The Cuttlefish Emulator

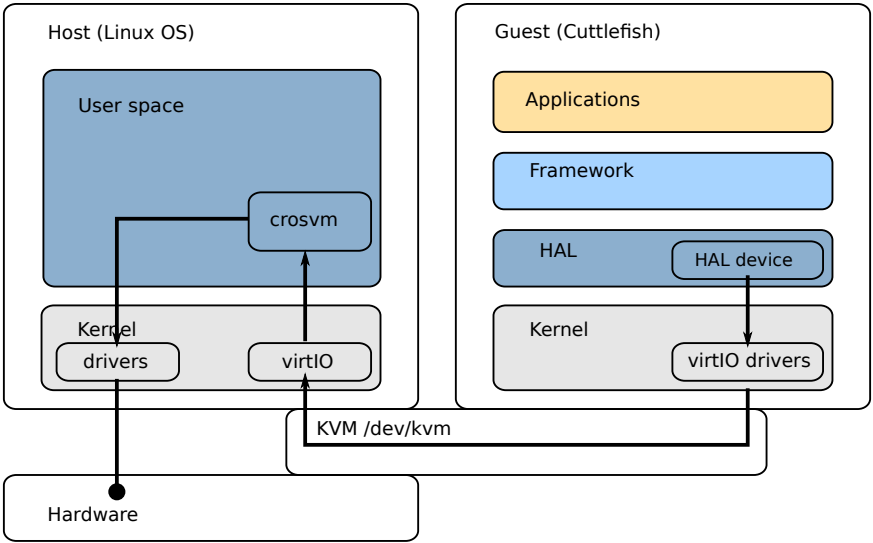
Overview

- Android emulators
- Setting up Cuttlefish
- Using Cuttlefish
- acloud

Goldfish and Cuttlefish

- Goldfish
  - well-known Android emulator, used for testing apps
  - usually installed along with with Android Studio ...
  - ... but also part of AOSP
- Cuttlefish (since P/9)
  - used to test the Android operating system
  - more accurate emulation of real hardware

Cuttlefish



Cuttlefish

- Headless: you communicate via ADB or remote desktop
- Most I/O is via Linux virtio drivers (block, net, serial, gpu)
- Uses crosvm (Chrome OS Virtual Machine Monitor) (*)
- Uses Linux KVM (Kernel-based Virtual Machine) for hardware acceleration

(*) Earlier versions were based on QEMU

https://source.android.com/setup/create/cuttlefish

The cuttlefish-common package

- cuttlefish-common is a Debian package that configures the environment for the Cuttlefish runtime
- Usually built from source (*) - for example

```
$ sudo apt install -y git devscripts config-package-dev debhelper-compat
$ git clone https://github.com/google/android-cuttlefish
$ cd android-cuttlefish
$ debuild -i -us -uc -b
$ sudo dpkg -i ../cuttlefish-common_*_*64.deb || sudo apt-get install -f
$ sudo usermod -aG kvm,cvdnetwork $USER
$ sudo reboot
```

(*) lots of dependencies, works on Ubuntu 20.04, other distros may require some extra steps

Launching Cuttlefish

- Cuttlefish runs a CVD (Cuttlefish Virtual Device), similar to AVD for Goldfish
- AOSP has these commands for manipulating CVDs

`launch_cvd`	start a CVD instance
`cvd_status`	status of running instances
`stop_cvd`	stop an instance

Runtime and log files

`launch_cvd` generates these directories in `$HOME`:

```
cuttlefish_assembly/     intermediate files
cuttlefish_runtime/      symbolic link to the current instance
cuttlefish_runtime.1/    the current instance
```

`cuttlefish_runtime.1/` contains these useful log files

```
launcher.log             log files from launch_cvd
kernel.log               console log, including messages from boot loader and kernel
logcat                   Android logcat
```

The CVD

- The runtime configuration for the CVD is written to `$HOME/cuttlefish_runtime.1/cuttlefish_config.json`
- Generated from fragments, including:

```
|-- device/google/cuttlefish/shared/config
    |-- config_auto.json
    |-- config_foldable.json
    |-- config_phone.json
    |-- config_tablet.json
    |-- config_tv.json
```

- The config fragment is selected based on the build target
- Override using `-config [name]`, for example

```
$ launch_cvd -config tv
```

The CVD

- Example CVD config

  `device/google/cuttlefish/shared/config/config_phone.json`

```
{
        "x_res" : 720,
        "y_res" : 1280,
        "dpi" : 320,
        "memory_mb" : 2048
}
```

VNC remote desktop

To see the Android screen using VNC, launch the CVD like so:

```
$ launch_cvd -start_vnc_server
```

The VNC port number will be 6444, so you can launch a VNC client

```
$ java -jar tightvnc-jviewer.jar localhost 6444
```

Note: 'java' must be run from a terminal that has had lunch

Web remote desktop

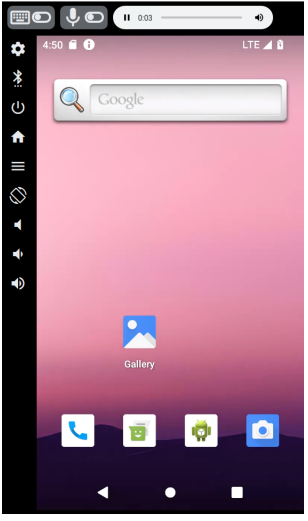
To see the Android screen via a WebRTC browser session launch the CVD like so:

```
$ launch_cvd --start_webrtc
```

Open a browser (Chrome/Chromium, not Firefox) using `https://localhost:8443` and accept security warning (about certificate)

The WebRTC view includes controls along the top and left side

You can add custom actions to the control panel, see `https://source.android.com/setup/create/cuttlefish-control-panel`



Serial console

To see the console output, launch with:

```
$ launch_cvd -console=true
```

Then attach to the console using a terminal emulator, such as screen:

```
$ screen $HOME/cuttlefish_runtime.1/console
```

Useful if you want to interact with the bootloader

Crib sheet for launch_cvd options

`-start_vnc_server`	Start VNC server on port 6444
`-start_webrtc`	Start web UI on `https://localhost:8443`
`-console=true`	Start console interface `cuttlefish_runtime/console`
`-daemon`	Daemon mode (run as a background process)
`-pause-in-bootloader=true`	Access bootloader via serial console
`-boot_slot`	a (default) or b
`-x_res`	default defined in the CVD
`-y_res`	default defined in the CVD
`-dpi`	default defined in the CVD
`-guest_enforce_security=false`	SELinux in permissive mode
`-extra_kernel_cmdline ""`	kernel command line
`-cpus`	default: 2
`-memory_mb`	default defined in the CVD
`-noresume`	Start a new runtime: factory reset
`-config=[name]`	Select CVD configuration [name]

acloud

- **acloud** is a front end to `launch_cvd` to simplify launching instances both locally and in the cloud
- Integrates with the Google Cloud Platform (using the CLI tool gcloud)
- You have to set up acloud first. If you only want to run local instances, add `--host`

```
$ acloud setup --host
```

acloud

Here is a simple example of using acloud to start a local instance running a locally generated image:

```
$ acloud create --local-instance 1 --local-image
Creating local AVD instance with the following details:
Image (local):
  /home/chris/aosp/out/target/product/marvin
hw config:
  cpu - 4
  ram - 6GB
  display - 720x1280 (320 DPI)

Waiting for AVD(s) to boot up ...
```

It will launch a copy of SSVNC Viewer (based on TightVNC viewer version 1.3.9) which was installed by acloud setup

Further reading

- My blog: "The Android Cuttlefish emulator"
  https://2net.co.uk/blog/cuttlefish-android12.html
- Code:
  https://android.googlesource.com/device/google/cuttlefish/
  - also has a "Cuttlefish Getting Started" guide

Summary

- Cuttlefish is designed to emulate a real world Android device a closely as possible
- Ideal for testing platform builds
- But, quite tricky to set up