COLLABORA

Cooking a Debian System: One, Two, Debos!

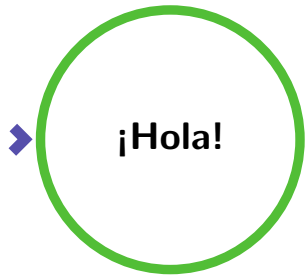
Ana Guerrero López

Collabora

October 22, 2018

COLLABORA

Who I am



- Free software user and enthusiast since 2001
- Debian Developer since 2006
- working at Collabora since earlier this year

COLLABORA

Presentation Outline

So ... What is debos?

- A tool written in Go to create Debian images. It also works with Debian derivatives
- It creates the images following sequentially the steps provide in a recipe file
- Every step must be an action known by debos, there are 12 different actions
- You don't need to be root to build the images

So ... What is debos?

- Tool designed to be easily integrated in CI systems
- Modular: new actions can be implemented easily
- Recipe is a YAML file which is pre-processed though Go's text templating engine
- It can be run in non-Debian systems with a docker container.

Architectures supported by debos

- Potentially can support every architecture that's both supported by Qemu and available in Debian
- Tested and working on: armhf, armel, arm64, i386, amd64, mips, mipsel, mips64el (debian ports of x86, ARM, MIPS)
- ... and riscv64 (`https://wiki.debian.org/RISC-V`)

Below debos: fakemachine and Qemu

- debos uses a library named fakemachine to create and spawn virtual machines for building images with debos
- fakemachine setups qemu-system using the `/usr` from the host system to run a virtual machine with the same architecture as the host
- This allows debos to work with root privileges inside this virtual machine
- debos setups qemu-user emulation to run binaries from foreign architectures
- For fakemachine to work, make sure your user has permission to use `/dev/kvm`

What is not debos

- It's not a build system!
- It's not the official way of installing Debian. It's debian-installer.

Other tools to create Debian images

- There are a bunch of other tools to create Debian images, too many to do a comparison!
- Check the Debian wiki if you're curious:
  https://wiki.debian.org/SystemBuildTools

Who is using debos? - Apertis

- Apertis is an Open Source infrastructure tailored to the automotive needs and fit for a wide variety of electronic devices.
- Amongst other things, Apertis is a Ubuntu/Debian-derived distribution with its own repositories
- debos is used to create the different images created by the project.
- `https://images.apertis.org/`

Who is using debos? - KernelCI.org

- KernelCI.org is a community based, open source distributed test automation system focused on upstream Linux kernel development.
- KernelCI builds many kernel trees automatically, boots them on a wide array of devices and runs test plans for a few subsystems.
- debos is used for creating the rootfs images used by the test plans.

Who is using debos? - other projects

- https://github.com/VitroTech/Vitrobian/ for the Vitro Crystal board
- https://github.com/ant9000/acmesystems-image-builder for the AcmeSystems boards
- https://gitlab.com/debian-pm/tools/rootfs-builder-debos Tools for maintaining the debian-pm packages

COLLABORA

Presentation Outline

A small debos recipe

```yaml
architecture: armhf

actions:
  - action: debootstrap
    suite: "stretch"
    components:
      - main
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: pack
    file: rootfs.tar.gz
    compression: gz
```

This should be saved as `small-recipe.yaml`

Running our small debos recipe

```
$ debos small-recipe.yaml
Running /debos --artifactdir /home/ana/code /home/ana/code/small-recipe.yaml
2018/09/28 11:23:29 ==== debootstrap ====
2018/09/28 11:23:29 Debootstrap | W: Unable to read /etc/apt/apt.conf.d/ - DirectoryExists (2: No such
2018/09/28 11:23:29 Debootstrap | I: Retrieving InRelease
2018/09/28 11:23:32 Debootstrap | I: Retrieving Release
2018/09/28 11:23:33 Debootstrap | I: Retrieving Release.gpg
2018/09/28 11:23:33 Debootstrap | I: Checking Release signature
2018/09/28 11:23:33 Debootstrap | I: Valid Release signature (key id 067E3C456BAE240ACEE88F6FEF0F382A1
2018/09/28 11:23:34 Debootstrap | I: Retrieving Packages
2018/09/28 11:23:37 Debootstrap | I: Validating Packages
2018/09/28 11:23:39 Debootstrap | I: Resolving dependencies of required packages...

...

2018/09/28 11:27:27 Debootstrap (stage 2) | I: Configuring libc-bin...
2018/09/28 11:27:27 Debootstrap (stage 2) | I: Configuring ca-certificates...
2018/09/28 11:27:37 Debootstrap (stage 2) | I: Base system installed successfully.
2018/09/28 11:27:38 ==== pack ====
2018/09/28 11:27:38 Compression to /home/ana/code/rootfs.tar.gz
Powering off.
2018/09/28 13:27:44 ==== Recipe done ====
```

Templating: parameters

- We can use Go's template package `text/template` to introduce parameters in the YAML recipe file.
- This allows one to reuse the same recipe to be build with different parameters.
- For example, if we want to use the same recipe for differents arch and Debian release:

```
$ debos -t suite:"stretch" -t arch:"amd64" recipe.yaml
$ debos -t suite:"buster" -t arch:"arm64" recipe.yaml
```

debos recipe with parameters

```
{{- $arch := or .arch "arm64" -}}
{{- $suite := or .suite "stretch" -}}
{{- $image := or .image (printf "%s-%s.tgz" $suite $arch) -}}

architecture: {{ $arch }}

actions:
  - action: debootstrap
    suite: {{ $suite }}
    components:
      - main
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: pack
    file: {{ $image }}
    compression: gz
```

Templating: conditional

```
{{- if eq $type "tools" "development" }}
- action: apt
  packages:
    - vim
    - git
{{- end -}}

...

- action: apt
  recommends: false
  packages:
    - adduser
    - sudo
{{- if eq $arch "armhf" "arm64" }}
    - python-libsoc
{{- end }}
```

debos actions

There are currently 12 actions that can be used by the recipes:

apt	image-partition	pack
debootstrap	ostree-commit	raw
download	ostree-deploy	run
filesystem-deploy	overlay	unpack

The former recipe used two actions:

- **debootstrap**: construct the target rootfs with debootstrap
- **pack**: create a tarball with the target filesystem

What is debootstrap?

- `debootstrap` allows the installation of a Debian base system from scratch.
- It works by downloading all the .deb files from a mirror site and carefully unpacking them into a directory which can eventually be chrooted into
- it's able to installs the system without requiring the availability of dpkg or apt, which mean you can use deboostrap from a non Debian system.
- More info at https://wiki.debian.org/Debootstrap

Action apt

```
- action: apt
  recommends: bool
  packages:
    - package1
    - package2
    - package3
```

- This action installs packages and their dependencies to the target rootfs with 'apt'.
- There is an option to not install `Recommends` packages.
- If the package is not in debian `main`, e.g. `linux-firmware` make sure `non-free` has been added in the debootstrap action.

Example: action apt

```
architecture: armhf

actions:
  - action: debootstrap
    suite: "stretch"
    components:
      - main
      - contrib
      - non-free
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: apt
    packages:
    - sudo
    - openssh-server
    - adduser
    - firmware-linux
```

Action download

```
- action: download
  url: https://example.domain/firmware.tgz
  name: firmware
  unpack: true
  compression: gz
```

- Download a single file from the internet and unpack it in place if needed.
- This action *doesn't place the file inside* into the target filesystem.

Action overlay

```
- action: overlay
  origin: name
  source: directory
  destination: directory
```

- Copy recursively directories and/or files into the target filesystem.
- The tree of files and directories is copied directly in the root directory of the image if `destination` isn't set.

Example: action overlay

```
{{- $firmware_version := or .firmware_version "1.20171029" -}}

...

  - action: download
    url:  https:// example.domain/{{ $firmware_version }}.tar.gz
    unpack: true   # Unpack downloaded file
    name: firmware # directory name used by other actions

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

  - action: overlay
    description: Log automatically on the serial console
    source: overlays/auto-login
```

Action run

- Allows to run a command or script
- ...in the target filesystem or in the fakemachine host
- In the target filesystem, it will be run with root privileges
- There a few variables defined in debos that can be used with this action when they're run in the fakemachine host:
  - `ROOTDIR` is the root of the target filesystem
  - `ARTIFACTDIR` is the artifact directory
  - `IMAGE` points to the image (if any)
  - `RECIPEDIR` is the recipe directory.

Example: action run

```
architecture: arm64

actions:
  - action: debootstrap
    suite: stretch
    components:
      - main
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: run
    description: Get package list
    chroot: true
    command: dpkg -l > list.txt

  - action: run
    description: Copy file with the list of packages
    chroot: false
    command: cp ${ROOTDIR}/list.txt ${ARTIFACTDIR}/list.txt
```

Example: action run

```
{{- $arch := or .arch "arm64" -}}
{{- $suite := or .suite "stretch" -}}

architecture: {{ $arch }}

actions:
  - action: debootstrap
    suite: stretch
    components:
      - main
    mirror: https://deb.debian.org/debian
    variant: minbase

  - action: run
    description: Set hostname
    chroot: true
    command: echo deb-{{ $suite }}-{{ $arch }} > /etc/hostname
```

Actions image-partition + filesystem-deploy

```
- action: image-partition
  imagename: image_name
  imagesize: size
  partitiontype: gpt
  gpt_gap: offset
  partitions:
    <list of partitions>
  mountpoints:
    <list of mount points>

- action: filesystem-deploy
  setup-fstab: bool
  setup-kernel-cmdline: bool
```

- `image-partition` create an image file, make partitions and format them
- `filesystem-deploy` deploy a root filesystem to an image previously created by `image-partition`

Example: image-partition + filesystem-deploy

```
- action: image-partition
  imagename: "apertis-armhf.img"
  imagesize: 4G
  partitiontype: gpt
  mountpoints:
    - mountpoint: /
      partition: root
      flags: [ boot ]
  partitions:
    - name: root
      fs: ext4
      start: 0%
      end: 100%

- action: filesystem-deploy
  description: Deploy the filesystem onto the image
```

debos recipe: bootable image for a Raspberry Pi 3 B+

debos recipe creating a bootable image for a Raspberry Pi 3 B+
https://github.com/ana/debos-recipes/tree/master/rpi3bplus

COLLABORA

Presentation Outline

Future plans

- There is no defined roadmap
- Development led by our needs and external contributions
- Some ideas for new actions and improvements are listed in the TODO file
- Continously improving the documentation!

Find more information

- debos at GitHub https://github.com/go-debos
- Documentation of all the actions
  https://godoc.org/github.com/go-debos/debos/actions
- Example used in this presentation https://github.com/go-debos/debos-recipes/

Thank you!