Taller de Proyecto II

2021

Informe de Avance

N10 - Clúster de Raspberry Pi’s con Kubernetes y Docker

Grupo de Desarrollo

- Cao, Agustin Leonardo - 1593/9
- Fausto, Simon Passerini - 1002/2

Informe de Avance 14/10/19

1.- Proyecto

El proyecto consiste en la creación de un clúster de kubernetes el cual permita correr aplicaciones dentro de contenedores de manera eficiente y balanceada entre sus nodos. Cada nodo está compuesto por una SBC Raspberry Py 3B+. Como prueba, se lanzarán dos aplicaciones contenerizadas, donde una obtendrá mediciones desde un sensor, y la otra mostrará la última lectura desde una página web simple.
El sistema propuesto por este proyecto despliega un clúster de Kubernetes para facilitar y, en la medida de lo posible, automatizar la administración y escalabilidad de las aplicaciones que corren sobre el servidor.

2.- Materiales y Presupuesto

Cantidad	Item	Precio unidad[1]	Valor[1]
2	Raspberry Pi 3B+ [2]	$ 7160	$ 14,320.00
2	memoria flash USB 32GB [2]	$ 500	$ 1,000.00
2	fuente de alimentación 3.6A USB[2]	$ 880	$ 1,760.00
1	sensor medidor de temperatura y humedad DHT11[2]	$ 290	$ 290.00
2	cable micro-USB	$ 450	$ 900.00
2	cable UTP Cat 5e 1.2m	$ 199	$ 398.00
1	router TpLink TL-WR741ND v4.22[3]	$ 1600	$ 1,600.00
1	conector múltiple eléctrico de 4 tomas	$ 890	$ 890.00
Total	-	-	$21,158.00

[1]En pesos argentinos. Los valores provienen de los sitios web https://www.mercadolibre.com.ar/ y https://ar.mouser.com/ buscados en la fecha 13/10/2021. En caso de ser productos importados, incluyen todos los impuestos hasta la fecha. No se incluyen gastos de envío ni transporte.
[2] Provisto/financiado por la cátedra.
[3] Reutilizado de otro proyecto.

3.- Esquema Gráfico del Proyecto

A partir de la configuración pre impuesta sobre las Raspberry Pi's, se decidió optar por el uso de una red cableada. Se impuso una IP estática en toda la interfaz eth0, permitiendo así la independencia entre el clúster y la estructura de la red si dichas direcciones IP estuvieran disponibles. La configuración está representada en la Fig.1. Dicha configuración permite al sistema funcionar en cualquier red cuya IP de Origen sea 192.168.0.1 y en las que las direcciones IP 192.168.0.201 y 192.168.0.202 se encuentren disponibles.

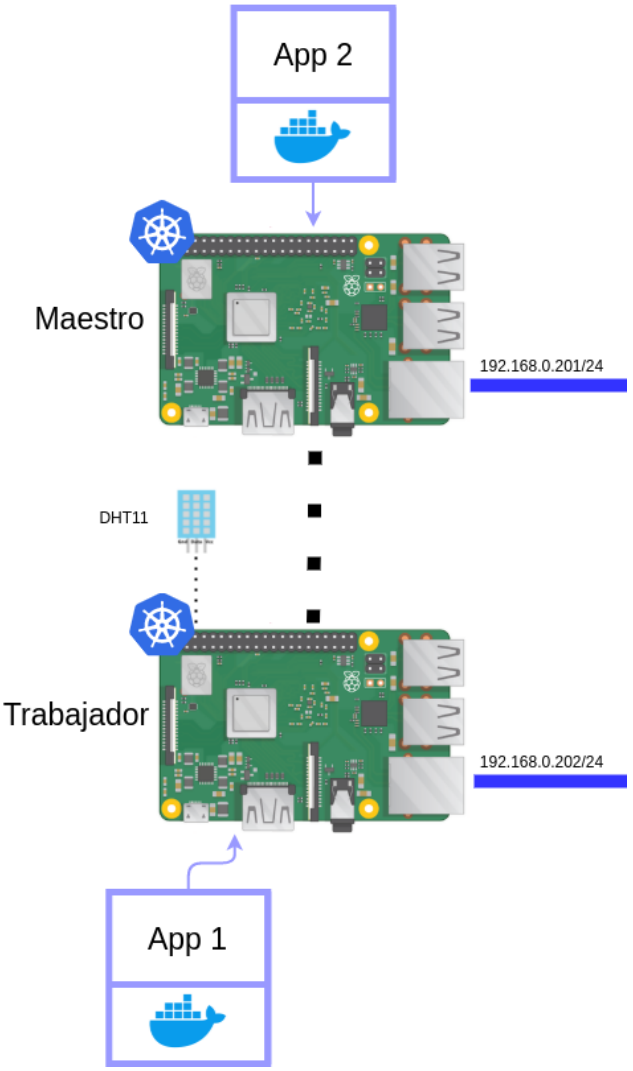

Fig.1: Red Independiente

La estructura de red actual en la que se ha instalado el clúster es como se muestra en la Fig. 2. El clúster está conectado a un conmutador de red con capacidades inalámbricas, que proporciona una conexión entre el clúster y un enrutador de red doméstica.

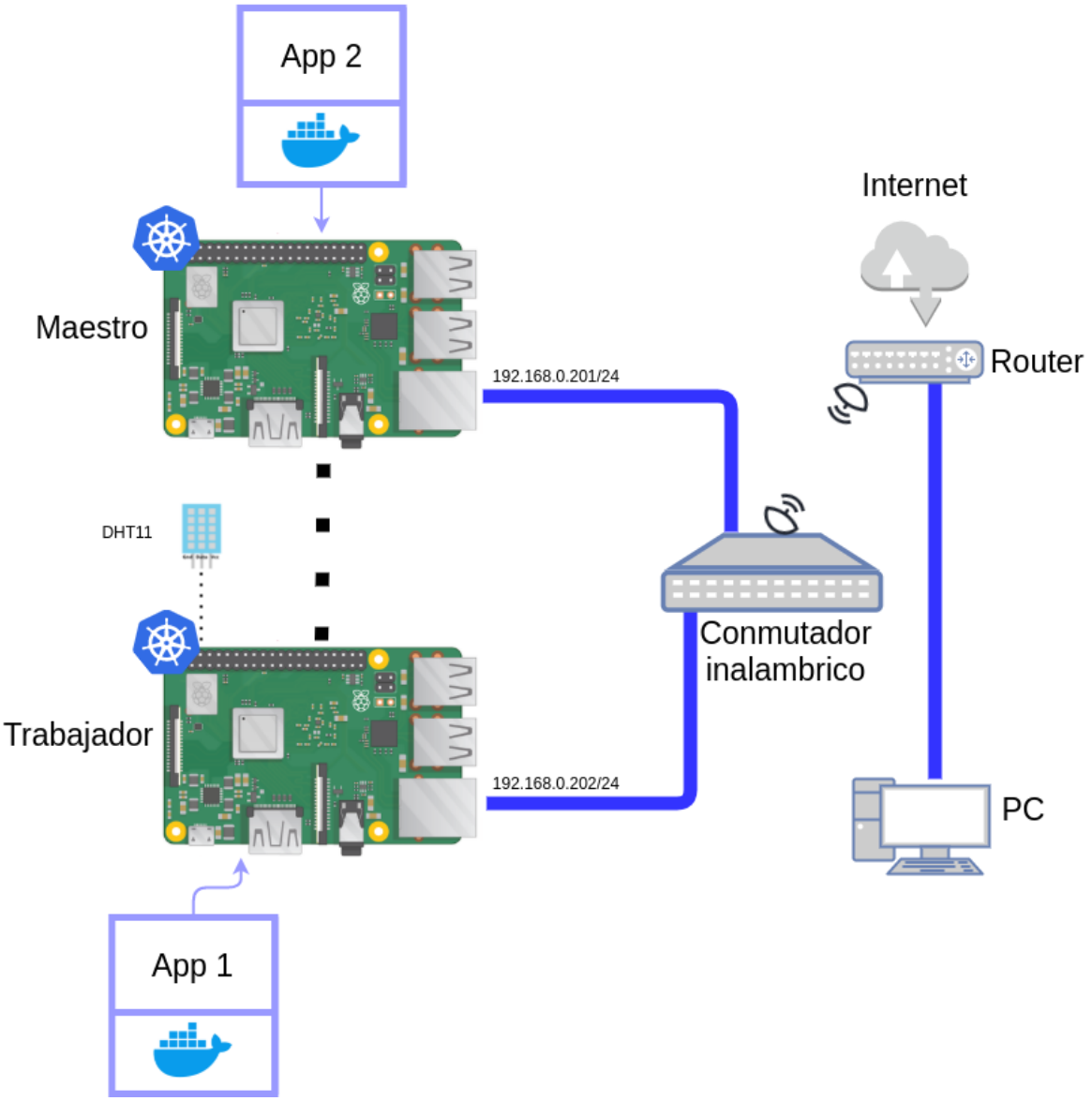

Fig. 2: Red actual.

4.- Grado de Avance, Problemas y Soluciones

En primer lugar, se procedió a investigar las tecnologías de contenedores y orquestación de contenedores, específicamente las más usadas en la industria: Docker y Kubernetes.

Docker

Docker es una plataforma open source que provee herramientas para el desarrollo y despliegue de aplicaciones, separándolas de la infraestructura sobre las que corren. En líneas generales, consiste en el aislamiento de una unidad de software en un contenedor, proceso al que se le llama dockerizar. Este contenedor es abstraído en una imagen, que luego puede ser instanciada la cantidad de veces que se necesite en el servidor donde vaya a correr. Para manejar varias instancias de contenedores se utilizan sistemas de administración, uno de los cuales se describe en la sección siguiente.
Otra gran ventaja de docker es Docker Hub, un sistema que centraliza imágenes de contenedores y permite tanto subirlas como bajarlas con gran facilidad. A fines de este proyecto, fue de gran utilidad para desplegar una base de datos sin necesidad de instalar y configurar un motor manualmente.
El sistema propuesto en este proyecto brinda servicios a aplicaciones dockerizadas para facilitar el desarrollo, despliegue y, gracias a Kubernetes, escalamiento del software que se corre.

Kubernetes

Kubernetes es un sistema open source para la automatización del manejo de contenedores, con un enfoque muy fuerte en lo que es la escalabilidad.
Los containers se agrupan en unidades lógicas (por ejemplo, una aplicación más su base de datos), llamadas Pods. Estos, a su vez, se alojan en las máquinas que las corren, ya sean físicas o virtuales, llamadas Nodes. Todos los nodes están coordinados por el Control Pane, una capa de administración que expone una API para supervisar y controlar el estado del Cluster. Cabe destacar que dentro de las funcionalidades del control pane yace un cliente que da funcionalidades de integración a servicios en la nube como AWS y Azure.

Cada nodo se puede duplicar ad infinitum, y haciendo uso de una herramienta automática llamada Load Balancer el sistema intenta que la carga sobre el servidor se distribuya equitativamente de acuerdo a la capacidad de procesamiento de cada trabajador.

Dentro del cluster se pueden configurar n cantidad de redes, cada nodo perteneciendo de 0 a n de ellas. No hay relación entre las direcciones y puertos internos, y los externos, por lo que el direccionamiento interno no tiene necesidad de ser consistente con el exterior.

Actualmente la red propuesta se encuentra configurada y en estado funcional (Fig. 3). Como conmutador inalámbrico se utilizó un router reciclado, el cual fue configurado en modo WDS Bridge a fin de lograr la conexión del clúster con el resto de la red doméstica.

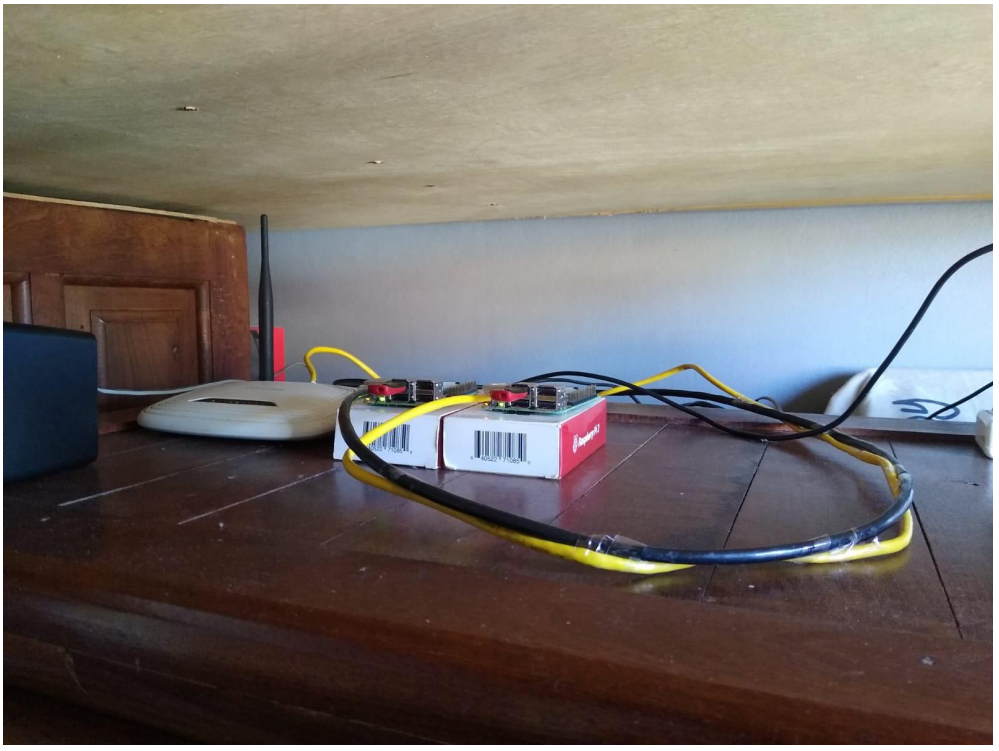

Fig. 3: Sistema Montado.

Una vez logrado esto, se procedió a realizar pruebas de práctica con contenedores en un ambiente separado del clúster con la finalidad de prevenir fallas en caso de que algo saliese mal. Una mala administración de imágenes y contenedores típicamente resulta en pérdida de almacenamiento, por lo que decidimos realizar algunas pruebas en Docker dentro de otra plataforma antes de operar sobre el clúster.

La instalación de Docker para realizar las pruebas no supuso dificultad al utilizar sistemas operativos basados en Debian o Red Hat. La misma consistió en utilizar un script provisto por Docker en get.docker.com (más detalles sobre el proceso en la bitácora). Sin embargo, el mismo no tiene soporte para distribuciones basadas en Arch Linux. En ese caso, la mejor solución es valerse de los repositorios AUR.

Gracias a las pruebas realizadas, se obtuvo un conocimiento práctico respecto al funcionamiento de Docker, “Docker Hub” y la creación de imágenes con programas personales utilizando el llamado “Dockerfile”.

También se obtuvo experiencia en el manejo de redes de contenedores con Docker y el manejo de puertos. De esta forma, se consigue crear redes virtuales de contenedores que posteriormente pueden interactuar con sistemas fuera de la red, utilizando únicamente la dirección IP del host.

La primera prueba realizada sobre el clúster consistió en lanzar un contenedor con un servidor de Nginx y comprobar la conexión del clúster desde una computadora dentro de la misma red. La misma resultó ser un éxito gracias a la investigación realizada previamente.

Actualmente, se encuentra en investigación el proceso para utilizar aplicaciones dentro de contenedores con acceso a los terminales de entrada y salida de una Raspberry Pi. Dado que los contenedores no son más que un programa que corre en un ambiente aislado del resto de los programas, suponemos que la solución a esta inquietud solo se reduce a una configuración especial del "Dockerfile". Por otro lado, como se dispone de un único sensor, solo es posible conectar una única Raspberry Pi con el mismo. A raíz de esto surgió la siguiente duda: ¿Qué ocurre si la aplicación del clúster encargada de leer el sensor no se encuentra funcionando en la SBC conectada al sensor? La misma permanece en incógnita. Suponemos que la aplicación encargada de la lectura debe estar siendo ejecutada sobre aquella SBC conectada al sensor. Una solución prematura consistiría en crear una aplicación que corra continuamente en una SBC especifica (la conectada al sensor) y que los contenedores hagan solicitudes a la misma. Otra sería establecer que el contenedor cuya imagen corresponda a la lectura del sensor sea forzado a correr sobre aquel nodo conectado al mismo.

La muestra de la información por parte del sensor debe hacerse desde una página web accesible desde la misma red. No suponemos que la misma suponga demasiada dificultad en cuanto a interfaz. Sin embargo, debemos evaluar cómo comunicar dos aplicaciones contenerizadas, y si existen diferencias en cuanto a comunicar contenedores utilizando Docker y comunicar pods utilizando kubernetes más adelante.

Respecto al uso de kubernetes, nos encontramos evaluando las diferentes opciones disponibles. Existen varios sistemas para orquestación de contenedores, de los cuales podemos destacar K8s como la más conocida de todas. Sin embargo, K8s resulta ser una herramienta demasiado compleja con un gran consumo de recursos, la cual podría traer problemas a nuestro clúster en etapas más avanzadas del proyecto.
Para ello, hemos optado por utilizar una versión que no demande tantos recursos. Entre las opciones vistas, K3s, una *"distribución de kubernetes de alta disponibilidad certificada diseñada para cargas de trabajo en sistemas independientes, de bajos recursos, ubicaciones remotas o dentro de aplicaciones IoT … Optimizado para ARM"* [4], parece ser la opción que mejor se adecua a nuestro proyecto. Queda por evaluar su implementación en el sistema.

Bitácora: https://github.com/tpII/2021-n10-kubernetes-rpi/wiki/Bit%C3%A1cora

[4] Lightweight Kubernetes - https://k3s.io/ y https://rancher.com/products/k3s/ | Rancher Labs

5.- Documentación en Formato Gráfico y Video

Video complementario:
https://drive.google.com/file/d/1Rxck_gs9F4YB8IOGzR7LYn9XADmjB5DP/view?usp=sharing