom distribution, can become a maintainance nightmare
- All packages are compiled on your machine
- High learning curve
- Why make things hard…

Why use Debian as a base?

- Traditionally seen as a desktop OS… recent years effort has gone into enabling embedded targets
- Released in 1993, widely used (DistroWatch top 10)
- 1,000s of volunteers shape Debian, all following the DFSG & social contract
- 51,000 popular packages & libraries (apt!)
- Great community, lots of tutorials, easy to get started
- Stable, testing and unstable (bleeding edge)
- Timely security updates
- No one company leads the development/direction
- Allows you to concentrate on the most important part: your application!

Debian releases

- *stable*, *testing* and *unstable*
- Bleeding-edge software is packaged into *unstable*
- Trickles into *testing*: usually ~2 weeks after upload so long as no major bugs are reported
- Most devs run *unstable*: essentialy a QA staging area for *testing*
- *unstable* doesn't mean buggy: usually means things can change without warning
- *stable* is frozen for two years: usually only security updates and minor releases of packages are included
- Recommend using *testing*, unless brave

Debian disadvantages

- Only cater for systemd (changing!) and glibc
- Designed with desktop/server use in mind
- Can be coservative of very new technologies
- Limited enterprise support
- Slow release cycle (not always bad!)

How to create custom Debian image

- Create an image (*dd*)
- Insert a partition table (*fdisk*)
- Format partitions (*mkfs.*****)
- Mount partitions in a loop device (*kpartx*)
- Chroot into the mounted image
  - create basic Debian filesystem (*debootstrap*)
  - install custom packages (*apt*)
  - set hostname, user accounts, configuration…
- Unmount image, cleanup loop devices
- Compress your image & save build logs
- Nice, until the fragile thing breaks, works on my machine…

Why not use x tool?

- Lots of other tools already out there…
- The many methods to build a Debian image by Riku Voipio summarises the most popular tools
- Other tools serve a very specific purpose
- Debos inherntly more flexible and robust against random failures
- Debos can generate a distro from one configuration file which can be stored in version control
- Debos is constantly evolving and improved by Collabora and Apertis (automotive Debian derivative)
- Get started with Debos quicker!

The solution: Debos!

- Runs under a VM on your machine (fakemachine)
- Disks are attached to the VM (no more loop devices)
- Recipe contains actions: steps to create your image
- Recipe is translated into commands which are ran inside the VM
- Actions abstract file changes & commands
- Where there is no action: run a shell cmd/script
- Easy cleanup even if things break: kill the VM
- Reproducable on your PC as well as the cloud

Who’s using Debos?

- Apertis is a Debian-based GNU/LInux platform tailored for automotive and consumer needs; uses debos to generate reference images for multiple platforms
- KernelCI, a Linux Foundation project, uses debos to generate Debian-based root filesystems for Continuous Integration of the Linux Kernel
- Radxa uses debos to generate reference images for their Rockchip-PX30 based board called the ROCK Pro PX30
- Mobian Project - Debian for Mobiles a project by Arnaud Ferraris uses debos to generate Debian images for PinePhone, PineTab and Librem 5
- Plasma Mobile use Debos to generate their Neon reference images
- Gemian: Debian for the Gemian PDA/Cosmo Communicator use debos to generate images
- Reproducible Builds use debos to make sure Debian packages can be independently verified

What is Debos?

- Core is written in Golang
  - No need to know Go, only to patch the core
  - Similar enough to C, low barrier of entry for most
- Fakemachine seperate library/tool handles VM
- Packages are in Debian stable (amd64 host)
- Docker container
- Install from source on other OS

Debos recipe

- YAML file defines the steps to create your image
- YAML is simple & can be version controlled
- Consists of:
  - header containing metadata (image architecture)
  - multiple actions which are ran sequentially, each having their own properties
- Comments prefixed with #
- Pre-processed through the Go templating engine
- Variables can be passed from the cmdline
- Basic scripting: if/else statements
- Recipes can include other recipes

Example: simple-ospack.yml

```yaml
# This recipe creates a tarball of a Debian system
architecture: amd64
actions:
  - action: debootstrap
    suite: testing
    components:
      - main
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: apt
    packages:
      - linux-image-amd64

  - action: run
    chroot: true
    command: echo simple-ospack > /etc/hostname

  - action: pack
    file: simple-ospack.tar.gz
    compression: gz
```

Example: simple-ospack.yml

```
$ apt install --yes docker
$ docker run --rm --interactive --tty \
    --device /dev/kvm \
    --user $(id -u) \
    --mount "type=bind,source=$(pwd),destination=/recipes" \
    --workdir /recipes
    --security-opt label=disable \
    godebos/debos simple-ospack.yaml

2020/10/09 11:12:04 ==== debootstrap ====
2020/10/09 11:12:05 Debootstrap | ...output removed...
2020/10/09 11:13:59 ==== apt ====
2020/10/09 11:13:59 apt | ...output removed...
2020/10/09 11:15:10 ==== run ====
2020/10/09 11:15:10 ==== pack ====
2020/10/09 11:15:10 Compressing to simple-ospack.tar.gz
Powering off.
2020/10/09 11:16:06 ==== Recipe done ====

$ ls
simple-ospack.tar.gz
```

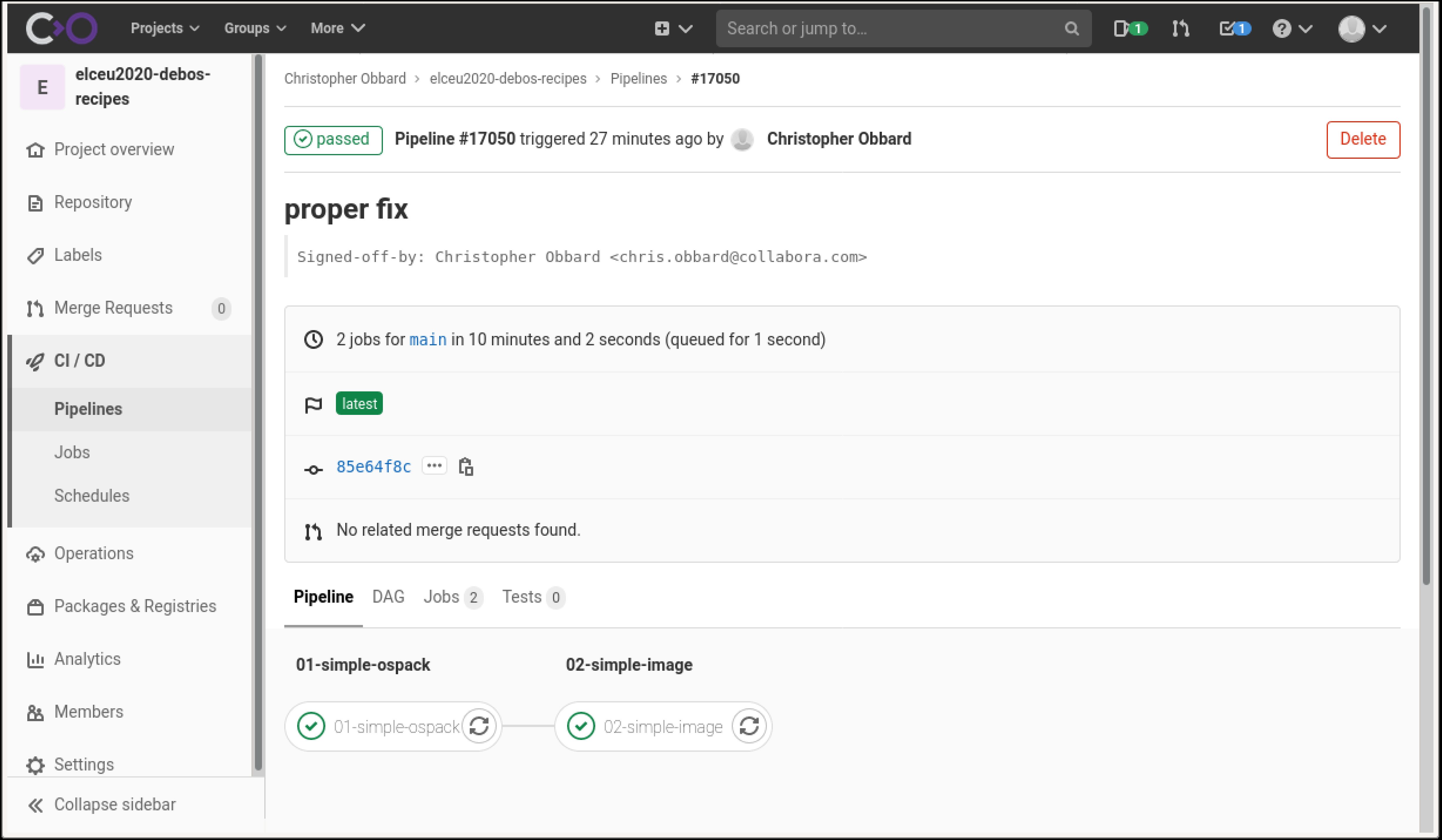
Example: simple-ospack.yml

```
architecture: amd64
actions:
  - action: debootstrap
    suite: testing
    components:
      - main
    mirror: https://deb...
    variant: minbase

  - action: apt
    packages:
      - linux-image-amd64

  - action: run
    chroot: true
    command: echo simple-ospack > ...

  - action: pack
    file: simple-ospack.tar.gz
    compression: gz
```

```
2020/10/09 11:12:04 ==== debootstrap ====
2020/10/09 11:12:05 Debootstrap | ...removed...
2020/10/09 11:13:59 ==== apt ====
2020/10/09 11:13:59 apt | ...removed...
2020/10/09 11:15:10 ==== run ====
2020/10/09 11:15:10 ==== pack ====
2020/10/09 11:15:10 Compressing to ospack.tar.gz
Powering off.
2020/10/09 11:16:06 ==== Recipe done ====
```

GitLab CI

```
stages:
  - simple-ospack

simple-ospack:
  stage: simple-ospack
  tags:
    - kvm
  image:
    name: godebos/debos:latest
    entrypoint: [ "" ]
  script:
    - debos simple-ospack.yml
  artifacts:
    expire_in: 4 weeks
    paths:
      - simple-ospack/out
```



elceu2020-debos-recipes

Christopher Obbard › elceu2020-debos-recipes › Pipelines › #17050

passed **Pipeline #17050** triggered 27 minutes ago by **Christopher Obbard**

proper fix

```
Signed-off-by: Christopher Obbard <chris.obbard@collabora.com>
```

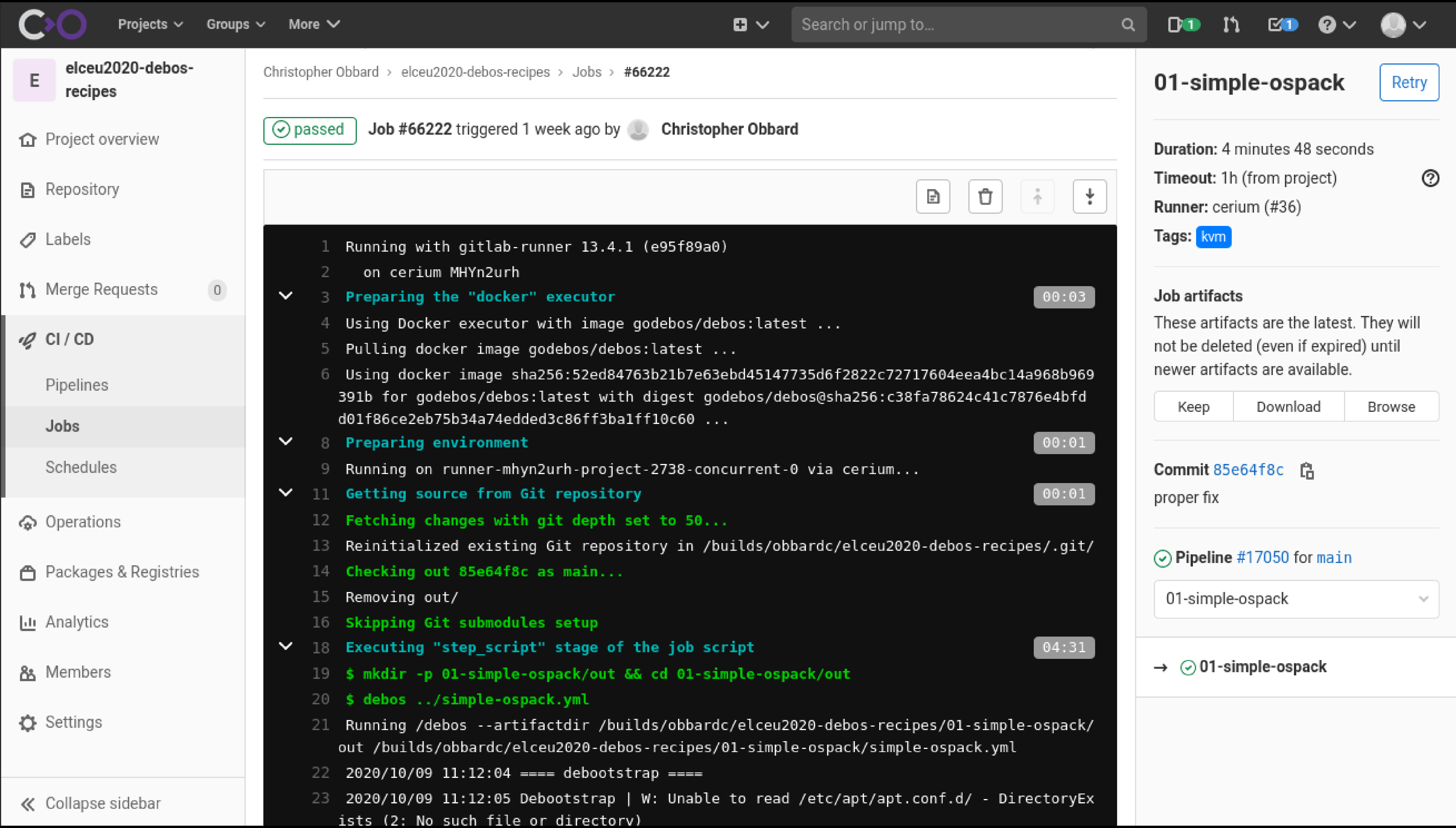
2 jobs for `main` in 10 minutes and 2 seconds (queued for 1 second)

latest

`85e64f8c`

No related merge requests found.

01-simple-ospack **02-simple-image**



Job #66222 triggered 1 week ago by Christopher Obbard

```
Running with gitlab-runner 13.4.1 (e95f89a0)
  on cerium MHYn2urh
Preparing the "docker" executor                                            00:03
Using Docker executor with image godebos/debos:latest ...
Pulling docker image godebos/debos:latest ...
Using docker image sha256:52ed84763b21b7e63ebd45147735d6f2822c72717604eea4bc14a968b969
391b for godebos/debos:latest with digest godebos/debos@sha256:c38fa78624c41c7876e4bfd
d01f86ce2eb75b34a74edded3c86ff3ba1ff10c60 ...
Preparing environment                                                      00:01
Running on runner-mhyn2urh-project-2738-concurrent-0 via cerium...
Getting source from Git repository                                         00:01
Fetching changes with git depth set to 50...
Reinitialized existing Git repository in /builds/obbardc/elceu2020-debos-recipes/.git/
Checking out 85e64f8c as main...
Removing out/
Skipping Git submodules setup
Executing "step_script" stage of the job script                           04:31
$ mkdir -p 01-simple-ospack/out && cd 01-simple-ospack/out
$ debos ../simple-ospack.yml
Running /debos --artifactdir /builds/obbardc/elceu2020-debos-recipes/01-simple-ospack/
out /builds/obbardc/elceu2020-debos-recipes/01-simple-ospack/simple-ospack.yml
2020/10/09 11:12:04 ==== debootstrap ====
2020/10/09 11:12:05 Debootstrap | W: Unable to read /etc/apt/apt.conf.d/ - DirectoryEx
ists (2: No such file or directory)
```

01-simple-ospack

Duration: 4 minutes 48 seconds
Timeout: 1h (from project)
Runner: cerium (#36)
Tags: kvm

Job artifacts
These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Commit 85e64f8c
proper fix

Pipeline #17050 for main

01-simple-ospack

→ 01-simple-ospack

Action: debootstrap

```
- action: debootstrap
  mirror: https://deb.debian.org/debian   # ubuntu: http://archive.ubuntu.com/ubuntu/
  suite: testing                  # e.g: stable, unstable, bullseye, sid, xenial...
  components:
    - main
    - contrib
    - non-free
  variant: minbase        # optional; minbase|buildd|fakechroot
```

- Sets up a basic Debian system in the target filesystem
- Mirror allows you to choose where packages come from
- /etc/sources.list is created
- Variant:
  - omit for a “full” Debian system
  - minbase (recommended) includes essential packages and apt

Action: apt

```
- action: apt
  recommends: false          # optional; default is false
  unauthenticated: false     # optional; default is false
  packages:
    - package1
    - package2
```

- Installs packages (and their dependencies) into the target filesystem
- “Recommends” pulls in packages which are not strictly required for a working minimal system (e.g. fonts for LibreOffice or ffmpeg codecs for Firefox)
- Under the hood just calls apt to install packages: handles dependencies the same

Action: pack/unpack

```
- action: pack
  file: filename.ext
  compression: gz              # optional; default is gz

- action: unpack
  file: file.ext
  compression: gz              # optional; default is gz
```

- Pack compresses the complete target filesystem to a tarball
- Unpack uncompresses a filesystem tarball into the target
- Useful for targets which have multiple image types
  - common usage is to create an “ospack” then from that multiple images for different targets
- Only tar.gz compression is supported currently

Action: image-partition

```
- action: image-partition
  imagename: "test.img"
  imagesize: 4G
  partitiontype: gpt   # or msdos
  partitions:
    - name: EFI
      parttype: C12A7328-F81F-11D2-…
      fs: fat32
      start: 6176s
      end: 256M
      flags: [ boot ]
    - name: root
      fs: ext4
      start: 256M
      end: 100%
  mountpoints:         # optional
    - partition: root
      mountpoint: /
    - partition: EFI
      mountpoint: /boot/efi
      options: [ x-systemd.automount ]
```

- Creates an image & partition table
- Formats filesystems (btrfs, f2fs)
- Attaches image to VM
- Mounts filesystems inside VM
- May only used once per recipe!
- Uses parted, mkfs…, fdisk, etc under the hood

Action: filesystem-deploy

```
- action: filesystem-deploy
  setup-fstab: bool                   # optional; default is true
  setup-kernel-cmdline: bool          # optional; default is true
  append-kernel-cmdline: arguments    # optional
```

- By default the root filesystem is not stored on the image
- This action copies the root filesystem to the image
- Subsequent actions are executed on the mounted image
- Can create /etc/fstab from image-partition action (from block UUID)
- Can create /etc/kernel/cmdline (parameters passed to bootloader from kernel-install script)

Action: overlay

```
- action: overlay
  source: directory
  destination: directory       # optional; default is /
```

- Recurisvely copies a directory into the target filesystem
- Source is relative to the recipe file
- Preserves permissions

Action: raw

```
- action: raw
  source: filename
  offset: bytes                 # optional; default is 0
  partition: partition name     # optional; if omitted writes to image
```

- Writes an image to a partition or the image itself
- Useful for:
  - installing bootloader to an image
  - copying pre-prepared images to a partition

Action: run

```
- action: run
  chroot: bool                # default is false
  postprocess: bool           # default is false
  command: command line
  script: script argument1 argument2
```

- Allows scripts **or** commands to be ran inside the VM
- Can be run inside the chroot
- Can run after the VM has been shutdown (postprocess)
- Scripts must be executable & relative to recipe (version control!)
- Presumes failure if exit code is not 0

Variables

```
{{ $architecture := or .architecture "arm64" }}
{{ $suite := or .suite "buster" }}
{{ $image := or .image (printf "debian-%s-%s.tgz" $suite $architecture) }}

architecture: {{ $architecture }}
actions:
  - action: debootstrap
    suite: {{ $suite }}
    components:
      - main
      - contrib
      - non-free
    mirror: https://deb.debian.org/debian
    variant: minbase
  - action: pack
    file: {{ $image }}
    compression: gz


$ debos -t architecture:armhf -t suite:sid test-variables.yaml
```

If/else statements

```
{{ $architecture := or .architecture "arm64" }}

architecture: {{ $architecture }}
actions:
  - action: apt
    packages:
{{ if eq $architecture "amd64" }}
        - linux-kernel-arm64
        - some-package-for-arm64
{{ else }}
        - linux-kernel-armhf
        - some-other-package-for-armhf
{{ end }}

$ debos -t architecture:armhf test-if-else.yaml
```

Action: recipe

```
- action: recipe
  recipe: path
  variables:
    key: value
```

- Include a recipe inside another recipe
- Abstract reusable things somewhere else
- Recipe must run standalone
- Variables from cmdline are passed along with extra defined variables
- Architecture must be the same (but the parent arch is passed)
- Components (e.g LibreOffice or Firefox recipe)

More examples!

- Debian
  - basic example: a good starting point!
  - Raspberry Pi 3/4 arm64 image
- Apertis
  - more scripting/if statements
  - ospack
  - Raspberry Pi 3/4 arm64 image

Future plans

- Documentation & getting more people using it!
- Q4-2020:
  - Automated testing
  - UML support (build images on GitHub without KVM)
  - More useful actions (e.g. install deb package)
- Q1-2021:
  - Support for Arch
  - More examples & documentation
  - Release v1.1.0?
- Fix all the bugs!

Thank you & questions!

```
- type: message
  priority: high
  body: Collabora is hiring...
  recipient: you
  calltoaction: https://col.la/join

- type: message
  priority: medium
  body: Ask questions!
  recipient: you
  calltoaction: The chatbox
```