Carcasa OGST • Monku R4 • Media Center • ESP 32 • Android Things

ODROID Magazine

Año Siete
Num. #73
Ene 2020

LA MAS RECIENTE Y POPULAR GENERACION DE ORDENADORES PORTATILES DE HARDKERNEL

ODROID GO *Advance*

KUBERNETES:
PLATAFORMA DE ORGANIZACION DE CONTENEDORES DE CODIGO ABIERTO

CAN BUS:
MICROCONTROLADORES

VIDEO VIGILANCIA:
MONITORIZACION VISUAL AVANZADA USANDO UN ODROID-C2

Montando una Carcasa ODROID GameStation Turbo (OGST) para tu ODROID-XU4

January 1, 2020

La razón por la que queremos utilizar una carcasa OGST más grande es porque viene con una placa de expansión que te permite añadir botones de encendido y reinicio, así como un espacio para guardar un disco duro USB externo. Esto te permitirá aumentar el espacio de almacenamiento disponible. Este ▶

Retro ESP32: La Mejor Imagen de Emulación para tu ODROID-GO

January 1, 2020

Retro ESP32 es el último lanzador repleto de funciones para ODROID-GO. El lanzador incluye esquemas de colores y temas inspirados en la popular interfaz de usuario del emulador RetroArch. En el momento de la publicación de este artículo, llegamos a pre-compilamos 11 emuladores incluido ROM/Game Manager. Además, cada emulador incluye ▶

Cómo Configurar y Usar el Bus CAN: Utilizando el ODROID-N2 con Microcontroladores

January 1, 2020

Cómo habilitar el bus CAN en ODROID-N2 a través de la interfaz HW SPI. También incluye instrucciones detalladas para obtener datos a través de una placa Monitor Bus MCP2515.

ODROID-GO Advance: La más Reciente y Famosa Generación de Ordenadores Portátiles de Hardkernel

January 1, 2020

¡Hemos continuado escuchando a los usuarios que querían jugar a juegos retro de 16 o 32 bits en un dispositivo portátil, así que ahora anunciamos el ODROID-GO Advance!

Monku R4 con un ODROID-N2 y Batocera Linux: La Mejor Consola de Juegos Retro que Puedes Montar por Alrededor de 100$

January 1, 2020

Este tutorial cubre el proceso de configuración de un ODROID-N2 de 2 GB de RAM y la instalación Batocera Linux para que podamos usar nuestro ODROID-N2 como una consola de juegos retro conectada a la TV

Fiesta de Desarrollo Kernel 5.4

January 10, 2020

Let's start the 5.4 kernel development party.

El Punto G: Tu Destino para Todas las Cuestiones Relacionadas con Juegos Android: ¡Google Dejo Caer la Pelota; Giphy es la Pelota; y ODROID-N2 lo Gana TODO!

January 1, 2020

Bueno, fue especial, ¿no? Si eres uno de los miles de suscriptores de Google Stadia Founder's Edition, entonces sabrás exactamente cómo Google dejó caer la pelota el día del lanzamiento. Este Tweet oficial de Google Stadia resume casi todo el enredo: "Aquí tienes la última actualización: si compraste y pagaste ▶️

Cluster Kubernetes Sobre un ODROID-N2

January 1, 2020

Introducción Kubernetes (o k8s para abreviar) es una plataforma extensible de organización de contenedores de código abierto diseñada para administrar cargas de trabajo en contenedores y servicios a escala. Ayuda con la implementación automatizada, el escalado y la administración de cargas de trabajo de aplicaciones centradas en contenedores en un ▶️

Sistema de Video Vigilancia en Movimiento Pearl Linux con Kodi: Monitorización Visual Avanzado Utilizando un ODROID-C2

January 10, 2020

@pearllinux ha creado una imagen de video vigilancia basada en Ubuntu 18.04 usando el kernel 3.16.75, con el software Motion Video Surveillance preinstalado y que se inicia en el primer arranque ejecutándose en modo de usuario normal. ¡Echale un vistazo!

Android Things

January 10, 2020

¿Alguna vez has intentado conectar un dispositivo periférico a los pines GPIO de tu SBC ODROID con el sistema operativo Android?

Montando una Carcasa ODROID GameStation Turbo (OGST) para tu ODROID-XU4

January 1, 2020 By Brian Ree Juegos, ODROID-XU4, Mecaniqueo

La razón por la que queremos utilizar una carcasa OGST más grande es porque viene con una placa de expansión que te permite añadir botones de encendido y reinicio, así como un espacio para guardar un disco duro USB externo. Esto te permitirá aumentar el espacio de almacenamiento disponible. Este tutorial te mostrará cómo ensamblar rápida y fácilmente la carcasa y asegurar el disco duro en la misma. No hace falta decir que, si está montando una consola de juegos retro, esto te dará la posibilidad de almacenar toneladas y toneladas de juegos.

Componentes necesarios

- Un ODROID-XU4/Monku Retro 3 funcional: https://bit.ly/3658Jrp - Una carcasa OGST ODROID-XU4: https://bit.ly/2Q6maC9

Herramientas necesarias

- Pequeño Juego de destornilladores - Tiras de Velcro permanente o desmontable - Disco duro externo USB 3

Revisando los componentes

Echemos un vistazo a los componentes con los que vamos a trabajar. La siguiente imagen muestra el hardware necesario para utilizar el ODROID-XU4 con OGST a modo de carcasa de juegos retro.

Figura 1

La configuración con la que he trabajado, tal y como se muestra en la imagen anterior, durante este tutorial es un ODROID-XU4 con Lakka. En realidad, puedes usar cualquier configuración que desees en tu ODROID-XU4. Si necesitas más espacio de almacenamiento, la OGST y este tutorial están pensados para ti. Un mando inalámbrico Game Sir. Una micro SD de 64 GB para la unidad de arranque. Un disco duro externo USB 3 KESU de 250 GB. El disco podría haber sido más grande, pero éste en concreto cuesta unos 40$, de modo que ésta es la configuración perfecta para montar consolas retro con un presupuesto limitado. Publicaré luego algunos enlaces donde centrarás algunos tutoriales sobre el montaje del ODROID-XU4 y de los componentes.

- Mando por cable Game Sir Wired (Hard Kernel) 17$ - Mando por cable Game Sir Wired (Amazon) 17$ - Mando por cable Game Sir Wired 17$ - Disco duro externo KESU 250GB USB3 37$

Ahora, para colocar el disco de forma segura dentro de la carcasa, necesitamos algunas tiras de Velcro®. Esto nos proporciona una forma segura pero fácilmente desmontable guardar el disco duro dentro de la carcasa. Tienes dos opciones, tiras permanentes súper fuertes o tiras fuertes pero desmontables. Si tienes pensado reutilizar el disco duro, puede optar por la segunda opción. Aquí tienes los dos tipos de tiras:

- Tiras de velcro muy resistentes $3 - Tiras de velcro desmontables $3

Figura 2

Una vez que hayas terminado con todo esto, estarás listo para montar tu carcasa preliminar. Recuerda que siempre puedes optar por un disco duro con más espacio. Siempre que tenga un tamaño similar al que aparecen en la imagen, encajará perfectamente en la carcasa. En mi caso opté por usar la versión súper fuerte de la cinta Velcro porque deseaba que la unidad se mantuviera firme en su lugar y que la posibilidad de que se soltara fuese mínima. Tampoco tenía la intención de usar el disco duro para otra cosa, ya que fue comprado únicamente para este este proyecto en concreto. A continuación, abriremos nuestra carcasa OGST y vemos que hay dentro.

Figura 3

Dentro de la carcasa, encontrarás una placa de expansión con una pequeña pantalla. Un conjunto de tapas de plástico, un cable plano, 2 cables USB y algunos tornillos; ah y, por supuesto, una carcasa. ¡Los cables son importantes! El cable USB A a USB Micro B es perfecto para la unidad USB externa. Tiene un conector de 90 grados que hará que sea mucho más fácil guardar la unidad dentro de la carcasa OGST. Echemos ahora un vistazo a nuestro hardware ODROID-XU4.

Figura 4

Nuevamente, usaremos el ODROID-XU4 con una tarjeta micro SD de 64GB. Necesitarás un destornillador llegado a este punto. Un juego de destornilladores electrónicos estándar debería serte suficiente. Si no sabe cómo configurar tu ODROID-XU4 con un sistema operativo, puedes recurrir a los siguientes recursos.

- Munku R3/ODROID-XU4 Ubuntu con consola multiusos Retroarch - Tutorial Parte 1 - Munku R3/Consola de juegos retro ODROID-XU4 Lakka - Tutorial Parte 1

A partir de aquí, asumiré que tiene tu dispositivo ODROID-XU4 configurado. Puesto que estoy montando esta carcasa para una consola de juegos retro y quiero usar todos y cada uno de los recursos del dispositivo para la emulación, decidí no configurar la segunda pantalla en la placa de expansión. Si quieres configurar la tuya, dependerá del sistema operativo que estés utilizando en tu dispositivo. Puede encontrar más información en estos enlaces:

- Información de la segunda pantalla de la carcasa OGST (https://bit.ly/2tdvwTl) - Información de la segunda pantalla de la carcasa OGST (https://bit.ly/2MCbNDP)

Preparando la parte superior de la carcasa

La placa se monta boca abajo por el lado superior de la carcasa. Quizás sea un poco raro, pero como observarás pronto, la carcasa está maravillosamente diseñada y es muy fácil de usar. Las dos imágenes que aparecen a continuación muestran la placa ODROID-XU4 montada en el parte superior de la carcasa OGST. CONSEJO: Tómate tu tiempo con los tornillos. Encajan un poco raro debido a que la rosca de los tornillos es más grande. Simplemente tómate tu tiempo y apriételos cuidadosamente hasta que la placa se fije bien. Intenta no apretarlos demasiado. La siguiente imagen muestra la placa ODROID-XU4, los tornillos, la parte superior de la carcasa y el destornillador que he utilizado.

Figura 5

Las siguientes imágenes muestran la placa en su lugar atornillada correctamente a la parte superior de la carcasa.

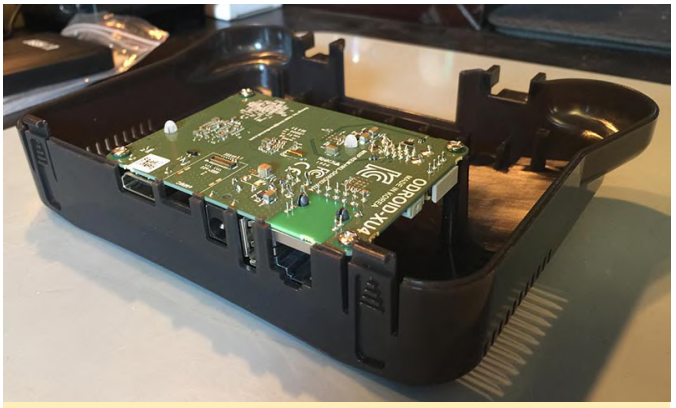


Figura 6

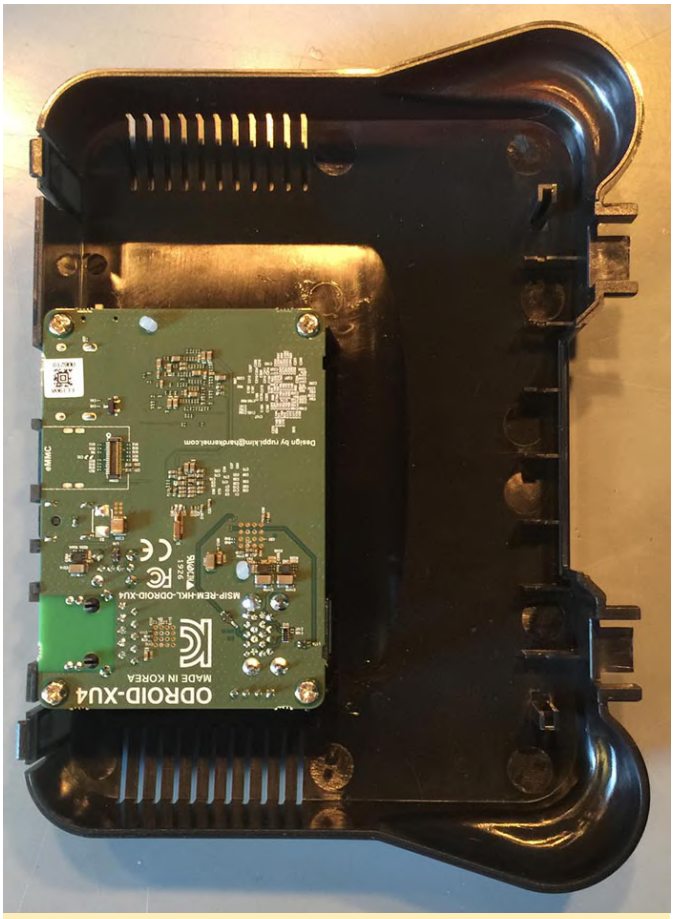


Figura 7

A continuación, configuraremos la placa de expansión y el cable plano. La siguiente imagen muestra los componentes que necesitarás para este paso.

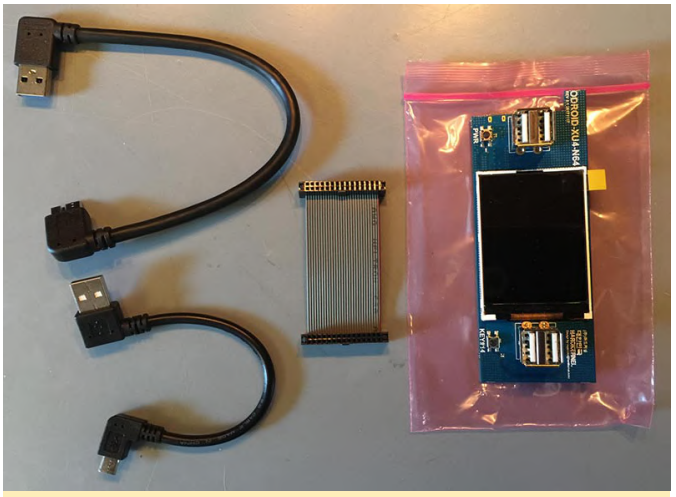


Figura 8

Conectaremos primero el cable plano a la placa montada. Asegúrate de tener el marcador de dirección en la dirección correcta. El marcador necesita espacio en el conector del cable plano. Alinea la ranura del conector del cable plano con la marca del propio cable. Asegúrate cuidadosamente de que el cable plano esté completamente dentro del

conector. Es posible que te lleve algo de tiempo, empujarlo hacia abajo de manera uniforme. SUGERENCIA: No ejerces demasiada presión sobre la placa cuando conectes el cable plano..

A continuación, conectaremos la placa de expansión. Nuevamente, asegúrate de que la marca del cable plano esté correctamente alineada con la ranura del conector del cable plano. Ten mucho cuidado al conectar el cable plano a la placa de expansión, ya que es muy fácil terminar presionando la pantalla llegando incluso a agrietarla. Tras de conectar la banda, retira la pegatina de protección de la pantalla y deja que la pantalla se asiente en las dos ranuras de guía que forman parte de la carcasa, como se muestra a continuación. De momento, no la empujes hasta el fondo de la carcasa.

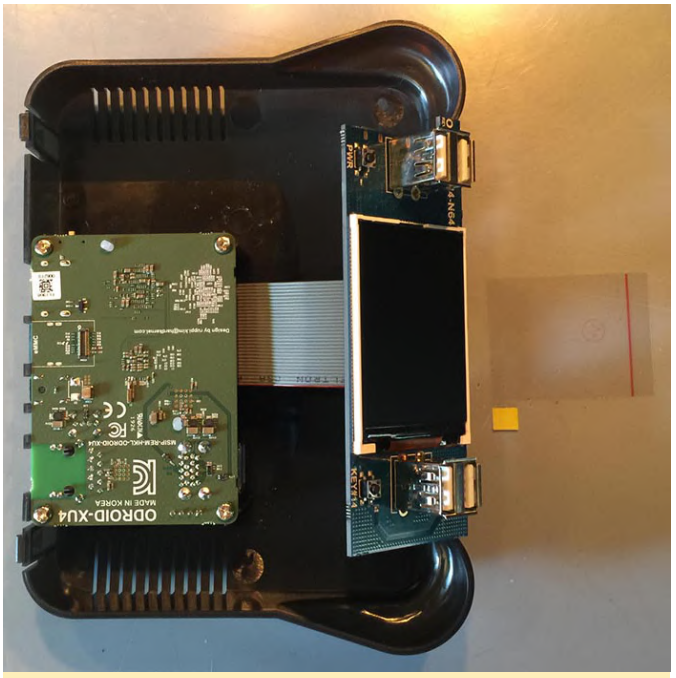


Figura 9

Ahora conectaremos el cable USB a USB. Esto dividirá un puerto USB de la placa ODROID-XU4 en 4 puertos USB que estarán disponibles en el frontal de la carcasa OGST. Ten en cuenta que queremos que el cable USB se ubique encima del cable plano tal y como se muestra a continuación. Estamos trabajando en la parte superior de la carcasa, de modo que todo lo que estamos haciendo se invertirá cuando hayamos terminado. El cable USB debe estar debajo del cable plano, esto facilitará el acceso en el caso de que quedamos desmontar la carcasa.

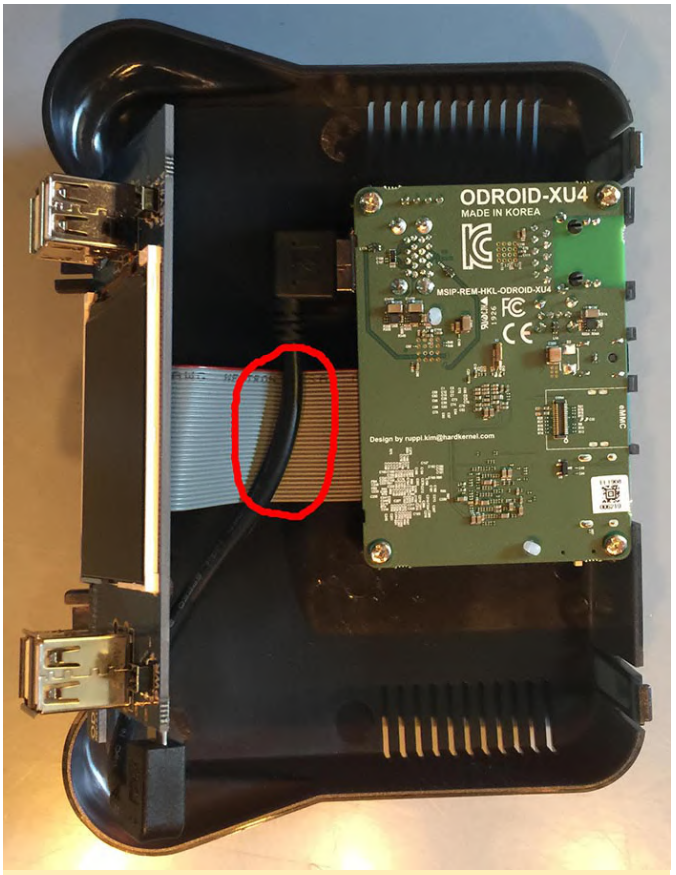


Figura 10

Terminando

Ahora vamos a preparar el disco duro. Sosteniendo el disco duro al revés, usa el cable USB de la unidad incluido para conectar éste a la placa de expansión. Ten en cuenta que estamos doblando el cable, en cierto modo imagina cómo se sitúa el disco duro dentro de la carcasa. Este es un buen momento para asegurarnos de que la unidad encaja y que todos los cables se ubican correctamente. Echa un vistazo a la siguiente imagen. En esta foto, compruebo dos veces que el cable está conectado correctamente a la unidad. Parece que se adapta muy bien a la carcasa

Figura 11

Figura 12

Ok, todavía no hemos terminado. Necesitamos hacer algo más para asegurarnos de que la unidad se asiente correctamente en la parte inferior de la carcasa. Mientras dejas la parte superior de la caja como está, ubica la parte inferior de la carcasa al lado y coloca, pero no pegues, las tiras de velcro donde quieras. Yo normalmente uso la parte del bucle en la unidad. Puedes configurarlo como quieras. Si usas tiras más anchas, la sujeción será más fuerte y tendrá más flexibilidad en la forma de colocar el disco duro en la parte de la cinta sobre la unidad. La siguiente foto muestra mi primer intento de colocar las tiras de velcro. Finalmente decidí doblar las tiras para agrandar la superficie de contacto. SUGERENCIA: asegúrate de tener en cuenta el cable de la unidad al fijar las tiras de velcro en la parte inferior de la carcasa.

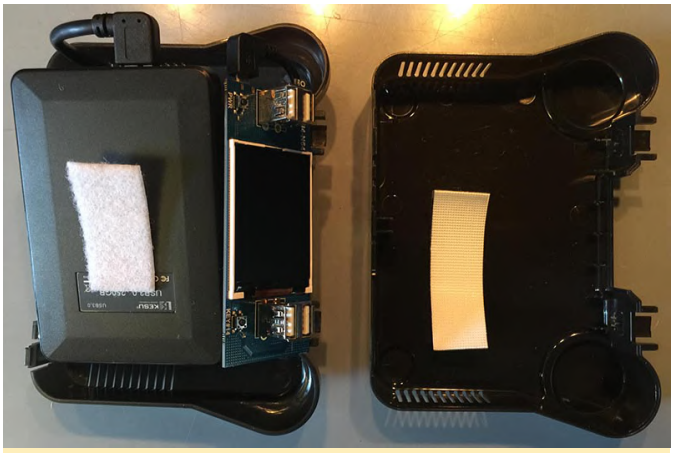


Figura 13

A continuación, ajustaremos la placa de expansión firmemente en su lugar. Ten cuidado y asegúrate de que entre de manera uniforme. Si un lado queda más abajo que otro, no podrá colocar la placa correctamente. Resiste el impulso de aplicar demasiada presión durante este paso. Simplemente aplica la fuerza necesaria para ajustarla correctamente. Ahora puede colocar la parte superior de la carcasa verticalmente en tu superficie de trabajo y colocar la unidad y la parte inferior de la carcasa al lado con el cable de la unidad USB conectado tal y como se muestra a continuación. Realiza una verificación final antes de proceder a cerrar la carcasa.

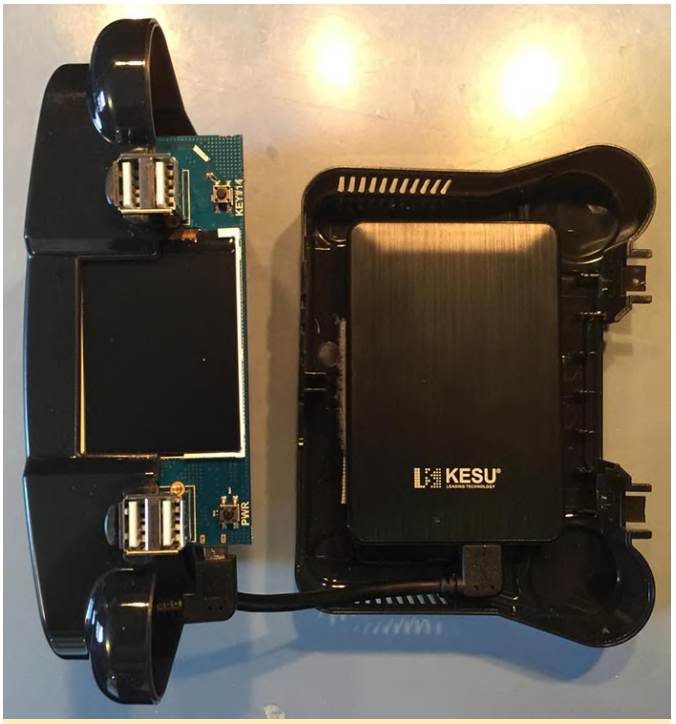


Figura 14

¡Finalmente, es hora de cerrar la carcasa! Junta los dos lados. Asegúrate de que los cables USB estén debajo del cable plano de la placa de expansión. Asegúrate también de que el cable USB del disco duro esté colocado y ajustado correctamente y que la unidad esté asegurada con las tiras de velcro. Cierra la caja y coloca las dos tapas de plástico gris sobre los puertos USB frontales. CONSEJO: asegúrate de alinear los dos pequeños pliegues de un lado de la tapa. Estos se alinean con pequeños pliegues de la carcasa.

Figura 15

¡Felicidades! Está todo listo. ¡Has añadido más almacenamiento a tu sistema ODROID-XU4! Las siguientes imágenes muestran la configuración completa. ¡A Disfrutar!

Figura 16

Figura 17

Referencias

http://middlemind.net/tutorials/odroid_go/mfb_build.html

Retro ESP32: La Mejor Imagen de Emulación para tu ODROID-GO

January 1, 2020 By @32teeth Juegos, ODROID-GO

Retro ESP32 es el último lanzador repleto de funciones para ODROID-GO. El lanzador incluye esquemas de colores y temas inspirados en la popular interfaz de usuario del emulador RetroArch. En el momento de la publicación de este artículo, llegamos a pre-compilamos 11 emuladores incluido ROM/Game Manager. Además, cada emulador incluye un menú en el juego permitiendo una mejor administración de la ROM.

Figura 1 – Lanzador ESP32 en acción en ODROID-GO

Instalación

La instalación de Retro ESP32 es muy simple.

1. Descárgate la última versión: **https://github.com/retro-esp32/RetroESP32/releases**
2. Descomprime el archivo
3. Copia RetroESP32.fw a la carpeta odroid/firmware de tu tarjeta SD (**https://github.com/retro-esp32/RetroESP32/blob/Software/SD%20Card/SDCARD.zip**)
4. Monta la tarjeta SD en tu ODROID-GO
5. Reinicia y mantén presionado el botón B
6. Selecciona Retro ESP32 de la lista de firmware
7. Siéntate y relájale mientras tu ODROID-GO se actualiza con el nuevo firmware

Emuladores Soportados

Retro ESP32 admite una amplia gama de emuladores para que puedas jugar en el ODROID-GO. Aquí tienes una lista de todos los emuladores soportados:

Nintendo Entertainment System Nintendo Game Boy Nintendo Game Boy Color Sega Master System Sega Game Gear Colecovision Sinclair Zx Spectrum 48k Atari 2600 Atari 7800 Atari Lynx PC Engine

Peticiones de los usuarios

¿Tienes una gran idea? ¿Deseas ver una característica añadida en el próximo lanzamiento? ¿O te topaste con algún problema? Usa nuestras secciones de Proyecto (**https://github.com/retro-esp32/RetroESP32/projects/1**) y (**https://github.com/retro-esp32/RetroESP32/issues**) para enviarnos la información.

Referencias

Este proyecto ha sido posible gracias al trabajo de los siguientes personas:

- Eugene Yevhen Andruszczenko - Trabajo inicial y en curso - **32teeth**
- Fuji Pebri - Consultor de Espressif IOT - **pebri86**

Este artículo es una adaptación del README.md de la página GitHub del proyecto. Para más información, visita el repositorio en: **https://github.com/retro-esp32/RetroESP32/**

Cómo Configurar y Usar el Bus CAN: Utilizando el ODROID-N2 con Microcontroladores

January 1, 2020 By Justin Lee, CEO of Hardkernel ODROID-N2, Mecaniqueo, Tutoriales

Este artículo explica cómo habilitar el bus CAN en ODROID-N2 a través de la interfaz HW SPI. También incluye instrucciones detalladas para obtener datos a través de una placa Monitor Bus MCP2515.

Fig. 01

Conexión H/W

Los siguientes artículos son necesarios para configurar el hardware:

- ODROID-N2
- Kit de pequeños ajustes.
- Módulo CAN MCP2515

Fig. 02

Fig. 03

Fig. 04

Circuito de referencia

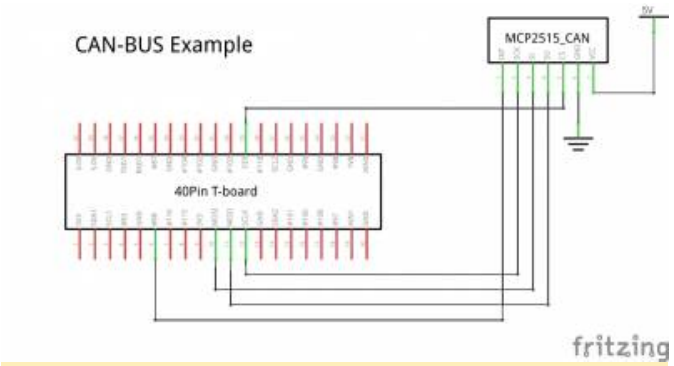


Fig. 05

Con kit de pequeños ajustes

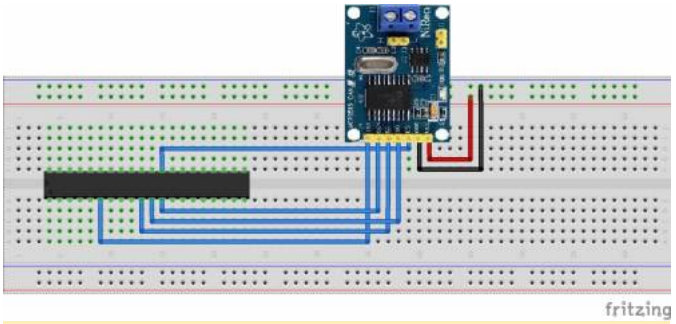


Fig. 06

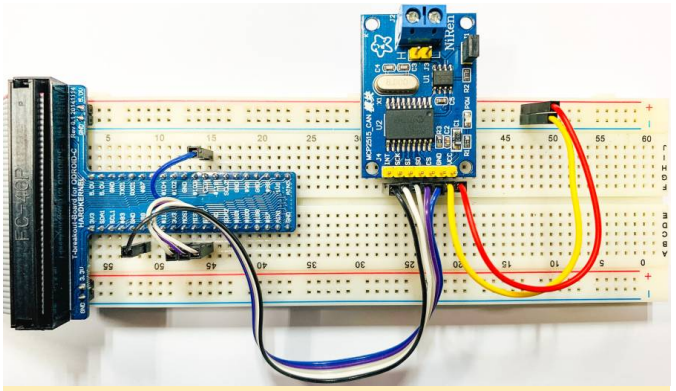


Fig. 07

Instalación del Software

Nota:

- Operación confirmada con imagen mínima de Ubuntu ODROID-N2 sobre el kernel 4.9.205-64.
- El ejemplo de can-bus usa el mismo pin cs que spidev, de modo que no deben habilitarse al mismo tiempo.
- Si spidev está activado, el can-bus puede no funcionar correctamente

Se recomienda encarecidamente actualizar el kernel. Todo esto funciona con Linux odroid 4.9.205-64 o versión superior

```
root@odroid:~# apt update && apt full-upgrade
```

Habilita los módulos usando **device-tree-compiler**

```
root@odroid:~# apt install device-tree-compiler
```

Cambia el estado a **okay** en los nodos SPI del árbol de dispositivos.

```
# SPICC0
root@odroid:~# fdtput -t s
/media/boot/meson64_odroidn2.dtb
                /soc/cbus@ffd00000/spi@13000
[status okay
root@odroid:~#
# can0
root@odroid:~# fdtput -t s
/media/boot/meson64_odroidn2.dtb

/soc/cbus@ffd00000/spi@13000/can@0
status okay
root@odroid:~#
```

Comprueba si el estado ha cambiado

```
# SPICC0
root@odroid:~# fdtget
/media/boot/meson64_odroidn2.dtb
                /soc/cbus@ffd00000/spi@13000
Status okay
root@odroid:~#
# can0
root@odroid:~# fdtget
/media/boot/meson64_odroidn2.dtb

/soc/cbus@ffd00000/spi@13000/can@0
Status okay
root@odroid:~#
```

Luego reinicia para aplicar los cambios. También puedes verificar si los módulos se han cargado correctamente.

```
root@odroid:~# lsmod | grep spi
spi_meson_spicc      20480  0
root@odroid:~# lsmod | grep mcp251x
mcp251x              24576  0
can_dev              24576  1 mcp251x
root@odroid:~#
```

Verificando la configuración de soporte CAN

Comprueba que el driver del host CAN esté registrado correctamente.

```
root@odroid:~# ls /sys/class/net/
can0  eth0  lo
root@odroid:~# ifconfig can0
can0: flags=128  mtu 16
      unspec 00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00  txqueuelen 10  (UNSPEC)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

root@odroid:~#
```

Enciende el hardware CAN. Ajusta la tasa de bits antes de realizar cualquier operación. Ejemplo: Fija la tasa de bits de la interfaz can0 a 125 kbps:

```
root@odroid:~# ip link set can0 type can bitrate
125000 triple-sampling on
root@odroid:~# ifconfig can0 up
root@odroid:~# ifconfig
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
      unspec 00-00-00-00-00-00-00-00-00-00-00-
00-00-00-00-00  txqueuelen 10  (UNSPEC)
      RX packets 0  bytes 0 (0.0 B)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 0  bytes 0 (0.0 B)
      TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500
      inet 192.168.10.8  netmask 255.255.255.0
broadcast 192.168.10.255
      inet6 fe80::e160:7710:5360:f82a  prefixlen
64  scopeid 0x20
      ether 02:00:00:0d:1d:01  txqueuelen 1000
(Ethernet)
      RX packets 24  bytes 6066 (6.0 KB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 54  bytes 6420 (6.4 KB)
      TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0
      device interrupt 22

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
      inet 127.0.0.1  netmask 255.0.0.0
      inet6 ::1  prefixlen 128  scopeid 0x10
      loop  txqueuelen 1  (Local Loopback)
```

```
      RX packets 129  bytes 10117 (10.1 KB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 129  bytes 10117 (10.1 KB)
      TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0

root@odroid:~#
```

Instalar las utilidades SocketCAN.

El paquete can-utils es un conjunto de drivers CAN y herramientas de red para Linux. Permite interactuar con dispositivos de bus CAN de forma similar a otros dispositivos de red.

```
sudo apt install can-utils
```

Necesitamos realizar una prueba de circuito cerrado en un solo puerto CAN. Configura el modo loopback en can0

```
ifconfig can0 down
ip link set can0 type can bitrate 125000 loopback
on
ifconfig can0 up
ip -details link show can0

root@odroid:~# ifconfig can0 down
root@odroid:~# ip link set can0 type can bitrate
125000 loopback on
root@odroid:~# ifconfig can0 up
root@odroid:~# ip -details link show can0
3: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 16 qdisc
fq_codel state UNKNOWN mode DEFAULT group default
qlen 10
  link/can  promiscuity 0
  can <LOOPBACK,TRIPLE-SAMPLING> state ERROR-
ACTIVE restart-ms 0
        bitrate 125000 sample-point 0.850
        tq 400 prop-seg 8 phase-seg1 8 phase-seg2
3 sjw 1
        mcp251x: tseg1 3..16 tseg2 2..8 sjw 1..4
brp 1..64 brp-inc 1
        clock 5000000numtxqueues 1 numrxqueues 1
gso_max_size 65536 gso_max_segs 65535
root@odroid:~#
```

El siguiente comando muestra el mensaje recibido desde el bus CAN

```
candump can0
```

En un segundo terminal, el siguiente comando envía 3 bytes sobre el bus (0x11, 0x22, 0x33) con el identificador 500.

```
cansend can0 500#11.22.33
```

Cómo probar el enlace CAN-bus entre 2 placas ODROID-N2

Conecta los pines CANL, CANH entre dos placas ODROID-N2

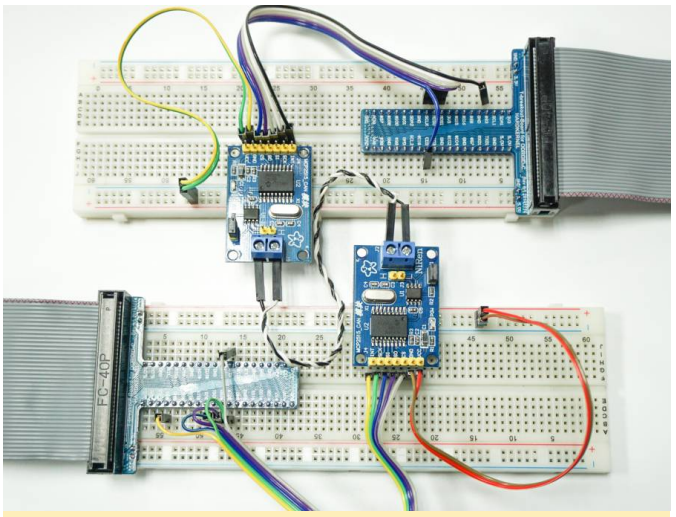


Fig. 08

Enciende ambas placas y escribe lo siguiente en el intérprete de comandos de ambas placas para configurar el dispositivo de bus CAN:

```
ip link set can0 type can bitrate 125000 triple-
sampling on
ifconfig can0 up
```

Escribe lo siguiente en el intérprete de comandos de la placa 1 (que se utiliza para probar /recibir en el dispositivo can0):

```
candump can0
```

Escriba lo siguiente en el intérprete de comandos de la placa 2 (que se utiliza para probar/enviar paquetes de datos a través del dispositivo can0):

```
cansend can0 500#11.22.33
```

En este punto, la placa 1 recibirá el paquete de datos que ha sido enviado desde la placa 2:

```
root@odroid:~# candump can0
  can0  500   [3]  11 22 33
  can0  500   [3]  11 22 33
```

Referencias

https://wiki.odroid.com/odroid-n2/application_note/gpio/can-bus

ODROID-GO Advance: La más Reciente y Famosa Generación de Ordenadores Portátiles de Hardkernel

January 1, 2020 By Justin Lee, CEO of Hardkernel Juegos, ODROID-GO Advance, ODROID-GO

Anunciamos el ODROID-GO en junio de 2018 para celebrar nuestro décimo cumpleaños. Ha sido increíble y divertido poder emular juegos retro de 8 bits de la vieja escuela con un rendimiento mejor de lo esperado con solo la MCU, más que una simple MPU de gama alta. El dispositivo ha sido muy usado no solo para jugar sino también a nivel educativo.

Figura 1 - ODROID-GO Advance

Figura 2 - ODROID-GO Advance

Hemos continuado escuchando a los usuarios que querían jugar a juegos retro de 16 o 32 bits en un dispositivo portátil con potencial y características más avanzados. De modo que, hemos estado investigamos es una nueva plataforma este año y nos hemos topado con una solución ideal, así que hemos pasado varios meses desarrollando un nuevo dispositivo Linux de 64 bits. Este nuevo dispositivo, llamado ODROID-GO Advance, cuenta con un moderno procesador ARM de cuatro núcleos de baja potencia de 64 bits, así como un LCD de 3.5 pulgadas con un amplio ángulo de visión.

Especificaciones de la ODROID-GO Advance

Procesador	CPU : RockChip RK3326(Quad-Core ARM Cortex-A35 1.3GHz) GPU : Mali-G31 Dvalin
Memoria	1GB (DDR3L 786Mhz, con bus de 32 Bits)
Almacenamiento	Flash SPI (arranque de 16Mbytes), ranura para tarjeta Micro SD (interfaz compatible con UHS-1)
Pantalla	LCD TFT 320 × 480 de 3,5 pulgadas (ILI9488, interfaz MIPI)
Audio	Clavija para auriculares, 0.5Watt 8Ω Mono
Batería	Li-Polymer 3.7V/3000mAh, hasta 10 horas de juego continuo
Clavija DC	Conector DC de 2.5 mm de diámetro: se incluye un cable de carga USB en el pack
E/S externa:	Un USB 2.0 Host, un puerto de 10 pines (I2C, GPIO, IRQ a 3.3Volt)
Botones de entrada	F1, F2, F3, F4, F5, F6, A, B, X, Y, Pad de dirección, botón superior izquierdo, botón superior derecho, joystick analógico
Consumo Energía	Consumo de energía de Emulación del juegos: 100 ~ 115mA, Modo de reposo: 5.3 ~ 5.8mA, Apagado: 0.1mA

En este momento, la imagen de BSP de prueba es compatible con los siguientes sistemas:

- Atari 2600
- Atari 5200
- Atari 7800
- Atari Lynx
- Gamegear
- Gameboy
- Gameboy Advance
- Gameboy Color
- Sega Master System
- Sega Genesis
- Nintendo
- PC Engine
- PC Engine CD
- Sony PlayStation
- Sega CD
- Super Nintendo
- Sony PlayStation Portable

Puedes ver algunos videos del ODROID-GO Advance en acción en https://youtu.be/okVJe6ywc4c y https://youtu.be/im46rlz0Nwg. Estará disponible por 55$ a finales de enero de 2020.

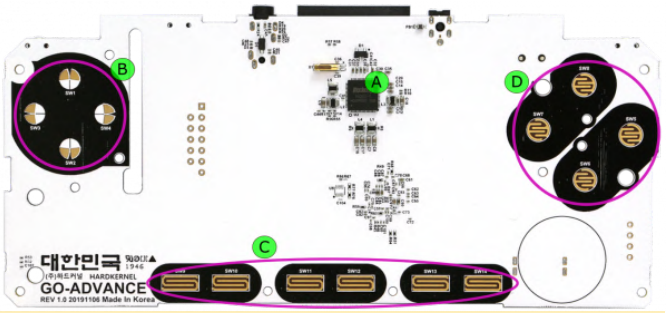


Figura 3 - Diagrama con notas del exterior del ODROID-GO Advance

A PMIC(RK817) incluyendo el cargado y audio B Botones D-pad C I ~ VI botones (F1, F2, F3, F4, F5, F6) D Botones X, Y, A, B

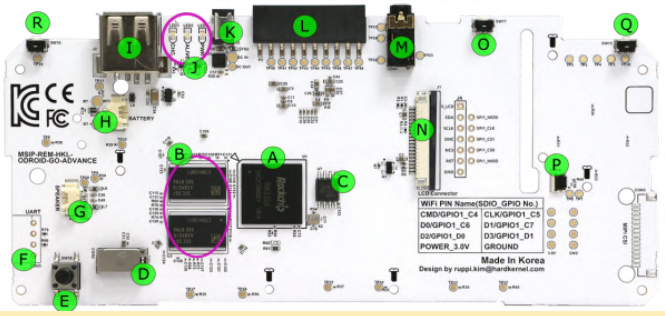


Figura 4 - Diagrama con notas del interior del ODROID-GO Advance

A CPU : Rockchip RK3326 B RAM : 1GB DDR3L C Flash SPI (16Mbytes de arranque) D Ranura tarejta MicroSD E Arranque forzado de tarjeta SD (sin spirom) F Puerto UART(But no montado por defecto) G Conetor altavoces H Conector Batería I USB 2.0 tipo A Host J Estados LED(carga, en funcionamiento, alimentación) K Clavija alimetnación DC L Puerto de expansión 10pin M Clavija Audio N Conector LCD 20pin O switch PWR P Conector joystick Analogico Q Botón superior izquierdo R Botón superior derecho

Cómo usarlo

Los siguientes enlaces proporcionan información sobre cómo usar ODROID-GO Advance:

- Montaje del kit ODROID-GO-Advance
- Instalar imagen de SO en tu tarjeta SD

Transferir roms de juegos a través del lector de tarjetas SD (PC-HOST Linux)

Inserta la tarjeta SD en tu PC HOST y luego copia las ROM del juego en la carpeta /roms, como se muestra en la Figura 5. Puedes copiar las ROM del juego en la carpeta /roms sin ningún permiso

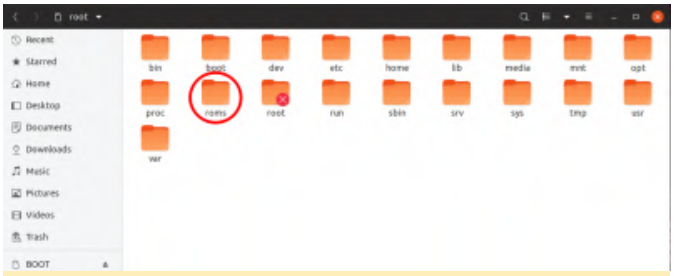


Figura 5 - La carpeta /roms de ODROID-GO Advance

Puesto que el sistema de archivos de la partición de datos de la tarjeta SD es EXT4, no puede acceder a él desde una PC con Windows, por lo que debemos preparar una forma de transferir archivos ROM desde un almacenamiento USB en el sistema. En primer lugar, comprueba tu entorno de red. Si tiene algún módulo de red USB, sigue estas instrucciones sobre **cómo conectar tu GO-Advance a una red inalámbrica con un adaptador WiFi USB adicional**. Los dongles WiFi compatibles se venden por separado (**módulo 0 WiFi**, **Módulo 3 WiFi**, **Módulo 5A WiFi**) Después de esto, podrás enviar las ROM de tus juegos al GO-Advance con el comando "scp" desde tu PC HOST:

```
$ sudo apt install ssh
$ scp  odroid@:/roms//
```

Por ejemplo:

```
$ ping 192.168.0.10
$ scp test.gba odroid@192.168.0.10:/roms/gba/
```

Para obtener más información, visita la Wiki de ODROID en **https://wiki.odroid.com/odroid_go_advance/start** .

Monku R4 con un ODROID-N2 y Batocera Linux: La Mejor Consola de Juegos Retro que Puedes Montar por Alrededor de 100$

January 1, 2020 By Brian Ree Juegos, ODROID-N2, Tutoriales

Componentes necesarios

Antes de empezar, debes reunir los siguientes elementos. Todos estos artículos se los puedes comprar a Hardkernel:

- ODROID-N2 (2GB de RAM) - Carcasa ODROID-N2 - 2 tarjetas Micro SD de 64GB - Un cable HDMI - Una Fuente de Alimentación 12V/2A - Un mando por cable GameSir - Un módulo de memoria eMMC de 16GB o 32GB - Un adaptador USB a Micro SD - Un adaptador eMMC a Micro SD

Introducción y objetivos del tutorial

Este tutorial cubre el proceso de configuración de un ODROID-N2 de 2 GB de RAM y la instalación de Batocera Linux para que podamos usar nuestro ODROID-N2 como una consola de juegos retro conectada a la TV. Con mucho amor llamo a este dispositivo Monku R4. Me centraré en la configuración del sistema operativo, la configuración de las ROM y los archivos de BIOS, y mostraré, además, como obtener imágenes de las cajas y capturas de pantalla de las ROM. En mi opinión, esta es la MEJOR consola de juegos retro que puedes montar por poco dinero. Ejecuta un montón de emuladores y todos los ejecuta muy bien. Echemos un vistazo a algunos de los emuladores que puede ejecutar. Personalmente he configurado una para ejecutar los siguientes emuladores.

- Atari 2600
- Atari 5200
- Atari 7800
- C64
- Colecovision
- DOOM

- Dreamcast
- FBA MAME
- Game Boy
- Game Boy Advance
- Game Boy Color
- Intellivision
- Jaguar
- Lynx
- Magnavox Odyssey
- MAME
- MS-DOS
- MSX 1/2
- N64
- NES
- NEO-GEO Pocket
- NEO-GEO Pocket Color
- PSP
- PS1
- ScummVM
- Sega 32X
- Sega CD
- Sega GameGear
- Sega Genesis
- Sega Master System
- Sega SG-1000
- SNES
- Turbo Grafx 16
- Turbo Grafx 16 CD
- Virtual Boy
- WonderSwan
- WonderSwan Color
- ZX Spectrum

Una nota sobre el coste: En el titulo he comentado que esta es la mejor consola de juegos retro que puedes montar por menos de 100$, sin embargo, para ello tendrías que obtener solo una tarjeta SD de 64 GB y el módulo eMMC más pequeño con el fin de mantenerte en los 100$. Recomiendo comprar un pack de dos tarjetas SD de 64 GB para hacer una copia de seguridad. Ahora que tienes una idea de con qué vas a trabajar, y para que quede claro, el ODROID-N2 es bastante más potente que el ODROID-XU4 (lo siento, por fanáticos del ODROID-XU4), vamos a adaptar este tutorial para usar un módulo eMMC como unidad de arranque del sistema operativo para lograr un mejor rendimiento del ODROID-N2. Puede decidir ejecutar las cosas completamente desde una tarjeta micro SD o un módulo eMMC más grande, si lo crees conveniente. No lo tratare en detalle, pero puedes coger diferentes partes del tutorial y aplicarlas solo con una tarjeta SD o solo un módulo eMMC. La separación que tenemos entre el sistema operativo y las ROM nos permite mantener la tarjeta SD independiente del sistema operativo y en un sistema de archivos Fat32. Esto significa que podemos extraer la tarjeta SD y conectarla a cualquier ordeandor y editarla, según sea necesario. Es un poco más complicado con un sistema de archivos ext4 Linux de arranque para Mac o un PC con Windows. CONSEJO: si optas por una instalación de almacenamiento única, solo eMMC o solo SD, lo más probable es que tenga que iniciar sesión en tu dispositivo a través de SSH en algún momento del proceso de configuración para configurarlo. La contraseña root por defecto para el dispositivo es Linux.

Configuración del módulo eMMC

Echemos un vistazo al hardware que necesitaremos para escribir en el módulo eMMC. A continuación, se muestra una imagen del adaptador eMMC a microSD. Debajo hay una imagen del adaptador y del módulo eMMC. Usaré un módulo de 32 GB, aunque realmente solo necesitas un módulo de 16 GB ya que solo lo usaremos para alojar el sistema operativo. Si tienes pensado realizar una configuración más avanzada con las ROMs en múltiples ubicaciones, eMMC y SD, entonces probablemente querrás usar un módulo eMMC de 32 GB o más.

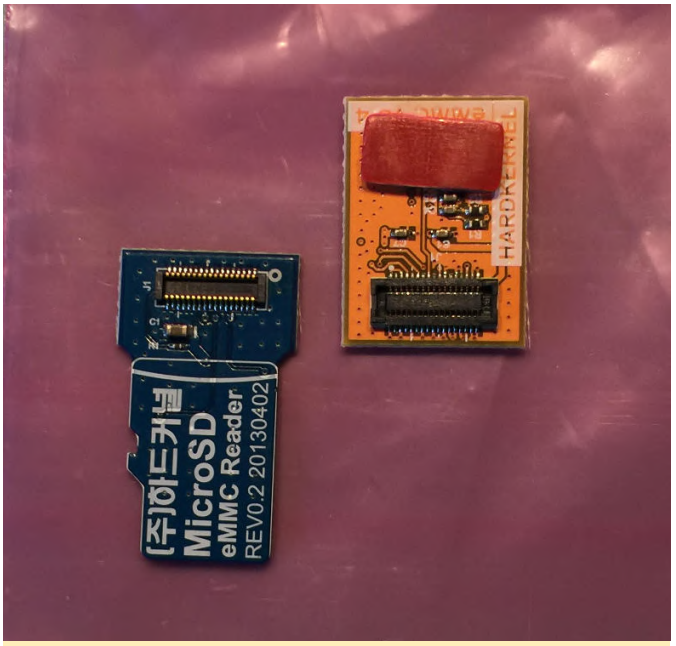


Figura 1 - Módulo eMMC y adaptador SD

Figura 2: módulo eMMC montado en el adaptador de tarjeta SD

Si eres nuevo con los módulos eMMC, te recomiendo que sigas el siguiente tutorial, ya que solo cubriré en este caso el proceso de escritura de imágenes y no los pasos específicos del módulo eMMC.

- Tutorial sobre el funcionamiento de los módulos eMMC

Necesitarás hacerte con una copia de la última versión de Batocera Linux para ODROID-N2. Batocera Linux está basado en Recalbox Linux, por lo que, si estás familiarizado con RecalBox, vas a ir por delante del resto. Usa los siguientes enlaces para localizar la última versión deBatocera Linux para ODROID-N2 y descárgatela. CONSEJO: Si bien este tutorial se centra en el ODROID-N2, puede usarlo como guía general para instalar Batocera Linux en cualquier otro hardware como el ODROID-XU4.

- Batocera: Página de descarga general

- Batocera: Página de descarga específica de ODROID-N2

Una vez que tenga tu imagen lista, es hora de conseguir algún software que puedas usar para actualizar el módulo eMMC. Si está utilizando una Mac, te recomiendo obtener Balena Etcher. Funciona muy bien y lo recomiendo encarecidamente. Si está utilizando Windows, puede obtener una copia de Win32 Disk Imager. Aunque no es tan bueno como Balena Etcher, Win32 Disk Imager funciona. Para los usuarios de Linux, éstos deberás realizar los siguientes pasos. No te preocupes, no es tan tán difícil.

1. Inserta tu tarjeta SD en tu ordenador 2. Localiza el dispositivo ejecutando sudo fdisk -l. Probablemente será el único disco del tamaño correcto. Anota el nombre del dispositivo; supongamos que es /dev/sdx. Si tiene alguna duda, retira la tarjeta, ejecuta sudo fdisk -l nuevamente y anota qué discos aparecen. Inserta la tarjeta SD nuevamente, ejecute sudo fdisk -l y se mostrará el nuevo disco. 3. Desmonta las particiones ejecutando sudo umount /dev/sdx *. Puede aparecer un error diciendo que el disco no está montado, es normal. Copia el contenido del archivo de imagen en la tarjeta SD ejecutando

```
$ sudo dd bs=1M if=your_image_file_name.img
of=/dev/sdx
```

Por supuesto, necesitas cambiar el nombre del archivo de imagen, según corresponda. También deberás ajustar el argumento de destino “of”, para que coincida con el dispositivo de destino de tu sistema. AVISO: asegúrate de que la unidad, el dispositivo y la letra de la unidad a escribir sean los

correctos. ¡Asegúrate de no sobrescribir otra unidad importante! adjuntaré capturas del proceso tal como se ve en un Mac. Es posible que te solicite que proporciones privilegios de administrador en Mac y Windows.

Selecciona el archivo de imagen que deseas grabar en el módulo eMMC. El archivo de imagen que se muestra a continuación no es el que usarás en este tutorial. Utilizarás tu archivo de imagen ODROID-N2 de Batocera Linux.

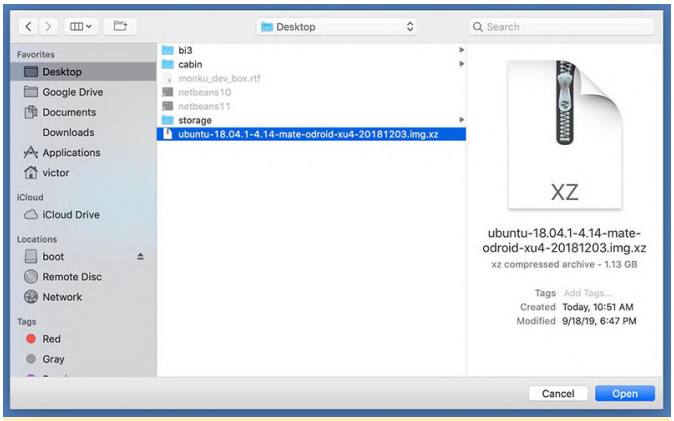


Figura 3 - El archivo de imagen comprimido

Figura 4: Responde a cualquier solicitud de privilegio de administrador

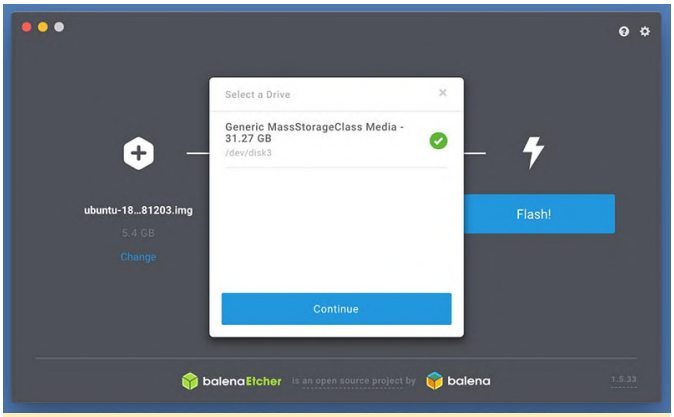


Figura 5 – Verifica que estás grabando en el dispositivo correcto y que tiene el tamaño aproximado correcto..

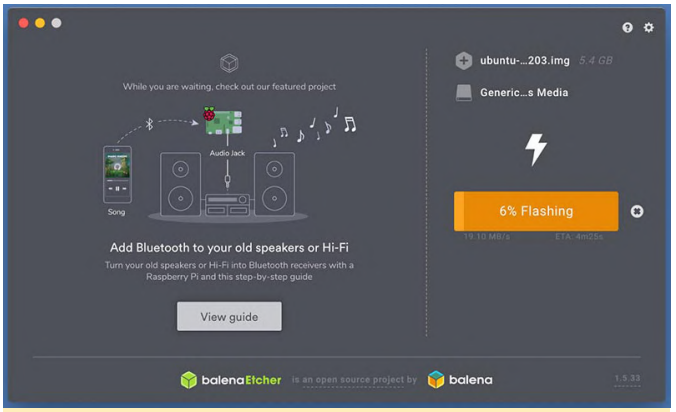


Figura 6: Empieza a grabar en el dispositivo y espera a que se complete el proceso.

En un PC con Windows, si está utilizando la herramienta de software mencionada anteriormente, verás algo como lo siguiente antes de hacer clic en el botón Write. Nuevamente espera a que la herramienta termine de escribir la imagen en el módulo eMMC. Como en el caso anterior, el archivo de imagen que se muestra a continuación no es el que usarás en este tutorial. Utilizará tu archivo de imagen ODROID-N2 de Batocera Linux. AVISO: asegúrate de elegir la letra de unidad correcta. ¡Comprueba hasta tres veces la letra para no sobrescribir otra unidad importante!

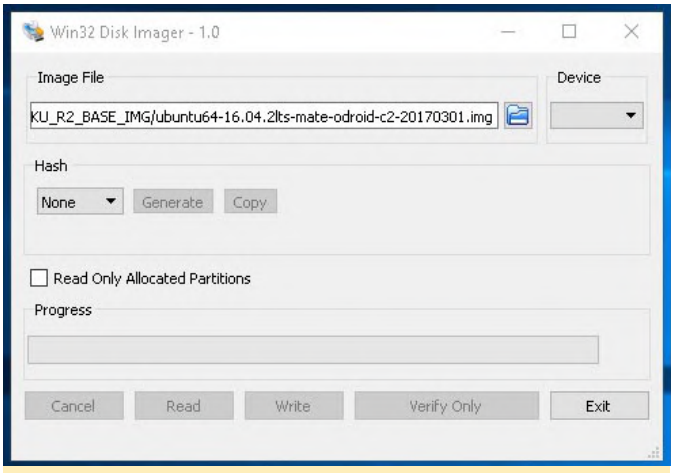


Figura 7 - Win32 Disk Imager

A continuación, iniciaremos el dispositivo y comprobaremos el sistema operativo. La imagen que aparece a continuación muestra el ODROID-N2 con una flecha roja al lado de la ranura del módulo eMMC y una flecha azul al lado de la ranura de la tarjeta SD. Si optas por la tarjeta micro SD, deberás retirar la tarjeta al mismo tiempo que colocas la carcasa.

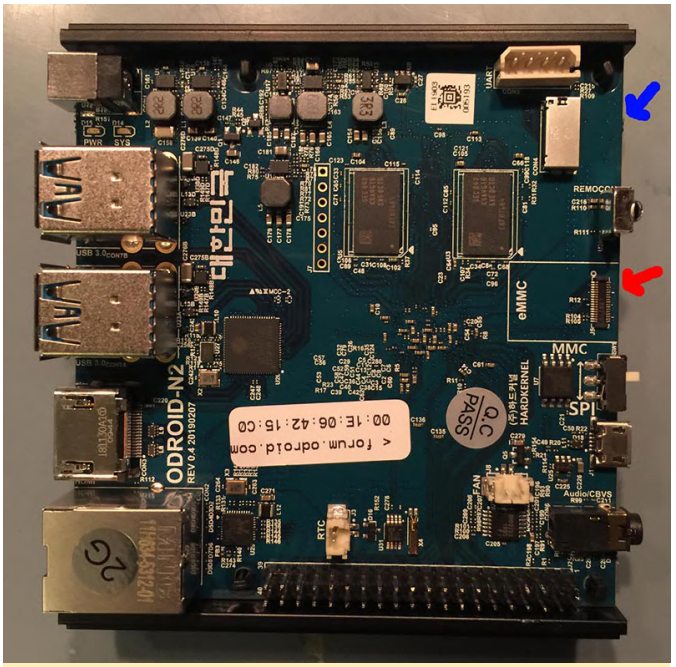


Figura 8: El módulo eMMC en el rojo y la tarjeta SD es el azul

Conecta el módulo eMMC al ODROID-N2 como se muestra a continuación y prepara una superficie libre de estática para el dispositivo. Desearás tener lista tu fuente de alimentación de 12V/2A.

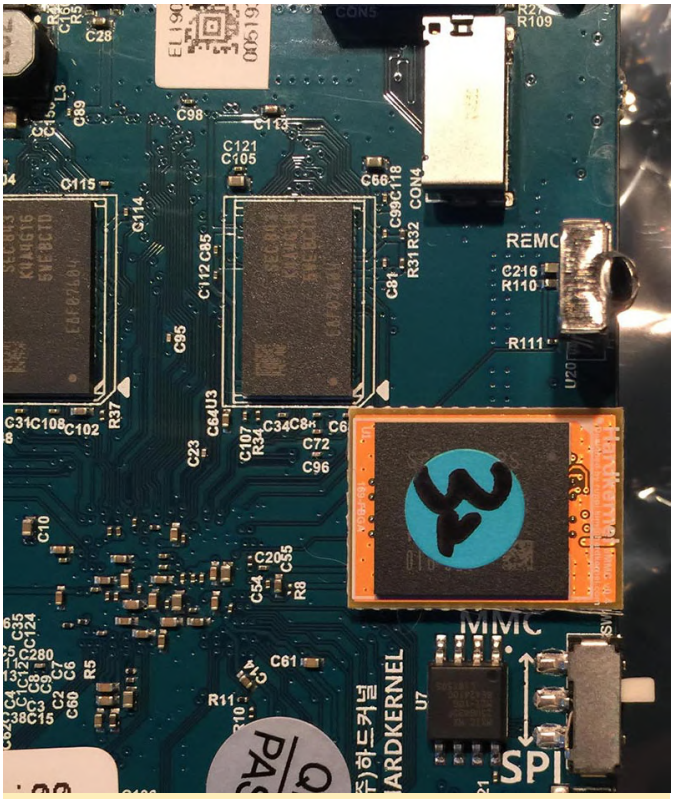


Figura 9: módulo eMMC de 32 GB cargado en el N2

Asegúrate de tener el interruptor blanco de la parte posterior del ODROID-N2 movido hacia la derecha para arrancar el módulo eMMC. Muévelo completamente hacia la izquierda para arrancar con la tarjeta micro SD. En nuestro caso, lo moveremos hacia la derecha para arrancar con el módulo eMMC. Seguiremos pudiendo acceder a la tarjeta micro SD como unidad.

Figura 10: el interruptor blanco se mueve a la derecha y se inserta la tarjeta SD

Conecta la fuente de alimentación y el cable HDMI, conten la respiración y, con suerte, verás una pantalla similar a la que se muestra a continuación.

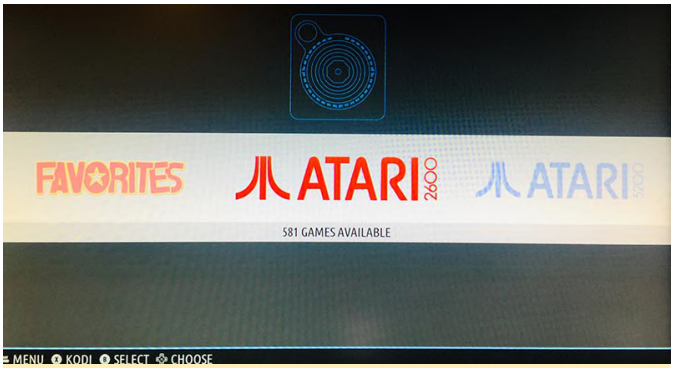


Figura 11 - Menú en funcionamiento

Apaga el ODROID-N2 saliendo del dispositivo usando un teclado. O, si tiene a mano tu mando de juego, puede configurarlo a través de la interfaz de usuario de Batocera Linux y luego apagar el dispositivo. Trataré cómo configurar las cosas en detalle en breve. Esto nos lleva al final de esta sección del tutorial. A continuación, montaremos la carcasa y luego pasaremos a algunas cuestiones de configuración y personalización.

Montar la carcasa

Veras que la carcasa está realmente diseñada con mucho ingenio. Algo que es genial es que todo descansa sobre un disipador de calor. Así es, la parte inferior del N2 es un gran disipador de calor, aunque también actúa como una base realmente sólida para el dispositivo. CONSEJO: Usa cinta aislante negra y coloca 4 piezas en la base del disipador térmico del ODROID-N2, las dos piezas principales de metal que realmente tocan la superficie sobre la que descansa. Coloca una pieza cerca de cada una de las cuatro

esquinas. Esto creará un contacto más suave con las diferentes superficies y también evitará el deslizamiento. Coloca el ODROID-N2 y las partes de la carcasa como se muestra a continuación.

Hay un pequeño pliegue en el borde izquierdo y derecho del ODROID-N2, coge la parte superior de la carcasa más pequeña, la que está a la izquierda en la imagen de arriba, y deslízala sobre el ODROID-N2 teniendo cuidado de mantenerla sobre los bordes de guía. La siguiente imagen muestra la parte frontal más pequeña de la carcasa en posición y las guía.

Figura 12: pequeño pliegue en el borde izquierdo y derecho del N2

Luego desliza la parte superior de la carcasa más grande hasta que las dos se encuentren. Asegúrate de mantenerla recta mientras la empujas suavemente a lo largo de los bordes de la guía. Las dos piezas superiores de la carcasa se encontrarán y harán clic con un pequeño pliegue. ¡La carcasa está montada!

Figura 13 – Carcasa montada

Figura 14 - ¡glamour!

Carcasa cerrada. Bien, ahora que tenemos el ODROID-N2 correctamente guardado y protegido, reemplaza tu tarjeta SD, si estás usando este método, de la ranura situada la parte posterior de la carcasa. Inicia el dispositivo una vez más para asegurarte de

que todo está en orden. Esto nos lleva al final de esta sección. A continuación, vamos a trabajar en la configuración de nuestra tarjeta micro SD como sistema de archivos externo que Batocera Linux usa para acceder a las ROM. También voy a tratar la configuración de los mandos, las opciones avanzadas de configuración de Batocera Linux y el diseño de las cajas de las ROMs en este tutorial.

Configuración del mando y de la tarjeta Micro SD

Primero configuremos el mando. Si tu mando no funciona de la forma esperada, puedes usar un teclado USB para navegar por el menú principal. Intenta presionar el botón de inicio o equivalente en tu mando, esto abrirá el menú principal. También puede usar un teclado y presionar la barra espaciadora para que aparezca el menú principal. El mando GameSir que recomendamos tiene un botón de inicio. Deberías ver algo similar a lo que se muestra en la imagen que aparece a continuación. Selecciona la opción del menú Controller Settings CONSEJO: si estás utilizando un teclado para navegar por el sistema de menús, la tecla Intro se usa para hacer selecciones, la tecla Esc se usa para volver a un menú anterior y la barra espaciadora se usa para cerrar/abrir el sistema de menús. A continuación, deberás seleccionar la opción de menú “Configure a Controller”. Al seleccionar esta opción, aparecerá una pantalla para configurar el mando. Puede salir de esta pantalla presionando el botón de inicio o la barra espaciadora por segunda vez si ya tienes tu controlador configurado o si accidentalmente seleccionaste esta opción después de entrar en la configuración. Se te pedirá que presiones en serie ciertos botones del mando y luego estará listo. Es así, es muy fácil de hacer. A continuación, se muestra una imagen de la configuración del mando. CONSEJO: Al presionar el botón azul del mando GameSir y el botón de inicio al mismo tiempo, saldrás del emulador actual. Es posible que necesites configurar esto de un modo diferente para diferentes mandos. SUGERENCIA: Si el sistema no reconoce tu mando GameSir con cable, mantenga presionado el botón azul durante unos segundos hasta que el pequeño cuadrado rojo en la parte frontal del controlador se mueva a la segunda posición. Si aún no se reconoce, prueba la posición tres, luego la posición cuatro.

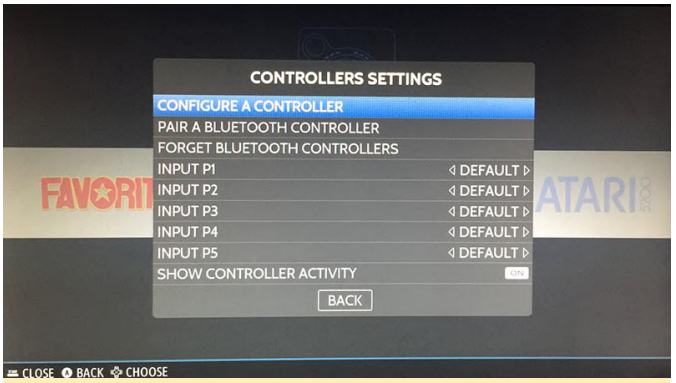


Figura 15

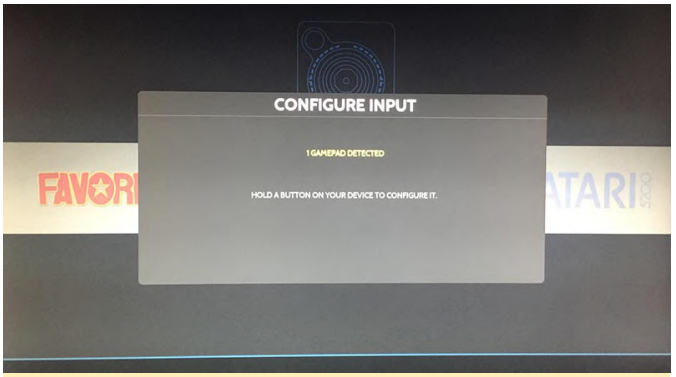


Figura 16

Lo siguiente que vamos a hacer es configurar la tarjeta micro SD para que funcione con Batocera Linux. Recomiendo usar una tarjeta de 64GB y contar con un pack de dos. Tiene al principio los enlaces de las que yo uso. Son asequibles y bastante confiables. Conecta la tarjeta micro SD y luego abre el menú principal con el botón de inicio o la barra espaciadora. Selecciona las opciones de Configuración del sistema tal y como se muestra a continuación.

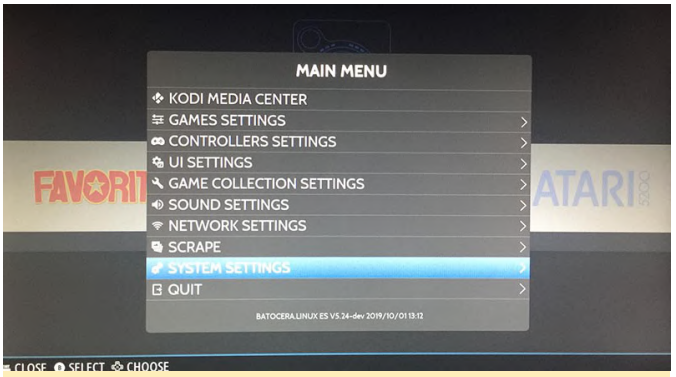


Figura 17: navega hacia abajo hasta la opción de menú Storage Device como se muestra a continuación.

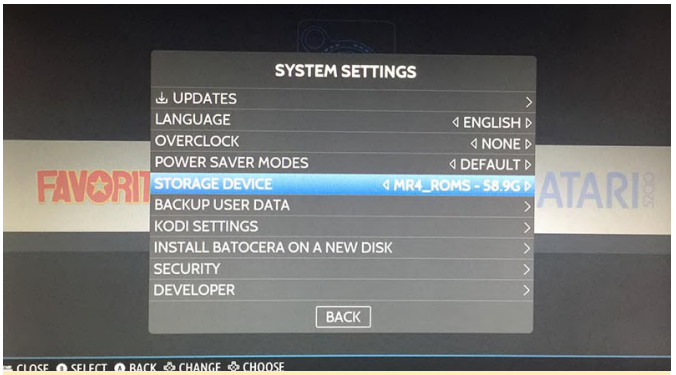


Figura 18

Aparecerá un cuadro de selección que te permitirá elegir entre algunas de las diferentes ubicaciones de almacenamiento. Elije la entrada que tiene el mismo nombre que la tarjeta micro SD que colocaste en el ODROID-N2. El sistema ahora se reiniciará después de seleccionar la unidad. Durante este reinicio, Batocera Linux creará, en tu tarjeta micro SD, un nuevo sistema de archivos que contendrá todas las ROM, archivos de BIOS y configuraciones para los emuladores que desea configurar. Esto nos lleva al final esta sección del tutorial. Ahora tenemos una versión de arranque de Batocera Linux ejecutándose en nuestro dispositivo ODROID-N2. Tenemos un mando de juego configurado y sabemos cómo navegar por el sistema de menús. También tenemos un sistema de archivos externo, nuestra tarjeta micro SD, preparada y lista para las ROMs y los archivos de BIOS.

Añadir ROM y archivos de BIOS

Ahora estamos listos para agregar archivos ROMs y BIOS a la tarjeta micro SD que configuramos con el sistema de archivos externo Batocera Linux en la última sección del tutorial. Necesitarás un adaptador para conectar la tarjeta micro SD a tu PC o Mac para poder copiar y pegar los archivos en el directorio adecuado. A continuación, se muestra una imagen del contenido de la carpeta root que se llama batocera.

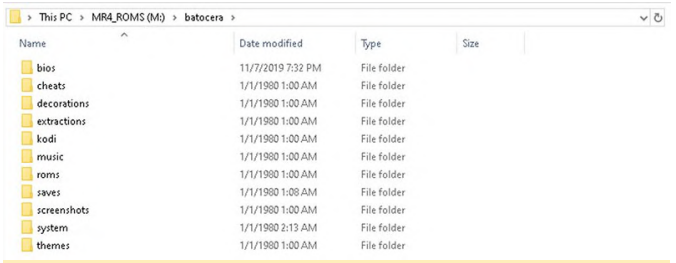


Figura 19

Ten en cuenta que tienes otras opciones que puedes usar en su dispositivo Batocera Linux como Kodi para reproducir música y videos. En nuestro caso, nos preocuparemos más de la carpeta ROMs. Abre la carpeta ROMs y verá un directorio para cada sistema compatible. Ahora no estoy 100% seguro de que, si cada emulador se puede ejecutar sobre cada componente de hardware en el que se puede instalar Batocera Linux, pero es cierto que los mejores lo hacen. Copia y pega tus ROMs en el directorio correspondiente. Si tiene alguna duda sobre dónde ubicar las ROMs de un determinado sistema, búscalo on line y podrás localizar la carpeta adecuada. También puede leer el archivo _info.txt de cada carpeta para ver qué sistema admite.

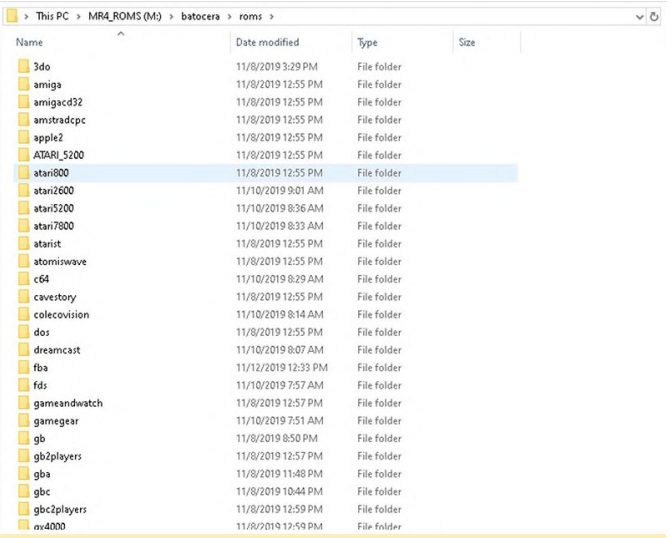


Figura 20

A continuación, tendrás que localizar los archivos BIOS correctos para cada sistema. No puedo publicarlos aquí, pero son lo suficientemente fáciles de encontrar en internet indagando un poco. Regresa de la carpeta roms y abre la carpeta bios. En la carpeta habrá un archivo de texto llamado readme.txt. Ábrelo para ver qué archivos de BIOS van en la carpeta bios. La mayoría debe colocarse directamente en la carpeta bios, algunos se colocarán en el mismo directorio que las ROM, el archivo readme.txt le dirá dónde colocarlos y qué archivos necesitas. CONSEJO: no todos los emuladores son delicados con los archivos BIOS y funcionarán bien con buenos archivos, incluso si no tienen el mismo hash MD5. Haz pruebas con cada sistema para ver cuáles son los que tienen problemas. Para los sistemas con problemas, revisa cuidadosamente sus

archivos BIOS. Tendrá que encontrar una herramienta o utilidad on line para obtener el hash MD5, los usuarios de Mac y Linux tienen un comando CLI MD5. Una vez que localices los archivos correctos, colócalos en la carpeta adecuada y vuelva a hacer pruebas con ese sistema hasta que funcione.

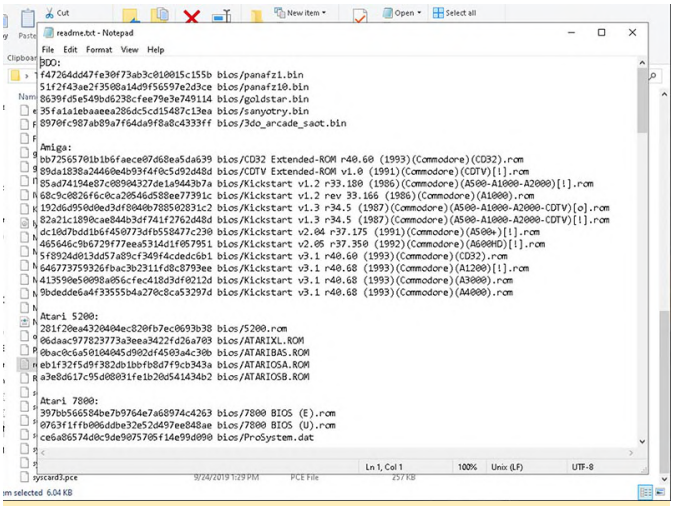


Figura 21

En realidad, puedes preguntarle a Batocera Linux qué archivos BIOS faltan. Dirigete a la opción de Configuración del juego del menú principal como se muestra a continuación.

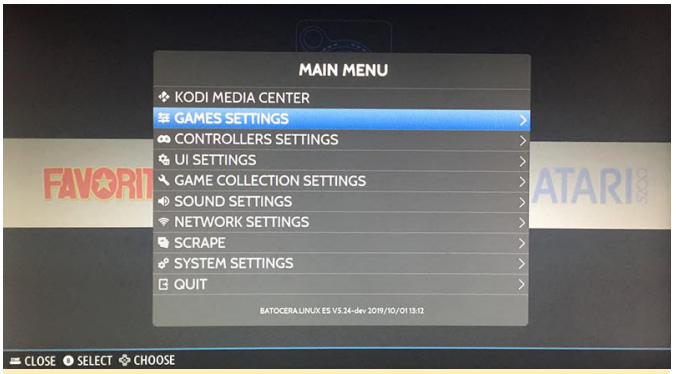
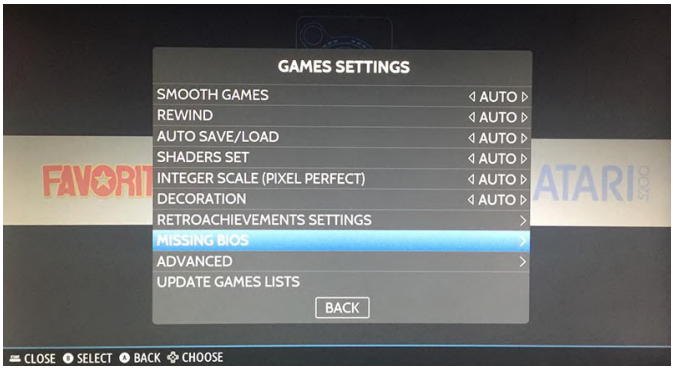


Figura 22

Desplácese hacia abajo en la opción del menú Configuración del juego y busca la opción del menú Missing Bios tal y como se muestra a continuación.



Aparecerá una ventana emergente con un desglose de los archivos BIOS que faltan para cada sistema. Puedes usarlo como referencia para los sistemas con los que tienes problemas.

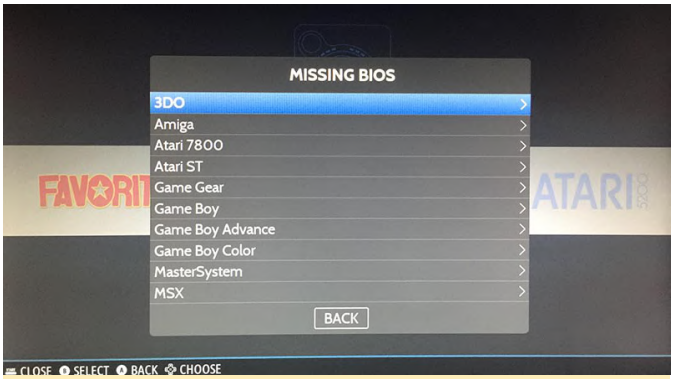


Figura 24

Hay otra cosa que quiero tratar en esta sección del tutorial, la configuración avanzada del sistema. Si vuelves al menú de Configuración del sistema. Navega hacia abajo hasta el menú Developer y selecciónalo, y te dirigirá a un menú donde puedes ajustar algunas configuraciones de nivel inferior. Recomiendo la siguiente configuración.

VRAM: 50MB Show Framerate: OFF VSYNC: OFF Preload UI: OFF Threaded Loading: OFF Async Image Loading: OFF Optimize Images VRAM Use: ON Optimize Video VRAM Use: ON

Las siguientes imagenes muestran el menú Developer y las opciones de configuración avanzada.

Figura 25

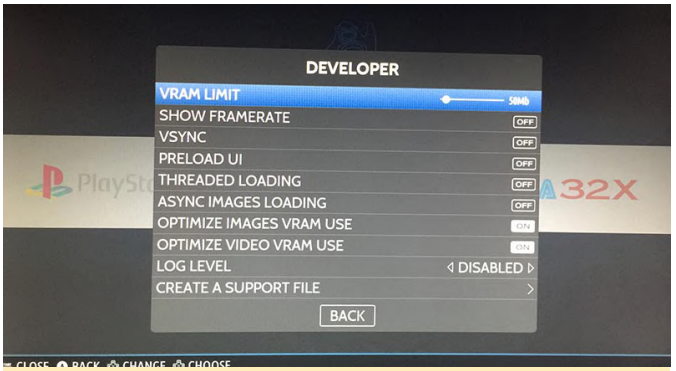


Figura 26

He descubierto que sin usar algunas de las configuraciones anteriores, el sistema puede bloquearse a veces durante el desplazamiento rápido. También opté por utilizar menos RAM para que los emuladores pudieran usar más RAM. La interfaz de usuario parece funcionar bien con 50MB, música de fondo e ilustraciones de las ROMs.

Obteniendo cajas de las rom y capturas de pantalla

El siguiente paso de nuestro proyecto es configurar todas las capturas de pantalla y cajas de las ROM. Dirigete a **https://www.skraper.net/** y descarga la última copia del software. También deberás hacerte con una cuenta de **https://www.screenscraper.fr/**. Por favor realiza una donación a ambos sitios, si te es posible. Son increíbles y realmente ayudan a que las consolas de juegos retro sean aún más increíbles al proporcionar acceso a cajas y capturas de pantalla para un sin fin de juegos. Una vez que tenga la configuración de tu cuenta, inicia el programa de interfaz de usuario de Skraper e introduce la información de tu cuenta screencraper.fr Prueba la cuenta para asegurarte de que funciona correctamente. Debería aparecer una pantalla similar a la que se muestra a continuación.

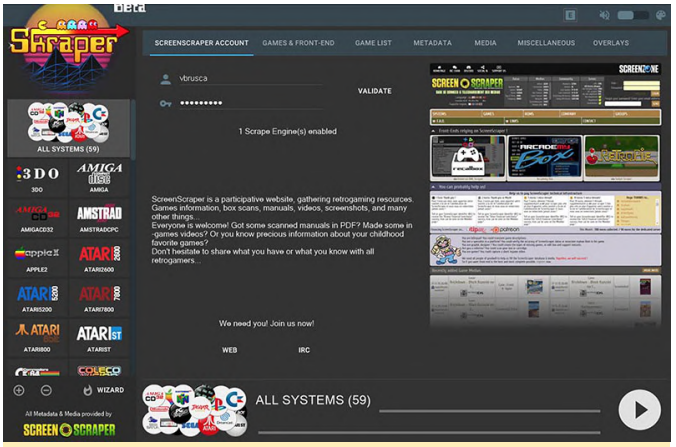


Figura 27

Haz clic en el botón del asistente en la parte inferior derecha de la IU del programa. Se te pedirá que entres en tu screencraper.fr y selecciona la carpeta de las ROMs de destino que quieres procesar. Debes tener tu tarjeta micro SD conectada a tu ordeandor y necesitar localizar y seleccionar la carpeta roms donde pegastes los archivos ROM. El software determinará automáticamente cada sistema que tenga ROMs y ejecutará una verificación con cada juego para ver si hay alguna caja disponible para ese juego. CONSEJO: no ejecutes el asistente con todos los sistemas. Las cosas podrían fallar y entonces no obtendrás el resulado deseado. Ejecuta en el software un solo sistema a mismo tiempo. A veces, si no obtienes resultados de las cajas de las ROMs, espera unas horas e intentarlo más tarde. Esta técnica me funcionó a mí y logre obtener buenos resultados para todos mis sistemas. El botón del asistente se muestra a continuación.

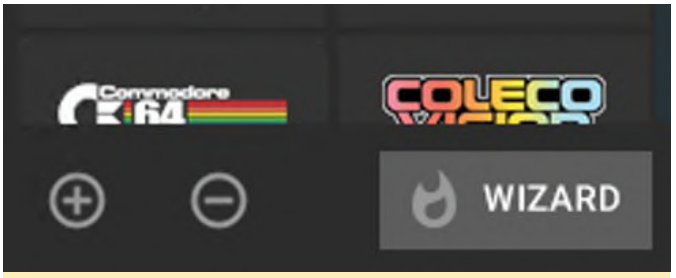


Figura 28

El tipo de caja que crea el software es realmente impresionante. Mira una muestra de los gráficos por defectro que genera.

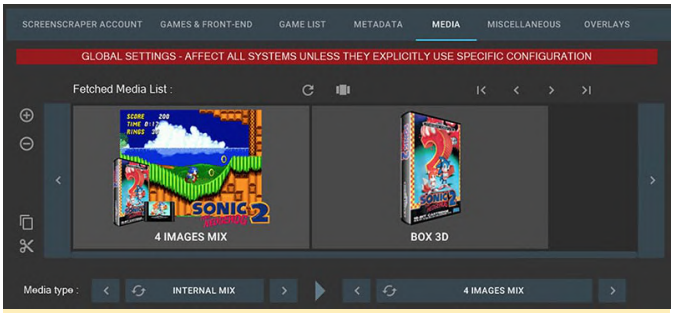


Figura 29

Una vez que haya recopilado todas las imágenes y capturas de pantalla de la ROMs, debes ajustar una opción de la Configuración de IU. Selecciona la opción del menú principal UI Settings y desplázate hacia abajo hasta la opción Parse Gamelists Only y actívala. Esto hará que la IU solo muestre los juegos en el archivo gamelist.xml. Si usas el software de respado de las cajas, tendrás buenos archivos XML y debería usar esta opción. Siempre puede configurar los juegos en los archivos XML a mano si fuera necesario.

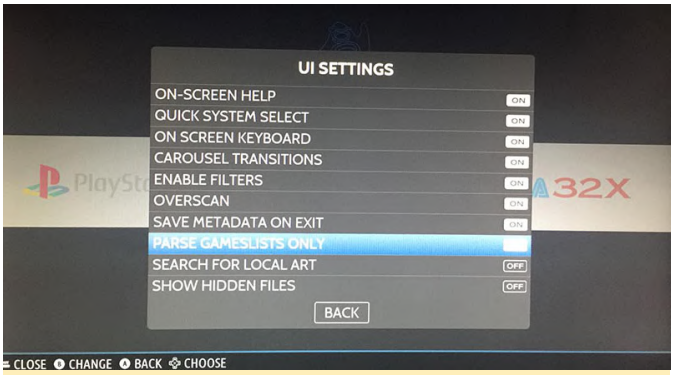


Figura 30

Esto concluye nuestro tutorial de compilación para la consola de juegos retro Monku R4/ODROID-N2. A continuación se muestra una captura de pantalla de cómo se ve la IU. Espero que hayas disfrutado y que te haya ayudado a montar tu increíble sistema de juegos retro. Para configuraciones avanzadas de emulador, dirigente a la siguiente sección del tutorial.

Figura 31

Configuración avanzada de emulador

Esta sección es un complemento y contiene información sobre cómo poner en marcha emuladores específicos y garantizar que sean estables. Sega CD: Problema: los juegos comienzan a cargarse y luego se bloquean después de unos segundos. Solución: Obtener el conjunto de archivos de BIOS de EE. UU. Y asegúrese de que tengan el mismo código hash que Batocera Linux requiere. Puedes verificar los problemas de BIOS utilizando las opciones de menú que hemos mencionado anteriormente para ver qué archivos espera el sistema. Atari 5200: Problema: los juegos se bloquean sin visualización o llegan a una pantalla de error del emulador que indica que hubo un problema con la carga. Solución: usa un solo juego para acceder al menú del emulador, donde puede cargar diferentes ROM y cambiar el tipo de cartucho. Para mí, el juego que mejor funcionó fue Asteroids. Aparecerá un error, pero en el menú del emulador puedes seleccionar el directorio para buscar ROM, este debe ser un directorio diferente al de Batocera Linux. Solo debes tener tu título de lanzamiento en el directorio de Batocera. También puede cambiar qué tipo de cartucho se debe usar para cargar una ROM usando las opciones del emulador. Elegir el estándar Atari generalmente soluciona cualquier problema de carga de ROM. Es un poco extraño, pero configurandolo de esta manera te asegurará que los juegos se cargan y tendrás un mejor control sobre cómo se cargan. Commodore 64: Problema: el emulador se ejecuta en un pequeño recuadrado en la parte inferior izquierda de la pantalla. Solución: cancela la carga del juego presionando el botón B. Debería poder acceder a las opciones del emulador. En la configuración de video/pantalla, elije pantalla completa. Luego regrese al menú principal y ve a la opción de administración de configuración para guardar tu configuración. Esto asegurará que el emulador se inicia en modo de pantalla completa. Sega Genesis: Problema: Tus juegos de Sega Genesis se bloquean y no son estables. Solución: carga la tarjeta SD en una máquina Windows o Mac. Dirigete a la carpeta batocera /system/batocera.conf. Desplázate hasta el final del archivo hasta que veas las entradas de megadrive. Agrega las siguientes líneas:

megadrive.core=picodrive megadrive.emulator-libretro Esto obligará al sistema a ejecutar las ROM utilizando un emulador diferente que no se puede seleccionar en los menús de la interfaz de usuario. Tus juegos serán más estables y no deberías tener más problemas. Nintendo 64: Problema: el emulador de Nintendo 64 se bloquea al salir y Batocera Linux no puede cargarse Solución: ajusta el emulador y el núcleo por defecto para las ROM de Nintendo 64 en emulador libretro y core parallel_n64. Esta combinación me ha funcionado muy bien. Puedo salir del emulador N64 y volver a Batocera Linux y elegir un nuevo sistema. Login SSH : Puede iniciar sesión en tu sistema a través de la red utilizando el inicio de sesión root por defecto, nombre de usuario: root, contraseña: linux.

Este artículo es una adaptación de middlemind.net, para obtener más información o para ver la fuente original, consulta: **http://middlemind.net/tutorials/odroid_go/mr4_build.html**

Fiesta de Desarrollo Kernel 5.4

January 10, 2020 By @memeka Desarrollo, Linux, ODROID-XU4

Empecemos con la fiesta de desarrollo del kernel 5.4. Hazte con mi rama 5.4 basada en 5.4.0 en **https://github.com/mihailescu2m/linux/tree/odroidxu4-5.4.y**.

Test tarjeta SD HC-1: * write

```
64+0 records in
64+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 22.0655
s, 24.3 MB/s
```

* read

```
64+0 records in
64+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 6.36601
s, 84.3 MB/s
```

TEst SSD HC-1: * write

```
64+0 records in
64+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 5.72404
s, 93.8 MB/s
```

* read

```
64+0 records in
64+0 records out
536870912 bytes (537 MB, 512 MiB) copied, 1.39248
s, 386 MB/s
```

1. Descargate la imagen oficial de Ubuntu desde cualquiera de los dos enlaces siguientes:

- Mate: **https://wiki.odroid.com/odroid-xu4/os_i ... 4/20190929**
- Minimal: **https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/20190910-minimal**

2. Graba la imagen usando Etcher en tu tarjeta SD/eMMC.

3. Inserta la tarjeta en tu Odroid XU4 para proceder a un arranque inicial. Se cambiará el tamaño de su sistema de archivos raíz para que se ajuste a la capacidad de tu memoria. A continuación, actualiza los paquetes:

```
$ apt update && apt full-upgrade -y
```

Si falla con un mensaje en el que aparece un problema de bloqueo, espera unos 5 o 10 minutos e inténtalo nuevamente.

4. Marca el paquete linux-kernel-5422 para que "no sea actualizado":

```
$ apt-mark hold linux-odroid-5422.
```

5. Reinicia, luego apaga y conecta tu tarjeta SD/eMMC a tu PC.

Compilando el kernel 5.4

1. Configura el entorno de compilación partiendo de esta guía: **https://wiki.odroid.com/odroid-xu4/soft ... ross-build**, luego descárgate la cadena de herramientas adecuada. Exporta las variables de entorno utilizando la cadena de herramientas..

2. Monta las particiones de arranque, rootfs de la tarjeta SD/eMMC en tu PC. Debería montarse automáticamente en Ubuntu.

3. Clona el núcleo 5.4 de memeka:

```
$ git clone
https://github.com/mihailescu2m/linux.git --depth
1 -b odroidxu4-5.4.y linux-kernel-odroidxu4-5.4.y.
```

4. Muévete al directorio clonado:

```
$ cd linux-kernel-odroidxu4-5.4.y.
```

5. Compila el Kernel 5.4

```
$ make odroidxu4_defconfig
$ make -j $(nproc)
```

6. Copia la imagen del kernel y el blob del árbol de dispositivos en la partición de arranque de la tarjeta. Reemplaza la ruta de destino por la tuya:

```
$ sudo cp -f arch/arm/boot/zImage
/media/joshua/boot
$ sudo cp -f arch/arm/boot/dts/exynos5422-
odroidxu4.dtb /media/joshua/boot
```

7. Instala los modulos:

```
$ sudo make -j $(nproc) modules_install ARCH=arm
INSTALL_MOD_PATH=/media/joshua/rootfs
$ sync
```

8. Desmonta tu tarjeta de arranque e insértala en el ODROID-XU4:

```
# uname -a
Linux odroid 5.4.0+ #2 SMP Wed Nov 27 08:17:23 UTC
2019 armv7l armv7l armv7l GNU/Linux
# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.3 LTS
Release:        18.04
Codename:       bionic
```

Para comentarios, preguntas y sugerencias, visita el artículo original en **https://forum.odroid.com/viewtopic.php?f=184&t=36947.**

El Punto G: Tu Destino para Todas las Cuestiones Relacionadas con Juegos Android: ¡Google Dejo Caer la Pelota; Giphy es la Pelota; y ODROID-N2 lo Gana TODO!

January 1, 2020 By Dave Prochnow Android, Juegos

Bueno, fue especial, ¿no? Si eres uno de los miles de suscriptores de Google Stadia Founder's Edition, entonces sabrás exactamente cómo Google dejó caer la pelota el día del lanzamiento. Este Tweet oficial de Google Stadia resume casi todo el enredo:

"Aquí tienes la última actualización: si compraste y pagaste la Founder's Edition, deberías tener tu código de acceso Stadia. Los pedidos anticipados y los códigos de acceso para Premiere Edition empezarán a enviarse a principios de la próxima semana. ¡Gracias por seguir con nosotros!"

Figura 1: Un logotipo de juego con un futuro brillante. Imagen cortesía de Google Stadia.

A priori, suena bastante inofensivo, pero este Tweet fue emitido tres días después del día del lanzamiento. Como probablemente sabrás, o habrás adivinado al leer el título de este artículo, un enorme grupo de usuarios de Stadia se toparon con la bolsa de servicio de juegos por streaming vacía sin juegos. Seguro que finalmente todo se solucionó, pero este no fue el único contratiempo que afectó al intento de Google

de "ofrecer juegos AAA en cualquier dispositivo, en cualquier momento y en cualquier lugar".

En otro Tweet (más tarde eliminado) del Google Stadia oficial se afirmaba que el próximo thriller AAA, Red Dead Redemption 2, se ejecutaría en calidad 4K. Del mismo modo, el jefe de Google Stadia, Phil Harrison ha declarado en comentarios anteriores, que TODOS los juegos se ejecutarán en calidad 4K a 60 fps. El único inconveniente con el que no encontramos con esto es que al menos dos desarrolladores de juegos (Destiny 2 y Red Dead Redemption 2) han declarado clara y explícitamente que sus títulos NO funcionan con estas especificaciones. De hecho, estos dos títulos están siendo actualizados para llegar al 4K a 60 fps.

Aunque ésta no es una evidencia para posibles conspiradores, sí indica claramente que este tipo de tecnología está en sus inicios y, como tal, aparecerán algunas dificultades en el camino hasta convertirse en un servicio de juegos por streaming totalmente viable que realmente funcione en cualquier dispositivo, en cualquier momento y en cualquier lugar.

Juegos Giphy

Si estás buscando una alternativa a los juegos 4K a 60 fps, entonces, ¿qué tal un juego retro tipo arcade? Patrocinado por el sitio web del motor de búsqueda de GIF on line, Giphy, un nuevo servicio de juego que presenta versiones cortas y de tamaño micro de algunos clásicos arcade. Por ejemplo, una versión Giphy de Asteroids (llamada "Blast 20 Asteroids" dentro de la lista de reproducción destacada "Gimme Space") acompañada de música y efectos de sonido se puede reproducir dentro de la mayoría de los navegadores en cualquier dispositivo conectado a la web.

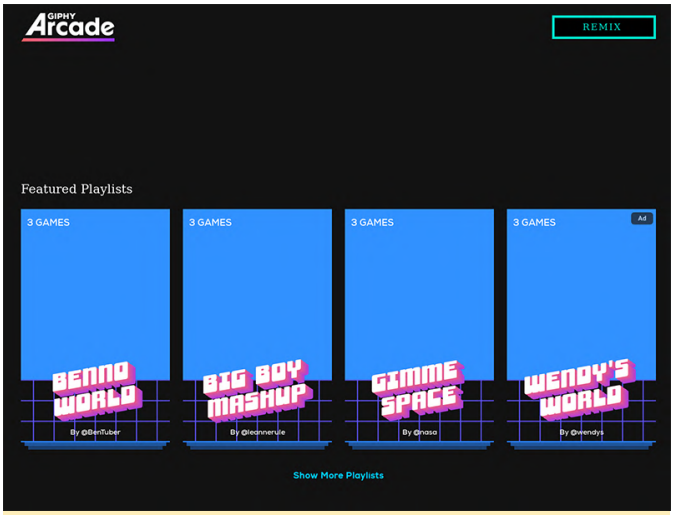


Figura 2: juegos de arcade GIF.

Llamado Giphy Arcade, el juego en este nuevo servicio es ridículamente corto y los gráficos están enormemente influenciados por GIF y emojis. Por ejemplo, volviendo al juego "Blast 20 Asteroids" mencionado con anterioridad, el cañón móvil de disparo láser que se encuentra en Asteroids es reemplazado por un GIF astronauta que camina por el espacio con un puño láser.

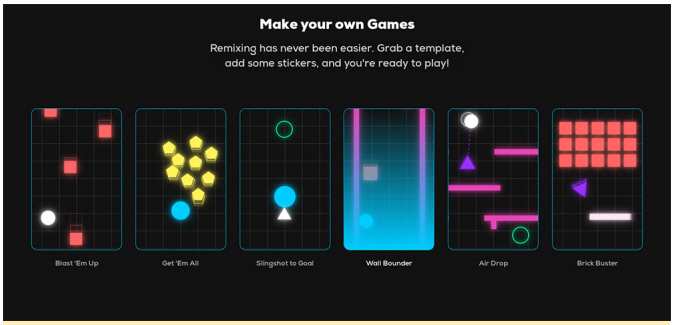


Figura 3: Desarrolla tu propio juego de arcade con la ayuda de algunas plantillas.

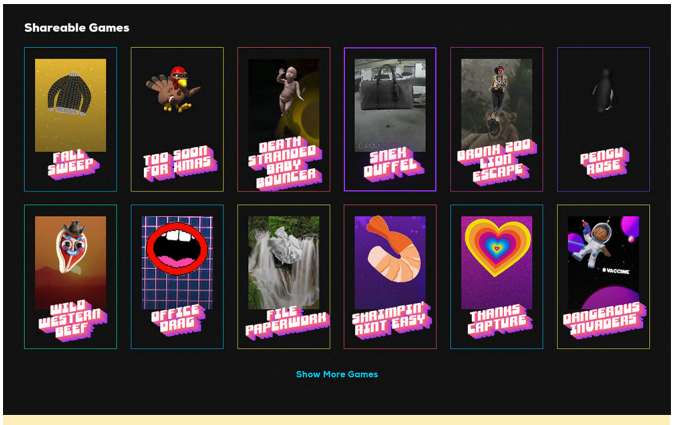


Figura 4: cuando hayas terminado tu obra maestra, compártela con el mundo.

Una vez que te canses de jugar a estos juegos con listas de reproducción, puedes probar suerte y crear tu propio juego. Algo así como una combinación de

GIF, música, emojis y plantillas, estas obras maestras hechas a mano se pueden jugar y compartir con otros visitantes de Giphy. Actualmente, las opciones para la creación de juegos son un poco toscas y complicadas, pero puedes optar por el nivel más básico de los juegos de estilo retro y arcade y convertirte en una "estrella de rock" de los juegos de Giphy Arcade.

Un honor al juego ODROID

Cualquiera que siga los juegos retro, los emuladores de juegos y los dispositivos portátiles de juegos le sonará el nombre de ETA Prime. Con una importante presencia de juegos en YouTube y con más de 350 mil suscriptores, ETA Prime también es uno de los editores de videos más prolíficos en YouTube con múltiples cargas cada semana.

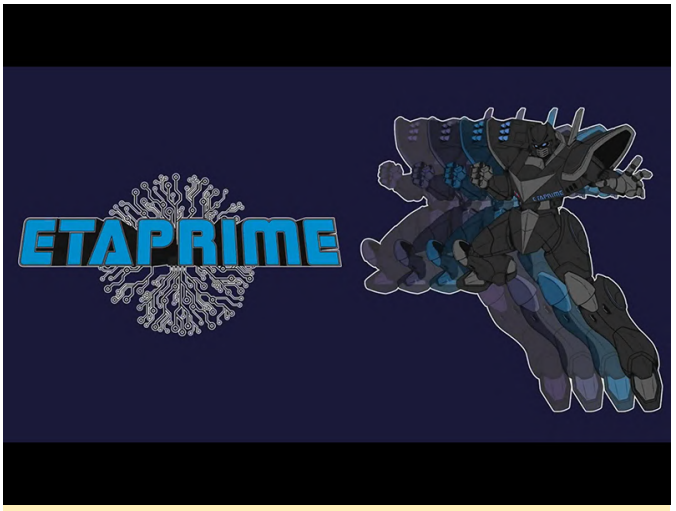


Figura 5: ETA Prime es una fuente de video para hacer comentarios sobre SBC.

Basta decir que cuando ETA Prime publica un video, MUCHAS personas prestan atención. Este fue el caso a mediados de noviembre cuando apareció en su canal un video titulado "Cuál es el mejor ordenador de placa reducida para Android".

Aunque el título del video mencionaba "... para Android", se trata de ETA Prime-speak para "Android Gaming" generalizado o más específicamente, consumo multimedia Android: un mercado único de video, películas y juegos que funciona bajo la apariencia de Android.

Durante el video, ETA Prime compara 12 ordenadores de placa reducida (SBC) diferentes. Cada uno de estos modelos son modernos y potentes SBC muy populares entre aficionados, diseñadores de sistemas e investigadores a nivel industrial de hoy en día. Ver el video de 7 minutos y 33 segundos te permitirá conocer de primera mano el estado actual de los SBC. Hay un total de 12 SBC que ETA Prime destaca y cada uno es de primera categoría con sus respectivas especificaciones y rendimiento.

Figura 6 - Los 12 SBC destacados; ¿Puedes identificarlos todos?

La emoción aumenta a medida que el video se acerca a su punto medio, donde aproximadamente a los 3 minutos y 38 segundos, se anuncia "la mejor placa

para ejecutar Android de 2019". Y ese SBC es el ODROID-N2. Con su típico y metódico formato, ETA Prime dedica el resto del video a respaldar esta afirmación con pruebas de rendimiento, lanzamientos de sistemas operativos de terceros y pruebas de juegos.

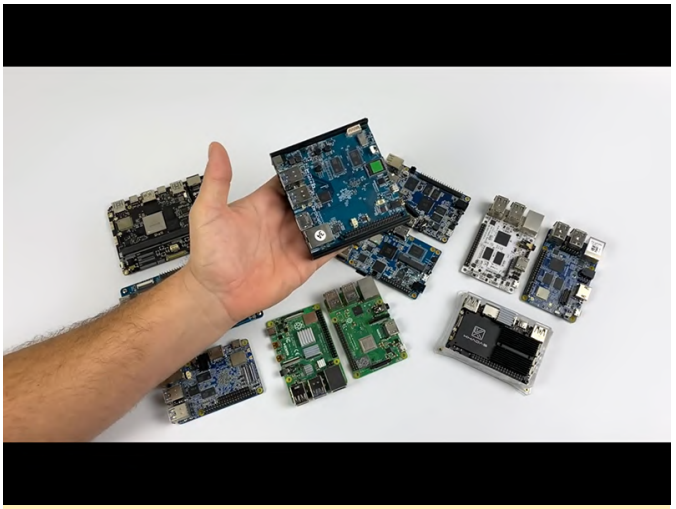


Figura 7 - Y el ganador es ... ODROID-N2.

Por lo tanto, si estás buscando un SBC Android, no busques más, el mejor es el ODROID-N2.

Cluster Kubernetes Sobre un ODROID-N2

January 1, 2020 By Swaminathan Bhaskar Desarrollo, ODROID-N2, Tutoriales

Introducción

Kubernetes (o k8s para abreviar) es una plataforma extensible de organización de contenedores de código abierto diseñada para administrar cargas de trabajo en contenedores y servicios a escala. Ayuda con la implementación automatizada, el escalado y la administración de cargas de trabajo de aplicaciones centradas en contenedores en un clúster de nodos (básicos, virtuales o en la nube) mediante la organización de la infraestructura del cálculo, la red y almacenamiento a partir de de esas cargas de trabajo de los usuarios.

Los dos tipos principales de nodos en un clúster de Kubernetes son:

Maestro: este nodo actúa a modo de control para todo el clúster. Es responsable de todas las implementaciones de carga de trabajo de las aplicaciones, planificación y decisiones, así como también de detectar y administrar cambios en el estado de las aplicaciones implementadas. Se compone de un almacén de valores clave, un servidor API, un planificador y un administrador de controladores.

Nodo(s) de trabajo: nodo(s) que realmente ejecutan los contenedores de aplicaciones. También se les conoce en ocasiones como Minion (s). El maestro también es un nodo, pero no está destinado a la implementación de aplicaciones. Se compone de un agente llamado kubelet, un proxy de red llamado kube-proxy y un motor de contenedores.

La siguiente Figura ilustra la descripción de la arquitectura de alto nivel de Kubernetes:

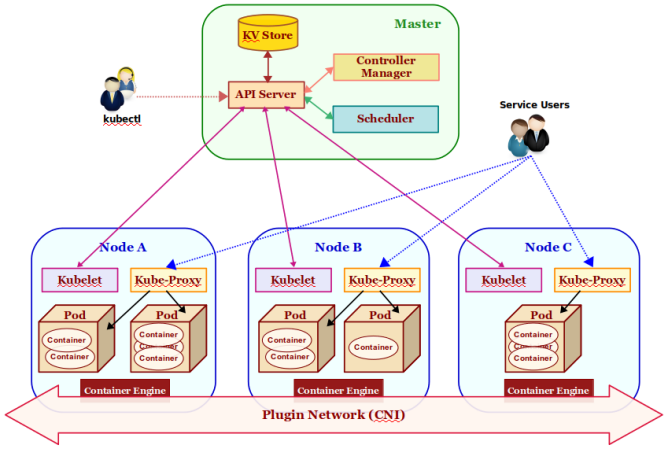


Figura 1

Los componentes básicos que forman un clúster de Kubernetes se describen a continuación:

KV Store: un almacén de datos valor-clave altamente confiable, distribuido y consistente que se utiliza para persistir y mantener información del estado sobre los diversos componentes del clúster de Kubernetes. Por defecto, Kubernetes usa etcd como el almacén de valores clave.

Servidor API: actúa como punto de entrada para el plano de control presentando un punto final API para todas las interacciones con y dentro del clúster de Kubernetes. A través del Servidor API se realizan solicitudes para la implementación, administración, gestión y operación de aplicaciones basadas en contenedores. Utiliza el almacén de valores clave para mantener la información de estado sobre todos los componentes del clúster de Kubernetes.

Pod(s): es la unidad de implementación más pequeña en Kubernetes. Uno o más contenedores se ejecutan dentro. Es algo así como un host lógico con red y almacenamiento compartidos. Los pods de aplicaciones están programados para ejecutarse en diferentes nodos de trabajo del clúster de Kubernetes en función de las necesidades de los recursos y las restricciones de la aplicación. Cada pod dentro del clúster tiene su propia dirección IP única. Los contenedores de aplicaciones dentro de un pod se comunican entre sí mediante localhost. Los Pods también son la unidad más pequeña de escalado en Kubernetes. Los Pod (s) son efímeros: pueden ir y venir en cualquier momento.

Scheduler: responsable de programar los pods de la aplicación para que se ejecuten en los nodos de trabajo seleccionados del clúster de Kubernetes partiendo de los requisitos de los recursos de la aplicación, así como las restricciones de afinidad específicas de la aplicación.

Service: proporciona un punto de red estable y lógico para un grupo de pod (s) (basado en una etiqueta relacionada con un pod de aplicación que se ejecuta en los nodos wor2ker del clúster de Kubernetes. Permiten el acceso a una aplicación a través del service-discovery y distribuyen las solicitudes mediante un simple equilibrio de carga. Para acceder a una aplicación, a cada servicio se le asigna una dirección IP interna:puerto en todo el clúster.

Controller Manager: gestiona diferentes tipos de controladores responsables de supervisar y detectar cambios en el estado del clúster de Kubernetes (a través del servidor API) y garantiza que el clúster cambie al estado deseado. Los diferentes tipos de controladores son:

- Node Controller => responsable de supervisar y detectar el estado y la solidez (arriba o abajo) de los nodos de trabajo en el clúster de Kubernetes.
- ReplicaSet => anteriormente denominado Controlador de replicación y es responsable de mantener el número deseado de réplicas de pod en el clúster.
- Endpoints Controller => responsable de detectar y gestionar cambios en los puntos de acceso al servicio de la aplicación (lista de direcciones IP:puerto).

Plugin Network: actúa como puente (red superpuesta) que permite la comunicación entre los pods que se ejecutan en diferentes nodos de trabajo del clúster. Hay diferentes implementaciones de este componente por parte de terceros como calico, franela, weave-net, etc. Todos ellos deben cumplir con una especificación común llamada Container Network Interface o CNI para abreviar.

kubelet: un agente que se ejecuta en cada nodo de trabajo del clúster de Kubernetes. Es responsable de crear e iniciar un pod de aplicación en el nodo de trabajo y asegurar de que todos los contenedores de aplicaciones estén en funcionamiento dentro del pod. Además, también es responsable de informar del

estado y solidez del nodo de trabajo, así como de todos los pods que están en ejecución al maestro a través del servidor API.

kube-proxy: un proxy de red que se ejecuta en cada uno de los nodos de trabajo del clúster de Kubernetes y actúa como un punto de entrada para acceder a los diversos puntos del servicio de aplicaciones. Dirige las solicitudes hacía los pods apropiados dentro del clúster.

Container Engine: un tiempo de ejecución de contenedor que se ejecuta en cada uno de los nodos de trabajo para administrar el ciclo de vida de los contenedores, como obtener las imágenes, iniciar y detener contenedores, etc. El motor de contenedores comúnmente utilizado es Docker.

kubectl: herramienta de línea de comandos utilizada para interactuar con el servidor API. Utilizada por los administradores (u operadores) para la implementación y el escalado de aplicaciones, así como para la gestión del clúster de Kubernetes.

Instalación y configuración del sistema

La instalación la realizaremos en un clúster ODROID-N2 de 5 nodos con Armbian Ubuntu Linux.

La siguiente Figura muestra un clúster ODROID-N2 con 5 nodos en funcionamiento:

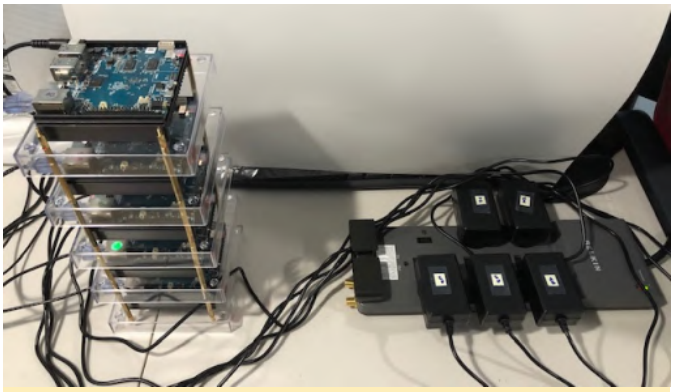


Figura 2

Para este tutorial, vamos a suponer que los 5 nodos en el clúster tienen los siguientes nombres de host y direcciones IP:

```
my-n2-1 192.168.1.51
my-n2-2 192.168.1.52
my-n2-3 192.168.1.53
my-n2-4 192.168.1.54
my-n2-5 192.168.1.55
```

Abra una ventana de Terminal y abre una pestaña para cada uno de los 5 nodos my-n2-1 a través de my-n2-5. En cada una de las pestañas de Terminal, conectate por ssh al nodo correspondiente.

Cada uno de los nodos my-n2-1 hasta my-n2-5 debe tener un identificador único para que el clúster funcione sin conflictos. El identificador de nodo único se encuentra en el archivo /etc/machine-id, vemos que todos los nodos my-n2-1 hasta my-n2-5 tienen el mismo valor. Esto hay que solucionarlo*. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos:

```
$ sudo rm -f /etc/machine-id
$ sudo dbus-uuidgen --ensure=/etc/machine-id
$ sudo rm /var/lib/dbus/machine-id
$ sudo dbus-uuidgen --ensure
$ sudo reboot now
```

Una vez más, en cada una de las pestañas de Terminal, conéctate por ssh al nodo correspondiente.

A continuación, necesitamos configurar el repositorio de paquetes para Docker. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-
certificates curl
software-properties-common -y
$ curl -fsSL
https://download.docker.com/linux/ubuntu/gpg |
sudo
apt-key add -
$ sudo apt-get update
$ sudo add-apt-repository "deb [arch=arm64]
-https://download.docker.com/linux/ubuntu xenial
stable"
$ sudo apt-get update
```

Para la versión 1.16 de Kubernetes (la versión en el momento de publicar este artículo), la versión recomendada de Docker es la 18.09.

ATENCIÓN: Para Docker CE 19.xx (y superior) Asegúrate de que la versión de Docker instalada sea * 18.09 *. De lo contrario, te encontrarás con el siguiente error: [ERROR SystemVerification]: unsupported docker version: 19.xx

Necesitamos verificar el último paquete de Docker 18.09 en el repositorio. En cualquiera de los nodos (elegiremos my-n2-1), ejecuta el siguiente comando:

```
$ apt-cache madison docker-ce
```

El siguiente sería un resultado típico:

```
<span style="font-weight: 400;">Output.1</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.5~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.4~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.3~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.2~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.1~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:19.03.0~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.9~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.8~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.7~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.6~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.5~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.4~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.3~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.2~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.1~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
5:18.09.0~3-0~ubuntu-xenial |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
18.06.3~ce~3-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
18.06.2~ce~3-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
18.06.1~ce~3-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
```

```
18.06.0~ce~3-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
18.03.1~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
18.03.0~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
17.12.1~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
17.12.0~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
17.09.1~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>

<span style="font-weight: 400;">docker-ce |
17.09.0~ce-0~ubuntu |
https://download.docker.com/linux/ubuntu
xenial/stable arm64 Packages</span>
```

En el resultado anterior, vemos que el último paquete para Docker 18.09 es 5: 18.09.9 ~ 3-0 ~ ubuntu-xenial.

A continuación, necesitamos instalar la versión elegida de Docker. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ sudo apt-get install docker-ce=5:18.09.9~3-
0~ubuntu-xenial -y
```

El siguiente sería un resultado típico:

```
Output.2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be
installed:
aufs-tools cgroupfs-mount containerd.io docker-ce-
cli git git-man liberror-perl pigz
Suggested packages:
git-daemon-run | git-daemon-sysvinit git-doc git-
el git-email git-gui gitk gitweb git-cvs git-
mediawiki git-svn
The following NEW packages will be installed:
aufs-tools cgroupfs-mount containerd.io docker-ce
docker-ce-cli git git-man liberror-perl pigz
0 upgraded, 9 newly installed, 0 to remove and 0
not upgraded.
Need to get 61.3 MB of archives.
After this operation, 325 MB of additional disk
space will be used.
Get:1 https://download.docker.com/linux/ubuntu
xenial/stable arm64 containerd.io arm64 1.2.10-3
[14.5 MB]
Get:2 http://ports.ubuntu.com/ubuntu-ports
bionic/universe arm64 pigz arm64 2.4-1 [47.8 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports
bionic/universe arm64 aufs-tools arm64
1:4.9+20170918-1ubuntu1 [101 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports
bionic/universe arm64 cgroupfs-mount all 1.4 [6320
B]
Get:5 http://ports.ubuntu.com/ubuntu-ports
bionic/main arm64 liberror-perl all 0.17025-1
[22.8 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports bionic-
updates/main arm64 git-man all 1:2.17.1-1ubuntu0.4
[803 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports bionic-
updates/main arm64 git arm64 1:2.17.1-1ubuntu0.4
[2941 kB]
Get:8 https://download.docker.com/linux/ubuntu
xenial/stable arm64 docker-ce-cli arm64
5:19.03.5~3-0~ubuntu-xenial [29.6 MB]
Get:9 https://download.docker.com/linux/ubuntu
xenial/stable arm64 docker-ce arm64 5:18.09.9~3-
0~ubuntu-xenial [13.3 MB]
Fetched 61.3 MB in 5s (11.6 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 156190 files and directories
currently installed.)
Preparing to unpack .../0-pigz_2.4-1_arm64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package aufs-
tools.
Preparing to unpack .../1-aufs-
tools_1%3a4.9+20170918-1ubuntu1_arm64.deb ...
Unpacking aufs-tools (1:4.9+20170918-1ubuntu1) ...
Selecting previously unselected package cgroupfs-
mount.
Preparing to unpack .../2-cgroupfs-
```

```
mount_1.4_all.deb ...
Unpacking cgroupfs-mount (1.4) ...
Selecting previously unselected package
containerd.io.
Preparing to unpack .../3-containerd.io_1.2.10-
3_arm64.deb ...
Unpacking containerd.io (1.2.10-3) ...
Selecting previously unselected package docker-ce-
cli.
Preparing to unpack .../4-docker-ce-
cli_5%3a19.03.5~3-0~ubuntu-xenial_arm64.deb ...
Unpacking docker-ce-cli (5:19.03.5~3-0~ubuntu-
xenial) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../5-docker-ce_5%3a18.09.9~3-
0~ubuntu-xenial_arm64.deb ...
Unpacking docker-ce (5:18.09.9~3-0~ubuntu-xenial)
...
Selecting previously unselected package liberror-
perl.
Preparing to unpack .../6-liberror-perl_0.17025-
1_all.deb ...
Unpacking liberror-perl (0.17025-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../7-git-man_1%3a2.17.1-
1ubuntu0.4_all.deb ...
Unpacking git-man (1:2.17.1-1ubuntu0.4) ...
Selecting previously unselected package git.
Preparing to unpack .../8-git_1%3a2.17.1-
1ubuntu0.4_arm64.deb ...
Unpacking git (1:2.17.1-1ubuntu0.4) ...
Setting up aufs-tools (1:4.9+20170918-1ubuntu1)
...
Setting up git-man (1:2.17.1-1ubuntu0.4) ...
Setting up containerd.io (1.2.10-3) ...
Created symlink /etc/systemd/system/multi-
user.target.wants/containerd.service â†’
/lib/systemd/system/containerd.service.
Setting up liberror-perl (0.17025-1) ...
Setting up cgroupfs-mount (1.4) ...
Setting up docker-ce-cli (5:19.03.5~3-0~ubuntu-
xenial) ...
Setting up pigz (2.4-1) ...
Setting up git (1:2.17.1-1ubuntu0.4) ...
Setting up docker-ce (5:18.09.9~3-0~ubuntu-xenial)
...
update-alternatives: using /usr/bin/dockerd-ce to
provide /usr/bin/dockerd (dockerd) in auto mode
Created symlink /etc/systemd/system/multi-
user.target.wants/docker.service â†’
/lib/systemd/system/docker.service.
Created symlink
/etc/systemd/system/sockets.target.wants/docker.so
cket â†’ /lib/systemd/system/docker.socket.
Processing triggers for systemd (237-3ubuntu10.33)
...
Processing triggers for man-db (2.8.3-2ubuntu0.1)
...
Processing triggers for libc-bin (2.27-3ubuntu1)
...
```

A continuación, debemos asegurarnos de que podemos ejecutar los comandos de Docker con el usuario conectado sin la necesidad de sudo. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos:

```
$ sudo usermod -aG docker $USER
$ sudo reboot now
```

Una vez más, en cada una de las pestañas de Terminal, conéctate por ssh al nodo correspondiente.

Para verificar la instalación de Docker, en cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ docker info
```

El siguiente sería un resultado típico:

```
Output.3
Client:
Debug Mode: false

Server:
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 18.09.9
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Log: awslogs fluentd gcplogs gelf journald json-
file local logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
```

```
Init Binary: docker-init
containerd version:
b34a5c8af56e510852c35414db4c1f4fa6172339
runc version:
3e425f80a8c931f88e6d94a8c831b9d5aa481657
init version: fec3683
Security Options:
seccomp
Profile: default
Kernel Version: 4.9.196-meson64
Operating System: Ubuntu 18.04.3 LTS
OSType: linux
Architecture: aarch64
CPUs: 6
Total Memory: 3.623GiB
Name: my-n2-1
ID:
QF32:QDZN:IQDM:34HX:NK3C:O3AP:Y6JZ:74DV:XXXL:KCBL:
7K5D:36B4
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
127.0.0.0/8
Live Restore Enabled: false
Product License: Community Engine
```

Luego, necesitamos configurar el repositorio de paquetes para Kubernetes. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos

```
$ curl -s
https://packages.cloud.google.com/apt/doc/apt-
key.gpg | sudo apt-key add -
$ echo "deb http://apt.kubernetes.io/ kubernetes-
xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
$ sudo apt-get update
```

A continuación, necesitamos instalar Kubernetes. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ sudo apt-get install -y kubeadm
```

El siguiente sería un resultado típico:

```
Output.4
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be
installed:
conntrack cri-tools ebtables kubectl kubelet
kubernetes-cni socat
The following NEW packages will be installed:
conntrack cri-tools ebtables kubeadm kubectl
kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 1
not upgraded.
Need to get 48.3 MB of archives.
After this operation, 280 MB of additional disk
space will be used.
Get:2 http://ports.ubuntu.com/ubuntu-ports
bionic/main arm64 conntrack arm64
1:1.4.4+snapshot20161117-6ubuntu2 [27.3 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports bionic-
updates/main arm64 ebtables arm64 2.0.10.4-
3.5ubuntu2.18.04.3 [74.2 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports
bionic/main arm64 socat arm64 1.7.3.2-2ubuntu2
[322 kB]
Get:1 https://packages.cloud.google.com/apt
kubernetes-xenial/main arm64 cri-tools arm64
1.13.0-00 [7965 kB]
Get:3 https://packages.cloud.google.com/apt
kubernetes-xenial/main arm64 kubernetes-cni arm64
0.7.5-00 [5808 kB]
Get:4 https://packages.cloud.google.com/apt
kubernetes-xenial/main arm64 kubelet arm64 1.16.3-
00 [18.5 MB]
Get:5 https://packages.cloud.google.com/apt
kubernetes-xenial/main arm64 kubectl arm64 1.16.3-
00 [8025 kB]
Get:6 https://packages.cloud.google.com/apt
kubernetes-xenial/main arm64 kubeadm arm64 1.16.3-
00 [7652 kB]
Fetched 48.3 MB in 5s (9383 kB/s)
Selecting previously unselected package conntrack.
(Reading database ... 157399 files and directories
currently installed.)
Preparing to unpack .../0-
conntrack_1%3a1.4.4+snapshot20161117-
6ubuntu2_arm64.deb ...
Unpacking conntrack (1:1.4.4+snapshot20161117-
6ubuntu2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.13.0-
00_arm64.deb ...
Unpacking cri-tools (1.13.0-00) ...
Selecting previously unselected package ebtables.
Preparing to unpack .../2-ebtables_2.0.10.4-
3.5ubuntu2.18.04.3_arm64.deb ...
```

```
Unpacking ebtables (2.0.10.4-3.5ubuntu2.18.04.3)
...
Selecting previously unselected package
kubernetes-cni.
Preparing to unpack .../3-kubernetes-cni_0.7.5-
00_arm64.deb ...
Unpacking kubernetes-cni (0.7.5-00) ...
Selecting previously unselected package socat.
Preparing to unpack .../4-socat_1.7.3.2-
2ubuntu2_arm64.deb ...
Unpacking socat (1.7.3.2-2ubuntu2) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.16.3-
00_arm64.deb ...
Unpacking kubelet (1.16.3-00) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../6-kubectl_1.16.3-
00_arm64.deb ...
Unpacking kubectl (1.16.3-00) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../7-kubeadm_1.16.3-
00_arm64.deb ...
Unpacking kubeadm (1.16.3-00) ...
Setting up conntrack (1:1.4.4+snapshot20161117-
6ubuntu2) ...
Setting up kubernetes-cni (0.7.5-00) ...
Setting up cri-tools (1.13.0-00) ...
Setting up socat (1.7.3.2-2ubuntu2) ...
Setting up ebtables (2.0.10.4-3.5ubuntu2.18.04.3)
...
Created symlink /etc/systemd/system/multi-
user.target.wants/ebtables.service â†’
/lib/systemd/system/ebtables.service.
update-rc.d: warning: start and stop actions are
no longer supported; falling back to defaults
Setting up kubectl (1.16.3-00) ...
Setting up kubelet (1.16.3-00) ...
Created symlink /etc/systemd/system/multi-
user.target.wants/kubelet.service â†’
/lib/systemd/system/kubelet.service.
Setting up kubeadm (1.16.3-00) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1)
...
Processing triggers for systemd (237-3ubuntu10.33)
...
```

Necesitamos reiniciar todos los nodos. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ sudo reboot now
```

Una vez más, en cada una de las pestañas de Terminal, conéctate por ssh al nodo correspondiente.

Para verificar la instalación de Kubernetes, en cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ kubeadm version
```

El siguiente sería un resultado típico:

```
Output.5
kubeadm version: &version.Info{Major:"1",
Minor:"16", GitVersion:"v1.16.3",
GitCommit:"b3cbbae08ec52a7fc73d334838e18d17e851274
9", GitTreeState:"clean", BuildDate:"2019-11-
13T11:20:25Z", GoVersion:"go1.12.12",
Compiler:"gc", Platform:"linux/arm64"}
```

A continuación, debemos asegurarnos de que los paquetes para Docker y Kubernetes no se actualicen en el futuro mediante el proceso de actualización de software. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ sudo apt-mark hold kubelet kubeadm kubectl
docker-ce
```

El siguiente sería un resultado típico:

```
Output.6
kubelet set on hold.
kubeadm set on hold.
kubectl set on hold.
docker-ce set on hold.
```

Por defecto, Docker usa cgroupfs como controlador cgroup. Kubernetes prefiere systemd como controlador cgroup. Necesitamos modificar la configuración del demonio Docker especificando las opciones en un archivo JSON llamado /etc/docker/daemon.json. En cada uno de los nodos my-n2-1 a través de my-n2-5, crea el archivo de configuración /etc/docker/daemon.json con el siguiente contenido:

```
/etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
```

```
"storage-driver": "overlay2"
}
```

A continuación, debemos reiniciar el demonio Docker para que la configuración surta efecto. En cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos:

```
$ sudo mkdir -p
/etc/systemd/system/docker.service.d
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

Nota: No usar el controlador systemd cgroup provocará el siguiente error: [preflight] Running pre-flight checks [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/

Para verificar que el demonio Docker se inició bien, en cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta el siguiente comando:

```
$ journalctl -u docker
```

El siguiente sería un resultado típico:

```
Output.7
-- Logs begin at Sat 2019-12-14 21:14:19 EST, end
at Sat 2019-12-14 21:49:26 EST. --
Dec 14 21:14:26 my-n2-1 systemd[1]: Starting
Docker Application Container Engine...
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.806496732-05:00" level=info
msg="systemd-resolved is running, so using
resolvconf: /run/systemd/res
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.821800611-05:00" level=info
msg="parsed scheme: "unix"" module=grpc
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.822661404-05:00" level=info
msg="scheme "unix" not registered, fallback to
default scheme" module
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.824226106-05:00" level=info
msg="parsed scheme: "unix"" module=grpc
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.824838344-05:00" level=info
msg="scheme "unix" not registered, fallback to
default scheme" module
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.828116839-05:00" level=info
msg="ccResolverWrapper: sending new addresses to
cc: [{unix:///run/cont
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.828945714-05:00" level=info
msg="ClientConn switching balancer to
"pick_first"" module=grpc
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.828101672-05:00" level=info
msg="ccResolverWrapper: sending new addresses to
cc: [{unix:///run/cont
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.830093104-05:00" level=info
msg="ClientConn switching balancer to
"pick_first"" module=grpc
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.832076285-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x400014e610, CONNECT
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.844251802-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40001343a0, CONNECT
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.846949059-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40001343a0, READY"
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.851896887-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x400014e610, READY"
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.857097768-05:00" level=info msg="
[graphdriver] using prior storage driver:
overlay2"
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.886090322-05:00" level=info
msg="Graph migration to content-addressability
took 0.00 seconds"
Dec 14 21:14:27 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:27.893602818-05:00" level=info
msg="Loading containers: start."
Dec 14 21:14:28 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:28.821256841-05:00" level=info
msg="Default bridge (docker0) is assigned with an
IP address 172.17.0.0
Dec 14 21:14:29 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:29.134364234-05:00" level=info
msg="Loading containers: done."
Dec 14 21:14:29 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:29.374311397-05:00" level=info
msg="Docker daemon" commit=039a7df
graphdriver(s)=overlay2 version=18.0
Dec 14 21:14:29 my-n2-1 dockerd[3347]: time="2019-
```

```
12-14T21:14:29.376444960-05:00" level=info
msg="Daemon has completed initialization"
Dec 14 21:14:29 my-n2-1 systemd[1]: Started Docker
Application Container Engine.
Dec 14 21:14:29 my-n2-1 dockerd[3347]: time="2019-
12-14T21:14:29.444607195-05:00" level=info
msg="API listen on /var/run/docker.sock"
Dec 14 21:49:11 my-n2-1 dockerd[3347]: time="2019-
12-14T21:49:11.323542665-05:00" level=info
msg="Processing signal 'terminated'"
Dec 14 21:49:11 my-n2-1 dockerd[3347]: time="2019-
12-14T21:49:11.328379659-05:00" level=info
msg="stopping event stream following graceful
shutdown" error="" m
Dec 14 21:49:11 my-n2-1 systemd[1]: Stopping
Docker Application Container Engine...
Dec 14 21:49:11 my-n2-1 systemd[1]: Stopped Docker
Application Container Engine.
Dec 14 21:49:11 my-n2-1 systemd[1]: Starting
Docker Application Container Engine...
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.499488062-05:00" level=info
msg="systemd-resolved is running, so using
resolvconf: /run/systemd/res
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.502141612-05:00" level=info
msg="parsed scheme: "unix"" module=grpc
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.502209240-05:00" level=info
msg="scheme "unix" not registered, fallback to
default scheme" module
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.502278577-05:00" level=info
msg="parsed scheme: "unix"" module=grpc
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.502295786-05:00" level=info
msg="scheme "unix" not registered, fallback to
default scheme" module
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.505887217-05:00" level=info
msg="ccResolverWrapper: sending new addresses to
cc: [{unix:///run/cont
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.506035600-05:00" level=info
msg="ClientConn switching balancer to
"pick_first"" module=grpc
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.506181190-05:00" level=info
msg="ccResolverWrapper: sending new addresses to
cc: [{unix:///run/cont
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.506446245-05:00" level=info
msg="ClientConn switching balancer to
"pick_first"" module=grpc
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.506671465-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40007a2230, CONNECT
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.506255319-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40008b0710, CONNECT
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.509814706-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40008b0710, READY"
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.511738887-05:00" level=info
msg="pickfirstBalancer: HandleSubConnStateChange:
0x40007a2230, READY"
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.525913142-05:00" level=info
msg="Graph migration to content-addressability
took 0.00 seconds"
Dec 14 21:49:11 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:11.529808838-05:00" level=info
msg="Loading containers: start."
Dec 14 21:49:12 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:12.258591473-05:00" level=info
msg="Default bridge (docker0) is assigned with an
IP address 172.17.0.0
Dec 14 21:49:12 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:12.540886055-05:00" level=info
msg="Loading containers: done."
Dec 14 21:49:12 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:12.614462758-05:00" level=info
msg="Docker daemon" commit=039a7df
graphdriver(s)=overlay2 version=18.0
Dec 14 21:49:12 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:12.614718313-05:00" level=info
msg="Daemon has completed initialization"
Dec 14 21:49:12 my-n2-1 dockerd[9629]: time="2019-
12-14T21:49:12.640530153-05:00" level=info
msg="API listen on /var/run/docker.sock"
Dec 14 21:49:12 my-n2-1 systemd[1]: Started Docker
Application Container Engine.
```

A continuación, debemos deshabilitar el intercambio basado en disco. Para ello necesitamos realizar dos cosas.

Primer paso, en cada uno de los nodos my-n2-1 a través de my-n2-5, edita el archivo /etc/default/armbian-zram-config y cambia la línea ENABLED=true por ENABLED=false.

Segundo paso, en cada uno de los nodos my-n2-1 a través de my-n2-5, ejecuta los siguientes comandos:

```
$ sudo systemctl disable armbian-zram-config
$ sudo reboot now
```

Una vez más, en cada una de las pestañas de Terminal, conéctate por ssh al nodo correspondiente.

Esto completa la instalación y la configuración del sistema de los nodos del clúster. Próxima parada: configuración de Kubernetes

Configuración de Kubernetes

Para empezar, designaremos el nodo my-n2-1 como nodo maestro y configuraremos el plano de control. Para hacer esto, ejecuta el siguiente comando en my-n2-1:

```
$ sudo kubeadm init
```

El siguiente sería un resultado típico:

```
Output.8
[init] Using Kubernetes version: v1.16.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [my-n2-1 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.51]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [my-n2-1 localhost] and IPs [192.168.1.51 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [my-n2-1 localhost] and IPs [192.168.1.51 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
W1215 11:58:08.359442 4811 manifests.go:214] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"
[control-plane] Creating static Pod manifest for "kube-scheduler"
W1215 11:58:08.366477 4811 manifests.go:214] the default kube-apiserver authorization-mode is "Node,RBAC"; using "Node,RBAC"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 25.513764 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.17" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --
```

```
upload-certs
[mark-control-plane] Marking the node my-n2-1 as
control-plane by adding the label "node-
role.kubernetes.io/master=''"
[mark-control-plane] Marking the node my-n2-1 as
control-plane by adding the taints [node-
role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token:
zcp5a6.w03lcuhx068wvkqv
[bootstrap-token] Configuring bootstrap tokens,
cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow
Node Bootstrap tokens to post CSRs in order for
nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow
the csrapprover controller automatically approve
CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow
certificate rotation for all node client
certificates in the cluster
[bootstrap-token] Creating the "cluster-info"
ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating
"/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
```

¡Tu plano de control Kubernetes se ha iniciado con éxito!

Para empezar a usar tu clúster, debes ejecutar lo siguiente como usuario normal:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Ahora debe implementar una red pod en el clúster. Ejecuta "kubectl apply -f [podnetwork] .yaml" con una de las opciones listadas en: https://kubernetes.io/docs/concepts/cluster-administration/addons/

Luego puedes unir cualquier número de nodos de trabajo ejecutando lo siguiente en cada uno como root:

kubeadm join 192.168.1.51:6443 --token zcp5a6.w03lcuhx068wvkqv --discovery-token-ca-cert-hash sha256:d2e38957f46a9eb089671924bca78ac4e02cdcc8db27e89677a014fe587b67c6

Para utilizar la herramienta de comandos kubectl como usuario no root en el nodo maestro (my-n2-1), ejecuta los siguientes comandos en my-n2-1:

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf
$HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Para hacer una lista de todos los nodos en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get nodes
```

El siguiente sería un resultado típico:

```
Output.9
NAME STATUS ROLES AGE VERSION
My-n2-1 NotReady master 2m37s v1.16.3
```

Para verificar que el clúster de Kubernetes se haya iniciado correctamente, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods -n kube-system -o wide
```

Lo siguiente sería un resultado típico

```
Output.10
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED
NODE READINESS GATES
coredns-6955765f44-4gk4f 1/1 Running 0 40m
10.32.0.3 my-n2-1
coredns-6955765f44-wskl4 1/1 Running 0 40m
10.32.0.2 my-n2-1
etcd-my-n2-1 1/1 Running 0 40m 192.168.1.51 my-n2-
1
kube-apiserver-my-n2-1 1/1 Running 0 40m
192.168.1.51 my-n2-1
kube-controller-manager-my-n2-1 1/1 Running 0 40m
192.168.1.51 my-n2-1
kube-proxy-tklp7 1/1 Running 0 40m 192.168.1.51
my-n2-1
kube-scheduler-my-n2-1 1/1 Running 0 40m
192.168.1.51 my-n2-1
```

En el resultado anterior, podemos ver que todos los componentes principales (servidor api, administrador de controladores, etc. y planificador) se encutran funcionando.

Ahora, necesitamos instalar una red de plugins superpuesta para la comunicación entre pod. Para nuestro clúster, elegiremos la implementación weave-net. Para instalar la red superpuesta en el nodo maestro (my-n2-1), ejecuta el siguiente comando:

```
$ kubectl apply -f
"https://cloud.weave.works/k8s/net?k8s-
version=$(kubectl version | base64 | tr -d '
')"
```

El siguiente sería un resultado típico:

```
Output.11
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net
created
clusterrolebinding.rbac.authorization.k8s.io/weave
-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net
created
daemonset.apps/weave-net created
```

Para verificar que la red de superposición Weave se ha iniciado correcatente, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods -n kube-system -l name=weave-
net -o wide
```

El siguiente sería un resultado típico:

```
Output.12
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED
NODE READINESS GATES
weave-net-2sjh4 2/2 Running 0 10m 192.168.1.51 my-
n2-1
```

Además, para verificar los registros de la red de superposición Weave, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl logs -n kube-system weave-net-ktjnv
weave
```

El siguiente sería un resultado típico:

```
Output.13
INFO: 2019/12/08 17:07:12.422554 Command line
options: map[conn-limit:200 datapath:datapath db-
prefix:/weavedb/weave-net docker-api: expect-
npc:true host-root:/host http-addr:127.0.0.1:6784
ipalloc-init:consensus=0 ipalloc-
range:10.32.0.0/12 metrics-addr:0.0.0.0:6782
name:9a:59:d0:9a:83:f0 nickname:my-n2-1 no-
dns:true port:6783]
INFO: 2019/12/08 17:07:12.422876 weave 2.6.0
INFO: 2019/12/08 17:07:12.780249 Bridge type is
bridged_fastdp
INFO: 2019/12/08 17:07:12.780350 Communication
between peers is unencrypted.
INFO: 2019/12/08 17:07:12.804023 Our name is
9a:59:d0:9a:83:f0(my-n2-1)
INFO: 2019/12/08 17:07:12.804267 Launch detected -
using supplied peer list: []
INFO: 2019/12/08 17:07:12.844222 Unable to fetch
ConfigMap kube-system/weave-net to infer unique
cluster ID
INFO: 2019/12/08 17:07:12.844324 Checking for pre-
existing addresses on weave bridge
INFO: 2019/12/08 17:07:12.853900 [allocator
9a:59:d0:9a:83:f0] No valid persisted data
INFO: 2019/12/08 17:07:12.866497 [allocator
9a:59:d0:9a:83:f0] Initialising via deferred
consensus
INFO: 2019/12/08 17:07:12.866684 Sniffing traffic
on datapath (via ODP)
INFO: 2019/12/08 17:07:12.872570 Listening for
HTTP control messages on 127.0.0.1:6784
INFO: 2019/12/08 17:07:12.873074 Listening for
metrics requests on 0.0.0.0:6782
INFO: 2019/12/08 17:07:13.540248 [kube-peers]
Added myself to peer list &{[{9a:59:d0:9a:83:f0
my-n2-1}]}
DEBU: 2019/12/08 17:07:13.558983 [kube-peers]
Nodes that have disappeared: map[]
INFO: 2019/12/08 17:07:13.661165 Assuming quorum
size of 1
10.32.0.1
DEBU: 2019/12/08 17:07:13.911144 registering for
updates for node delete events
```

Para este tutorial, designamos que los nodos my-n2-2 a través de my-n2-5 sean los nodos de trabajo de este clúster de Kubernetes. En el resultado 8, podemos determinar el comando de combinación kubeadm para usar en cada nodo de trabajo. Para cada uno de los nodos my-n2-2 a través de my-n2-5 (en su respectiva pestaña Terminal), ejecuta el siguiente comando:

```
$ sudo kubeadm join 192.168.1.51:6443 --token
zcp5a6.w03lcuhx068wvkqv --discovery-token-ca-cert-
hash
```

```
sha256:d2e38957f46a9eb089671924bca78ac4e02cdcc8db2
7e89677a014fe587b67c6
```

El siguiente sería un resultado típico:

```
Output.14
[preflight] Running pre-flight checks
[preflight] Reading configuration from the
cluster...
[preflight] FYI: You can look at this config file
with 'kubectl -n kube-system get cm kubeadm-config
-oyaml'
[kubelet-start] Downloading configuration for the
kubelet from the "kubelet-config-1.17" ConfigMap
in the kube-system namespace
[kubelet-start] Writing kubelet configuration to
file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file
with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform
the TLS Bootstrap...
```

Este nodo se ha unido al clúster: * Se envió una solicitud de firma de certificado a un servidor y se recibió una respuesta. * El Kubelet ha sido informado de los nuevos detalles de conexión segura.

Ejecuta 'kubectl get node' en el plano de control para ver como este nodo se une al clúster.

Para hacer una relación de todos los nodos activos en este clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1) (después de esperar unos 30 segundos):

```
$ kubectl get nodes -o wide
```

El siguiente sería un resultado típico:

```
Output.15
NAME STATUS ROLES AGE VERSION INTERNAL-IP
EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-
RUNTIME
my-n2-1 Ready master 51m v1.17.0 192.168.1.51
Ubuntu 18.04.3 LTS 4.9.196-meson64 docker://18.9.9
my-n2-2 Ready  2m58s v1.17.0 192.168.1.52  Ubuntu
18.04.3 LTS 4.9.196-meson64 docker://18.9.9
my-n2-3 Ready  2m38s v1.17.0 192.168.1.53  Ubuntu
18.04.3 LTS 4.9.196-meson64 docker://18.9.9
my-n2-4 Ready  2m35s v1.17.0 192.168.1.54  Ubuntu
18.04.3 LTS 4.9.196-meson64 docker://18.9.9
my-n2-5 Ready  2m21s v1.17.0 192.168.1.55  Ubuntu
18.04.3 LTS 4.9.196-meson64 docker://18.9.9
```

¡Eso es! Esto completa toda la configuración necesaria para este clúster de Kubernetes.

Práctica con Kubernetes

Para hacer un lista de todos los pod (s) que se ejecutan en el clúster de Kubernetes (incluidos los pods del sistema), ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods --all-namespaces -o wide
```

El siguiente sería un resultado típico:

```
Output.16
NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE
NOMINATED NODE READINESS GATES
kube-system coredns-6955765f44-4gk4f 1/1 Running 0
52m 10.32.0.3 my-n2-1
kube-system coredns-6955765f44-wskl4 1/1 Running 0
52m 10.32.0.2 my-n2-1
kube-system etcd-my-n2-1 1/1 Running 0 52m
192.168.1.51 my-n2-1
kube-system kube-apiserver-my-n2-1 1/1 Running 0
52m 192.168.1.51 my-n2-1
kube-system kube-controller-manager-my-n2-1 1/1
Running 0 52m 192.168.1.51 my-n2-1
kube-system kube-proxy-9zxfj 1/1 Running 0 3m36s
192.168.1.55 my-n2-5
kube-system kube-proxy-c7mns 1/1 Running 0 3m53s
192.168.1.53 my-n2-3
kube-system kube-proxy-dv52p 1/1 Running 0 4m13s
192.168.1.52 my-n2-2
kube-system kube-proxy-mpwkb 1/1 Running 0 3m50s
192.168.1.54 my-n2-4
kube-system kube-proxy-tklp7 1/1 Running 0 52m
192.168.1.51 my-n2-1
kube-system kube-scheduler-my-n2-1 1/1 Running 0
52m 192.168.1.51 my-n2-1
kube-system weave-net-2sjh4 2/2 Running 0 21m
192.168.1.51 my-n2-1
kube-system weave-net-68lcd 2/2 Running 0 3m50s
192.168.1.54 my-n2-4
kube-system weave-net-7fh98 2/2 Running 1 4m13s
192.168.1.52 my-n2-2
kube-system weave-net-krdtz 2/2 Running 1 3m36s
192.168.1.55 my-n2-5
kube-system weave-net-ljm6k 2/2 Running 0 3m53s
192.168.1.53 my-n2-3
```

Como es evidente en el resultado 16 anterior, vemos una entrada para API Server, etcd, Controller Manager, Scheduler y Plugin Network (weave-net) que indica que están en funcionamiento.

Para mostrar información detallada sobre cualquier pod (digamos el Controller Manager) en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl describe pod kube-controller-manager-my-n2-1 -n kube-system
```

El siguiente sería un resultado típico:

```
Output.17
Name: kube-controller-manager-my-n2-1
Namespace: kube-system
Priority: 2000000000
Priority Class Name: system-cluster-critical
Node: my-n2-1/192.168.1.51
Start Time: Sun, 15 Dec 2019 11:58:39 -0500
Labels: component=kube-controller-manager
tier=control-plane
Annotations: kubernetes.io/config.hash:
536dc7132dfd0d2ca1d968c9ede1e024
kubernetes.io/config.mirror:
536dc7132dfd0d2ca1d968c9ede1e024
kubernetes.io/config.seen: 2019-12-
15T11:58:35.86446527-05:00
kubernetes.io/config.source: file
Status: Running
IP: 192.168.1.51
IPs:
IP: 192.168.1.51
Controlled By: Node/my-n2-1
Containers:
kube-controller-manager:
Container ID:
docker://63b0d105457f52849afa38d2e914b53e68b7e21786fc41cda322bb21bc5b86a4
Image: k8s.gcr.io/kube-controller-manager:v1.17.0
Image ID: docker-pullable://k8s.gcr.io/kube-controller-
manager@sha256:0438efb5098a2ca634ea8c6b0d804742b733d0d13fd53cf62c73e32c659a3c39
Port:
Host Port:
Command:
kube-controller-manager
--authentication-
kubeconfig=/etc/kubernetes/controller-manager.conf
--authorization-
kubeconfig=/etc/kubernetes/controller-manager.conf
--bind-address=127.0.0.1
--client-ca-file=/etc/kubernetes/pki/ca.crt
--cluster-signing-cert-
file=/etc/kubernetes/pki/ca.crt
--cluster-signing-key-
file=/etc/kubernetes/pki/ca.key
--controllers=*,bootstrapsigner,tokencleaner
--kubeconfig=/etc/kubernetes/controller-
manager.conf
--leader-elect=true
--requestheader-client-ca-
file=/etc/kubernetes/pki/front-proxy-ca.crt
--root-ca-file=/etc/kubernetes/pki/ca.crt
--service-account-private-key-
file=/etc/kubernetes/pki/sa.key
--use-service-account-credentials=true
State: Running
Started: Sun, 15 Dec 2019 11:58:22 -0500
Ready: True
Restart Count: 0
Requests:
cpu: 200m
Liveness: http-get https://127.0.0.1:10257/healthz
delay=15s timeout=15s period=10s #success=1
#failure=8
Environment:
Mounts:
/etc/ca-certificates from etc-ca-certificates (ro)
/etc/kubernetes/controller-manager.conf from
kubeconfig (ro)
/etc/kubernetes/pki from k8s-certs (ro)
/etc/ssl/certs from ca-certs (ro)
/usr/libexec/kubernetes/kubelet-
plugins/volume/exec from flexvolume-dir (rw)
/usr/local/share/ca-certificates from usr-local-
share-ca-certificates (ro)
/usr/share/ca-certificates from usr-share-ca-
certificates (ro)
Conditions:
Type Status
Initialized True
Ready True
ContainersReady True
PodScheduled True
Volumes:
ca-certs:
Type: HostPath (bare host directory volume)
Path: /etc/ssl/certs
HostPathType: DirectoryOrCreate
etc-ca-certificates:
Type: HostPath (bare host directory volume)
Path: /etc/ca-certificates
```

```
HostPathType: DirectoryOrCreate
flexvolume-dir:
Type: HostPath (bare host directory volume)
Path: /usr/libexec/kubernetes/kubelet-
plugins/volume/exec
HostPathType: DirectoryOrCreate
k8s-certs:
Type: HostPath (bare host directory volume)
Path: /etc/kubernetes/pki
HostPathType: DirectoryOrCreate
kubeconfig:
Type: HostPath (bare host directory volume)
Path: /etc/kubernetes/controller-manager.conf
HostPathType: FileOrCreate
usr-local-share-ca-certificates:
Type: HostPath (bare host directory volume)
Path: /usr/local/share/ca-certificates
HostPathType: DirectoryOrCreate
usr-share-ca-certificates:
Type: HostPath (bare host directory volume)
Path: /usr/share/ca-certificates
HostPathType: DirectoryOrCreate
QoS Class: Burstable
Node-Selectors: < none >
Tolerations: :NoExecute
Events: < none >
```

Para hacer una relación de todos los pod(s) de aplicación que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods
```

El siguiente sería un resultado típico:

```
Output.18
No resources found in default namespace.
```

Para hacer una lista de todos los servicios que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get services
```

El siguiente sería un resultado típico:

```
Output.19
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.96.0.1  443/TCP 64m
```

Crearemos una simple aplicación web Python para mostrar el nombre del host, así como la dirección IP cuando se invoque a través de HTTP. El siguiente contenido pertenece a la aplicación web Python almacenada en el directorio /tmp en el nodo maestro (my-n2-1):

```
web-echo.py
from flask import Flask
import socket

app = Flask(__name__)

@app.route("/")
def index():
host_name = socket.gethostname()
host_ip = socket.gethostbyname(host_name)
return 'Hello from container -> ' + host_name + '
[' + host_ip + ']'

if __name__ == "__main__":
app.run(host='0.0.0.0', port=8888)
```

El siguiente es el contenido de Dockerfile para crear una imagen Docker para la simple aplicación web de Python almacenada en el directorio /tmp en el nodo maestro (my-n2-1):

```
Dockerfile
FROM python:3.7.5-alpine3.9
RUN pip install flask
ADD web-echo.py /web-echo.py
CMD ["python", "/web-echo.py"]
```

Para crear una imagen de Docker llamada py-web-echo con la etiqueta v1.0, ejecuta los siguientes comandos en el nodo maestro (my-n2-1):

```
cd /tmp
docker build -t "py-web-echo:v1.0" .
```

El siguiente sería un resultado típico:

```
Output.20
Sending build context to Docker daemon 3.072kB
Step 1/4: FROM python:3.7.5-alpine3.9
3.7.5-alpine3.9: Pulling from library/python
0362ad1dd800: Pull complete
9b941924aae3: Pull complete
fd7b3613915d: Pull complete
078d60b9b97e: Pull complete
7059e1dd9bc4: Pull complete
Digest:
sha256:064d9ce3e91a59535c528bc3c38888023791d9fc78b
a9e5070f5064833f326ff
```

```
Status: Downloaded newer image for python:3.7.5-alpine3.9
---> 578ec6233872
Step 2/4: RUN pip install flask
---> Running in d248e23dd161
Collecting flask
Downloading https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43bbb1d3b/Flask-1.1.1-py2.py3-none-any.whl (94kB)
Collecting Jinja2>=2.10.1
Downloading https://files.pythonhosted.org/packages/65/e0/eb35e762802015cab1ccee04e8a277b03f1d8e53da3ec3106882ec42558b/Jinja2-2.10.3-py2.py3-none-any.whl (125kB)
Collecting Werkzeug>=0.15
Downloading https://files.pythonhosted.org/packages/ce/42/3aeda98f96e85fd26180534d36570e4d18108d62ae36f87694b476b83d6f/Werkzeug-0.16.0-py2.py3-none-any.whl (327kB)
Collecting itsdangerous>=0.24
Downloading https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting click>=5.1
Downloading https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl (81kB)
Collecting MarkupSafe>=0.23
Downloading https://files.pythonhosted.org/packages/b9/2e/64db92e53b86efccfaea71321f597fa2e1b2bd3853d8ce658568f7a13094/MarkupSafe-1.1.1.tar.gz
Building wheels for collected packages: MarkupSafe
Building wheel for MarkupSafe (setup.py): started
Building wheel for MarkupSafe (setup.py): finished with status 'done'
Created wheel for MarkupSafe: filename=MarkupSafe-1.1.1-cp37-none-any.whl size=12629 sha256=8a200864ca113d03b4de2d951ae4a1d0806a3ff84128349770dfe3fb018a6458
Stored in directory: /root/.cache/pip/wheels/f2/aa/04/0edf07a1b8a5f5f1aed7580fffb69ce8972edc16a505916a77
Successfully built MarkupSafe
Installing collected packages: MarkupSafe, Jinja2, Werkzeug, itsdangerous, click, flask
Successfully installed Jinja2-2.10.3 MarkupSafe-1.1.1 Werkzeug-0.16.0 click-7.0 flask-1.1.1 itsdangerous-1.1.0
Removing intermediate container d248e23dd161
---> 4ee40e66a655
Step 3/4: ADD web-echo.py /web-echo.py
---> 31a0341bf9d7
Step 4/4: CMD ["python", "/web-echo.py"]
---> Running in 1ee52ea10ad3
Removing intermediate container 1ee52ea10ad3
---> 7cd037d24ef7
Successfully built 7cd037d24ef7
Successfully tagged py-web-echo:v1.0
```

Para hacer una relación de todas las imágenes de Docker en el nodo maestro (my-n2-1), ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ docker images
```

El siguiente sería un resultado típico:

```
Output.21
REPOSITORY TAG IMAGE ID CREATED SIZE
py-web-echo v1.0 7cd037d24ef7 3 minutes ago 119MB
k8s.gcr.io/kube-proxy v1.17.0 ac19e9cffff5 7 days ago 114MB
k8s.gcr.io/kube-apiserver v1.17.0 aca151bf3e90 7 days ago 166MB
k8s.gcr.io/kube-controller-manager v1.17.0 7045158f92f8 7 days ago 156MB
k8s.gcr.io/kube-scheduler v1.17.0 0d5c120f87f3 7 days ago 93.7MB
python 3.7.5-alpine3.9 578ec6233872 4 weeks ago 109MB
weaveworks/weave-npc 2.6.0 1c672c2f5870 5 weeks ago 36.6MB
weaveworks/weave-kube 2.6.0 81393394d17d 5 weeks ago 111MB
k8s.gcr.io/coredns 1.6.5 f96217e2532b 5 weeks ago 39.3MB
k8s.gcr.io/etcd 3.4.3-0 ab707b0a0ea3 7 weeks ago 363MB
k8s.gcr.io/pause 3.1 6cf7c80fe444 24 months ago 525kB
```

Ten en cuenta que creamos la imagen de Docker en el nodo maestro (my-n2-1). Dado que los pods se implementarán en los nodos de trabajo, debemos asegurarnos de que las imágenes docker necesarias estén presentes en los nodos de trabajo.

Para cada uno de los nodos de trabajo my-n2-2 hasta my-n2-5 (en su respectiva pestaña Terminal), ejecuta el siguiente comando:

```
$ docker pull python:3.7.5-alpine3.9
```

Para cada uno de los nodos de trabajo my-n2-2 a través de my-n2-5, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ docker save py-web-echo:v1.0 | bzip2 | ssh
polarsparc@192.168.1.52 'bunzip2 | docker load'
$ docker save py-web-echo:v1.0 | bzip2 | ssh
polarsparc@192.168.1.53 'bunzip2 | docker load'
$ docker save py-web-echo:v1.0 | bzip2 | ssh
polarsparc@192.168.1.54 'bunzip2 | docker load'
$ docker save py-web-echo:v1.0 | bzip2 | ssh
polarsparc@192.168.1.55 'bunzip2 | docker load'
```

!!! AVISO IMPORTANTE !!!

No tener las imágenes de Docker en los nodos de trabajo hará que los pod se queden atascados en el estado de creación de los contenedores.

En Kubernetes, un pod es lo que encapsula los contenedores Docker. Para implementar nuestra aplicación web Docker py-web-echo:v1.0 en nuestro clúster de Kubernetes, necesitamos un archivo manifiesto de pod en formato YAML.

Los siguientes son los contenidos del archivo manifiesto del pod llamado web-echo-pod.yaml almacenado en el directorio /tmp en el nodo maestro (my-n2-1):

```
web-echo-pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
name: web-echo-pod
labels:
app: web-echo
spec:
containers:
- name: web-echo
image: py-web-echo:v1.0
imagePullPolicy: Never
ports:
- containerPort: 8888
```

A continuación explicaremos los elementos del archivo de manifiesto web-echo-pod.yaml:

- apiVersion: especifica la versión de la API (v1 en este ejemplo)
- kind: especifica el tipo de objeto Kubernetes a utilizar (Pod en este ejemplo)
- metadata: asocia un nombre (web-echo-pod en este ejemplo) con el tipo de objeto Kubernetes. Además, permite marcar algunas etiquetas, que son simples pares clave-valor, con los Kubernetes.
- object. En este ejemplo, tenemos una etiqueta con la aplicación clave que tiene un valor de web-echo
- spec: especifica lo que hay en el pod. En este ejemplo, queremos implementar la imagen Docker py-web-echo: v1.0 que se presenta a través del puerto de red 8888.
- imagePullPolicy: indica a Kubernetes que no tire de la imagen del contenedor.

Para implementar el pod en nuestro clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl apply -f /tmp/web-echo-pod.yaml
```

El siguiente sería un resultado típico:

```
Output.22
pod/web-echo-pod created
```

Para hacer una lista de todos los pods de aplicaciones que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods -o wide
```

El siguiente sería un resultado típico:

```
Output.23
1
```

En el resultado 23, vemos que nuestro pod de aplicación se ha implementado en el nodo my-n2-2 de nuestro clúster Kubernetes.

Para mostrar información detallada sobre el pod de aplicación web-echo-pod implementado, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl describe pods web-echo-pod
```

El siguiente sería un resultado típico:

```
Output.24
Name: web-echo-pod
Namespace: default
Priority: 0
```

```
Node: my-n2-2/192.168.1.52
Start Time: Sun, 15 Dec 2019 14:58:21 -0500
Labels: app=web-echo
Annotations: kubectl.kubernetes.io/last-applied-
configuration:
{"apiVersion":"v1","kind":"Pod","metadata":
{"annotations":{},"labels":{"app":"web-
echo"},"name":"web-echo-
pod","namespace":"default"},"spe...
Status: Running
IP: 10.44.0.1
IPs:
IP: 10.44.0.1
Containers:
web-echo:
Container ID:
docker://0af2c99fd074b5ee3c0b9876eb9ad44ca446400c2
190b4af6fa1a18543bff723
Image: py-web-echo:v1.0
Image ID:
docker://sha256:7cd037d24ef7c842ffe005cfcb548a802f
c13661c08c8bb4635c365f77e5a3aa
Port: 8888/TCP
Host Port: 0/TCP
State: Running
Started: Sun, 15 Dec 2019 14:58:23 -0500
Ready: True
Restart Count: 0
Environment:
Mounts:
/var/run/secrets/kubernetes.io/serviceaccount from
default-token-tvl5x (ro)
Conditions:
Type Status
Initialized True
Ready True
ContainersReady True
PodScheduled True
Volumes:
default-token-tvl5x:
Type: Secret (a volume populated by a Secret)
SecretName: default-token-tvl5x
Optional: false
QoS Class: BestEffort
Node-Selectors:
Tolerations: node.kubernetes.io/not-
ready:NoExecute for 300s
node.kubernetes.io/unreachable:NoExecute for 300s
Events:
Type Reason Age From Message
---- ------ ---- ---- -------
Normal Scheduled 7m39s default-scheduler
Successfully assigned default/web-echo-pod to my-
n2-2
Normal Pulled 7m38s kubelet, my-n2-2 Container
image "py-web-echo:v1.0" already present on
machine
Normal Created 7m38s kubelet, my-n2-2 Created
container web-echo
Normal Started 7m37s kubelet, my-n2-2 Started
container web-echo
```

En el anterior resultado 23 (así como en el resultado 24), vemos que la dirección IP de la aplicación web implementada es 10.44.0.1.

Para probar la aplicación web desplegada utilizando el comando curl, ejecuta el siguiente comando en cualquiera de los nodos my-n2-1 a través de my-n2-5:

```
$ curl http://10.44.0.1:8888
```

El siguiente sería un resultado típico:

```
Output.25
Hello from container -> web-echo-pod [10.44.0.1]
```

Para mostrar los registros log de la aplicación web implementada web-echo-pod, ejecuta el siguienta comando en el nodo maestro (my-n2-1):

```
$ kubectl logs web-echo-pod
```

El siguiente sería un resultado típico:

```
Output.26
* Serving Flask app "web-echo" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use
it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8888/ (Press CTRL+C to
quit)
10.32.0.1 - - [15/Dec/2019 20:11:33] "GET /
HTTP/1.1" 200 -
10.36.0.0 - - [15/Dec/2019 20:11:58] "GET /
HTTP/1.1" 200 -
```

Para eliminar la aplicación web desplegada web-echo-pod, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl delete pod web-echo-pod
```

El siguiente sería un resultado típico:

```
Output.27
pod "web-echo-pod" deleted
```

NOT es muy común al implementar un único Pod. Es más normal implementar un objeto Kubernetes de nivel superior llamado ReplicaSet. Un ReplicaSet define cuántas réplicas de un Pod deben implementarse y mantenerse en el clúster de Kubernetes.

Los siguientes son los contenidos del archivo manifiesto de ReplicaSet llamado web-echo-rs.yaml almacenado en el directorio /tmp en el nodo maestro (my-n2-1):

```
web-echo-rs.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
name: web-echo-rs
spec:
replicas: 3
selector:
matchLabels:
app: web-echo
template:
metadata:
labels:
app: web-echo
spec:
containers:
- name: web-echo
image: py-web-echo:v1.0
imagePullPolicy: Never
ports:
- containerPort: 8888
```

Explicaremos ahora algunos de los elementos del archivo manifiesto web-echo-rs.yaml:

apiVersion: especifica la versión de la API (apps/v1 en este ejemplo) replicas: indica las instancias deseadas del Pod que se ejecutarán en el clúster de Kubernetes selector: identifica y selecciona un grupo de objetos de Kubernetes con la misma etiqueta de valor clave (aplicación clave y valor web-echo en este ejemplo)

template: es la especificación interna de un Pod

Para implementar ReplicaSet en nuestro clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl apply -f /tmp/web-echo-rs.yaml
```

El siguiente sería un resultado típico:

```
Output.28
replicaset.apps/web-echo-rs created
```

Para hacer una lista de todos los ReplicaSet implementados que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get replicasets -o wide
```

El siguiente sería un resultado típico:

```
Output.29
NAME DESIRED CURRENT READY AGE CONTAINERS IMAGES
SELECTOR
web-echo-rs 3 3 3 7m web-echo py-web-echo:v1.0
app=web-echo
```

Para mostrar información detallada sobre el ReplicaSet implementado llamado web-echo-rs, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl describe replicasets web-echo-rs
```

El siguiente sería un resultado típico:

```
Output.30
Name: web-echo-rs
Namespace: default
Selector: app=web-echo
Labels:
Annotations: kubectl.kubernetes.io/last-applied-
configuration:
{"apiVersion":"apps/v1","kind":"ReplicaSet","metad
ata":{"annotations":{},"name":"web-echo-
rs","namespace":"default"},"spec":
{"replicas":3,...
Replicas: 3 current / 3 desired
Pods Status: 3 Running / 0 Waiting / 0 Succeeded /
0 Failed
Pod Template:
Labels: app=web-echo
Containers:
web-echo:
Image: py-web-echo:v1.0
Port: 8888/TCP
Host Port: 0/TCP
```

```
Environment:
Mounts:
Volumes:
Events:
Type Reason Age From Message
---- ------ ---- ---- -------
Normal SuccessfulCreate 14m replicaset-controller
Created pod: web-echo-rs-xn94l
Normal SuccessfulCreate 14m replicaset-controller
Created pod: web-echo-rs-9x9b9
Normal SuccessfulCreate 14m replicaset-controller
Created pod: web-echo-rs-tbd49
```

Para hacer una lista de todos los pods de aplicaciones que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get pods -o wide
```

El siguiente sería un resultado típico:

```
Output.31
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED
NODE READINESS GATES
web-echo-rs-9x9b9 1/1 Running 0 63s 10.42.0.1 my-
n2-4
web-echo-rs-tbd49 1/1 Running 0 63s 10.44.0.1 my-
n2-2
web-echo-rs-xn94l 1/1 Running 0 63s 10.36.0.1 my-
n2-3
```

En el resultado 31, vemos que nuestros pods de aplicación se han implementado en los 3 nodos my-n2-2, my-n2-3 y my-n2-4 con direcciones IP únicas de 10.44.0.1, 10.36 .0.1 y 10.42.0.1 respectivamente.

Como se ha mencionado al principio, los pods de aplicación son efímeros. Pueden aparecer y desaparecer en cualquier momento. Esto significa que sus direcciones IP pueden cambiar en cualquier momento. Necesitamos una abstracción de nivel superior que proporcione una dirección IP estable para que otros pod (s) de aplicaciones puedan usarla. Aquí es donde resula muy útil un objeto de Servicio. Proporciona una única dirección IP estable para que otras aplicaciones la usen y distribuye la carga a través de los diferentes pod (s) de aplicaciones del back-end que están delante.

Hay 3 tipos de servicio(s) en Kubernetes:

- ClusterIP: expone el Servicio en una dirección IP que es interna al clúster de Kubernetes. Esto significa que se puede acceder al Servicio *SOLO* desde dentro del clúster de Kubernetes. Este es el tipico por defecto.
- NodePort: expone al Servicio en la dirección IP de cada nodo de trabajo en un puerto alto en el rango de 30000 a 32767. Las aplicaciones externas al clúster de Kubernetes pueden acceder al Servicio en la dirección IP del nodo de trabajo y el puerto de nodo asignado
- LoadBalancer: Expone el Servicio externamente usando un Load Balancer de proveedores en la nube como AWS, Azure o Google Cloud

El siguiente es el contenido del archivo de manifiesto del Servicio basado en ClusterIP llamado web-echo-svc-cip.yaml almacenado bajo el directorio /tmp en el nodo maestro (my-n2-1):

```
web-echo-svc-cip.yaml
---
apiVersion: v1
kind: Service
metadata:
name: web-echo-svc-cip
spec:
selector:
app: web-echo
ports:
- name: http
protocol: TCP
port: 8888
```

Para implementar el Servicio en nuestro clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl apply -f /tmp/web-echo-svc-cip.yaml
```

El siguiente sería un resultado típico:

```
Output.32
service/web-echo-svc created
```

Para hacer una lista de todos los Servicios que se ejecuten en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get services -o wide
```

El siguiente sería un resultado típico:

```
Output.33
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
SELECTOR
kubernetes ClusterIP 10.96.0.1  443/TCP 9h
web-echo-svc ClusterIP 10.96.238.16  8888/TCP 105s
app=web-echo
```

En el resulado 33 anterior, vemos que se puede acceder a la aplicación web-echo desde cualquier lugar del clúster a través de la dirección IP 10.96.238.16 y el puerto 8888.

Para probar el punto final del Servicio implementado utilizando el comando curl, ejecuta el siguiente comando 5 veces en cualquiera de los nodos my-n2-1 a través de my-n2-5:

```
$ curl http://10.96.238.16:8888
```

El siguiente sería un resultado típico:

```
Output.34
Hello from container -> web-echo-rs-xn94l
[10.36.0.1]
Hello from container -> web-echo-rs-9x9b9
[10.42.0.1]
Hello from container -> web-echo-rs-tbd49
[10.44.0.1]
Hello from container -> web-echo-rs-9x9b9
[10.42.0.1]
Hello from container -> web-echo-rs-tbd49
[10.44.0.1]
```

Para mostrar información detallada sobre el punto final del servicio etiquetado web-echo-svc, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl describe service web-echo-svc
```

El siguiente sería un resultado típico:

```
Output.35
Name: web-echo-svc
Namespace: default
Labels:
Annotations: kubectl.kubernetes.io/last-applied-
configuration:
{"apiVersion":"v1","kind":"Service","metadata":
{"annotations":{},"name":"web-echo-
svc","namespace":"default"},"spec":{"ports":
[{"name":"ht...
Selector: app=web-echo
Type: ClusterIP
IP: 10.96.238.16
Port: http 8888/TCP
TargetPort: 8888/TCP
Endpoints:
10.36.0.1:8888,10.42.0.1:8888,10.44.0.1:8888
Session Affinity: None
Events:
```

Para eliminar el objeto desplegado web-echo-svc, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl delete service web-echo-svc
```

El siguiente sería un resultado típico:

```
Output.36
service "web-echo-svc" deleted
```

Los siguientes son los contenidos del archivo de manifiesto de servicio basado en NodePort llamado web-echo-svc-nop.yaml almacenado bajo el directorio /tmp en el nodo maestro (my-n2-1):

```
web-echo-svc-nop.yaml
---
apiVersion: v1
kind: Service
metadata:
name: web-echo-svc
spec:
type: NodePort
selector:
app: web-echo
ports:
- name: http
protocol: TCP
port: 8888
```

Para implementar el Servicio en nuestro clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl apply -f /tmp/web-echo-svc-nop.yaml
```

El siguiente sería un resultado típico:

```
Output.37
service/web-echo-svc created
```

Para hacer una lista de todos los Servicios que se ejecutan en el clúster de Kubernetes, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl get services -o wide
```

El siguiente sería un resultado típico

```
Output.38
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
SELECTOR
kubernetes ClusterIP 10.96.0.1  443/TCP 9h
web-echo-svc NodePort 10.96.144.75  8888:32546/TCP
38m app=web-echo
```

Para mostrar información detallada sobre el punto del servicio etiquetado web-echo-svc, ejecuta el siguiente comando en el nodo maestro (my-n2-1):

```
$ kubectl describe service web-echo-svc
```

El siguiente sería un resultado típico:

```
Output.39
Name: web-echo-svc
Namespace: default
Labels:
Annotations: kubectl.kubernetes.io/last-applied-
configuration:
{"apiVersion":"v1","kind":"Service","metadata":
{"annotations":{},"name":"web-echo-
svc","namespace":"default"},"spec":{"ports":
[{"name":"ht...
Selector: app=web-echo
Type: NodePort
IP: 10.96.144.75
Port: http 8888/TCP
TargetPort: 8888/TCP
NodePort: http 32546/TCP
Endpoints:
10.36.0.1:8888,10.42.0.1:8888,10.44.0.1:8888
Session Affinity: None
External Traffic Policy: Cluster
Events:
```

En el resultado 39 anterior, vemos que el puerto del nodo de Servicio implementado es 32546.

Abre un navegador y acceda a la url http://192.168.1.53:32546. La siguiente imagen es la pantalla típica del navegador:

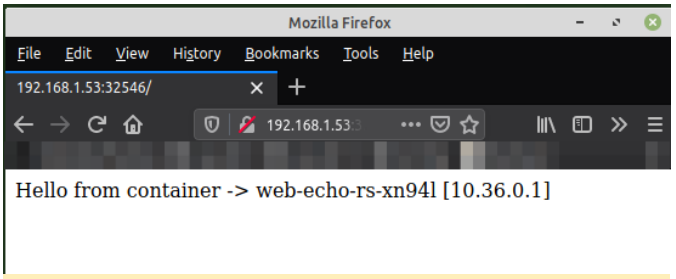


Figura 3

BINGO: ¡todo funciona como era de esperar!

Y con esto concluimos los ejercicios básicos que hemos realizado en nuestro grupo de Kubernetes.

Referencias

https://kubernetes.io/docs/home/?path=browse
https://www.weave.works/docs/net/latest/overview/
https://docs.docker.com/
https://www.polarsparc.com/xhtml/Practical-K8S-N2.html

Sistema de Video Vigilancia en Movimiento Pearl Linux con Kodi: Monitorización Visual Avanzado Utilizando un ODROID-C2

January 10, 2020 By @pearllinux ODROID-C2, Tutoriales

He creado una imagen de video vigilancia basada en Ubuntu 18.04 usando el kernel 3.16.75, con el software Motion Video Surveillance preinstalado y que se inicia en el primer arranque ejecutándose en modo de usuario, no root. Incluye un servidor web Apache preinstalado y WebMin para administrar su sistema a través de la interfaz web, y se inicia en el escritorio Lightweight MATE de Pearl con la mayoría de las características de la versión 7.0 de Pearl.

Características

- Imagen Linux Pearl de Odroid C2 con Kernel 3.16.75
- Motion Video Surveillance preinstalado y activo al inicio
- Webmin y Apache Web Server preconfigurados y activos al inicio
- Entorno de escritorio MATE más reciente

En primer lugar, descárgate la imagen y luego usa Etcher (**https://etcher.io**) para grabar la imagen en tu tarjeta SD o módulo eMMC. En el primer arranque, el sistema cambiará automáticamente el tamaño y usará todo el espacio disponible de tu dispositivo. Dale al sistema unos 2-3 minutos una vez que la pantalla se quede en blanco, luego desconecta la alimentación a C2 y vuelve a conectarla.

Para usar KODI, cierra sesión y vuelva a iniciar sesión en una sesión de Kodi en lugar de Mate. No se requiere contraseña para iniciar sesión en el gestor de pantalla lightdm. El nombre de usuario es "root" u "odroid", y la contraseña es "odroid". Puede cambiar la contraseña desde el símbolo del sistema o en el panel de control.

Video Surveillance

Esta versión de Pearl viene con Motion Video Surveillance versión 4.2.2-1pearl7.3. He creado un

paquete Debian .deb y añadido un Basic Camera Viewer (uno para 2 y otro para 4). La imagen está lista para ser usada, con la única excepción de que tu dirección IP real de tu ordenador puede no ser la misma que la configurada de nuestra LAN. Nosotros hemos utilizando la red 192.168.1.0, que es una de las más comunes (otras pueden ser 10.0.0.1 o 192.168.0.1).

La ubicación del monitor de la cámara esta es /opt/pearl/mcm. Allí, encontrarás 2 archivos para el monitor 2 y 2 archivos para 4. En el archivo HTML cambia la dirección IP si no es la misma. La monitorización de 2 y 4 está configurada para monitorizar únicamente una cámara que está conectada al ODROID-C2. El resto serán cogidas desde otros ordenadores de tu LAN que ejecutan el software de movimiento. Puede descargar e instalar nuestra versión de Motion desde nuestros repositorios en **http://apt.pearllinux.com/pool/main/m/motion/motion_4.2.2-1pearl7.3_arm64.deb**. Tienes más imágenes disponibles en **http://apt.pearllinux.com/pool/main/m/motion/**.

Notas de versión

Puesto que la video vigilancia se indica en el arranque, el directorio es /var/lib/motion. Añade imágenes y videoclips cortos a ese directorio automáticamente. Si usa una tarjeta Micro SD de 16GB, querrás ver ese directorio porque puedes llenar la tarjeta rápidamente.

Puedes cambiar las directivas, incluida la creación automática de estos archivos en el archivo de configuración de Motion principal ubicado en /etc/motion/montion.conf.

Para comentarios, preguntas y sugerencias, visita la publicación original en **https://sourceforge.net/projects/odroid-c2-motionvideo/** o el sitio web de Pearl Linux en **https://www.pearllinux.com**.

Android Things

January 10, 2020 By @Luke.go Android, Desarrollo, Tutoriales

¿Alguna vez has intentado conectar un dispositivo periférico a los pines GPIO de tu SBC ODROID con el sistema operativo Android? Por ejemplo, intentar conectar un interruptor para iniciar una aplicación o conectar un sensor de atenuación. El primer problema con el que te enfrentarás será la dificultad para manejar los pines GPIO desde tu aplicación o servicio Android y tal vez te topes con problemas de permisos para acceder a GPIO, PWM o I2C, ya que a una aplicación genérica de Android se le niega el acceso a recursos de hardware. La solución pasa por exportar una librería de bajo nivel, como wiringPi basado en C/C ++, pero será necesario que se comunique con tu aplicación a través de JNI (Interfaz Nativa de Java) usando NDK. Aun así, tienes que resolver el problema de los permisos.

Google ha presentado otro sistema operativo (SO) conocido como "Android Things", diseñado para ejecutarse en dispositivos integrados livianos y proporciona un entorno de trabajo con Java para manejar periféricos. Mi idea era incorporar el entorno de trabajo de Android Things en el software ODROID permitiendo a los usuarios usar los pines de expansión con facilidad. Sin embargo, el problema es que este sistema operativo no es de código abierto, por lo tanto, tuve que implementar el código en Android para ODROID. Afortunadamente, Google tiene abiertas las APIs del entorno de trabajo con su documento y Android Things SDK. Este hecho me animó a implementar la pila completa del entorno de trabajo que funciona como Android Things, de abajo a arriba.

Utilicé algunas APIs de las partes de gestión periférica de Android Things. Tiene muchas otras características, pero no son necesarias para nuestra tarea. Cree la interfaz para usar la API de Android Things. Para procesar y administrar la solicitud desde la capa de usuario a través de la API, compilé la arquitectura del servidor y cliente y la conecté a la capa de hardware a través de wiringPi para controlar

el propio hardware. Para empezar, se implementaron las funciones GPIO, I2C y PWM, porque son las usadas con más frecuencia por las personas en lugar de otras funciones como SPI y UART. Explicar toda la implementación es lo mejor, aunque solo te mostraré cómo usarla.

Puesto que utilicé el proceso de ingeniería inversa. Mi solución puede volverse incompatible con el sistema operativo Android Things real y/o degradar su rendimiento. Sin embargo, espero que los usuarios que con anterioridad querían usar pines GPIO en Android le resulte más fácil trabajar en C gracias a mi trabajo. Déjame mostrarte un ejemplo de Android Things sobre GPIO, I2C y PWM para aprender cómo usarlo.

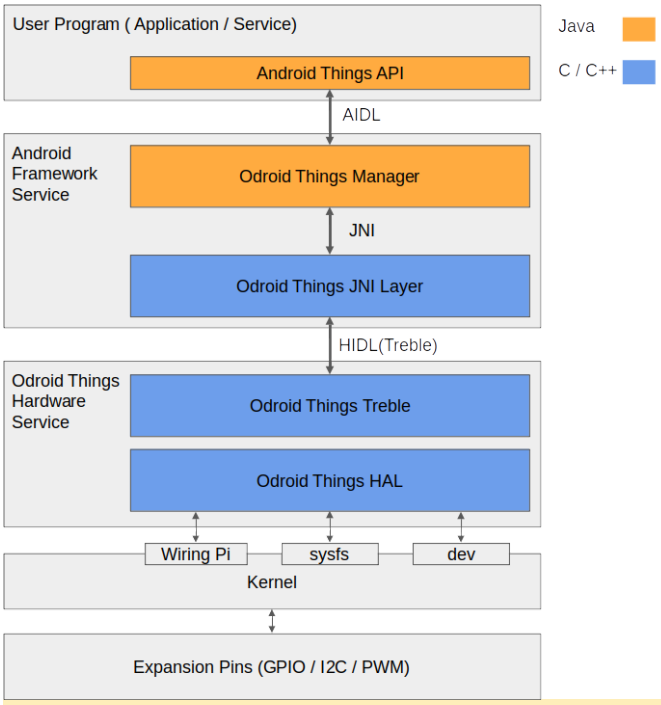


Fig. 01 - Arquitectura

No hay nada tan simple como usar Android Studio para crear, compilar y probar una aplicación o servicio que contiene Android Things. Simplemente necesita instalar Android Studio y agregar la opción oficial y el código oficial para usar Android Things e instalar un paquete en ODROID a través del puerto otg. y ejecutar un paquete ¡Simplemente funciona! Eso es todo. No necesitas hacer nada más.

Subí todo el código de ejemplo a mi repositorio de github (**https://github.com/xiane/thingsGpioExample**). Y cada uno de los ejemplos está separado por ramas. En la rama maestra, puedes controlar el pin GPIO. En la rama i2c_16x4, puede usar 16x4 lcd a través de I2C. En la rama PWM, puede controlar el PWM. En la rama i2c_weather_board, puede usar una placa meteorológica. Úsalo y pruébalo en tus propios proyectos.

Todo el código subyacente se basa en el sitio oficial de Android Things. Consulta el sitio oficial en: **https://developer.android.com/things**.

Manifesto

Antes de probar mis ejemplos, debe agregar las siguientes líneas a tu manifiesto:

Por ejemplo, aquí, **https://bit.ly/2spndDW**.

Debe añadir dependencias a un archivo:

```
compileOnly
'com.google.android.things:androidthings:1.0'
```

GPIO

El siguiente es el Pin# GPIO y Pin Map.

Pin name	Pin map		Pin name
	1 (3.3V)	2 (5.0V)	
	3 (I2C)	4 (5.0V)	
	5 (I2C)	6 (GND)	
7	7	8 (UART)	
	9 (GND)	10 (UART)	
11	11	12	12
13	13	14 (GND)	
15	15	16	16
	17 (3.3V)	18	18
19	19	20 (GND)	
21	21	22	22
23	23	24	24
25	25	26	26
	27 (I2C)	28 (I2C)	
29	29	30 (GND)	
31	31	32	32
33	33	34 (GND)	
35	35	36	36
	37 (ADC)	38 (1.8V)	
	39 (GND)	40 (ADC)	

Fig. 02 - Pin map GPIO

La tabla de mapeo anterior se basa en la wiki de: **https://bit.ly/37dXwFi**.

Primero, debe obtener PeripheralManager. Puede obtener una instancia de GPIO y una lista disponible de GPIO desde la instancia del gestor.

```
import
com.google.android.things.pio.PeripheralManager;
import com.google.android.things.pio.Gpio;
…
PeripheralManager manager =
PeripheralManager.getInstance();
```

Puedes obtener una lista GPIO disponible a través del método getGpioList. Este método proporciona una lista de nombres GPIO disponible. De este modo los puedes seleccionar de la lista para utilizarlos. Cada pin tiene un nombre que proviene de un número de pin físico. Sí, el nombre del pin GPIO es el número de pin. Puede obtener una instancia GPIO a través del método openGpio con el nombre del pin como parámetro.

```
List gpioList = manager.getGpioList();

Gpio gpio = manager.openGpio(gpioList.get(0));
// or Gpio gpio = manager.openGpio("7");
```

En este ejemplo, te mostraré cómo usar un pin GPIO como salida. En el ejemplo, quiero usar el pin #7 como salida y si presiono el botón en mi aplicación, se encenderá un LED que está conectado al pin #7 GPIO. Al igual que antes, después de obtener una instancia GPIO, puedes fijar la dirección E/S del pin GPIO. Puede configurar la dirección mediante el método setDirection y los valores de dirección son DIRECTION_IN, DIRECTION_OUT_INITIALLY_HIGH y DIRECTION_OUT_INITIALLY_LOW. Elegí DIRECTION_OUT_INITIALLY_LOW para que el valor GPIO sea low.

Luego puedes establecer el valor mediante el método setValue. Si deseas que el valor de salida sea high o 1, necesitas pasar el parámetro True o puede pasar el parámetro false como low o 0. En este ejemplo, obtengo información desde botón de la aplicación. De este modo, cuando haces clic en el botón, se ilumina un LED.

```
gpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW
);
Switch gpioSwitch = find
ViewById(R.id.gpio_switch);

gpioSwitch.setOnClickListener(new
View.OnClickListener() {
  @Override
  public void onClick(View v) {
    try {
      Switch gpioSwitch = (Switch) v;
      if (gpioSwitch.isChecked()) {
        gpio.setValue(true);

      } else {
        gpio.setValue(false);
      }
    } catch (IOException io) {
      io.printStackTrace();
    }
  }
});
```

Android Things también proporciona otros métodos como getValue, setActiveType, setEdgeTriggerType y registerGpioCallback. Puede aprenderlos en la página web oficial. Sin embargo, el ODROID aún no proporciona registerGpioCallback correctamente. En particular, la configuración de devolución de llamada utilizando el controlador no se ha implementado todavía. Espero que lo hagan pronto.

Referencia al metodo GPIO - https://bit.ly/2tXHUHI.

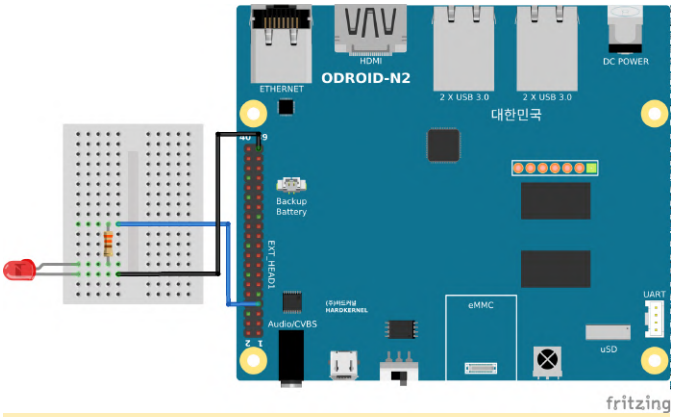


Fig. 03

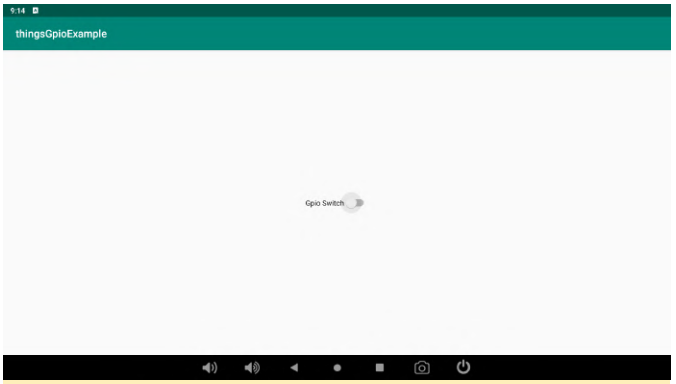


Fig. 04

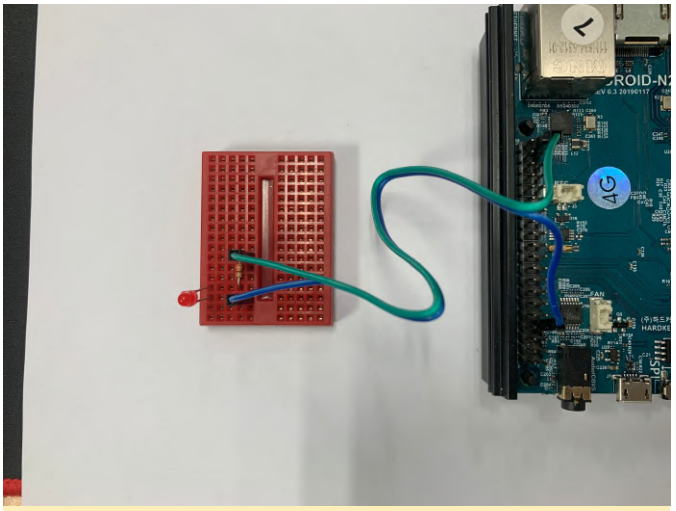


Fig. 05

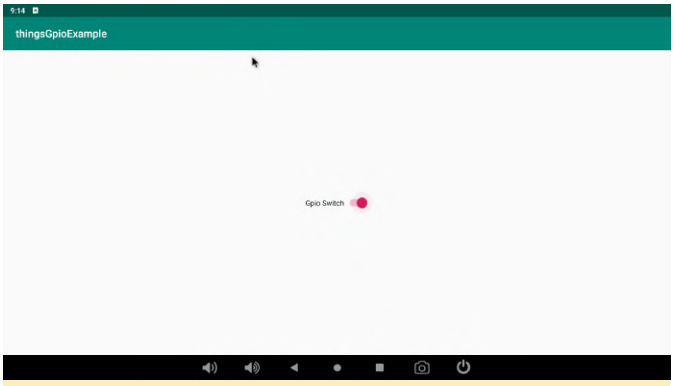


Fig. 06

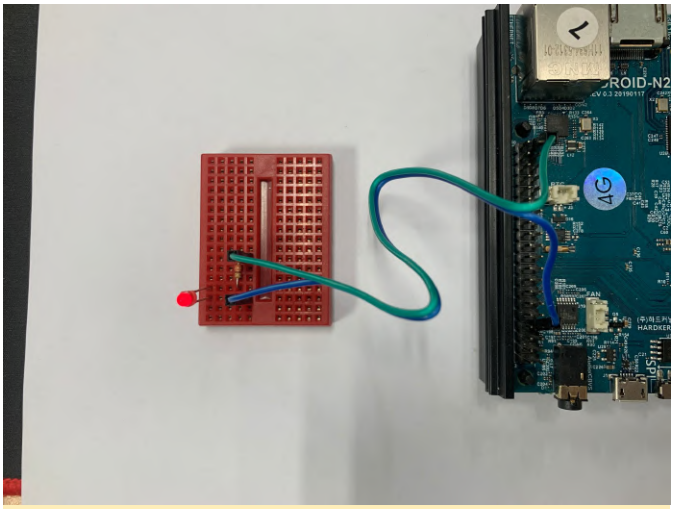


Fig. 07

Puedes controlar I2C y PWM revisando un ejemplo de mi github. Además, puedes obtener información sobre cada API periférica desde el sitio web.

I2C - https://developer.android.com/things/sdk/pio/i2c
PWM - https://developer.android.com/things/sdk/pio/pwm

¡Espero que esto te ayude a utilizar mejor tus periféricos de Android!

Referencias

https://developer.android.com/things
https://forum.odroid.com/viewtopic.php?f=178&t=37101