Ubuntu Core 101

From the bootloader to snaps

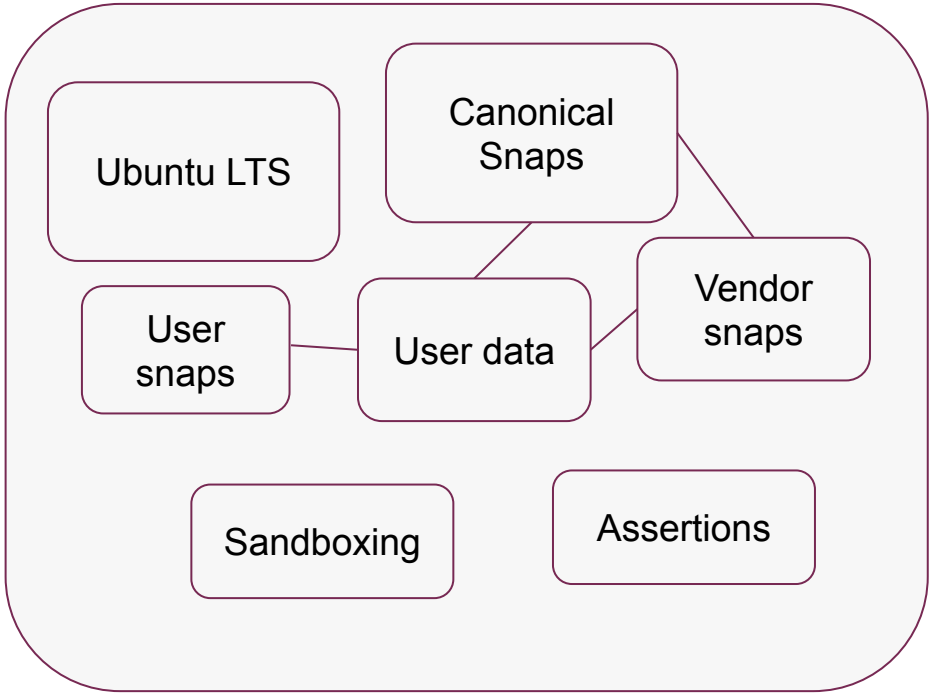
Maciej Borzęcki (snapd)

CANONICAL ubuntu

What?

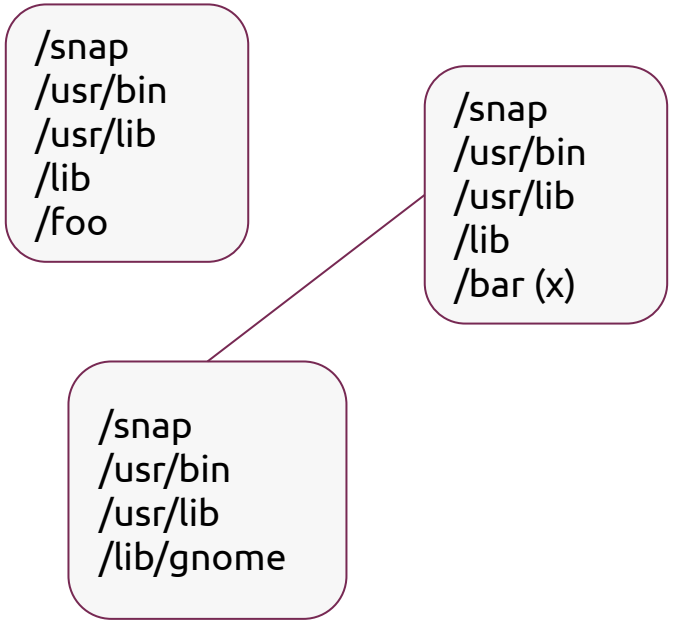
What is Ubuntu Core?

- Operating system based on stable Ubuntu releases
- Comprised of snaps
- Some snaps are maintained by Canonical
- Some are maintained by vendor or user
- User data can be encrypted
- Snaps are isolated from one another
- Snaps can access systems and other snaps through interfaces



What is a snap?

- Immutable image of of apps/libraries/data
- A single binary blob (squashfs)
- Built-in compression (xz, lzo)
- Snaps can share data
- Snaps are confined and sandboxed
- Hooks that respond to snap life-cycle events (install, remove, update, configure)




```
$ snap list
$ snap install hello-world
$ ls -l /snap/hello-world/current/
```

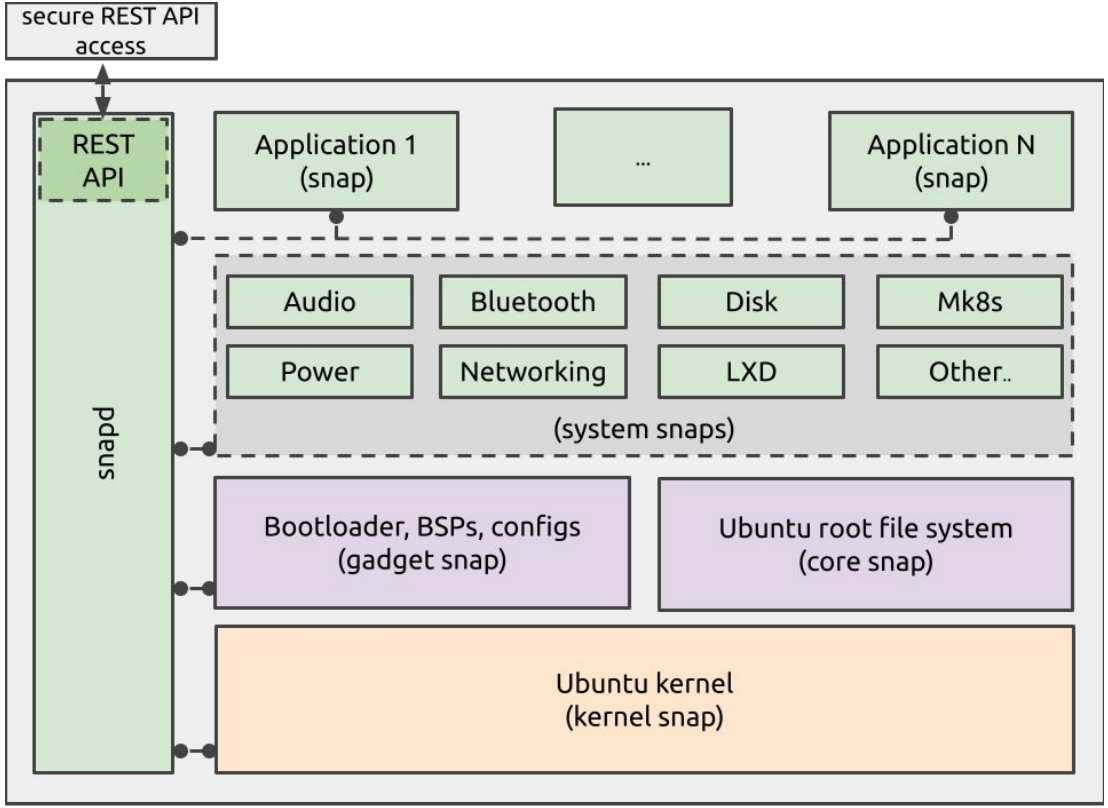
Why?

Why?

- Security
- Reproducibility
- Isolate vendors from common pitfalls
- Content delivery
- Configuration management
- Declarative

How?

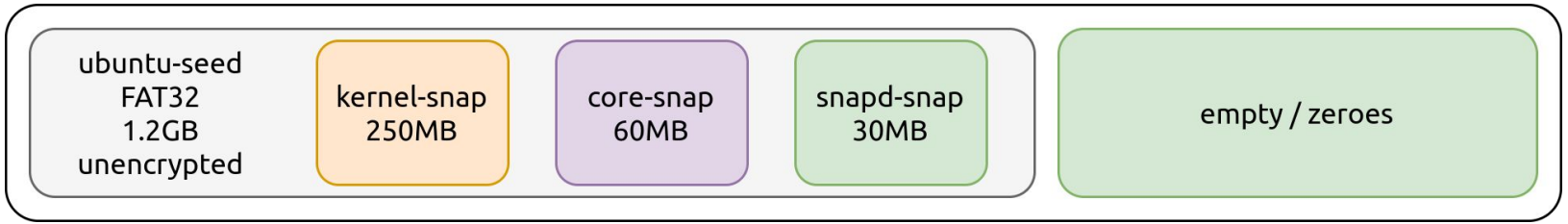
What is Ubuntu Core?



Ubuntu Core image for Raspberry Pi

Installer image partition layout (Raspberry Pi)

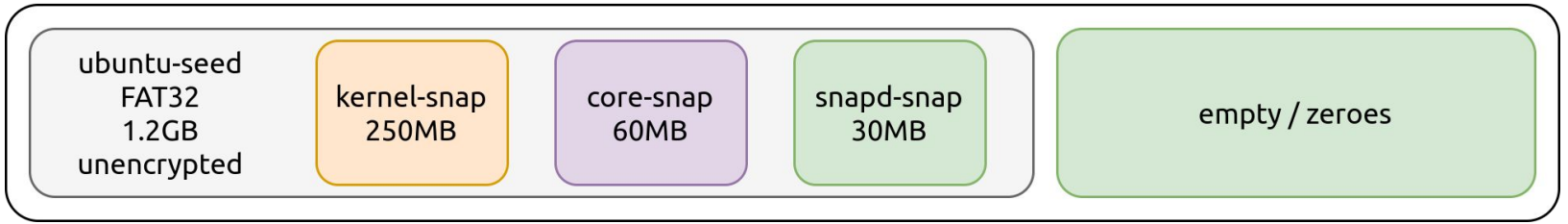
Total size = 3.4GB



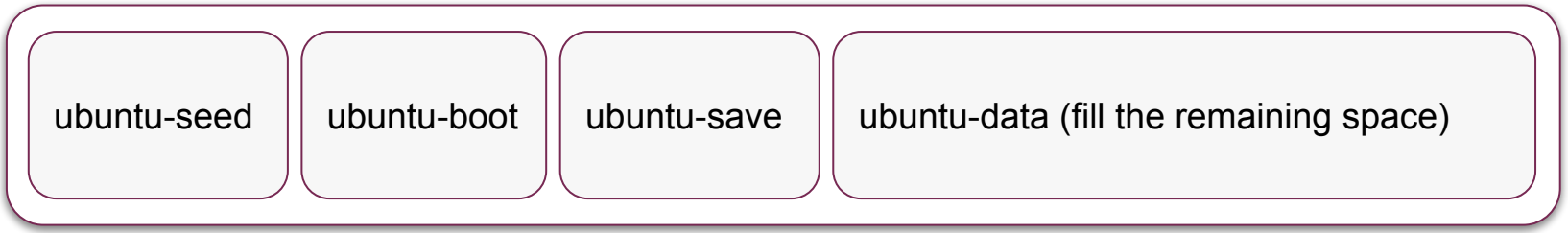
Ubuntu Core disk

Installer image partition layout (Raspberry Pi)

Total size = 3.4GB



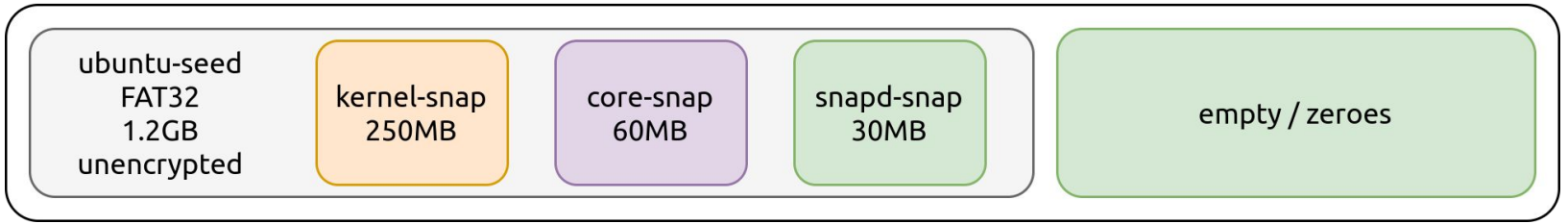
Installed disk layout



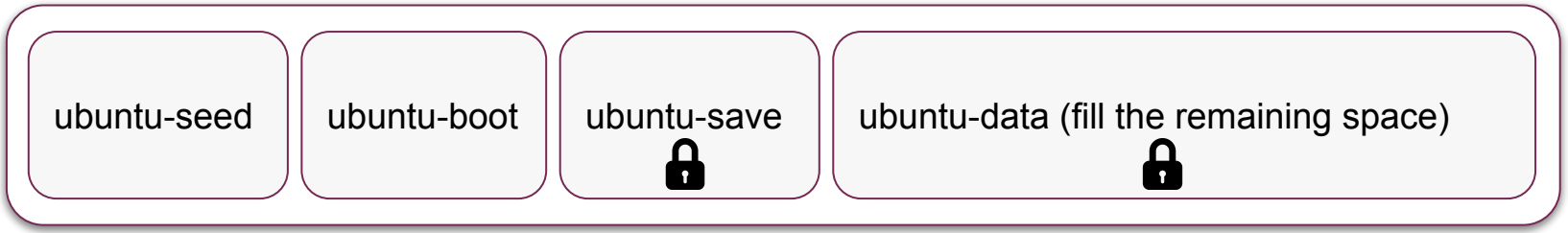
Ubuntu Core disk

Installer image partition layout (Raspberry Pi)

Total size = 3.4GB



Installed disk layout (with data encryption)



Installation

```
Nov 22 15:43:42 ubuntu snapd[1460]: secboot_tpm.go:75: checking if secure boot is enabled...
Nov 22 15:43:42 ubuntu snapd[1460]: secboot_tpm.go:80: secure boot is enabled
Nov 22 15:43:42 ubuntu snapd[1460]: secboot_tpm.go:82: checking if TPM device is available...
Nov 22 15:43:42 ubuntu snapd[1460]: secboot_tpm.go:96: TPM device detected and enabled
Nov 22 15:43:42 ubuntu snapd[1460]: handlers_install.go:277: create and deploy partitions
Nov 22 15:43:42 ubuntu snapd[1460]: install.go:71: installing a new system
Nov 22 15:43:42 ubuntu snapd[1460]: install.go:72:         gadget data from: /snap/pc/x1
Nov 22 15:43:42 ubuntu snapd[1460]: install.go:74:         encryption: on
Nov 22 15:43:43 ubuntu snapd[1460]: install.go:155: created new partition /dev/vda3 for structure #3 ("ubuntu-boot") (size
750 MiB) role system-boot
Nov 22 15:43:43 ubuntu snapd[1460]: install.go:155: created new partition /dev/vda4 for structure #4 ("ubuntu-save") (size
16 MiB) role system-save
Nov 22 15:43:43 ubuntu snapd[1460]: install.go:165: encrypting partition device /dev/vda4
Nov 22 15:43:44 ubuntu snapd[1460]: install.go:187: encrypted device /dev/mapper/ubuntu-save
Nov 22 15:43:44 ubuntu snapd[1460]: install.go:155: created new partition /dev/vda5 for structure #5 ("ubuntu-data") (size
1.02 GiB) role system-data
Nov 22 15:43:44 ubuntu snapd[1460]: install.go:165: encrypting partition device /dev/vda5
Nov 22 15:43:45 ubuntu snapd[1460]: install.go:187: encrypted device /dev/mapper/ubuntu-data
Nov 22 15:43:45 ubuntu snapd[1460]: handlers_install.go:333: make system runnable
```

Snap

- Declarative
- meta/snap.yaml
- Ships the apps and dependencies
- Some dependencies provided by base (libc et al.)
- Dependencies can be provided by content snaps
- Apps run confined in a sandbox environment
- Snap can modify their layout

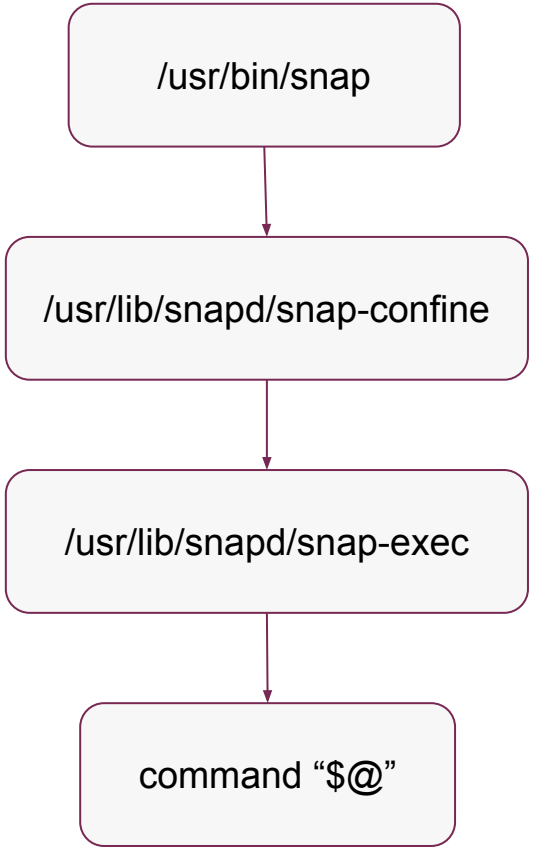
```
name: app
version: 1.0
summary: my snap
base: core20
confinement: strict
environment:
  USE_THIS_VAR: '1'
apps:
  my-app:
    command: bin/myapp
    plugs:
      - home
      - network
      - serial-port:
          path: /dev/ttyS1
plugs:
  gnome-3-34-1804:
    interface: content
    target: $SNAP/gnome-platform
    default-provider: gnome-3-34-1804
```

Snap, and how to make one?

- **snapcraft**
  - LXD
  - Multipass (VM)
  - Destructive (use the current host)
  - Can build from source and packages
- snap pack

```
$ unsquashfs -d pi-snap pi_107.snap
Parallel unsquashfs: Using 16 processors
33 inodes (207 blocks) to write

[==================================================|] 207/207
100%

created 32 files
created 7 directories
created 0 symlinks
created 0 devices
created 0 fifos
created 0 sockets
$ cd pi-snap
$ ls -l pi-snap
total 11708
drwxr-xr-x 2 maciek maciek     4096 07-07 00:54 boot-assets
-rw-r----- 1 maciek maciek     4096 07-07 00:54 boot.sel
drwxr-xr-x 4 maciek maciek     4096 07-07 00:54 meta
drwxr-xr-x 3 maciek maciek     4096 07-07 00:54 snap
-rw-r--r-- 1 maciek maciek        0 07-07 00:54 uboot.conf
$ snap pack pi-snap
built: pi_20-1_arm64.snap
```

How a snap runs?



/snap/bin/hello-world -> /usr/bin/snap

Set up confinement and sandbox

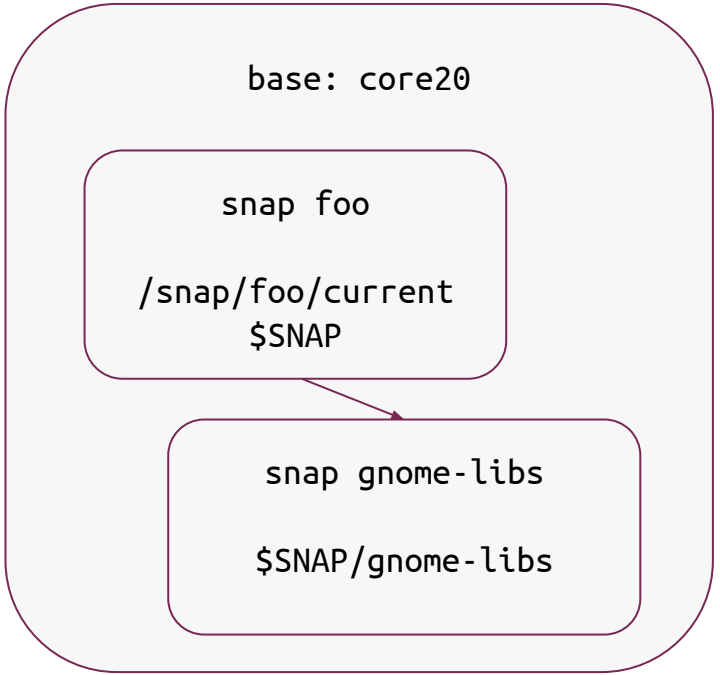
Executes in snap mount namespace

Interfaces, what are they and how to connect them?

- Interfaces provide access to system features or other snaps
- Organized as plugs (consumer) and slots (provider)
- Eg. opengl interface
  - Allow access to dri device nodes (/dev/dri/*)
  - Allow access to video related char devices (c 226:*)
  - Allow access to opengl libraries (libGLX_*, vulkan)
- Eg. content
  - Allow access to specific mount locations from providers inside the consumers mount namespace
- Eg. serial-port
  - Allow access to specific device nodes (/dev/ttyS*, /dev/ttyUSB*, /dev/ttyMX*..)
  - Allow access to related char devices (c 4:*, …)
- Eg. process control
  - Allow access to /usr/bin/{p}kill
  - Allow nice, setpriority, sched_set* syscalls

Mount namespace

- Each app/service runs in a separate mount namespace
- Pivot to the base snap
- Detached from host (/var/lib/snapd/hostfs)
- Specific entries are introduced into the mount ns
- Event propagation is MS_SLAVE for most
- Specific entries have MS_SHARED (eg. /media)
- Private /tmp
- Mount ns is captured and preserved at /run/snapd/ns/foo.mnt
- snap-confine and snap-update-ns



AppArmor

- Currently the only major LSM supported
- Each app/service gets a profile
- Interfaces contribute to the content of a profile
- pivot_root ensures proper operation

```
#include <tunables/global>

profile snap-update-ns.ohmygiraffe (attach_disconnected) {
...
  # Common devices accesses
  /dev/null rw,
  /dev/full rw,
...
  # Allow reading the command line (snap-update-ns uses
  # it in pre-Go bootstrap code).
  @{PROC}/@{pid}/cmdline r,

  # Allow reading file descriptor paths
  @{PROC}/@{pid}/fd/* r,
...
}
```

Seccomp & syscall filtering

- Seccomp program attached for each snap
- Lists allowed syscalls and their parameters (if possible)
- Uses libseccomp to generate the binary program
- /usr/lib/snapd/snap-seccomp

```
# LP: #1446748 - support syscall arg filtering
# for mode_t with O_CREAT
open
...
# Daemons typically run as 'root' so allow chown
# to 'root'. DAC will prevent non-root from
# chowning to root.
chown - u:root g:root
chown32 - u:root g:root
fchown - u:root g:root
fchown32 - u:root g:root

# enforce pid_t is 0 so the app may only change
# its own scheduler and affinity. Use
# process-control interface for controlling other
# pids.
sched_setaffinity 0 - -
sched_setparam 0 -
```

Cgroup and device access

- Use udev to tag allowed devices
- Supports both v1 and v2 (unified hierarchy)
- v1: separate controller /sys/fs/cgroup/devices/snap.foo.bar
- v2: /sys/fs/bpf/snap/snap_foo_bar
  - process tracking through systemd
  - custom eBPF program

```
/sys/fs/cgroup/devices/snap.foo.bar/devices.list:

c 1:9
c 1:5

/sys/fs/bpf/snap/snap_foo_bar:

key: 63 0a 00 00 00 ef 00 00  00  value: 01
key: 63 01 00 00 00 03 00 00  00  value: 01
key: 63 01 00 00 00 07 00 00  00  value: 01
```

Base snap

- This is your rootfs for both the snap and the system
- When in a system’s rootfs, apps of the base snap are unconfined
- Ubuntu: core, core18 (18.04), core20 (20.04), core22 (22.04)
- PoC of other base snaps: Fedora, Freedesktop.org
- Not all bases are bootable

Kernel snap

- Provides the kernel binaries
- Grub is capable load kernel directly from squashfs images
- Kernel image may need to be extracted
- Can ship boot assets (eg. DTBs)
- Can provide encryption hooks

```
name: pi-kernel
version: 5.4.0-1047.51
summary: The Canonical Raspberry Pi kernel
description: The Canonical Raspberry Pi kernel
architectures:
- arm64
assumes:
- kernel-assets
confinement: strict
grade: stable
type: kernel
```

Gadget snap

- Declares support for given host
- Defines slots
- Can contribute to kernel command line (or override it)
- Integration hooks for registration
- Describes the storage

```
name: pi
version: 20-1
summary: Raspberry Pi gadget
description: |
  Support files for booting Raspberry Pi.
  This gadget snap supports the Raspberry Pi 2B, 3B, .,..
architectures:
- arm64
assumes:
- kernel-assets
base: core20
confinement: strict
grade: stable
slots:
  bcm-gpio-0:
    interface: gpio
    number: 0
  bcm-gpio-1:
    interface: gpio
    number: 1
  bcm-gpio-10:
    interface: gpio
```

Gadget snap (meta/gadget.yaml)

- Volumes
- ... carrying structures
- ... carrying content
- https://github.com/snapcore/pi-gadget
- https://github.com/snapcore/dragonboard-gadget

```
volumes:
  pi:
    schema: mbr
    bootloader: u-boot
    structure:
      - name: ubuntu-seed
        role: system-seed
        filesystem: vfat
        type: 0C
        size: 1200M
        content:
          - source: $kernel:dtbs/dtbs/broadcom/
            target: /

      - name: ubuntu-boot
        role: system-boot
        filesystem: vfat
        type: 0C
        # whats the appropriate size?
        size: 750M
      - name: ubuntu-data
        role: system-data
        filesystem: ext4
        type: 83,0FC63DAF-8483-4772-8E79-3D69D8477DE4
        # XXX: make auto-grow to partition
        size: 1500M
```

Assertions

- Signed documents that describe and assert parts of the system
- snap-declaration (what a snap is)
- snap-revision (binds hash of the snap image to a revision)
- account (publisher)
- account-key (publisher signing keys)
- **model (describes a system)**
- serial (device’s serial number)

Model assertion

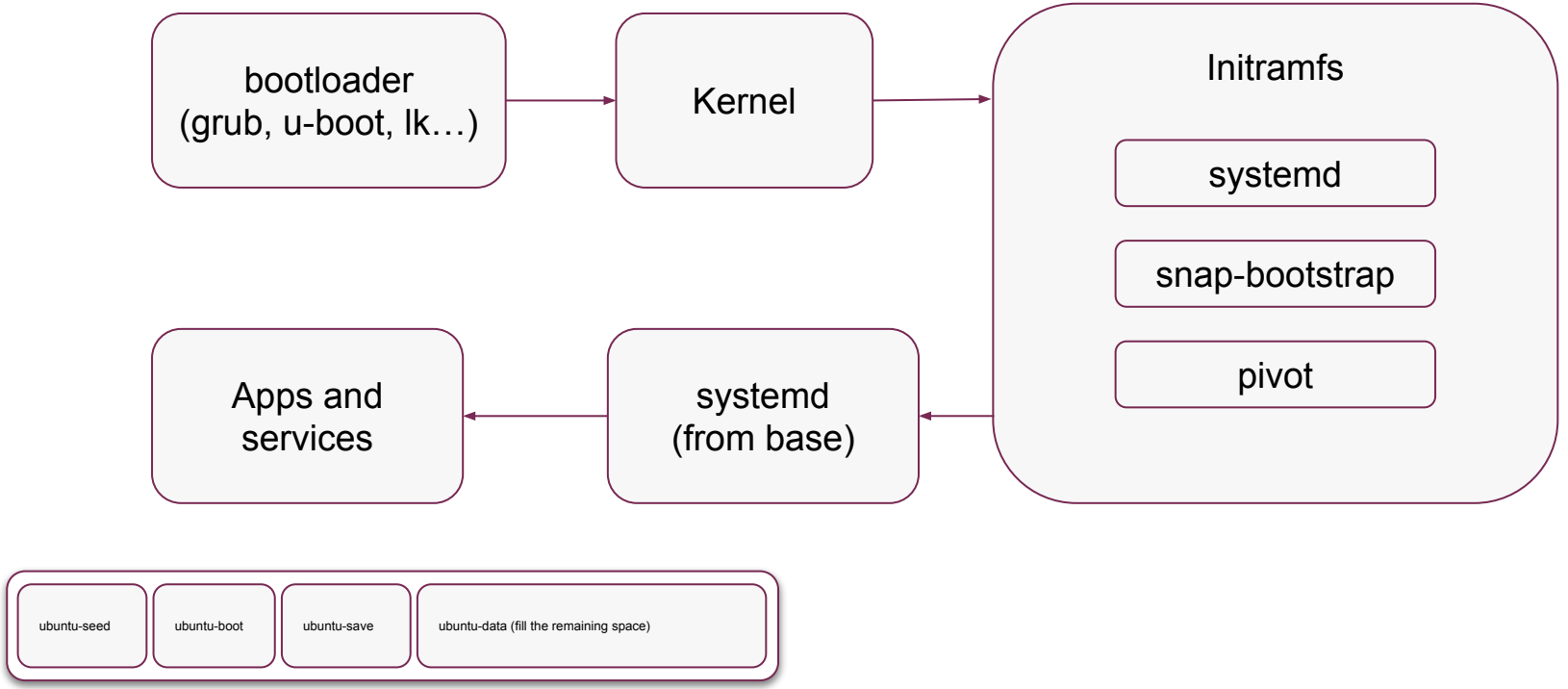
```
type: model
authority-id: canonical
brand-id: canonical
model: ubuntu-core-20-pi-arm64
architecture: arm64
base: core20
grade: signed
snaps:
 - default-channel: 20/stable
   name: pi
   type: gadget
 - default-channel: 20/stable
   name: pi-kernel
   type: kernel
timestamp: 2020-03-31T12:00:00.0Z
sign-key-sha3-384: 9tydnLa6MTJ-jaQTFUXEwHl1yRx7ZS4K5cyFDhYDcPzhS7uyEkDxdUjg9g08BtNn

AcLBXAQAAQoABgUCXpYTHgAKCRDgT5vottzAEvM/D/9z50cQ+T2pyz8/Lrlq99GfrjNvx34VbV4u
5KDBJ6D6TGPHBsImLATyWY+LW9Z5933I/K2wpETIxI3QYhbU4anS+vODRF2HnXTfgqt1/Yy173Py
...
```

Assertion chain

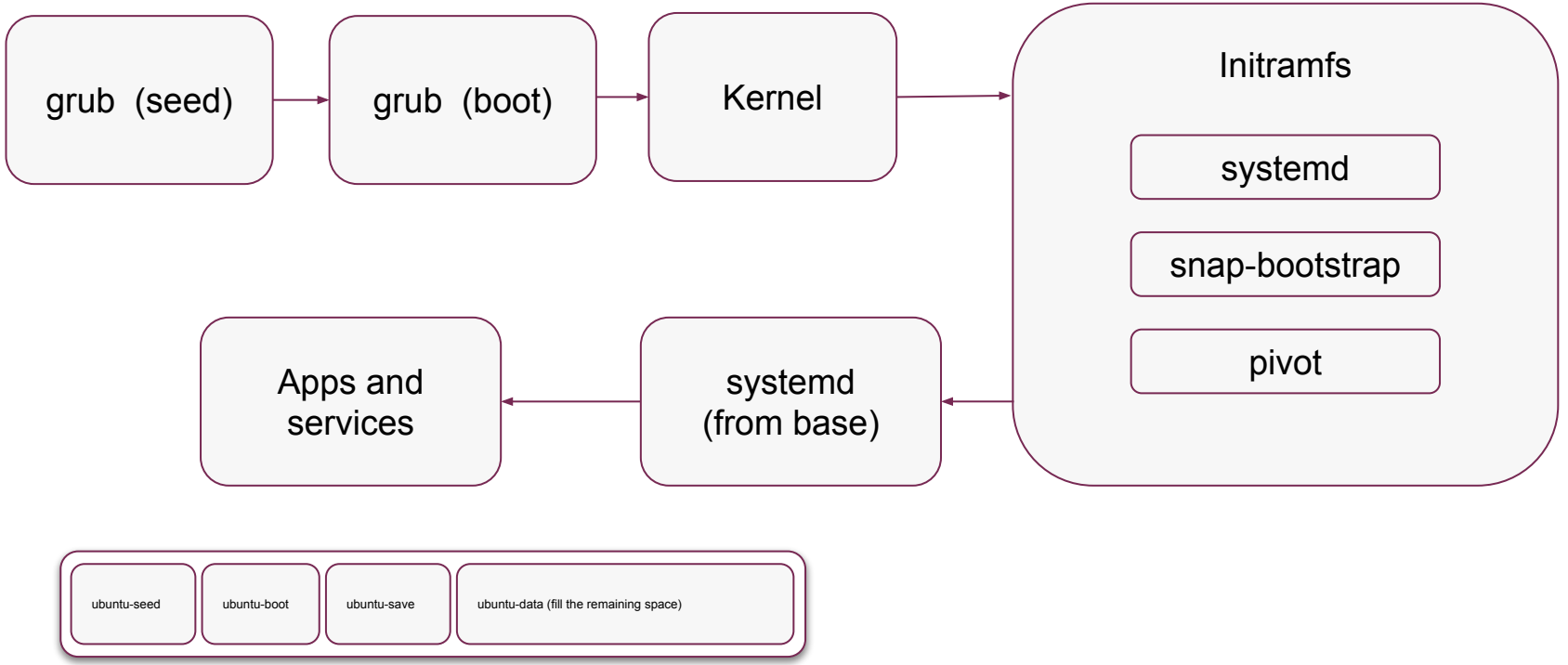
```
$ snap info graphics-debug-tools-bboozzoo
summary:   collection of utilities for debugging graphics/compute issues
...
snap-id: YcngvJw4tWJLG9sYThpH1cTGrlzbdig5
$ snap known --remote snap-declaration series=16 snap-id=YcngvJw4tWJLG9sYThpH1cTGrlzbdig5
type: snap-declaration
authority-id: canonical
snap-id: YcngvJw4tWJLG9sYThpH1cTGrlzbdig5
publisher-id: FpA2tgNE8bUrUKSOhE6SOpsMdlcEJkDL
sign-key-sha3-384: BWDEoaqyr25nF5SNCvEv2v7QnM9QsfCc0PBMYD_i2NGSQ32EF2d4D0hqUel3m8ul

AcLBUgQAAQoABgUCWq+PLwAAuFsQAFCtk5X8xJ/RKfOTrB/mza5JeB49ckR1zL8azj6TJeLJLept
...
$ snap known  --remote account account-id=FpA2tgNE8bUrUKSOhE6SOpsMdlcEJkDL
type: account
authority-id: canonical
account-id: FpA2tgNE8bUrUKSOhE6SOpsMdlcEJkDL
display-name: Maciek Borzecki
timestamp: 2017-10-26T07:41:29.654058Z
username: maciek-borzecki
sign-key-sha3-384: BWDEoaqyr25nF5SNCvEv2v7QnM9QsfCc0PBMYD_i2NGSQ32EF2d4D0hqUel3m8ul

483i8kxRZDc6a7Au0mr0ud1/VZMe3SyG33QNVCu3K6orJDEiw/LBcbdUJkpV59eQP99ihStE+OvC
...
```

How (does it boot)?

The boot process



The boot process

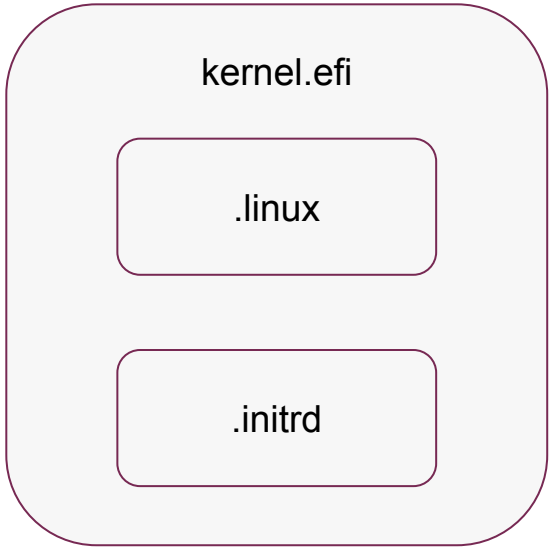


bootloader (grub)

```
set kernel=kernel.efi

if [ "$kernel_status" = "try" ]; then
    # a new kernel got installed
    set kernel_status="trying"
    save_env kernel_status
    # use try-kernel.efi
    set kernel=try-kernel.efi
elif [ "$kernel_status" = "trying" ]; then
    # nothing cleared the "trying snap" so the boot failed
    # we clear the mode and boot normally
    set kernel_status=""
    save_env kernel_status
elif [ -n "$kernel_status" ]; then
    # ERROR invalid kernel_status state, reset to empty
    echo "invalid kernel_status!!!"
    echo "resetting to empty"
    set kernel_status=""
    save_env kernel_status
fi
```

Kernel and initramfs (EFI)

```
$ objdump -h pc-kernel-snap/kernel.efi

pc-kernel-snap/kernel.efi:     file format pei-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00007500  0000000000004000  0000000000004000  00000400  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .reloc        0000000a  000000000000c000  000000000000c000  00007a00  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .data         00002128  000000000000d000  000000000000d000  00007c00  2**5
                  CONTENTS, ALLOC, LOAD, DATA
  3 .dynamic      00000110  0000000000010000  0000000000010000  00009e00  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  4 .rela         00000e58  0000000000011000  0000000000011000  0000a000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .dynsym       00000378  0000000000012000  0000000000012000  0000b000  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  6 .sbat         000000ff  0000000000050000  0000000000050000  0000b400  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .linux        00b3c980  0000000002000000  0000000002000000  0000b600  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .initrd       01947295  0000000003000000  0000000003000000  00b48000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
```

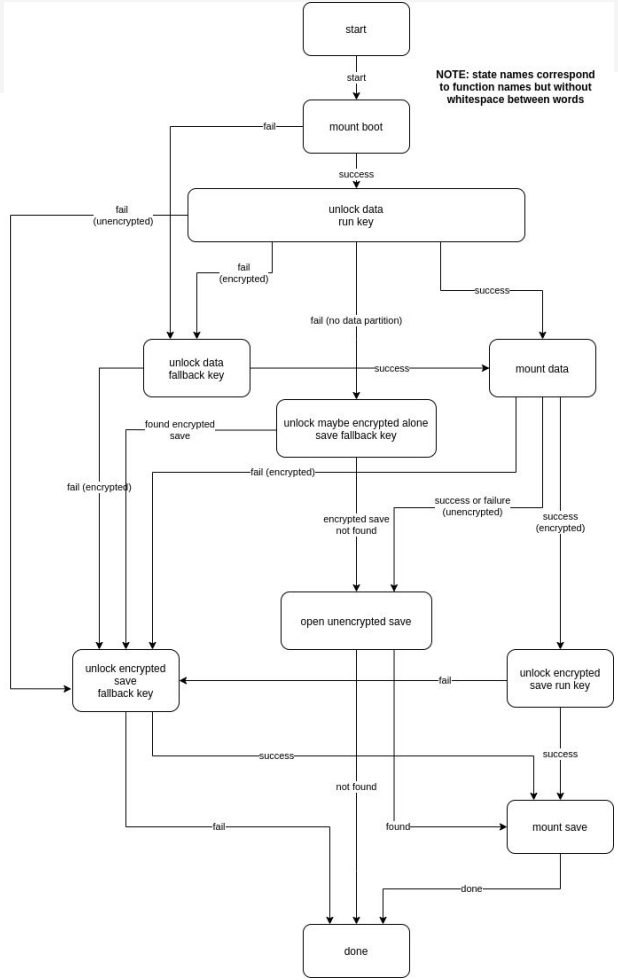


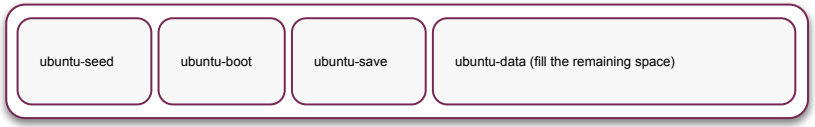
Kernel and initramfs (non-EFI)

```
$ ls -l pi-kernel/*.img
-rw-r--r-- 1 maciek maciek 20850017 11-19 08:59 pi-kernel/initrd.img
-rw------- 1 maciek maciek  8409083 11-12 15:38 pi-kernel/kernel.img

$ file pi-kernel/*.img
pi-kernel/initrd.img: LZ4 compressed data (v0.1-v0.9)
pi-kernel/kernel.img: gzip compressed data, max compression, from Unix, original size modulo 2^32
23843328

$ lz4cat pi-kernel/initrd.img > pi-kernel/initrd
$ file pi-kernel/initrd
initrd: ASCII cpio archive (SVR4 with no CRC)
```

snap-bootstrap

- Look at bootloader state data
- Identify boot partition
- Identify data and save partitions
- Unlock data and save volumes
  - Interact with TPM, pass measurements
  - Try to unseal encryption key
  - Attempt recovery key if needed
- Update boot status





A/B updates

- Kernel boot status updated by the boot script
- Base boot status updated by snap-bootstrap
- On the next successful boot
  - Check which kernel booted
  - Check which base booted
  - Fail/proceed with updates

```
$ snap debug boot-vars --uc20 --root-dir
/run/mnt/ubuntu-seed
snapd_recovery_mode=run
snapd_recovery_system=20211122
snapd_recovery_kernel=
snap_kernel=
snap_try_kernel=
kernel_status=
recovery_system_status=
try_recovery_system=
```

Secure boot

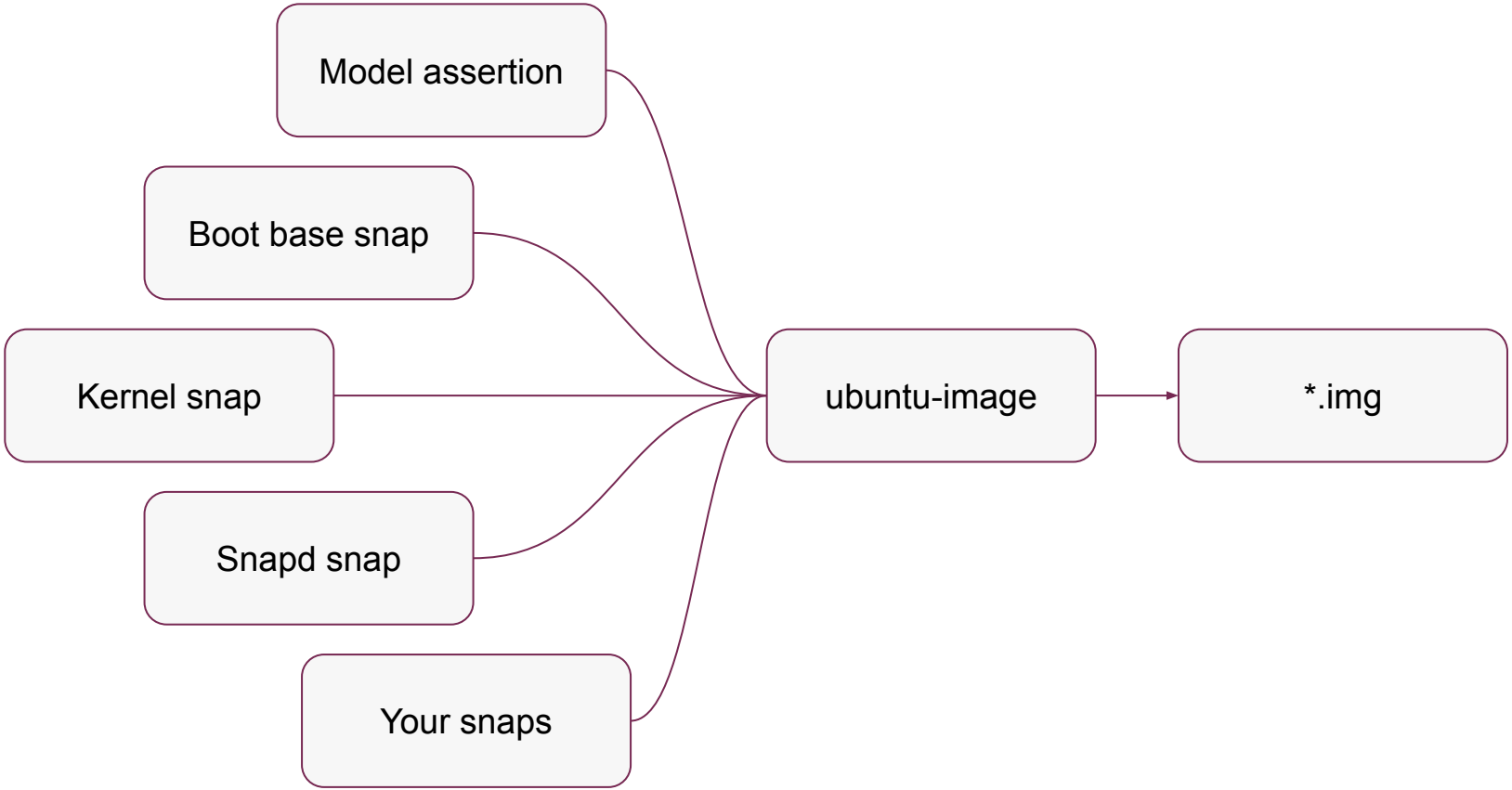
- https://github.com/snapcore/secboot
- System properties
  - Model
  - Kernel
  - Boot assets (grubx86.efi..)
  - Boot chains (order in which boot assets are loaded)
  - Recovery system(s)
  - Kernel command lines

Secure boot

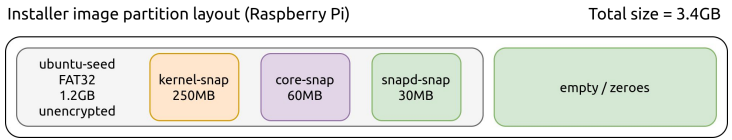
```
$ cat /var/lib/snapd/modeenv
mode=run
current_recovery_systems=20211122
good_recovery_systems=20211122
base=core20_1242.snap
current_kernels=pc-kernel_x1.snap
model=canonical/ubuntu-core-20-amd64-dangerous
grade=dangerous
model_sign_key_id=9tydnLa6MTJ-jaQTFUXEwHl1yRx7ZS4K5cyFDhYDcPzhS7uyEkDxdUjg9g08BtNn
current_trusted_boot_assets={"grubx64.efi":["9c5f188987104899164c64dbd7bd12f7b90af1250dec6643f54bddf26286391
d3afa6f75086a87e803c264323a673001"]}
current_trusted_recovery_boot_assets={"bootx64.efi":["088dac3572811618a29e329c71f0c754797fa07affdb4ef28fd5e1
72ccbc5d173ba33adfc58b9c07ac902ec6dbf8ac77"],"grubx64.efi":["9c5f188987104899164c64dbd7bd12f7b90af1250dec664
3f54bddf26286391d3afa6f75086a87e803c264323a673001"]}
current_kernel_command_lines=["snapd_recovery_mode=run console=ttyS0 console=tty1 panic=-1"]
```

How (do you build it)?

Ingredients



Building




```
$ ubuntu-image snap my.model
$ ubuntu-iamge snap my.model --snap foo.snap
$ file image-home/pc.img
image-home/pc.img: DOS/MBR boot sector, extended partition table (last)
$ parted image-home/pc.img p
WARNING: You are not superuser.  Watch out for permissions.
Model:  (file)
Disk /home/maciek/work/canonical/image/image-home/pc.img: 3155MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name         Flags
 1      1049kB  2097kB  1049kB               BIOS Boot    bios_grub
 2      2097kB  1260MB  1258MB  fat32        ubuntu-seed  boot, esp
```

Running

```
$ qemu-system-x86_64 -enable-kvm -snapshot -m 2048 \
    -device virtio-net-pci,netdev=mynet0 \
    -netdev user,id=mynet0,hostfwd=tcp:127.0.0.1:59467-:22 \
    -serial telnet:127.0.0.1:59468,server,nowait \
    -monitor telnet:127.0.0.1:59469,server,nowait \
    -vga virtio -display gtk,gl=on \
    -object rng-random,filename=/dev/urandom,id=rng0 \
    -device virtio-rng-pci,rng=rng0 \
    -drive file=image-home/pc.img,if=virtio,index=0 \
    -smp 4 \
    -bios /usr/share/ovmf/x64/OVMF_CODE.fd
```

Demo

Docs

- Snap & snapd docs

  https://snapcraft.io/docs
- Ubuntu Core docs
  https://ubuntu.com/core/docs

Questions?