ubuntu

Ubuntu Core 16

Training for SAGE X3

CANONICAL

CANONICAL

Overview

We are the company behind Ubuntu.

About Ubuntu & Canonical

Ubuntu is an open-source operating system, currently established on server, cloud, personal & IoT products.

Canonical has been developing operating systems since 2004, and is now extending the Ubuntu OS on mobile and IoT devices.

2004
FOUNDED

750+
EMPLOYEES

30+
COUNTRIES


ubuntu
is everywhere!

Ubuntu in your everyday life

Ubuntu phones from Meizu and BQ

Anyone can install Ubuntu on Google Nexus tablet or phone

Are all running on Ubuntu

Empowering social media

Yep. More Ubuntu.

Snapchat · Pinterest · Reddit · Instagram

Enabling your daily tasks

Shopping at Walmart? Everyday prices. Everyday Ubuntu.

Ubuntu is serious business for Bloomberg

Paypal and Dropbox are both served by Ubuntu

Watching a movie? WETA Digital and Netflix run Ubuntu

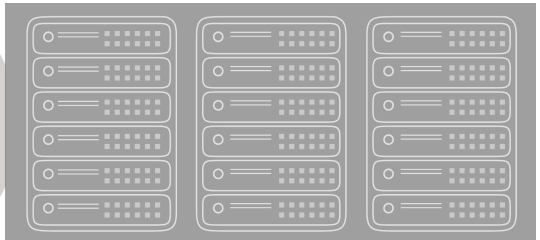
8-18 billion page views per month and hosted on Ubuntu

Controlling BYU Mars Rover

Ubuntu is everywhere!

Behind the largest supercomputer

- Tianhe-2. 33.86 PFLOPS
- All crunching on Ubuntu!



20 million

launches of Ubuntu instances in 2015 in:

public cloud
AWS, Microsoft Azure, Google Compute Engine, Rackspace, Oracle Cloud, VMware

private cloud
OpenStack - Including some of the world's largest private clouds, like Deutsche Telekom

bare metal
Ubuntu at scale on bare metal with MAAS

There's also plenty of Ubuntu in:

Kubernetes and Apache Mesos

Cloud Foundry and Heroku

Running within the International Space Station

In November 2015

2 million
new Ubuntu Cloud instances launched

Ubuntu Cloud instances:

- 67,000 Ubuntu Cloud instances launched in 24 hours
- 28,000 Ubuntu Cloud instances launched in 1 hour
- 46 Ubuntu Cloud instances launched each minute
- Approx. 1 Ubuntu Cloud instances launched each second

Ubuntu Core

What is Ubuntu Core?

A minimal version with the same bits as today's Ubuntu

Ubuntu Core with **transactional updates**

Applications confined by technologies lead by Canonical

Safe, reliable, worry free **updates** with tests and **rollback**

Amazing developer experience with **snapcraft**

Easily extensible

Easily create **app stores** for all your devices

Ubuntu Core

Build

Choose from a variety of SoC, boards and existing libraries.

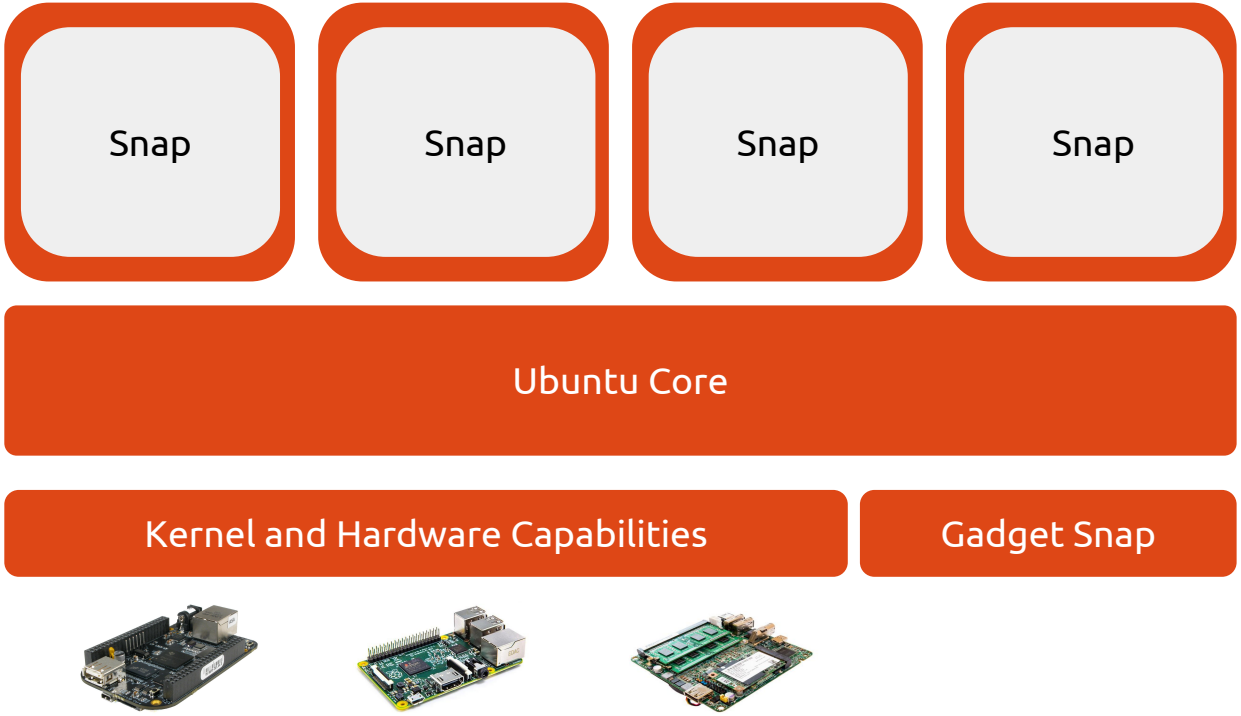
Differentiate

Monetize after device sales

Maintain

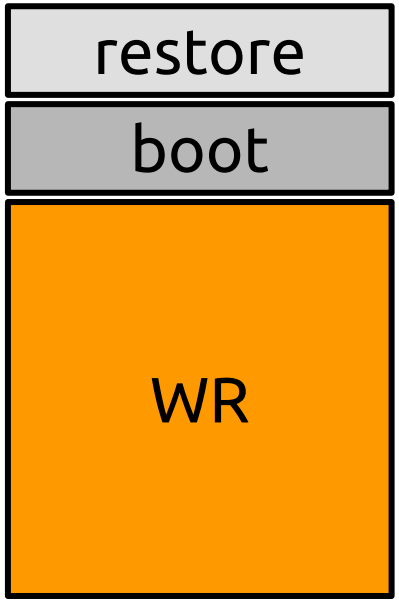
Over-the-Air updates with health-checks and rollback

Ubuntu Core architecture



Ubuntu Core Partition Layout

The partition layout on Ubuntu Core 16 has the following structure:

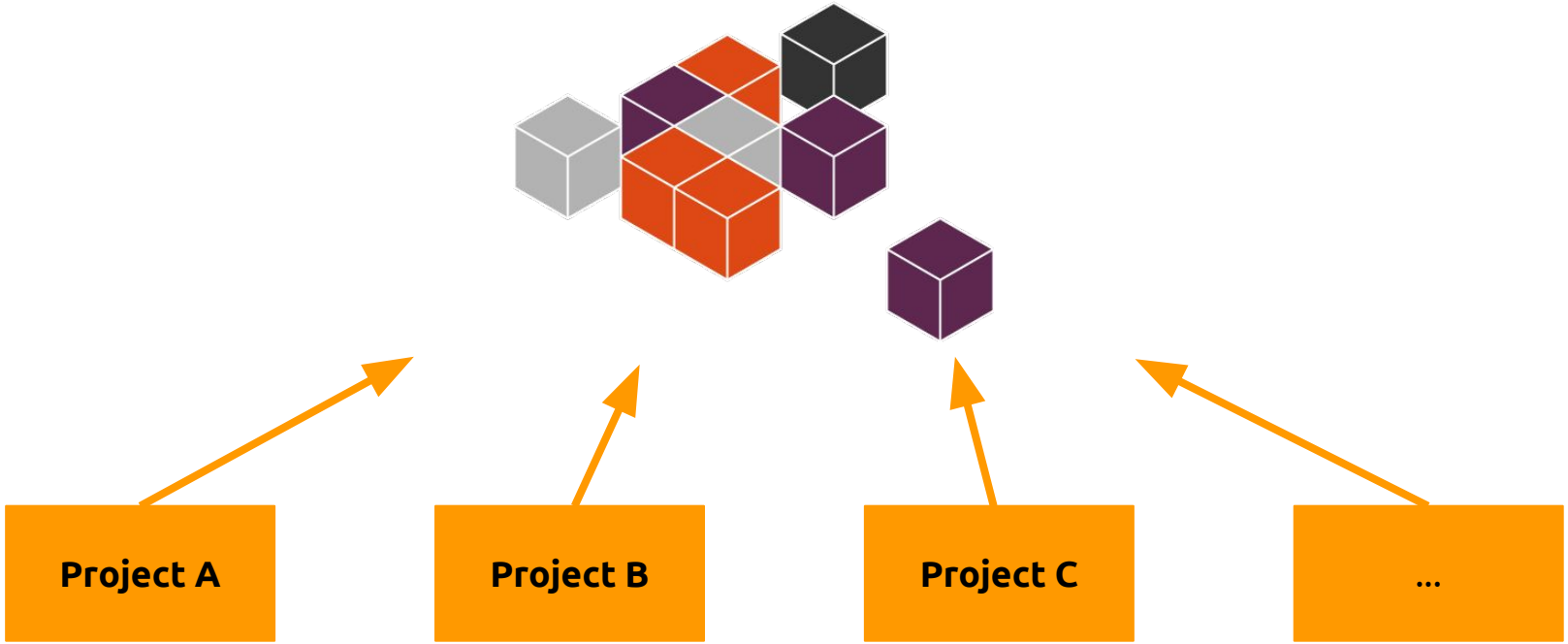



```
                                Disk: /dev/sda
                 Size: 3.6 GiB, 3899999744 bytes, 7617187 sectors
           Label: gpt, identifier: 3A72624E-AE14-4B07-86A9-1F387A3E6F79

   Device                Start        End    Sectors     Size Type
   Free space             2048       8191       6144       3M
   /dev/sda1              8192      16383       8192       4M BIOS boot
   /dev/sda2             16384     147455     131072      64M EFI System
>> /dev/sda3            147456    7614463    7467008     3.6G Linux filesystem
   Free space          7614464    7617153       2690     1.3M


  Partition name: writable
  Partition UUID: 1BBDC693-9E02-4124-8C30-DE3DC5E60FE4
  Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)
      Filesystem: ext4
Filesystem label: writable
 Filesystem UUID: bca5fa13-04af-4ccb-8cf1-1d0fdc717de2
      Mountpoint: /writable (mounted)

   [ Delete ]  [  Quit  ]  [  Type  ]  [  Help  ]  [  Write ]  [  Dump  ]
```

This partition layout introduces a significant change from the previous release as no longer uses a dual partition schema for the system.

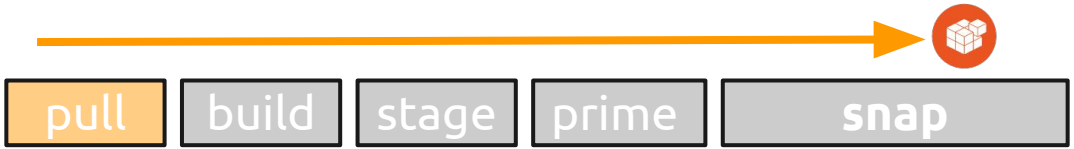
Developer tools: Snapcraft

Snapcraft lets developers assemble their snap from existing projects, leveraging different technologies.



Developer tools: Snapcraft

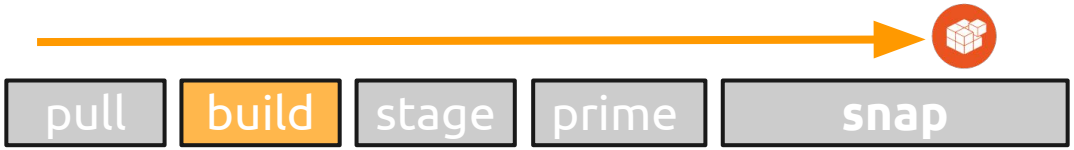
Snapcraft 2.x



The pull phase takes care of the downloading / cloning of the remote files needed for this part.

Developer tools: Snapcraft

Snapcraft 2.x

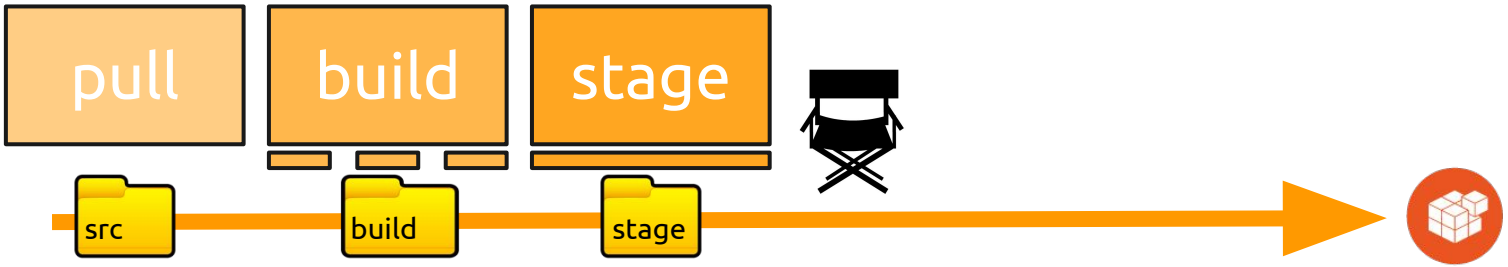


The build phase builds the parts of the downloaded code.

Developer tools: Snapcraft

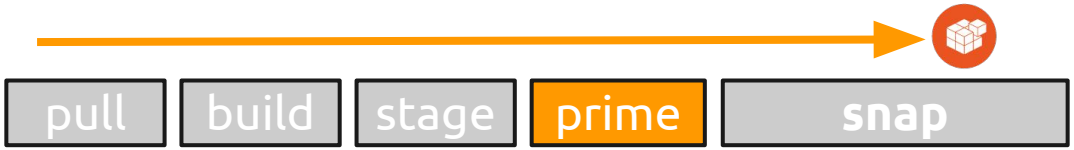
Snapcraft 2.x

The stage phase copies the installed files into a user-visible stage/folder. All parts share the same file layout.

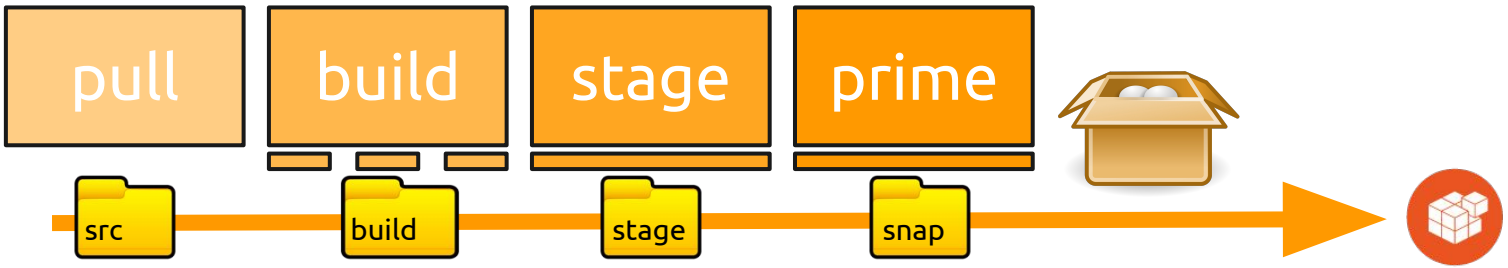


Developer tools: Snapcraft

Snapcraft 2.x

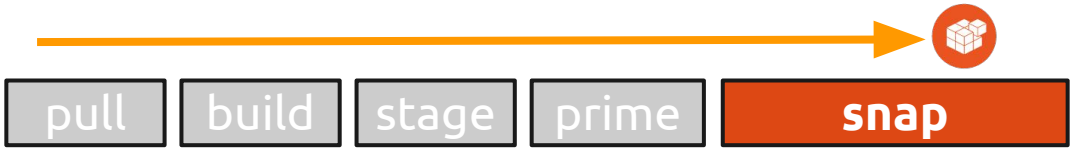


The prime phase performs the final copy and preparation for the snap, removing the parts that are not needed.

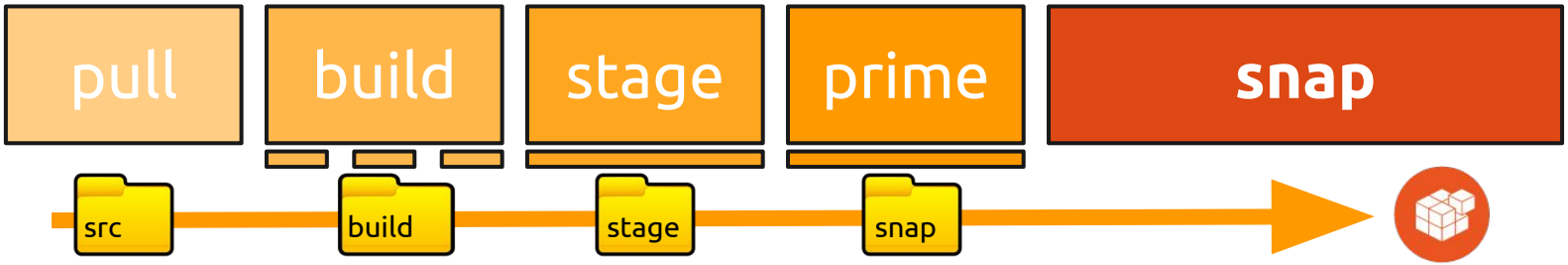


Developer tools: Snapcraft

Snapcraft 2.x



The snap phase finalises the snapcraft lifecycle creating the snap, that is can now be shared with your devices or make it available to the world.



Developer tools: Snapcraft

- Snapcraft allows developers to create snaps that run on any Ubuntu Core device as snaps allow decoupling of OS and applications
- With Snapcraft developers have full control of the version of the parts they leverage. OS updates won't affect their applications (and vice versa)
- Snapcraft allows developers to define security and access policies of their applications, making it easier to create confined and secure apps.

Developer tools: Snapcraft plugins

Snapcraft

Snapcraft supports several technologies through the current plugins available.

Snapcraft is extensible and new plugins to leverage existing technologies can be developed.

Java, Python, ROS, Go, Maven, QML, NodeJS, make, kernel are just a few examples of the languages and technologies that can be used.

We can reuse deb packages from ubuntu

Developer tools: Snapcraft

Snapcraft allows you to init your project creating a template:

```
127 pcoca@haswell16:~/snapcraft/mqtt-example⟫ snapcraft --version
2.9
pcoca@haswell16:~/snapcraft/mqtt-example⟫ snapcraft init
Created snapcraft.yaml.
pcoca@haswell16:~/snapcraft/mqtt-example⟫ tree
.
└── snapcraft.yaml

0 directories, 1 file
pcoca@haswell16:~/snapcraft/mqtt-example⟫ cat snapcraft.yaml
name: # the name of the snap
version: # the version of the snap
summary: # 79 char long summary
description: # a longer description for the snap
confinement: devmode # use "strict" to enforce system access only via declared interfaces
```

Developer tools: Snapcraft

MQTT broker snap yaml

```
name: mosquitto-mqtt-broker-dev
version: 0.1
summary: mosquitto broker
description: MQTT broker example
confinement: devmode

apps:
  mosquitto:
    command: usr/sbin/mosquitto -c $SNAP/mosquitto.conf
    daemon: simple
    plugs: [network, network-bind]

parts:
  mosquitto:
    plugin: copy
    stage-packages: [mosquitto]
    files:
      mosquitto.conf: mosquitto.conf
```

Developer tools: Snapcraft

MQTT publisher snap yaml

```
name: mqtt-publisher-dev
version: 0.1
summary: MQTT publisher
description: MQTT publisher example
confinement: devmode

apps:
  publish:
    command: bin/publish
    plugs: [network, network-bind]

parts:
  mqtt-publisher:
    plugin: copy
    files:
      publish.py: bin/publish
    after: [mqtt-paho-py3]
```

Developer tools: Snapcraft

MQTT subscriber snap yaml

```
name: mqtt-subscriber-dev
version: 0.1
summary: MQTT subscriber
description: MQTT subscriber example
confinement: devmode

apps:
  subscribe:
    command: bin/subscribe
    plugs: [network, network-bind]

parts:
  mqtt-subscriber:
    plugin: copy
    files:
      subscribe.py: bin/subscribe
    after: [mqtt-paho-py3]
```

Developer tools: Snapcraft

MQTT broker snapcraft lifecycle

The command snapcraft on our directory will go through the whole snapcraft cycle and will create the package in squashFS format.

The result of running snapcraft will be the MQTT broker snap.

```
'libc-bin', 'libc6', 'libcap2', 'libcap2-bin', 'libcomerr2', 'libcryptsetup4', 'libdb5.3', 'libdebconfclient0', 'libdevmapper1.02.1', 'libgcc1', 'libgcrypt20', 'libgpg-er
ror0', 'libgpm2', 'libkmod2', 'liblocale-gettext-perl', 'liblzma5', 'libmagic1', 'libmount1', 'libncurses5', 'libncursesw5', 'libpam-modules', 'libpam-modules-bin', 'libp
am-runtime', 'libpam0g', 'libpcre3', 'libprocps3', 'libreadline6', 'libselinux1', 'libsemanage-common', 'libsemanage1', 'libsepol1', 'libslang2', 'libsmartcols1', 'libss2
', 'libstdc++6', 'libsystemd0', 'libtext-charwidth-perl', 'libtext-iconv-perl', 'libtext-wrapi18n-perl', 'libtinfo5', 'libudev1', 'libusb-0.1-4', 'libustr-1.0-1', 'libuui
d1', 'locales', 'login', 'lsb-base', 'makedev', 'manpages', 'manpages-dev', 'mawk', 'mount', 'multiarch-support', 'ncurses-base', 'ncurses-bin', 'passwd', 'perl-base', 'p
rocps', 'readline-common', 'sed', 'sensible-utils', 'systemd', 'systemd-sysv', 'sysv-rc', 'sysvinit-utils', 'tar', 'tzdata', 'ubuntu-keyring', 'udev', 'util-linux', 'zlib
1g']
Get:1 http://gb.archive.ubuntu.com/ubuntu xenial/main amd64 libwrap0 amd64 7.6.q-25 [46.2 kB]
Get:2 http://gb.archive.ubuntu.com/ubuntu xenial/main amd64 gcc-6-base amd64 6.0.1-0ubuntu1 [14.3 kB]
Get:3 http://gb.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libssl1.0.0 amd64 1.0.2g-1ubuntu4.1 [1,122 kB]
Get:4 http://gb.archive.ubuntu.com/ubuntu xenial/universe amd64 libuv1 amd64 1.8.0-1 [57.4 kB]
Get:5 http://gb.archive.ubuntu.com/ubuntu xenial/universe amd64 libev4 amd64 1:4.22-1 [26.3 kB]
Get:6 http://gb.archive.ubuntu.com/ubuntu xenial/universe amd64 libwebsockets7 amd64 1.7.1-1 [61.0 kB]
Get:7 http://gb.archive.ubuntu.com/ubuntu xenial/universe amd64 mosquitto amd64 1.4.8-1build1 [108 kB]
Fetched 1,435 kB in 0s (0 B/s)
Pulling mosquitto
Preparing to build mosquitto
Building mosquitto
Staging mosquitto
Stripping mosquitto
Snapping mosquitto-mqtt-broker-dev_0.1_amd64.snap
Parallel mksquashfs: Using 4 processors
Creating 4.0 filesystem on mosquitto-mqtt-broker-dev_0.1_amd64.snap, block size 131072.
[=============================================================================================================================================================-] 86/86 100%

Exportable Squashfs 4.0 filesystem, xz compressed, data block size 131072
        compressed data, compressed metadata, compressed fragments, no xattrs
        duplicates are removed
Filesystem size 1403.19 Kbytes (1.37 Mbytes)
        37.20% of uncompressed filesystem size (3772.32 Kbytes)
Inode table size 1198 bytes (1.17 Kbytes)
        32.09% of uncompressed inode table size (3733 bytes)
Directory table size 1162 bytes (1.13 Kbytes)
        45.16% of uncompressed directory table size (2573 bytes)
Number of duplicate files found 0
Number of inodes 112
Number of files 62
Number of fragments 5
Number of symbolic links  6
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 44
Number of ids (unique uids + gids) 1
Number of uids 1
        root (0)
Number of gids 1
        root (0)
Snapped mosquitto-mqtt-broker-dev_0.1_amd64.snap
pcoca@haswell16:~/snapcraft/mqtt-broker$
u*  16.04 0:-  1:-# 2:-#- 3:-* 4:-  5:-  6:-  7:-  8:-  9:-                                                                    1! 7h24m 1.62 4x2.0GHz 7.7GB6% 2016-06-02 16:54:46
```

Developer tools: Snapcraft

MQTT publisher snapcraft lifecycle

The command snapcraft on our directory will go through the whole snapcraft cycle and will create the package in squashFS format.

The result of running snapcraft will be the MQTT publisher snap.

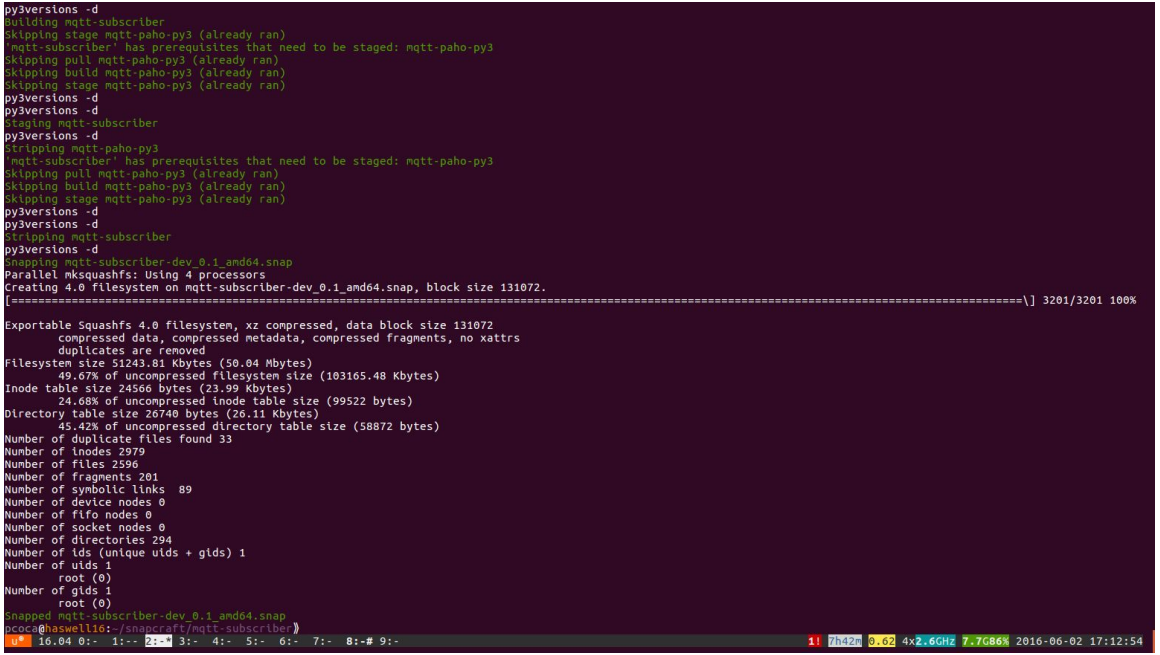
```
py3versions -d
Building mqtt-publisher
Skipping stage mqtt-paho-py3 (already ran)
'mqtt-publisher' has prerequisites that need to be staged: mqtt-paho-py3
Skipping pull mqtt-paho-py3 (already ran)
Skipping build mqtt-paho-py3 (already ran)
Skipping stage mqtt-paho-py3 (already ran)
py3versions -d
py3versions -d
Staging mqtt-publisher
py3versions -d
Stripping mqtt-paho-py3
'mqtt-publisher' has prerequisites that need to be staged: mqtt-paho-py3
Skipping pull mqtt-paho-py3 (already ran)
Skipping build mqtt-paho-py3 (already ran)
Skipping stage mqtt-paho-py3 (already ran)
py3versions -d
py3versions -d
Stripping mqtt-publisher
py3versions -d
Snapping mqtt-publisher-dev_0.1_amd64.snap
Parallel mksquashfs: Using 4 processors
Creating 4.0 filesystem on mqtt-publisher-dev_0.1_amd64.snap, block size 131072.
[=============================================================================================================================================================|] 3201/3201 100%

Exportable Squashfs 4.0 filesystem, xz compressed, data block size 131072
        compressed data, compressed metadata, compressed fragments, no xattrs
        duplicates are removed
Filesystem size 51243.64 Kbytes (50.04 Mbytes)
        49.67% of uncompressed filesystem size (103164.81 Kbytes)
Inode table size 24610 bytes (24.03 Kbytes)
        24.73% of uncompressed inode table size (99522 bytes)
Directory table size 26728 bytes (26.10 Kbytes)
        45.40% of uncompressed directory table size (58868 bytes)
Number of duplicate files found 33
Number of inodes 2979
Number of files 2596
Number of fragments 201
Number of symbolic links  89
Number of device nodes 0
Number of fifo nodes 0
Number of socket nodes 0
Number of directories 294
Number of ids (unique uids + gids) 1
Number of uids 1
        root (0)
Number of gids 1
        root (0)
Snapped mqtt-publisher-dev_0.1_amd64.snap
pcoca@haswell16:~/snapcraft/mqtt-publisher)
u*  16.04 0:-  1:-*  2:--  3:-  4:-  5:-  6:-  7:-  8:-# 9:-                    1! 7h39m 1.09 4x2.3GHz 7.7GB6% 2016-06-02 17:10:13
```

Developer tools: Snapcraft

MQTT subscriber snap yaml

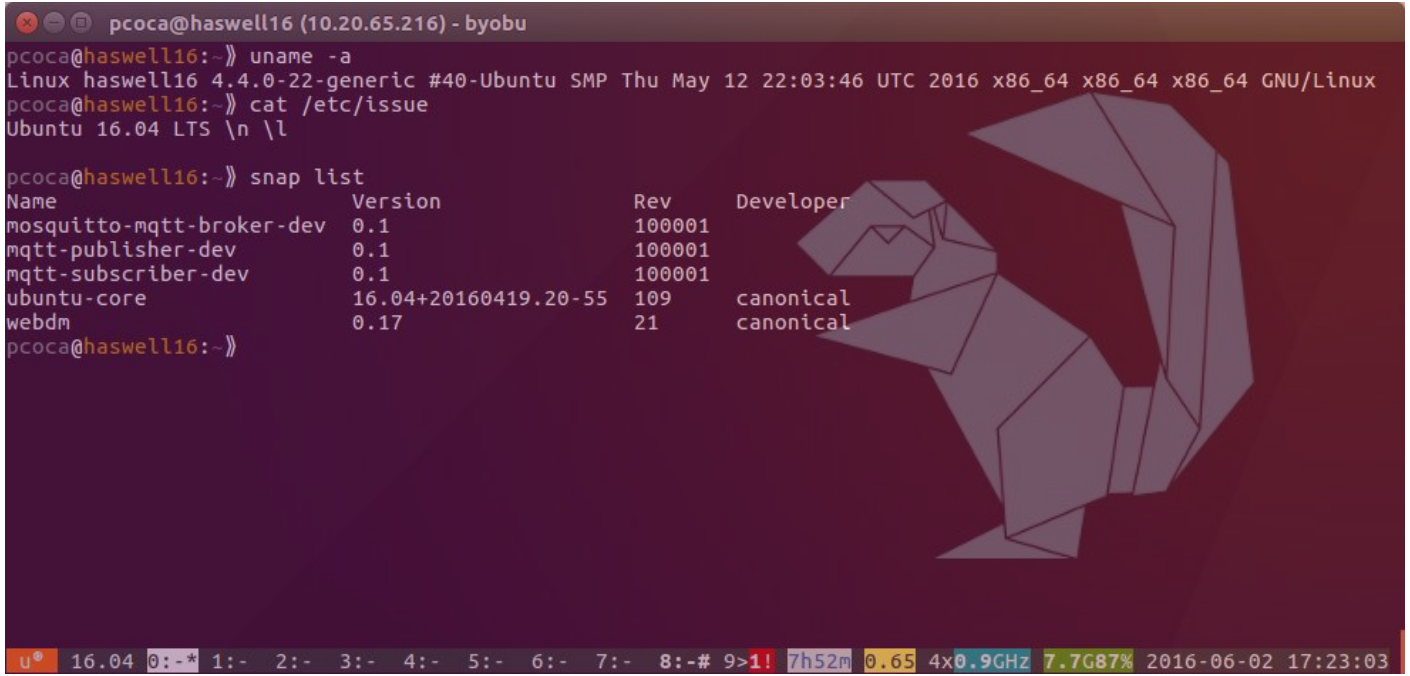
The command snapcraft on our directory will go through the whole snapcraft cycle and will create the package in squashFS format.

The result of running snapcraft will be the MQTT subscriber snap.



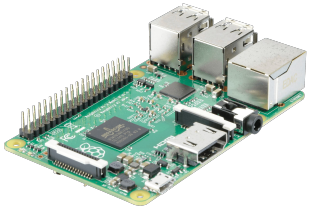
MQTT snaps setup on Ubuntu Classic

We can now install the snaps everywhere!



MQTT setup on different devices

MQTT Publisher	MQTT Broker	MQTT Subscriber
Ubuntu Core	Ubuntu Core	Ubuntu Core
Raspberry Pi 2	Intel NUC	Dragon Board 410c



MQTT testing

MQTT Publisher

```
pcoca@haswell16:~/snapcraft/mqtt-subscriber⟫ mqtt-subscriber-dev.subscribe --help
usage: subscribe [-h] [host] [port] topic

positional arguments:
  host        The IP or hostname of the MQTT server. Defaults to localhost.
  port        The port of the MQTT server. Defaults to 1883.
  topic       The topic to subscribe to.
```

MQTT Broker

MQTT Subscriber

```
pcoca@haswell16:~/snapcraft/mqtt-publisher⟫ mqtt-publisher-dev.publish --help
usage: publish [-h] [host] [port] topic payload

positional arguments:
  host        The IP or hostname of the MQTT server. Defaults to localhost.
  port        The port of the MQTT server. Defaults to 1883.
  topic       The topic to publish to.
  payload     The payload to send to the topic.
```

MQTT testing

MQTT Publisher

```
⟫ mqtt-publisher-dev.publish temperature 11
```

MQTT Broker


```
pcoca@haswell16:~/snapcraft/mqtt-subscriber⟫ mqtt-subscriber-dev.subscribe "temperature"
MQTT subscriber connected.
temperature b'11'
```

MQTT Subscriber

Ubuntu Core Interfaces

Used to grant access to resources, e.g. to give permission to do something or to control something.

Snaps can either offer (implement) or use (consume) interfaces

Interfaces are a key component in secure and flexible access to feature implemented in system or third party snaps.

- Resource interfaces: physical devices or similar.
  `front-camera, green-led...`
- Behaviour interfaces: associated with the demeanour of the snap
  `network-listening, timezone-managing...`

Ubuntu Core Interfaces

Check the interfaces on your Ubuntu Core system

```
pcoca@haswell16:~/ubuntu-device-flash-workspace⟫ snap interfaces
Slot                  Plug
:firewall-control     -
:home                 -
:locale-control       -
:log-observe          -
:mount-observe        -
:network              mosquitto-mqtt-broker-dev,mqtt-publisher-dev,mqtt-subscriber-dev,webdm
:network-bind         mosquitto-mqtt-broker-dev,mqtt-publisher-dev,mqtt-subscriber-dev,webdm
:network-control      -
:network-manager      -
:network-observe      -
:opengl               -
:snapd-control        webdm
:system-observe       -
:timeserver-control   -
:timezone-control     -
:unity7               -
:x11                  -
pcoca@haswell16:~/ubuntu-device-flash-workspace⟫
```

Hardware requirements: **15.04** VS **16**

	Ubuntu Core 15.04	Ubuntu Core 16
Processor / Architecture	x86 or ARM (32/64bits)	x86: 32 & 64bits ARM 32bits: RPi2 ARM 64bits: Qualcomm dragonboard
Memory	**256 MB**	**256 MB**
Storage	**4 GB**	**2 GB***
Available Connectivity	**WiFi, Ethernet, USB, BT 4.0...**	**WiFi, Ethernet, USB, BT 4.0...**

*The partition layout on UC16 has been modified.

Ubuntu Core: Build solutions easily

- Open Source
- Amazing developer tools
- Leverage multiple technologies
- Simple and decoupled architecture
- Simple packages: snaps
- Atomic updates and rollbacks

Ubuntu Core

Build

Choose from a variety of SoC, boards and existing libraries.

Differentiate

Monetize after device sales

Maintain

Over-the-Air updates with health-checks and rollback

Stores

- snap store as a service
- Full control over the store content
- Review queue for third parties
- Endless possibility

Your packages

New package

Manage available highlights

Edit Store preferences

	Package name	Version	Progress			
:-)	facedetectionweb	0.22	Published	Review	Feedback	Stats

Get help

Publishing an app
Get started
Packaging click apps
Application states
Choosing a license
Creating a good icon
Other forms of submitting apps
Security policy for click packages

Participate on AskUbuntu ›

A collaboratively-edited question and answer site for Ubuntu users and developers. 100% free, no registration required

Ask a question now ›

Report a bug on this site

Ubuntu Core Brand Store Management

- Branded store can be managed in the following ways:
    - Stores can inherit or exclude the content of other stores (branded or public)
    - Store owners can define their own policies regarding:
        - Security
        - Reviews
- A device can only point to a single store -- however that single store can inherit content from other stores.

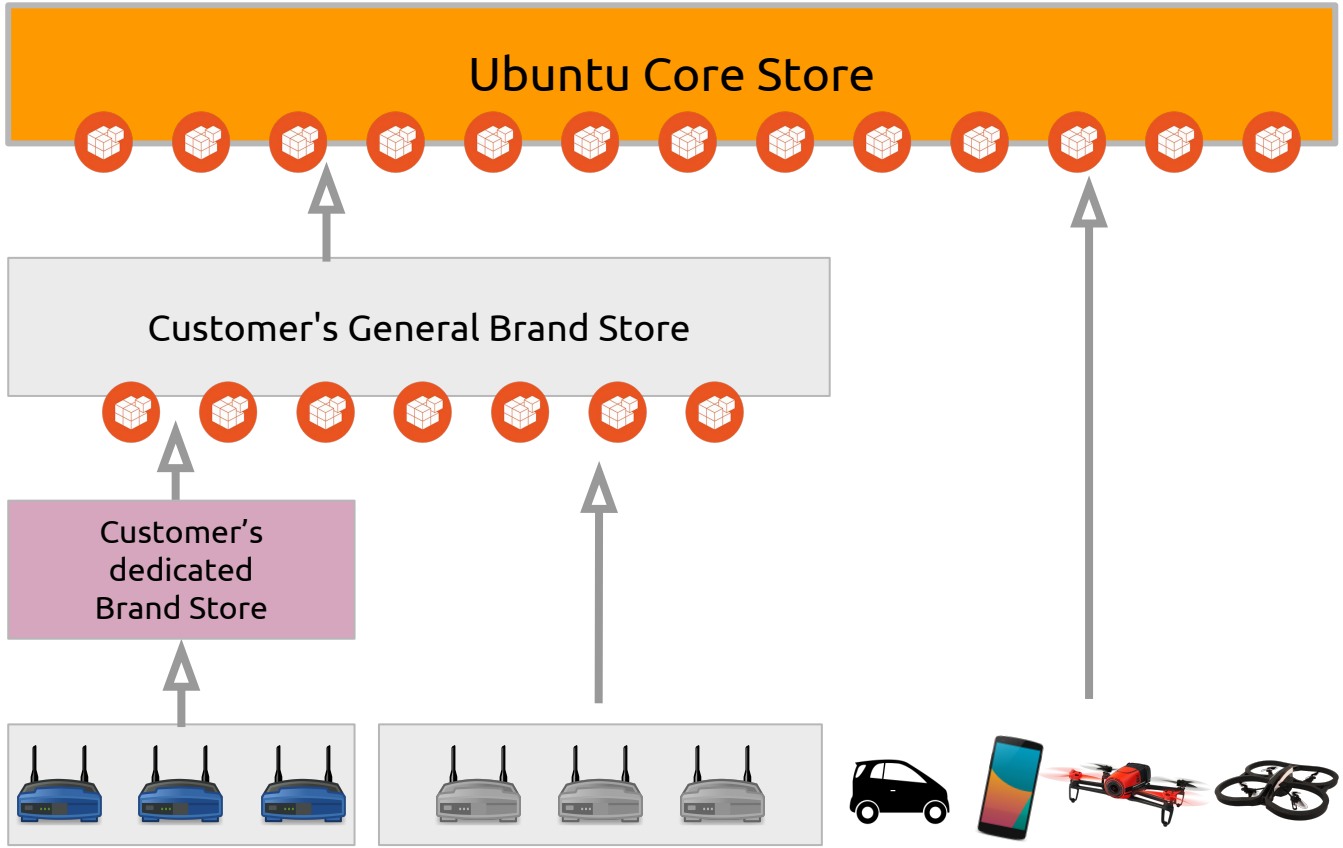
Ubuntu Core Brand Stores Set Up

- Setting a branded one is as easy as point the store/id entry in the gadget snap

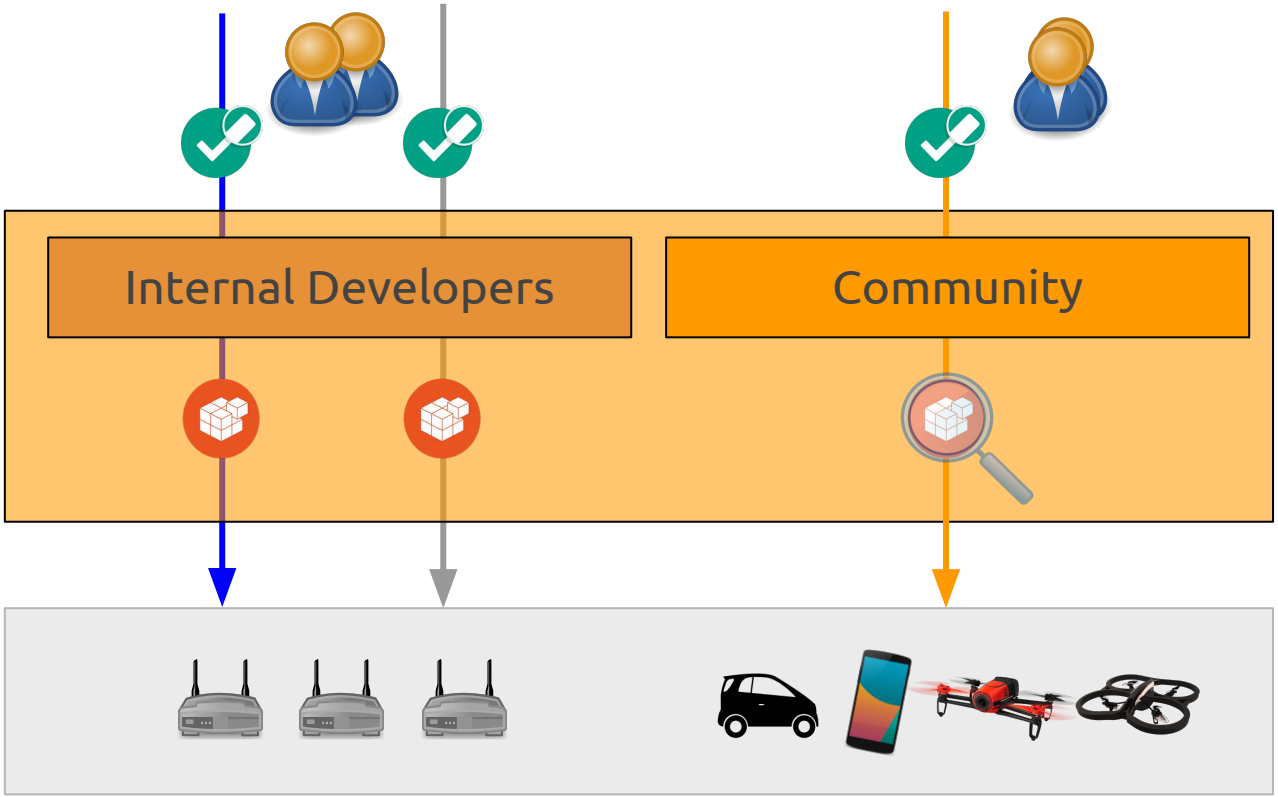
- Device authentication could allow to enforce the connection to branded stores.

- The gadget snap is used to setup and personalize the system according to certain rules and pre-installed software, for instance, pointing to a branded store.

Ubuntu Core Brand Store



Brand Store Example (II)



Ubuntu Core: Differentiate your product

- Revenue opportunities
- Design & Brand leverage
- Define your security and review policies
- Full control of your software stack

Ubuntu Core

Build

Choose from a variety of SoC, boards and existing libraries.

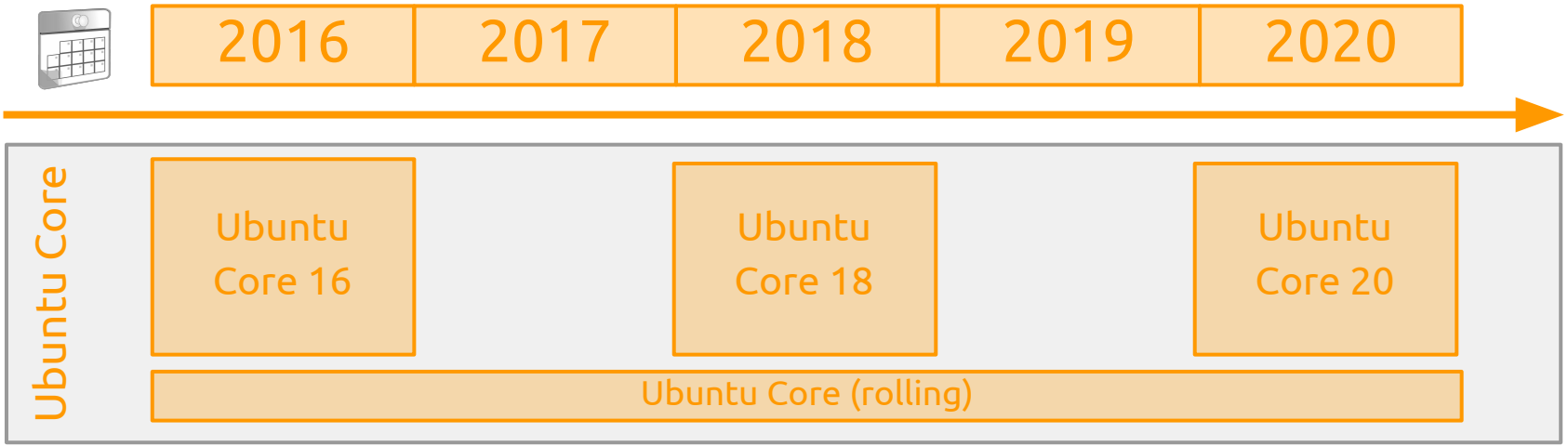
Differentiate

Monetize after device sales

Maintain

Over-the-Air updates with health-checks and rollback

Ubuntu Core release versions



Point in time releases (16,18, 20…) continue for a defined period. On those releases, we make only conservative and very stable changes.

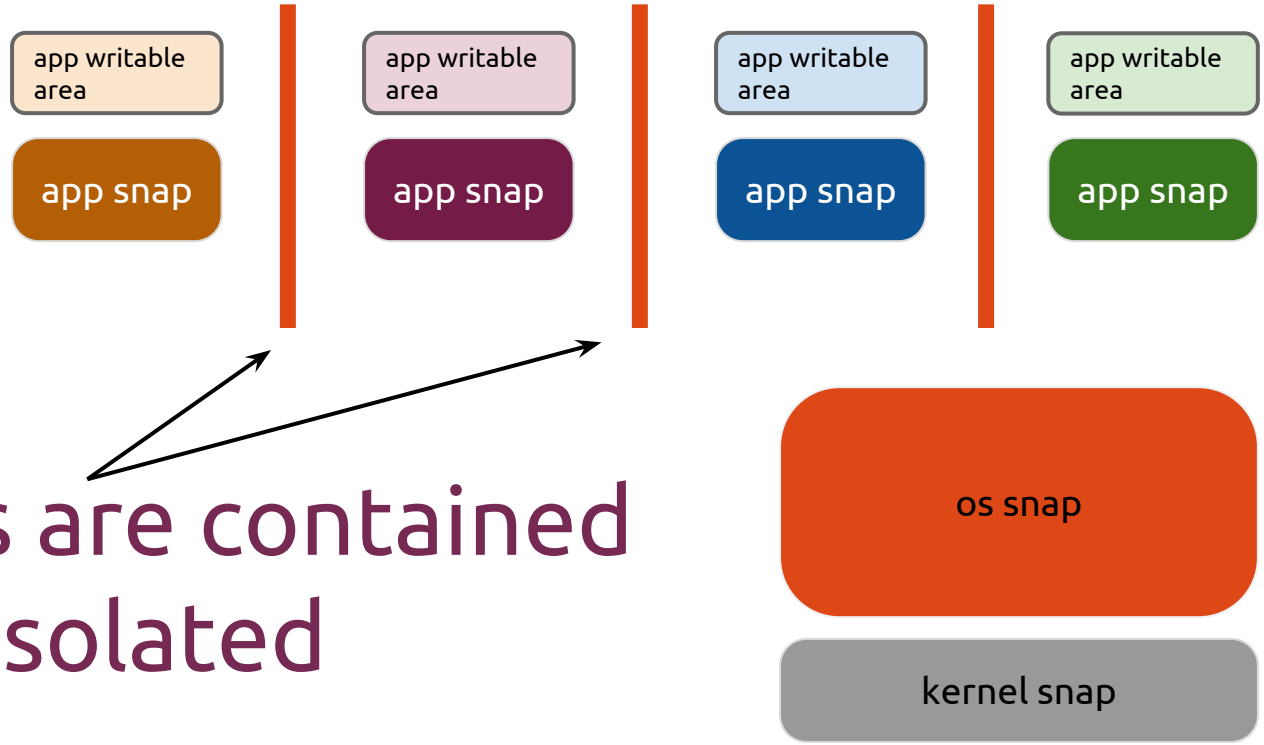
There is also a rolling release, where significant changes land in anticipation of the next stable releases.

LogJam, Shellshock, Heartbleed, libc, etc.

Ubuntu Core

Ubuntu Core apps and Ubuntu Core itself can be upgraded atomically and rolled back if needed.

A bulletproof approach that is perfect for deployments where predictability and reliability are paramount. It’s called “transactional” or “image-based” systems management.



Apps are contained and isolated

Apps confinement: Trust model

By default the **application snaps** are untrusted by the OS and:

- cannot access other applications' data
- cannot access non-app-specific user data
- cannot access privileged portions of the OS

Trusted by the OS **vs** Untrusted by the OS

Apps confinement: Technologies

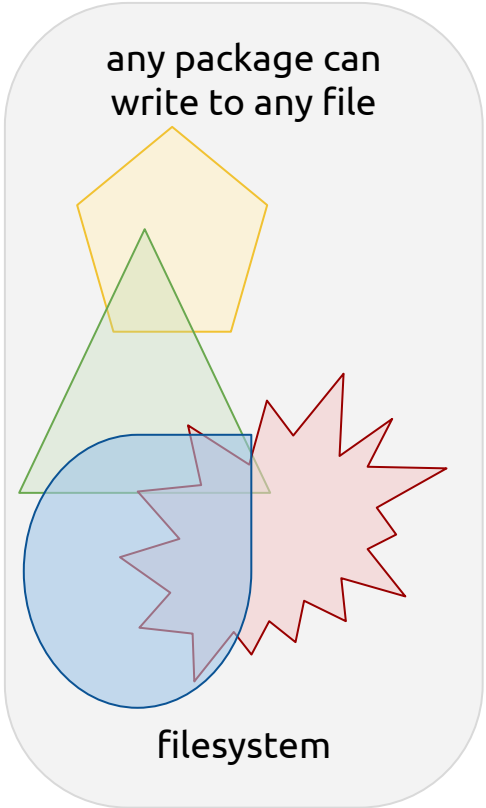
Several technologies are used by Ubuntu Core to:

- Implement the security sandboxing
- Implement the application isolation

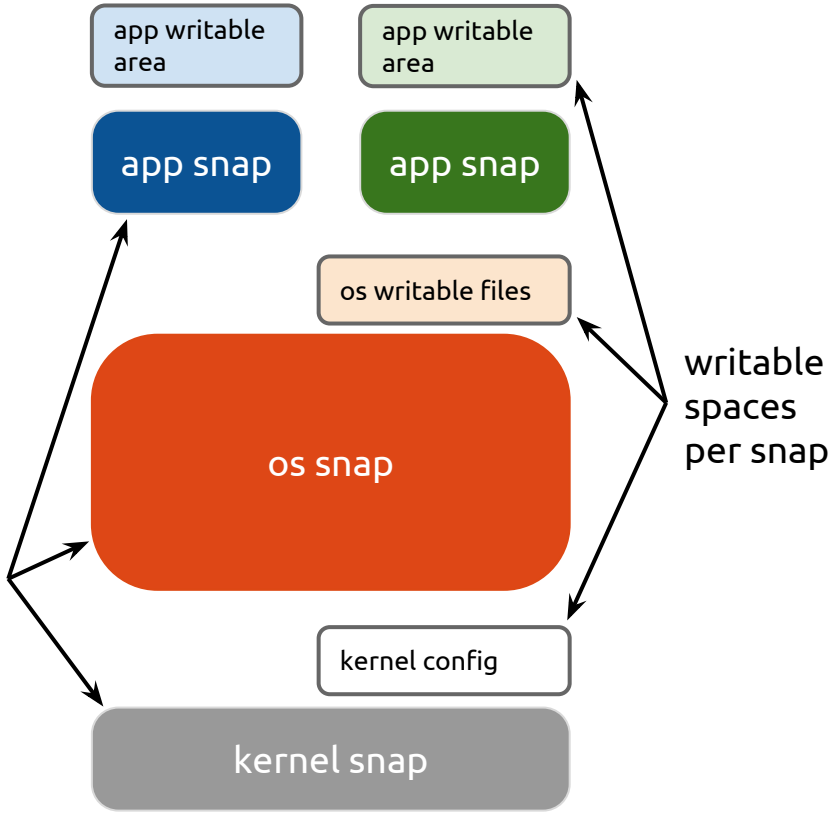
These technologies are mainly:

- **AppArmor**: A **Mandatory Access Control** system to confine programs and processes to a limited set of resources. (Application Isolation)
- **Seccomp**: A **secure computing mode** that provides an application sandboxing mechanism.

Classic Ubuntu



Ubuntu Core



Health Checks

System health checks are a lightweight & fast (asynchronous) mechanism to get the status of an Ubuntu Core system.

With health checks, Ubuntu Core can:

- check that all applications and services provided by each snap are up and running
- check if snaps that aren't working as expected, and rollback problematic snaps to their last known working version

With health checks, Ubuntu Core stores can:

- alert developers when their snaps don't work as expected
- stop bad releases from hitting more devices

Canary and phased updates

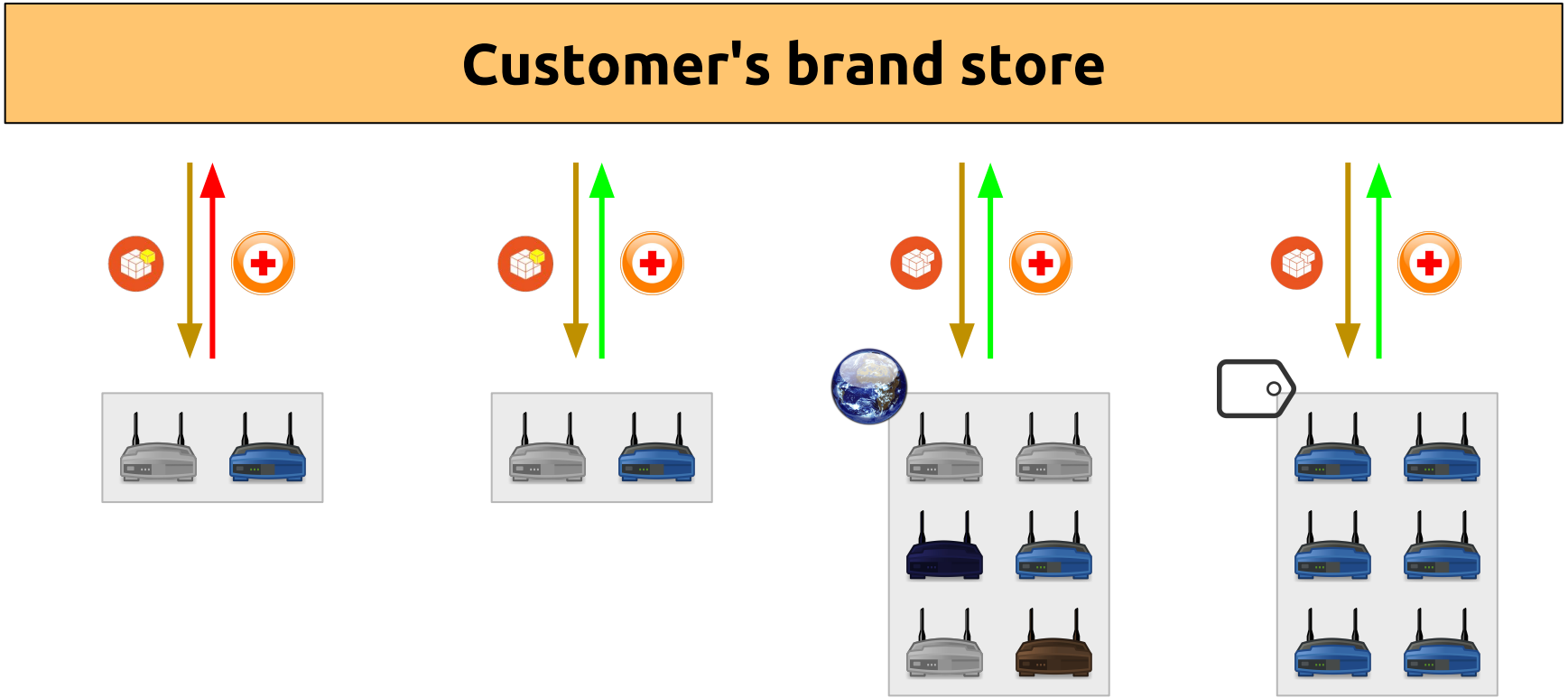
Canary updates are updates sent to a subset of devices in order to evaluate the impact of that update.

Canary updates and health checks help us to assess the updates and plan accordingly so we can spread the updates over time, over device subsets, etc.

Phased updates allow to send out updates to an increasing number of devices sub set.

Phased updates can for instance update 1% of the device set, get information and update the 5%, 25%, etc.

Health Checks, Canary and Phased Updates



Ubuntu Core: Effortless maintenance

- Redundancy and reliability
- Canary updates, Health Checks and phased updates
- Application sandboxing
- Resource isolation
- Mission Critical Security
- Reliable cadence and LTS releases

Ubuntu Core: Getting involved

Ubuntu Core: Getting involved

- **Ask a question on Ask Ubuntu**
  - If you're stuck on a problem, someone else has probably encountered it too and they can help you. Take a look at the "ubuntu-core" tag on Ask Ubuntu or ask a question.
- **Join our real time chat** (#snappy on freenode.net)
  - Share your projects and ask other developers for support. This high-bandwidth IRC channel is a good place when you are looking for a quick answer to a single question.
- **For app developers**
  - Reach out to other snap developers by using the"snapcraft" tag on Ask Ubuntu, join the snapcraft mailing list and make sure to join the Ubuntu App Developers Google+ community.
- **Ubuntu Core Clinics**
  - Our Ubuntu Core Clinics are video sessions where we want to inform you about what's new, show some demos, answer questions and where you can bring a problem and we are going to look at it together. Watch the sessions which already happened and check out which new ones are planned.

Ubuntu Core: References

Ubuntu in your everyday life

Ubuntu phones from Meizu and BQ

Anyone can install Ubuntu on Google Nexus tablet or phone

Are all running on Ubuntu

Empowering social media

Yep. More Ubuntu.

Snapchat · Pinterest · Reddit · Instagram

Enabling your daily tasks

Shopping at Walmart? Everyday prices. Everyday Ubuntu.

Ubuntu is serious business for Bloomberg

Paypal and Dropbox are both served by Ubuntu

Watching a movie? WETA Digital and Netflix run Ubuntu

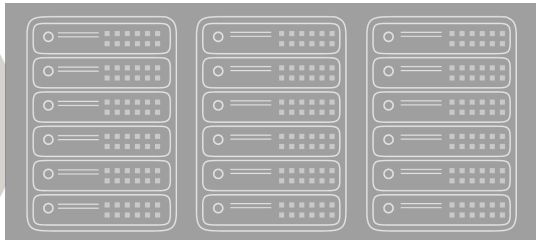
8-18 billion page views per month and hosted on Ubuntu

Controlling BYU Mars Rover

Ubuntu is everywhere!

Behind the largest supercomputer

- Tianhe-2. 33.86 PFLOPS
- All crunching on Ubuntu!



20 million

launches of Ubuntu instances in 2015 in:

public cloud
AWS, Microsoft Azure, Google Compute Engine, Rackspace, Oracle Cloud, VMware

private cloud
OpenStack - Including some of the world's largest private clouds, like Deutsche Telekom

bare metal
Ubuntu at scale on bare metal with MAAS

There's also plenty of Ubuntu in:

Kubernetes and Apache Mesos

Cloud Foundry and Heroku

Running within the International Space Station

In November 2015

2 million
new Ubuntu Cloud instances launched

Ubuntu Cloud instances:

- 67,000 Ubuntu Cloud instances launched in 24 hours
- 28,000 Ubuntu Cloud instances launched in 1 hour
- 46 Ubuntu Cloud instances launched each minute
- Approx. 1 Ubuntu Cloud instances launched each second

Ubuntu Core 16

Q&A

ubuntu.com/things