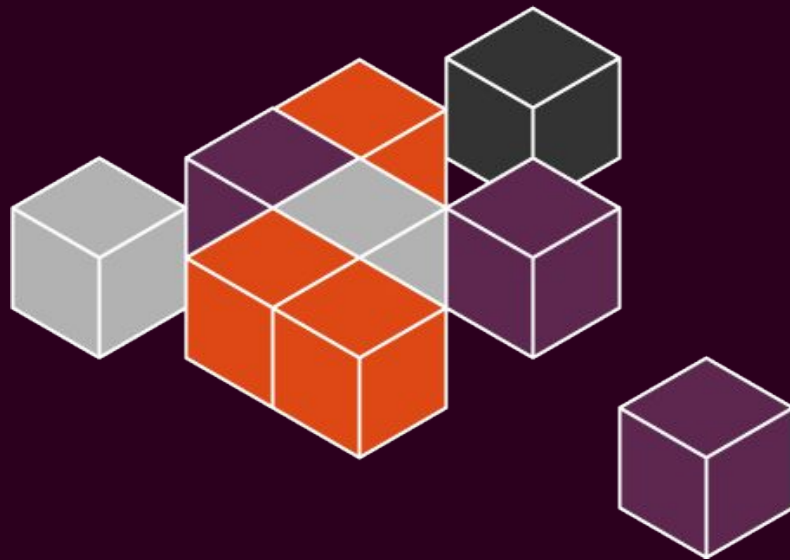


Snappy Ubuntu Core

Enabling secure devices with app stores

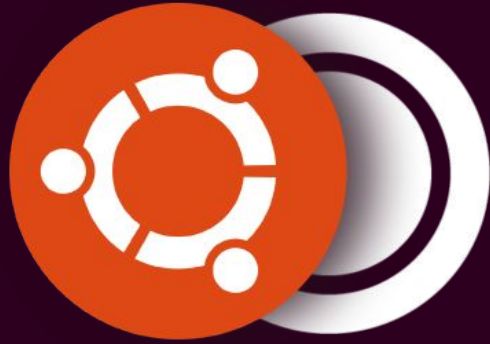


XiaoGuo, Liu
xiaoguo.liu@canonical.com

Agenda



- Company introduction
- Intro to snappy Ubuntu Core for IoT
 - architecture
 - security
 - store infrastructure
- Docker on snappy ubuntu core
- Demos



We are the company
behind Ubuntu

Canonical and Ubuntu | Introduction

Ubuntu is an open-source operating system, currently established on server, cloud, desktop and mobile devices.

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

2004
FOUNDED

700+
EMPLOYEES

30+
COUNTRIES



Ubuntu: where are we now?



The world's 3rd most popular PC OS

90% of the Linux market



30,000,000 users

and counting ...



#1 Open Source Desktop OS



#1 Guest OS in Public Clouds

Azure, AWS, Google Compute..

#1 Openstack OS (65% of prod)

A converged story

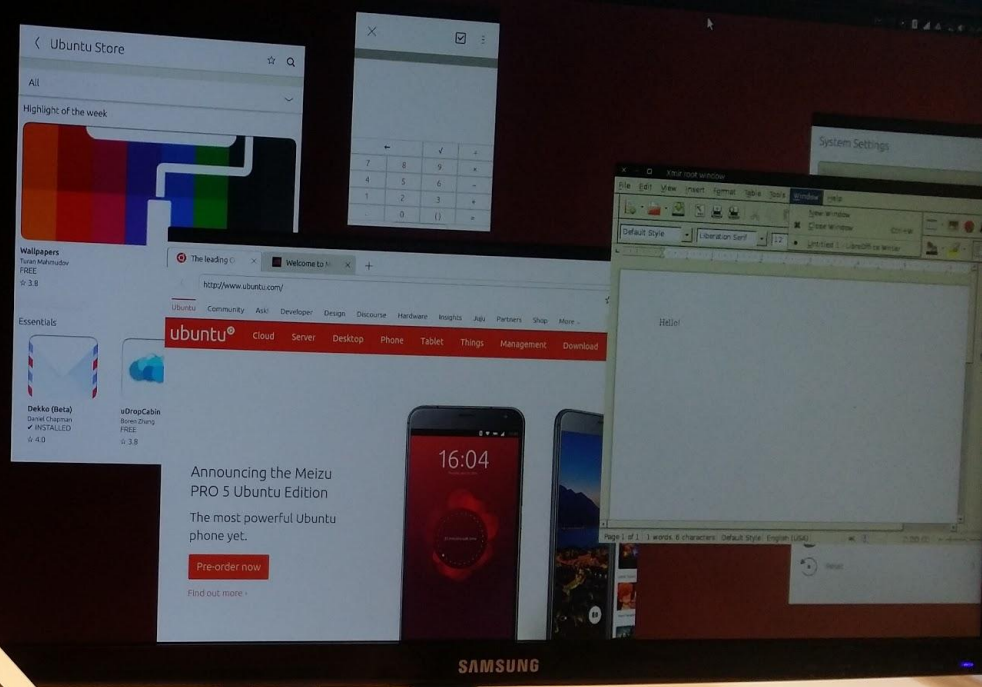
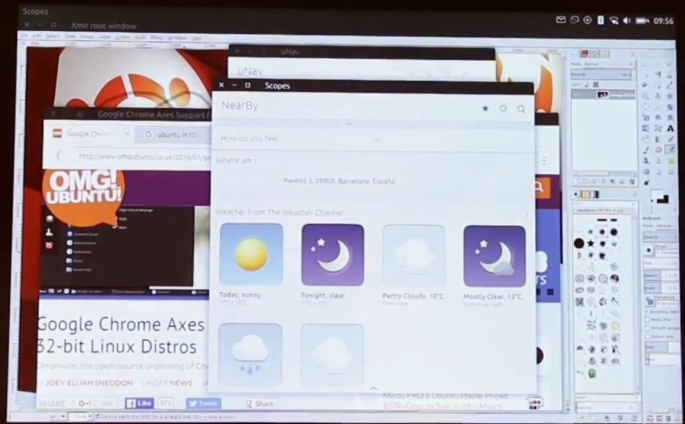
A single platform
for multiple form factors

A fully converged OS
for desktops, smartphones,
tablets and IoT



A development framework
for creating cross-product
applications and services

And also...
Centralized device management
User identity management
Store infrastructure



A tablet in desktop display mode



A phone or a computer?

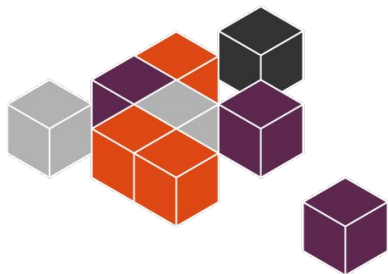


snappy Ubuntu Core

www.ubuntu.com/internet-of-things

Snappy Ubuntu Core is small, fast, efficient

Ubuntu's minimal expression



Faster, more reliable, and stronger security guarantees



Transactionally updated Ubuntu for clouds and devices



A new, simpler application packaging system

s Snappy Ubuntu Core targets



POS



KIOSK



Signage



Gaming



Thin Client



Automation

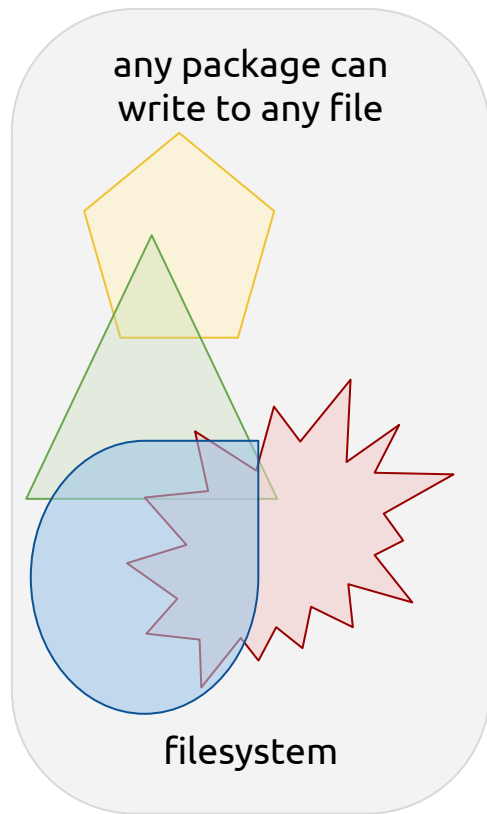


Panel PC

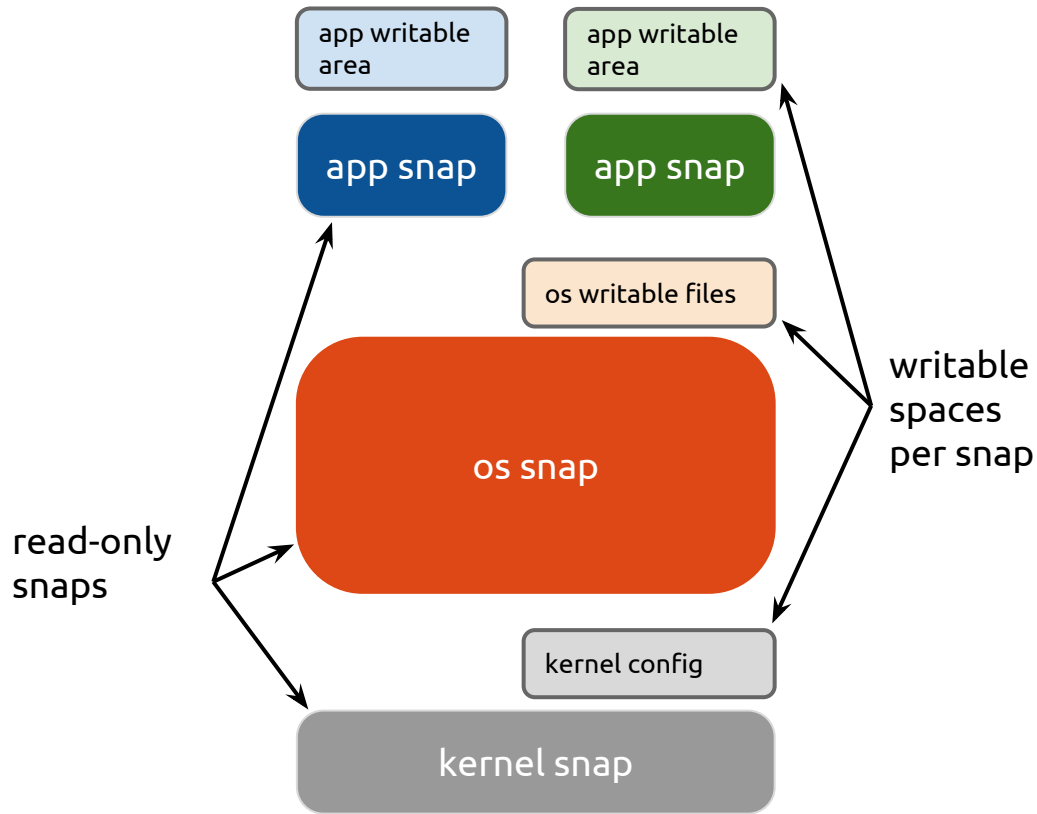


IoT

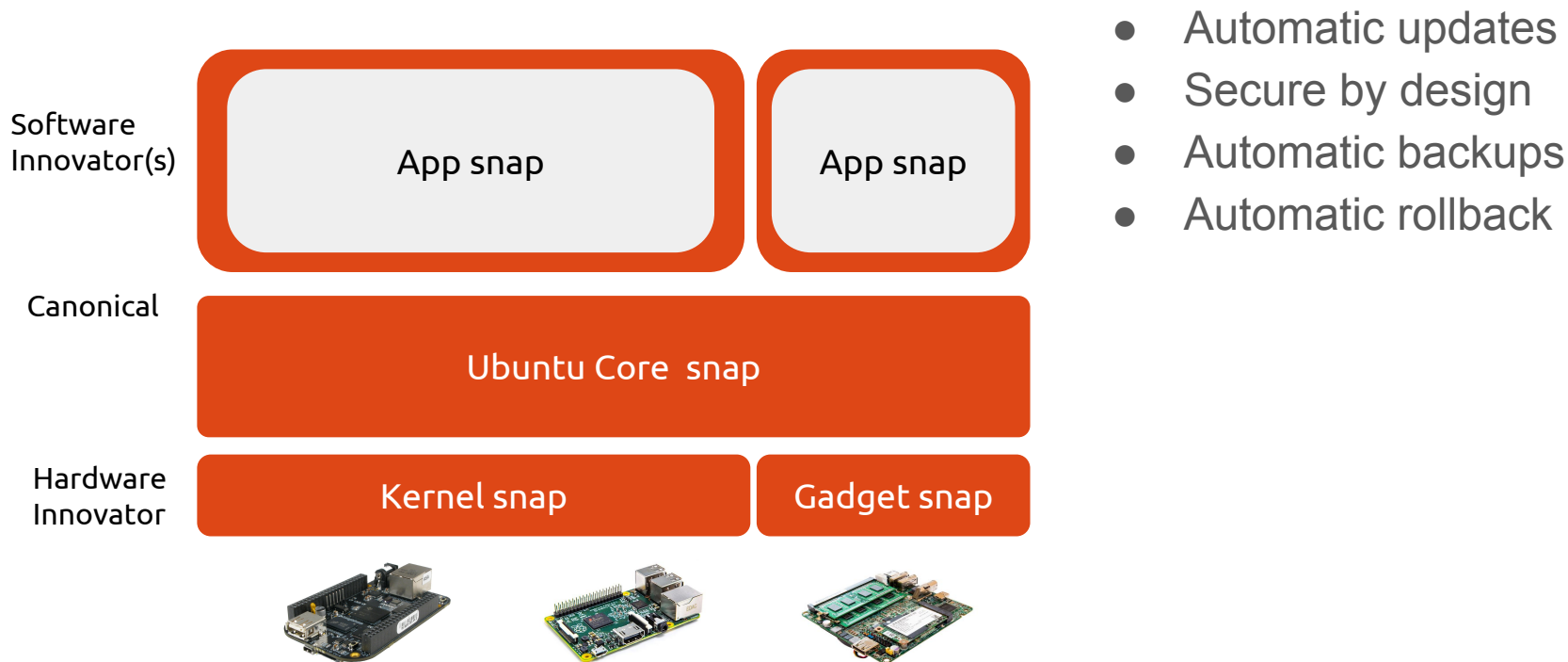
classic ubuntu



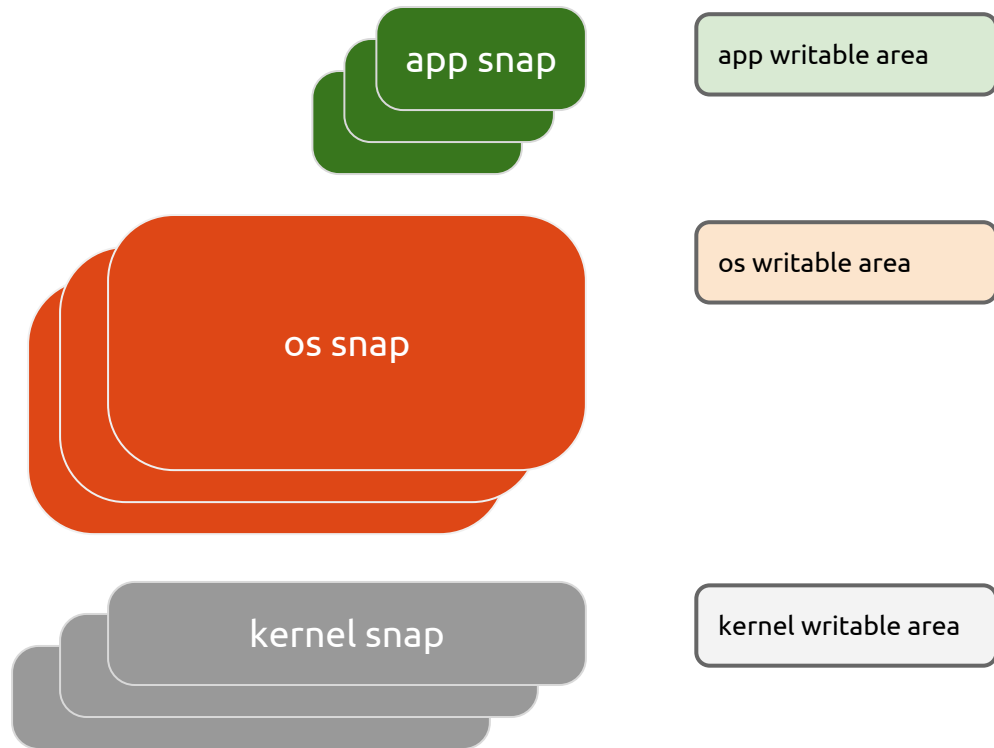
snappy ubuntu

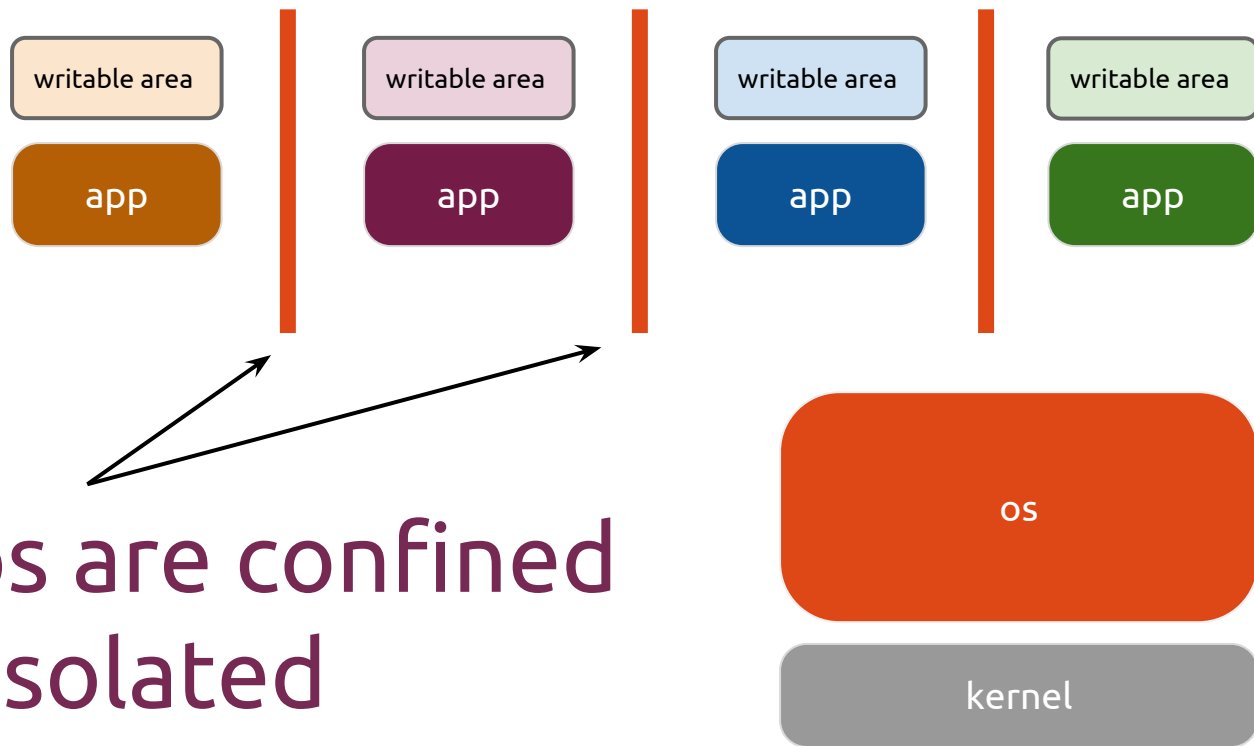


Snappy system architecture



A, B & Factory snap versions





Snaps are confined
and isolated

Snap files after installation



A snap packages everything needed into a single snap file. For example, python runtime env needs to be packaged into the snap package for python apps

Snap locations after installation

Mount point	Writable	Description
/apps/<app-name>/<version>	No	Read only location for application data, binaries and libraries that are immutable.
/var/lib/apps/<app-name>/<version>/	Yes	Writable location which is mounted through to the writable partition, this is where applications write out during their runtime.
/home/user/apps/<app-name>/<version>	Yes	User specific directory that contains all configuration and user-specific data pertaining to the application in question.

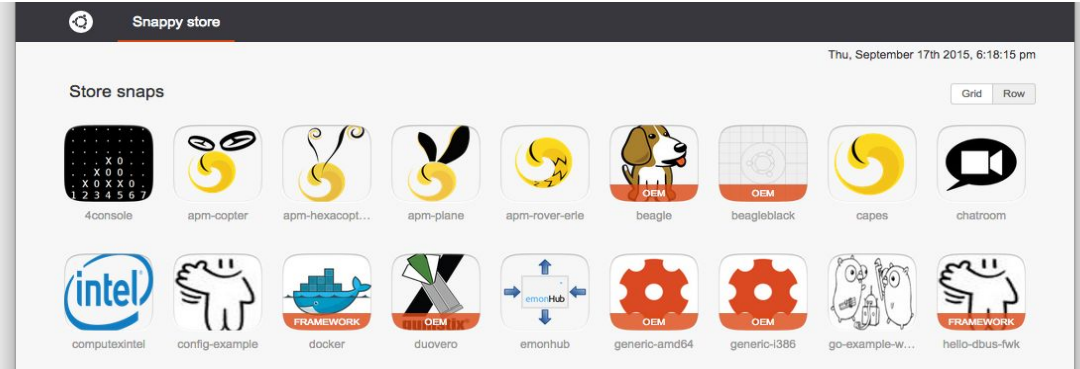
data from app with root can be written to **var/lib/apps/<app-name>/<version>/**

However, if an app does not have root privs, the best place for dumping data is the user specific directory **/home/<user>/apps/<app-name>/<version>**

Apps, Services and SaaS



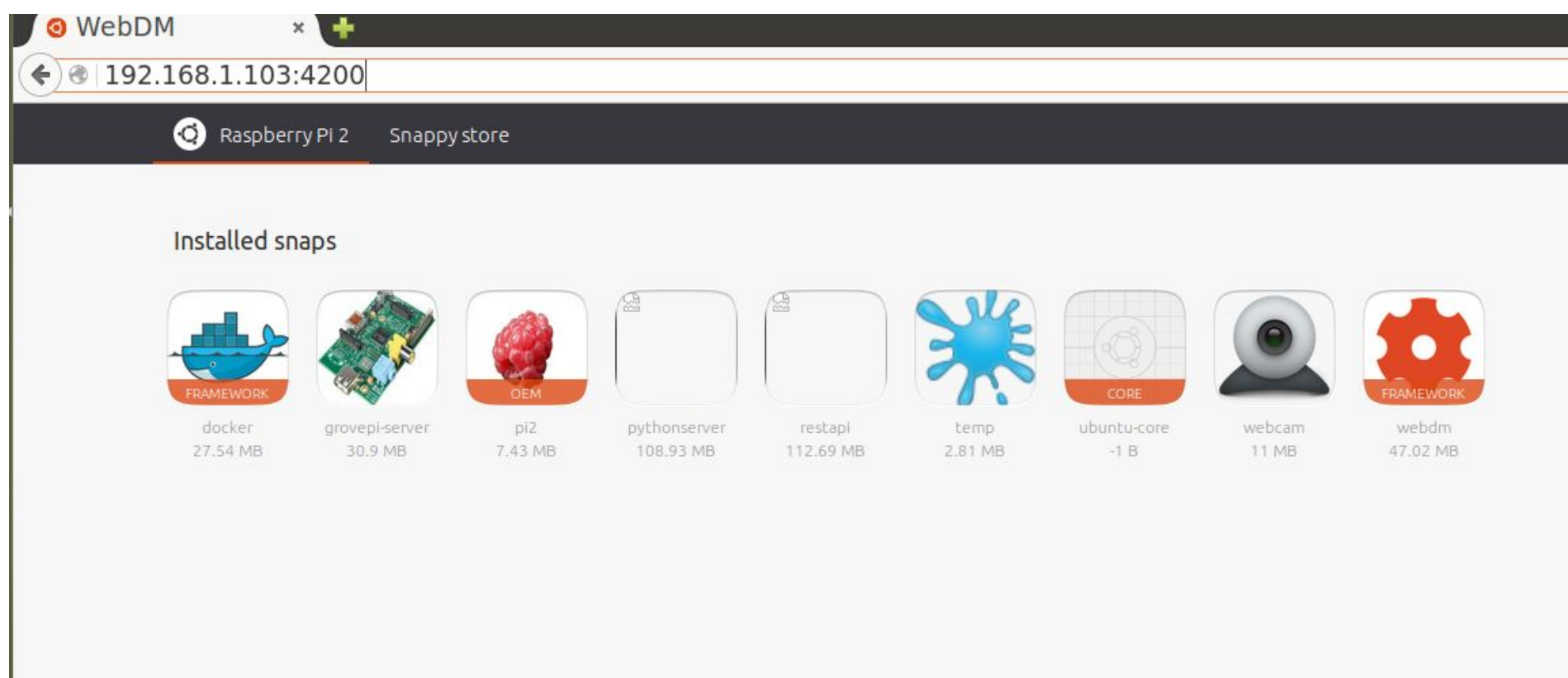
\$600 drone (hardware)



Infrastructure

Certification. Support. Assurance. Security.
Maintenance. Compatibility.

Snappy ubuntu core store



<https://myapps.developer.ubuntu.com/dev/click-apps/?format=snap>

A custom app store for any device



Publishers control snap distribution and updates directly



Snaps can extend the base operating system



Base operating system is free and built on familiar Ubuntu

Snap development with snapcraft

Snapcraft introduction

What is [snapcraft](#)?

Snapcraft is a tool which lets developers :

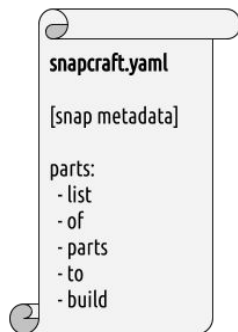
- package their software as a snap
- incorporate components from different sources and build technologies or solutions
- **snapcraft is a tool running on Ubuntu OS instead of ubuntu core**

What is a snap package?

Snaps are the packaging mechanism used in **snappy Ubuntu Core**. Snaps are:

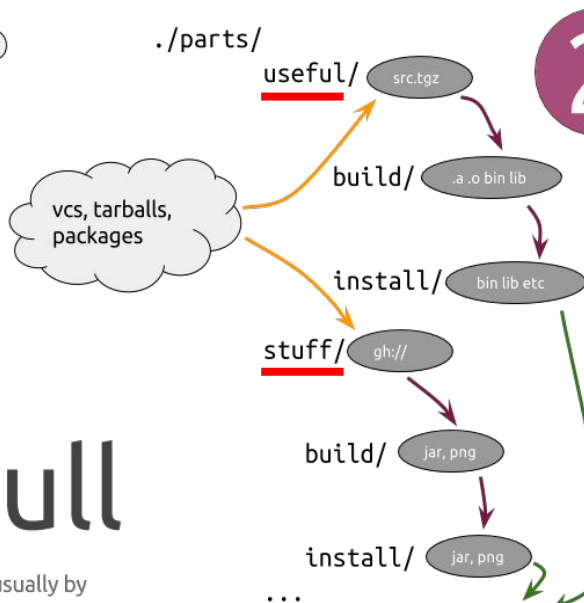
- self contained: developers can include all required dependencies in their snaps so as to remove any dependency on system libraries.
- made from reusable components called **parts**: developers can leverage existing open source projects by integrating them as part of their snap.

Snapcraft package build



1 Pull

Populates the parts, usually by fetching source.



2 Build

Builds each part individually in its own build/ directory and installs it in its own install/ directory.

3 Stage

Consolidates built files from all parts into a "stage" tree. A staging area.



4 Snap

Brings only desired shipping files from ./stage/ to ./snap/ for final testing and compression.



Developer tools: Snapcraft 2.0 lifecycle

Snapcraft 2.x





```
snapcraft

Usage:
  snapcraft [--version | --help] [options] COMMAND [ARGS ...]

Options:
  -h --help          show this help message and exit
  -v --version        show program version and exit
  -V --verbose        print additional information about command execution

The available commands are:
  list-parts          List available parts which are like "source packages" for snaps.
  list-plugins        List the available plugins that handle different types of part.
  init                Initialize a snapcraft project.
  add-part             Add a part to your snapcraft.yaml, interactively presenting
                      options.
  help                Obtain help for a certain plugin or topic
```



```
The available lifecycle commands are:
  clean              Remove content - cleans downloads, builds or install artifacts.
  pull               Download or retrieve artifacts defined for a part.
  build              Build artifacts defined for a part.
  stage              Stage the part's built artifacts into the common staging area.
  strip              Final copy and preparation for the snap.
  snap               Create a snap.
```

See 'snapcraft COMMAND --help' for more information on a specific command.

For more help, visit the documentation:
<http://developer.ubuntu.com/snappy/snapcraft>

Developer tools: Snapcraft plugins

Snapcraft supports several technologies through the current plugins available.

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

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

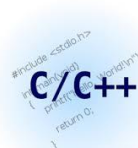
Supported languages on snappy ubuntu core



[Link](#)



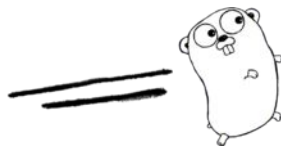
[Link](#)



[Link](#)



[Link](#)



Go [Link](#)

Or bring your own plugin for
your preferred framework!!



[Link](#)

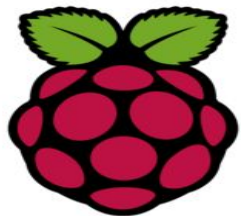


[Link](#)

<http://bazaar.launchpad.net/~snappy-dev/snapcraft/core/files/head:/examples/qmldemo/>

Supported targets

Snappy runs everywhere: on development boards, Internet-enabled devices, on cloud instances or even locally in a Virtual Machine. Where do you want to install it?



RaspberryPi

Ubuntu Core allows you to quickly install apps on your board in just a few clicks, get started [on the Raspberry Pi 2](#) >



Ubuntu Core can also be easily installed on other architectures like Intel® 64 bits, get started on [Intel® NUC device](#) >

Snappy can also be installed on the PandaBoard, BeagleBone, Gumstix and Odroid boards...

[See all enabled devices](#) >

Microsoft Azure

Try Ubuntu Core on the [Microsoft Azure cloud](#) >

 Google Cloud Platform

Try Ubuntu Core on the [Google Compute Engine cloud](#) >

 amazon
web services™

Try Ubuntu Core on the [Amazon Elastic Compute Cloud](#) >

ubuntu®

Try Ubuntu Core running on bare metal [x86 devices](#) >

 KVM  VAGRANT

Try Ubuntu Core on a virtual environment [locally with KVM](#). Alternatively try our and [Vagrant](#) images.

Snapcraft example

name: **piglow-snappy**

version: 1.0

vendor: XiaoGuo, Liu <xiaoguo.liu@canonical.com>

summary: Piglow API for controlling the piglow lights

description: This is the webserver API to control the piglow. The piglow can be controlled by install "piglow" on ubuntu phone

icon: icon.png

services:

piglow:

start: bin/piglow

description: Start a piglow-server for serving REST APIs

caps:

- network-client
- network-service

parts:

piglow:

plugin: go

source: ./src/piglow

Snappy ubuntu core security policies

System policy:

Policy vendor: ubuntu-core

Policy version: 15.04

Templates:

default

unconfined

Caps:

network-admin

network-client

network-firewall

network-service

network-status

networking

snapped

Framework policy:

Templates:

Caps:

docker_client

Success stories



<http://www.dji.com/product/phantom>



Open Source Robotics Foundation





mycroft



An Artificial Intelligence for Everyone

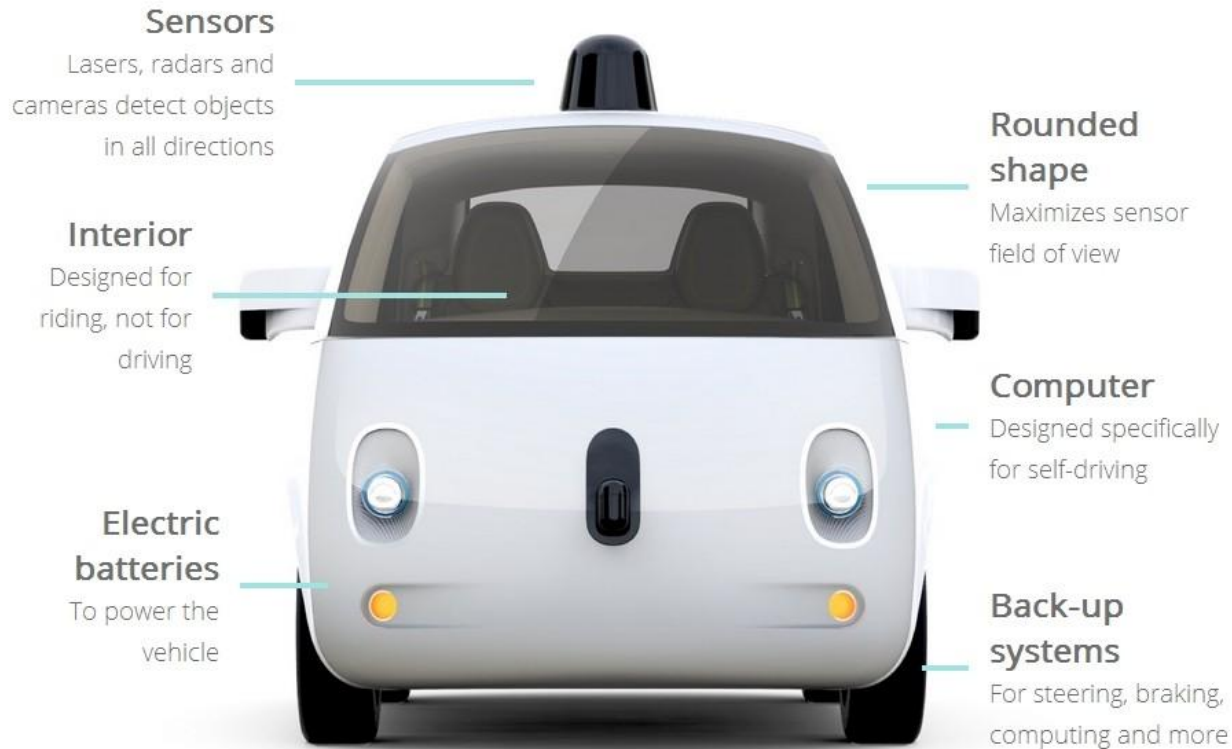
<https://mycroft.ai/>



<https://firstbuild.com/smart-refrigerator/>

<http://erlerobotics.com/>









Minimum system requirements

Processor Architecture

Bay Trail-I, Broxton (with 16.04), Core i3/5/7, Xeon

Memory

256MB

Flash Storage

2GB storage

Available Connectivity types

WiFi, Ethernet, USB, BT4.0 BLE, ..

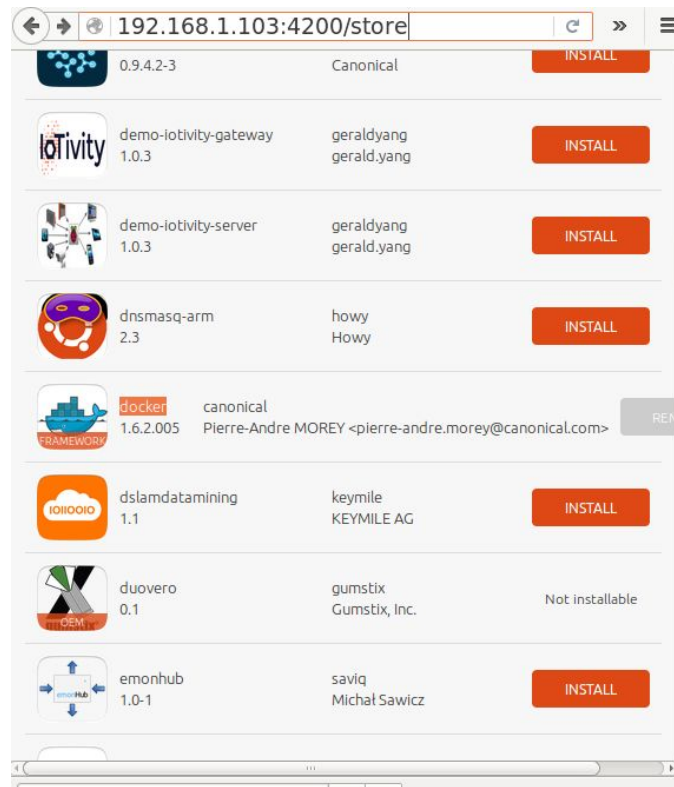
Docker on snappy ubuntu core

We can install docker onto snappy using the following command

```
$ sudo snappy install docker
```

```
(RaspberryPi2)ubuntu@localhost:~$ snappy list -v
```

Name	Date	Version	Developer
ubuntu-core	2015-11-13	3	ubuntu*
ubuntu-core	2016-01-20	6	ubuntu
docker	2015-12-04	1.6.2.005	canonical*
grovepi-server	2016-01-25	ILEJILdPdTLU	sideload
grovepi-server	2016-01-26	ILHPPaYRKJPQ	sideload*
pythonserver	2015-12-09	IHQfYWLQUdCG	sideload*

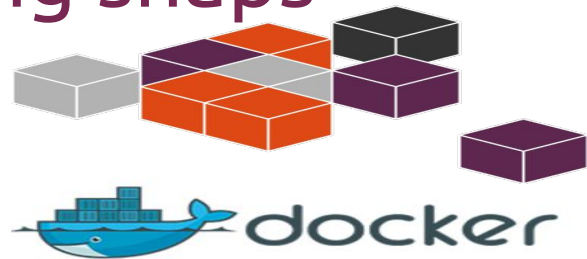


Docker usage example for building snaps

- Cross-compiling is not possible for now
- Build snap on a natively targeted ubuntu system
 - Inside snappy ubuntu, install docker framework
 - Inside docker, install a target ubuntu container
 - Inside the container, build snaps for apps
- Install the snap onto the device

[How to compile snappy apps for armhf](#)

如何为我们的Snappy Ubuntu应用编译并打包Snap (2)



.snap



Snappy Ubuntu Core Progress & Roadmap

15.04

UEFI Capsule Updates

UEFI Secure Boot

TPM 2.0 towards end of October
(contingent on Intel schedule)

Management (Local) over REST API

Branded Store with Device
Authentication

Snapcraft

16.04 LTS

ROS 2.0

Classic dimension on snappy devices

Factory process

Device Identity, Authentication and Store
Authorization

Network Manager support

Everything as a snap (Kernel, OS)

Snaps as blobs on disk

Network App Confinement

Disk Encryption

Snappy Recovery System

LXD framework on snappy

snappy ubuntu core technical support

The screenshot shows an XChat window titled "XChat: liuxg_ @ FreeNode / #snappy (+cnt)". The interface is divided into three main sections: a channel list on the left, a central chat log, and a user list on the right.

Channel List (Left): A list of IRC channels including #pes-meeting, #phablet, #premium, #sdk, #snappy (highlighted with a red checkmark), #snappy-internal, #SnappyChina, #tech, #u1-internal, #webapps, #yue, Kay, freenode, ##unavailable, #qt-labs, #qt-quick, #snappy-training, and #ubuntu.

Chat Log (Center): Displays the history of messages in the #snappy channel. The messages include:

- A header: 'snappy/ || Snappy for Bugs: https://bugs.launchpad.net/snappy :P'
- A timestamped message: [10:35] * Loaded log from Wed Jan 13 10:27:55 2016
- A timestamped message: [10:35] * Now talking on #snappy
- A timestamped message: [10:35] * Topic for #snappy is: Our 4 Forces: Snappy for Things: http://www.ubuntu.com/things || Snappy for Cloud: http://www.ubuntu.com/cloud/tools/snappy || Snappy for App Devs and Porters: https://developer.ubuntu.com/en/snappy/ || Snappy for Bugs: https://bugs.launchpad.net/snappy :P
- A timestamped message: [10:35] * Topic for #snappy set by asac!~asac@debian/developer/asac at Wed May 20 19:38:30 2015
- A timestamped message: [10:36] * liuxg has quit (Ping timeout: 276 seconds)
- A timestamped message: [10:41] * dvladar (~nogradpit@cpe-77.38.73.200.cable.t-1.si) has joined #snappy
- A timestamped message: [10:46] * dvladar has quit (Ping timeout: 265 seconds)
- A timestamped message: [10:58] * travnewmatic has quit (Ping timeout: 272 seconds)
- A timestamped message: [11:03] * iahmad (~iahmad@39.33.222.92) has joined #snappy

User List (Right): A list of users currently in the channel, including abner, alecu, alesage, alex-abreu, andyrock, apw, Armenta, asac, balloons, barry, bellyfeel, benoitc, beowulf, beuno, bigcat_, BjornT, and blr. The status "0 ops, 143 total" is shown at the top of this list.

The bottom of the window shows the input area with the text "liuxg_" and an empty text box for typing.

snappy support mailinglist: snappy-app-devel@lists.ubuntu.com

File a bug: <https://bugs.launchpad.net/snapcraft/+filebug>

References

- [Snappy Ubuntu Core - Application Developer Manual 15.04](#)
- [snapcraft master release](#)
- [Snappy ubuntu at blog.csdn.net](#)
- [https://code.launchpad.net/~snappy-dev/snappy-hub/snappy-examples](#)
- [https://github.com/liu-xiao-guo?tab=repositories](#)
- [https://github.com/ubuntu-core/snapcraft/tree/master/examples](#)
- File a bug at [https://bugs.launchpad.net/snapcraft/+filebug](#)
- Snappy mailing list: [snappy-app-devel@lists.ubuntu.com](#)
- snapcraft doc (2.x): [https://github.com/ubuntu-core/snapcraft/tree/master/docs](#)
- snapcraft doc (1.x) [https://github.com/ubuntu-core/snapcraft/blob/1.x/docs/get-started.md](#)
- [利用snapcraft为我们的Snappy Ubuntu应用打包](#)
- [http://www.ubuntu.com/internet-of-things](#)



snappy

ubuntu.com/snappy