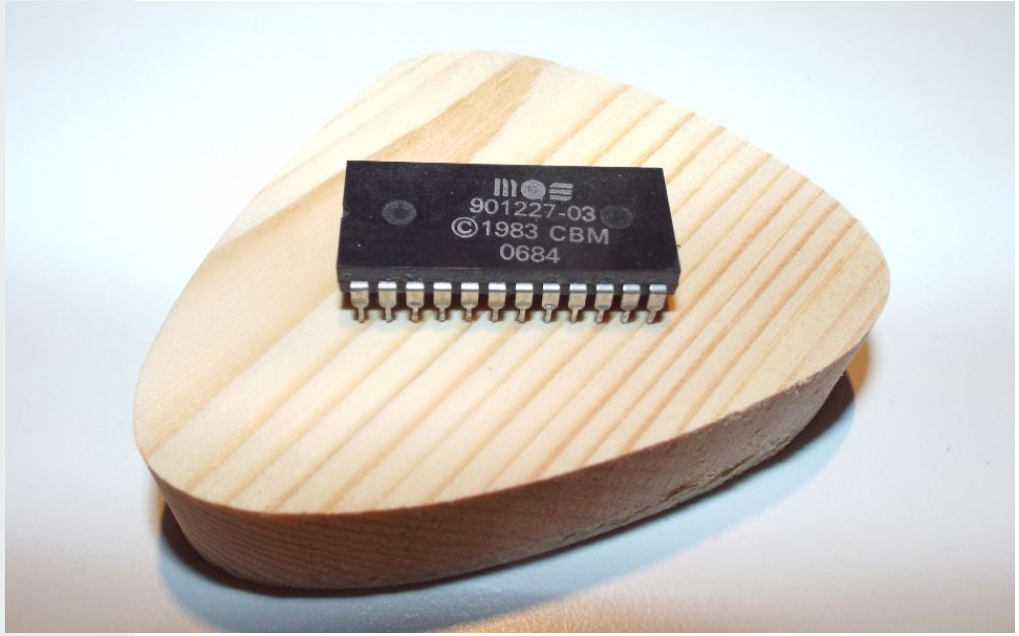
diskimage-builder:

Building Linux Images for Cloud / Virtualization / Container

Let’s start with a little bit of history:

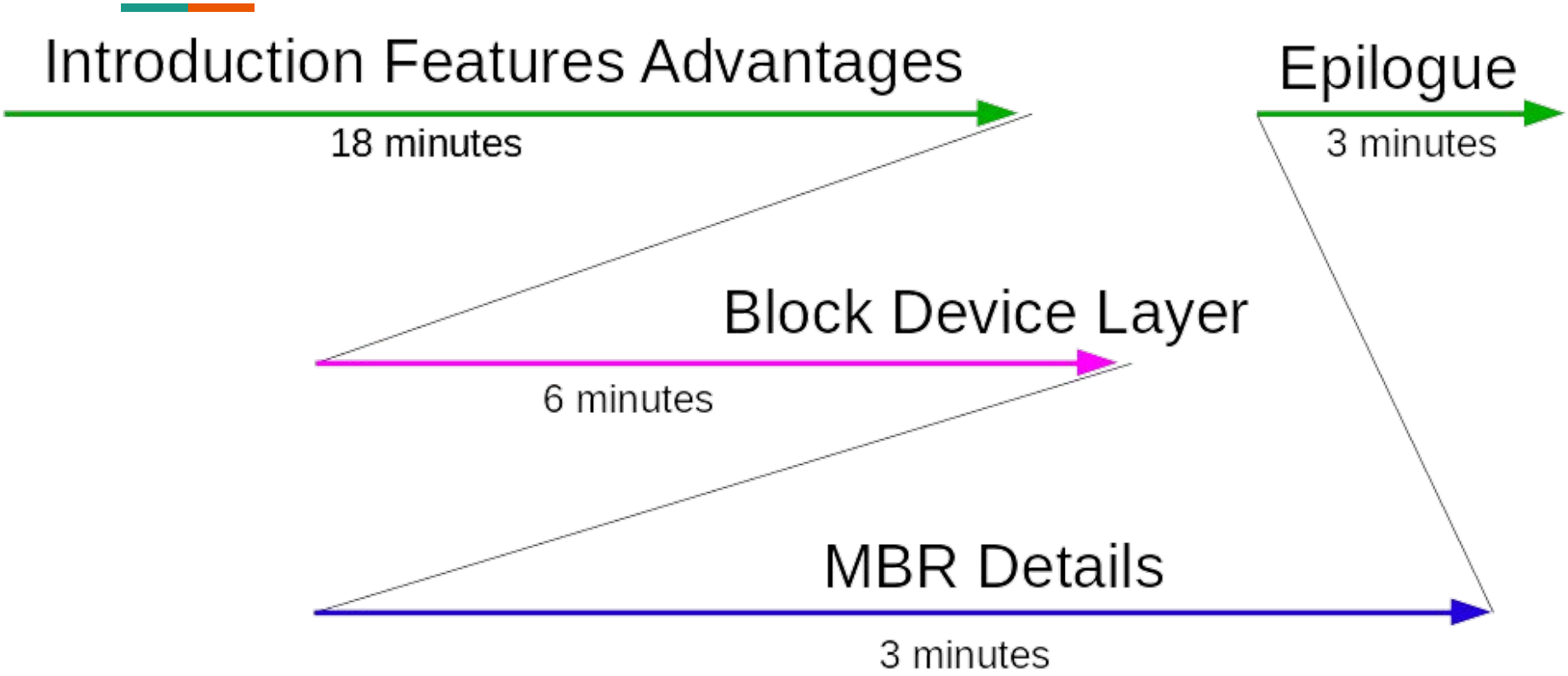
Once upon a time...



About the Author

Andreas Florath andreas@florath.net

Mathematician (RWTH Aachen)
Currently living in East-Belgium in Deutschsprachigen Gemeinschaft.
Professional software developer since 1994.
Preferred languages: C++, python.
Active Free and Open Software supporter since 1992.
Current projects: diskimage-builder (core developer), rmtoo and creating wooden things with my DIY CNC router.


Introduction Features Advantages
18 minutes
Epilogue
3 minutes
Block Device Layer
6 minutes
MBR Details
3 minutes

OS Images

Operating System (OS) Images are a copy of a pre-installed operating system.

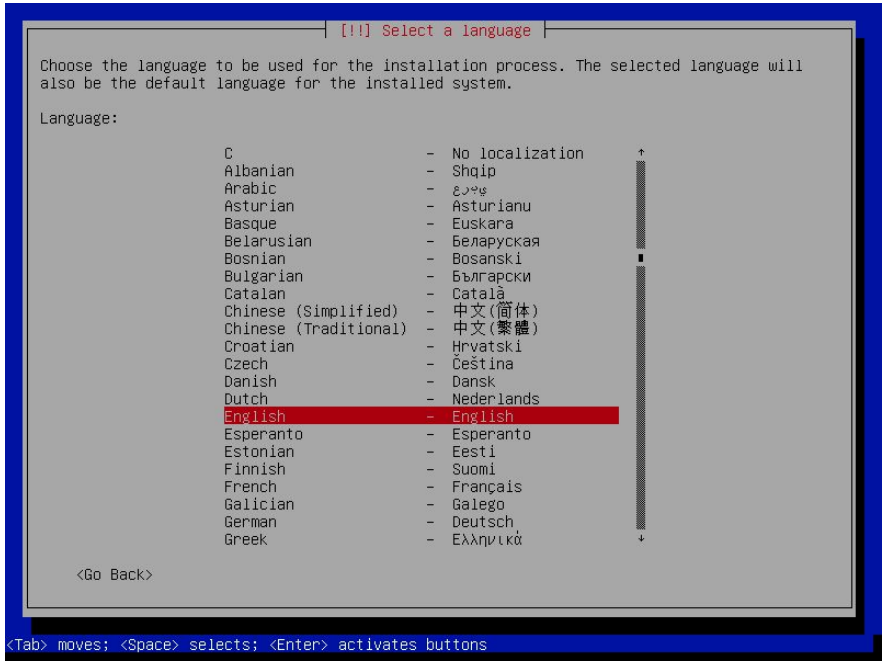
Other names: Golden Image, Template OS Image

OS Images are used in virtualization, cloud and container environments.

Building OS Images: Installer

Install OS in traditional way and copy the resulting data (manual / automatic)

Example:
kickstart for RedHat / CentOs / Fedora based systems



Building OS Images: utility

Use a program that directly creates an OS image / tree

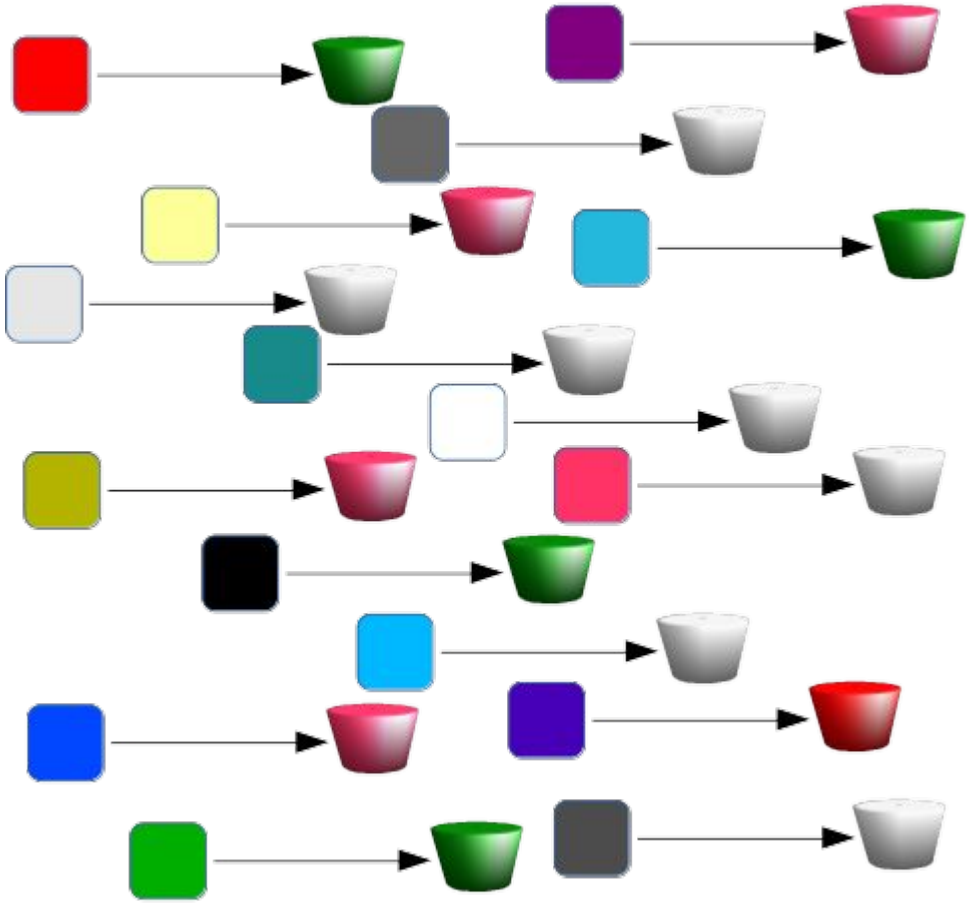
Example:
debootstrap for Debian / Ubuntu

```
root:/tmp/debian# debootstrap stretch .
I: Retrieving InRelease
I: Retrieving Release
I: Retrieving Release.gpg
I: Checking Release signature
I: Valid Release signature (key id 067E3C456BAE240
ACEE88F6FEF0F382A1A7B6500)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
```

```
I: Configuring libgnutls30:amd64...
I: Configuring wget...
I: Configuring tasksel...
I: Configuring tasksel-data...
I: Configuring libc-bin...
I: Configuring systemd...
I: Base system installed successfully.
root:/tmp/debian#
```

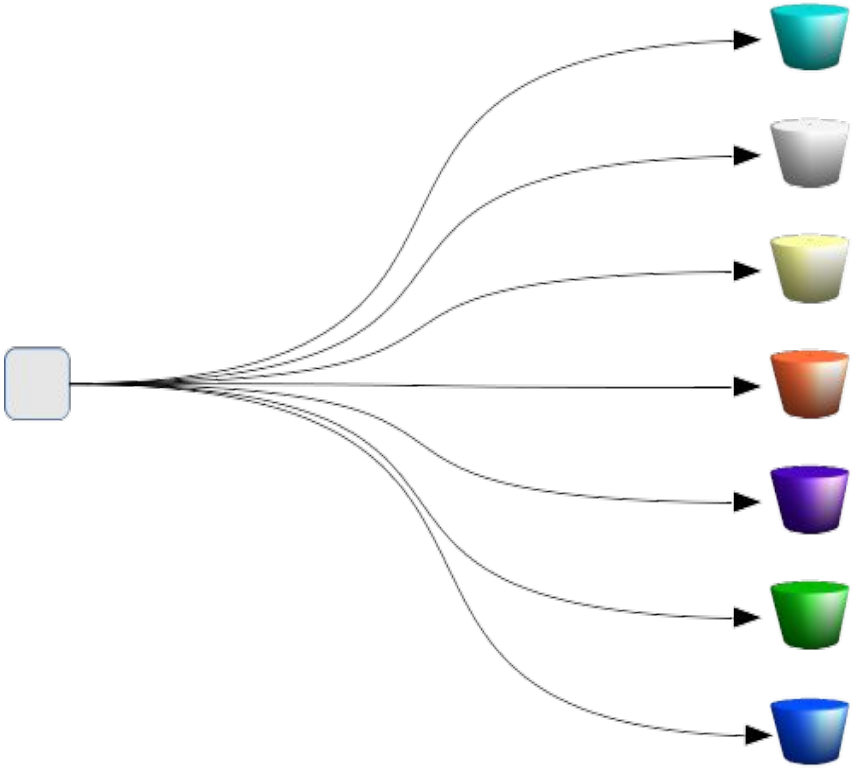
Many-To-Many

Simultaneously targeting multiple OSes, architectures and environments (virtualization, container, cloud, ...) requires one configuration for each combination.



One to Rule Them All

diskimage-builder solves this problem: it creates images for different distributions or architectures for different target platforms based on a single unified configuration.



Examples

```
disk-image-create debian-minimal vm
disk-image-create fedora-minimal vm
disk-image-create centos-minimal vm
disk-image-create -o docker fedora-minimal
```

Under the hood: diskimage-builder requires to have system utilities (rpm, debootstrap, ...) installed - and calls / uses them.

Support Matrix

Distributions:

Debian, Ubuntu, Fedora, RedHat, Centos, OpenSuse, Gentoo
(typically the stable and the last-stable version)

Architectures:

arm64, i386, amd64/x86_64, powerpc
(cross build is supported)

Environments:

VmWare, OpenStack, KVM, AWS EC2, Docker, Bare-Metal.

qemu is used to execute target binaries (as needed in pre- and postinstall scripts) on the host system.

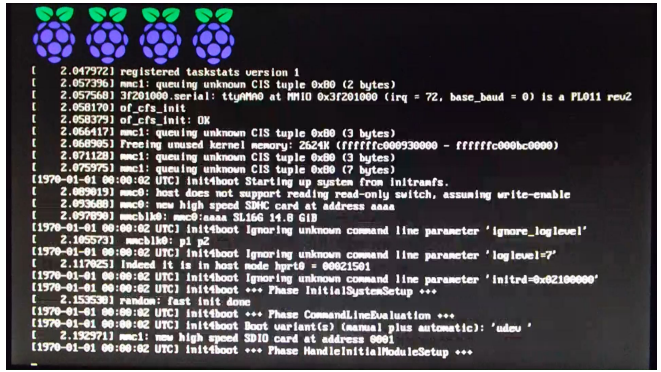
Images can be converted into mostly any format - `qemu convert` is your friend.

“Elements”: Batteries Included

diskimage-builder comes with about 100 predefined (so called) “elements” for additional features, configuration, adaptions and scripting.
Usage: `disk-image-create debian-minimal vm puppet-master`

Many additional elements are available in the Internet - one example: ready to use element for building a Raspberry Pi Image.
Usage: `disk-image-create debian-minimal vm rpi3`

Configurable via environment variables.



Example Elements

- baremetal
- cloud-init
- devuser
- docker
- epel
- growroot
- manifests
- pip_and_virtualenv
- ssh-server
- proliant-tools
- selinux-permissive
- sysctl
- uboot
- vm

Some packages that clash with others or are highly environment / hardware specific. Read the element's documentation!

An Element is...

```
README.rst
element-deps
package-installs.yaml
environment.d
root.d
```

An Element is...

```
README.rst
element-deps
package-installs.yaml
environment.d
root.d
```

```
debootstrap
dib-python
pkg-map
```

```
linux-image-amd64:
  arch: amd64
linux-image-686:
  arch: i386
linux-image-arm64:
  arch: arm64
netbase:
```

```
export DISTRO_NAME=debian
export \
DIB_RELEASE=${DIB_RELEASE:-stable}
```

```
#!/bin/bash
apt-get update
apt-get clean
apt-get dist-upgrade -y
...
```

Block Device Layer (1/2)

Level 4	fstab handling
Level 3	Mounting
Level 2	File system generation; mkfs (ext, xfs, fat, …)
Level 1	Combine / split level 0 / 1 block devices; partitioning, LVM; possible: RAID, cryptsetup, ...
Level 0	Provides disk space; loop device; possible: (raw) disk devices, iSCSI, ...

Block Device Layer (2/2)

```
- local_loop:
    name: image0

- partitioning:
    base: image0
    label: mbr
    partitions:
      - name: root
        flags: [ boot, primary ]
        size: 100%
```

```
- mkfs:
    base: root
    mount:
      mount_point: /
      fstab:
        options: "defaults"
        fsck-passno: 1
```

Block Device Layer MBR Module

It's about writing 72 bytes to the correct position in the Master Boot Sector!

```
root:~# dd if=/dev/sda skip=440 bs=1 count=72 2>/dev/null | hexdump
0000000 ccfb ee86 0000 2080 0021 fe83 ffff 0800
0000010 0000 a800 03b9 0000 0000 0000 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
*
0000040 0000 0000 0000 aa55
0000048
```

~~Idea: Use existing tool like fdisk, sfdisk, parted, ...~~

A small Python class war created to write MBRs:
Short (~150 LOC), open source, tested, and actually does what you tell it.

Development Insights

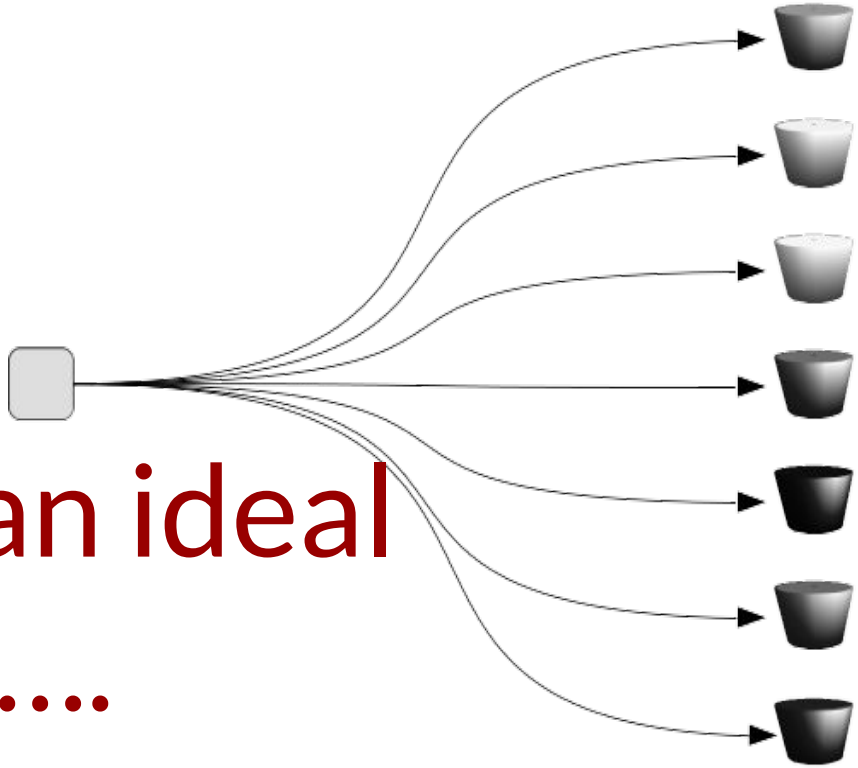
- ‘Small size’ project:
  - 12500 LOC: ~7000 bash, ~3000 python , ~2500 yaml
  - block device layer: ~2150 python, ~800 yaml
- Many adaptions / workarounds for (old and known) problems of various distributions.
- Design should be improved.
- Slow development cycle (CI slow / no developers).
- Many contributors focus on their own feature/bug-fix, rarely review other contributions.

One to Rule Them All

RECAP

diskimage-builder solves this problem: it creates images for different distributions or architectures for different target platforms based on one configuration.

This is for an ideal world only….



target \ host	centos-7	debian-buster	debian-jessie	debian-stretch	fedora-25	fedora-26	fedora-27	gentoo-latest	opensuse-42.2	opensuse-42.3	ubuntu-artful ...
hostgen	success	success	success	success	success	failed	success	failed	success	failed	success
tox	success	failed	failed	success	success	- 3 -	failed	- 3 -	failed	- 3 -	failed
centos-7	success	success	success	success	success	- 3 -	success	- 3 -	success	- 3 -	success
debian-buster	failed	success	success	success	success	- 3 -	success	- 3 -	- 4 -	- 3 -	success
debian-jessie	failed	success	failed	success	success	- 3 -	failed	- 3 -	success	- 3 -	success
debian-stretch	failed	failed	success	success	success	- 3 -	success	- 3 -	success	- 3 -	success
fedora-25	success	failed	success	success	success	- 3 -	success	- 3 -	failed	- 3 -	success
fedora-26	success	success	success	success	success	- 3 -	success	- 3 -	success	- 3 -	success
fedora-27	success	success	success	success	failed	- 3 -	success	- 3 -	success	- 3 -	success
gentoo-latest	failed	failed	failed	failed	failed	- 3 -	failed	- 3 -	failed	- 3 -	failed
opensuse-42.2	failed	- 1 -	failed	- 1 -	failed	- 3 -	success	- 3 -	success	- 3 -	failed
opensuse-42.3	failed	- 1 -	failed	- 1 -	failed	- 3 -	success	- 3 -	success	- 3 -	failed
ubuntu-artful	failed	success	- 2 -	- 2 -	- 2 -	- 3 -	success	- 3 -	- 2 -	- 3 -	success
ubuntu-precise	failed	success	success	success	failed	- 3 -	failed	- 3 -	failed	- 3 -	failed
ubuntu-trusty	failed	success	success	success	success	- 3 -	success	- 3 -	success	- 3 -	success
ubuntu-xenial	failed	success	- 6 -	success	success	- 3 -	success	- 3 -	success	- 3 -	success
ubuntu-zesty	failed	success	- 5 -	success	success	- 3 -	success	- 3 -	- 5 -	- 3 -	success

- 0 - runuser not available
- 1 - zypper not available
- 2 - No such script: /usr/share/debootstrap/scripts/artful
- 3 - missing dependency
- 4 - No such script: /usr/share/debootstrap/scripts/buster
- 5 - No such script: /usr/share/debootstrap/scripts/zesty
- 6 - No such script: /usr/share/debootstrap/scripts/xenial

Advantages / Disadvantages

+ Speed (with HTTP / packet) caching: 2-3 minutes
+ One configuration for all targets
+ Supports many distributions, architectures, host and target systems

- Only limited set of functions / systems are tested during CI
- Large docker images with unused packages are created

Best Practice: What to put into an OS image?

- Be as general and minimal as possible
  Don’t install a very specific application that rarely needed.
- Don’t do any hardening
  Hardening is a steady process that should be done by a configuration management system (puppet, chef, ansible, …)
- Get the disk layout as needed during OS image build
  You don’t want to mess around creating partitions / LVM later on.

General rule of thumb: Do things as late as possible.

diskimage_builder/lib/disk-image-create:main:500 : trap EXIT

This is the End.

The Doors

Thank You!

Alanis Morissette

References / Resources

Raspberry Pi 3 diskimage-builder element: https://github.com/florath/dib-element-raspberrypi3
rmtoo: free and open source requirements management system: http://rmtoo.florath.net/
diskimage-builder docker matrix build: https://review.openstack.org/#/c/414347/
diskimage-builder docker matrix build results:
https://etherpad.openstack.org/p/DIBMaxtrixDockerBuild

diskimage-builder @ OpenStack:
https://docs.openstack.org/diskimage-builder/latest/
https://git.openstack.org/cgit/openstack/diskimage-builder/
https://review.openstack.org/#/q/project:openstack/diskimage-builder

License / Contact

Feel free to contact me for questions, remarks or discussions:
andreas@florath.net