

Taller de Proyecto II

Ingeniería en Computación

Informe final

Monitoreo de Raspberry Pi con Prometheus y Grafana PS.4-2

Arreche Cristian - 01515/4

Blasco Federico - 01678/4

07 de diciembre de 2020

Facultad de Informática
Universidad Nacional de La Plata



1. Proyecto

El proyecto desarrollado se basa en la recolección y análisis de métricas de múltiples Raspberry Pi (RPi). El enfoque principal del proyecto es la configuración y uso de las herramientas de software Grafana y Prometheus.

El objetivo principal del proyecto es poder monitorear varias RPi en tiempo real, donde se pueda obtener información de uso y recursos de hardware de cada una de ellas. Esta información es representada de manera agradable mediante la utilización de gráficos y paneles, donde también se podrían configurar alertas y/o notificaciones de las métricas en base a eventos definidos.

Algunas de las métricas monitoreadas son:

- Uso de la CPU.
- Memoria RAM disponible.
- Memoria RAM total del sistema.
- Uso de la red.
- Espacio disponible en el disco
- Tamaño total del disco

Objetivos alcanzados:

- Se instaló y configuró un contenedor de docker-compose para los servicios de monitoreo y análisis en la máquina servidor.
- Se instalaron las herramientas Grafana y Prometheus en el contenedor de docker en la máquina servidor para no perder performance de las RPi.
- Se simularon las máquinas virtuales mediante el software VirtualBox.
- Se instaló la herramienta node-exporter en cada una de las RPi, para exportar sus métricas.
- Se configuró la herramienta Prometheus para que tome las métricas exportadas por node-exporter en las RPi a través de LAN.
- Se configuró la herramienta Grafana para que utilice las métricas obtenidas por Prometheus.
- Se configuraron paneles y gráficos en Grafana para la visualización de las métricas.

2. Materiales y presupuesto

Se trata de un proyecto casi exclusivamente de software, donde las herramientas de software a utilizar son gratuitas. Para el desarrollo del proyecto no es necesario ningún elemento de hardware adicional provisto por la cátedra. De la misma manera, tampoco es necesario la suscripción de servicios en la nube.

De este modo, no se necesita ningún tipo de presupuesto, el desarrollo del proyecto se puede llevar a cabo de una forma totalmente gratuita. El único hardware necesario son las computadoras de ambos integrantes del grupo que desarrollan el proyecto.

Entonces, el hardware a utilizar será:

- Una PC (PC1) con el software VirtualBox donde se simularán las RPi.
- Una PC (PC2) usada como servidor con los servicios de Prometheus y Grafana.

Entonces, el hardware simulado serán varias RPi, no solo por el costo que conlleva comprar varias RPi, sino también por la facilidad para compartir los avances y trabajar a distancia entre ambos integrantes del grupo. De esta manera, más allá de no contar con el hardware, se puede lograr una buena aproximación de cómo debería funcionar en el hardware real.

3. Descripción del proyecto

A continuación se explican brevemente las herramientas utilizadas para el desarrollo del proyecto:

Docker:

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Más información en <https://docs.docker.com/>

Grafana:

Es una herramienta open source para visualizar bases de datos de series de tiempo. A partir de una serie de datos recolectados se obtiene un panorama gráfico de la situación muy simple de interpretar y visualizar, y se utiliza para el monitoreo de infraestructura de IT, mediciones de aplicaciones, control de procesos, entre otros usos.

Más información en <https://grafana.com/docs/grafana/latest/>

Prometheus:

Es una aplicación de software libre utilizada para el monitoreo de eventos y alertas, la cual obtiene y almacena métricas en tiempo real en una base de datos de series de tiempo. Tiene una interfaz gráfica accesible a través del navegador web y a través de la cual se pueden hacer consultas en la base de datos.

Más información en <https://prometheus.io/docs/introduction/overview/>

Node exporter:

Es una herramienta que expone una gran variedad de métricas relacionadas con el hardware y el kernel de un sistema. Estas métricas son obtenidas por el software Prometheus y almacenadas en una base de datos para su posterior utilización.

Más información en https://github.com/prometheus/node_exporter

El proyecto se realizó siguiendo el diseño del esquema de la Figura 1, el mismo esquema fue presentado en el informe de avance y no tuvo modificaciones.

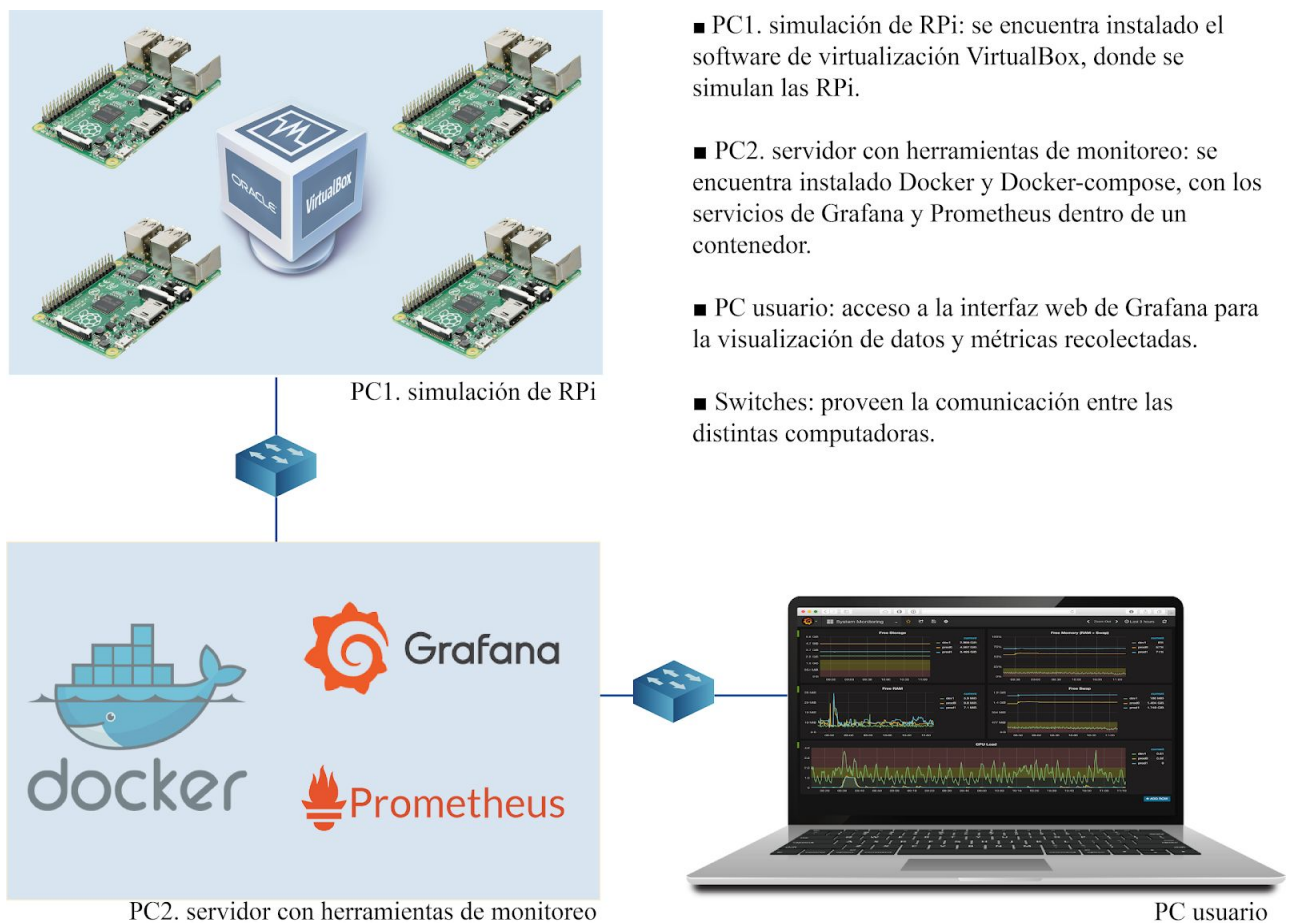


Figura 1. Esquema gráfico del proyecto

Simulación de las RPi

En este proyecto fue simulado el subsistema compuesto por las RPi, a fin de que sea mucho más simple que ambos integrantes del grupo trabajen sobre una misma plataforma al contar con la portabilidad de las instancias simuladas, además del costo económico que generaría tener que comprar más de una RPi.

En esta ocasión, ambas RPi se simularon con el software VirtualBox, con el cual ya habíamos trabajado con anterioridad, lo que no implicaba una complejidad extra estudiar su utilización. Primero se generó una instancia de la MV con el sistema operativo Raspberry Pi OS y se le instaló la herramienta node-exporter, para que se puedan exportar sus métricas hacia Prometheus, así como también se configuraron los puertos de la MV para que se pueda acceder al servicio de manera remota. Luego, esta instancia fue clonada para simular más de una RPi.

Las diferencias con el sistema físico son mínimas, ya que la instalación y configuración de las herramientas no varían en absoluto. Una posible diferencia se puede dar al visualizar los valores de las métricas en Grafana, ya que la utilización de CPU, memoria RAM total, espacio en disco, etc, probablemente varíen con respecto a la utilización del hardware real. De esta manera, si se contara con el sistema físico real, todas las instalaciones y/o configuraciones realizadas son totalmente válidas y no se requeriría de grandes cambios, solo constaría de la conexión física del hardware y realizar la misma configuración establecida en las instancias de la MV.

Para ver los avances de manera cronológica, y más capturas de los ítems mencionados anteriormente, visitar la Wiki de trabajo en:

<https://github.com/tp11/2020-ps-4.2-prometheus-grafana/wiki/Bit%C3%A1cora>

4. Guía de instalación: Proyecto y ambiente de desarrollo

Ambiente de desarrollo

El proyecto no cuenta con código fuente, sino con archivos de configuración “.yaml” para el despliegue de las herramientas de monitoreo Prometheus y Grafana. A continuación, se encuentran los 2 archivos de configuración utilizados:

```
# docker-compose.yml
version: "2"
services:
  grafana:
    image: grafana/grafana:latest
    container_name: monitoring_grafana
    ports:
      - "3000:3000"
    volumes:
      - ./data/grafana:/var/lib/grafana
    links:
      - prometheus
    restart: always

  prometheus:
    image: prom/prometheus:latest
    container_name: monitoring_prometheus
    volumes:
      - ./data/prometheus/config:/etc/prometheus
    command:
      - '--config.file=/etc/prometheus/prometheus.yml'
    ports:
      - "9090:9090"
    restart: always
```

A su vez, para la comunicación con el exportador de métricas de Prometheus, es necesario un segundo archivo de configuración:

```
# prometheus.yml
global:
  scrape_interval: 5s          # scrape targets
  evaluation_interval: 5s      # scrape targets
  external_labels:
    monitor: 'my-project'
rule_files:

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['192.168.0.6:9100']
        labels:
          hostname: Raspi-1
      - targets: ['192.168.0.6:9101']
        labels:
          hostname: Raspi-2
```

Como se puede observar, en la sección “scrape_configs”, se especifican las IP de las RPi a monitorear.

Proyecto

El proyecto puede dividirse en dos secciones, el Servidor por un lado y las Raspberry Pi simuladas por el otro, las cuales van a ser explicadas por separado:

RPi simuladas

El primer paso antes de comenzar el proyecto es simular las RPi con el software VirtualBox, para esto se deberán hacer los siguientes pasos:

1. Descargar VirtualBox desde la página oficial (<https://www.virtualbox.org/wiki/Downloads>) e instalarlo.
2. Descargar la imagen de Raspberry Pi OS de la página oficial (<https://www.raspberrypi.org/software/>)
3. Crear una máquina virtual
 - a. El nombre de la máquina no interesa
 - b. La cantidad de memoria RAM recomendada es de 1024MB, que es la cantidad de RAM que tiene una Raspberry Pi 3
 - c. Crear un disco duro virtual de mínimo 8GB
4. Iniciar la máquina virtual con la imagen de Raspberry Pi OS
5. Instalar Raspberry Pi OS con toda su configuración por defecto

Luego de tener creada la máquina virtual, se debe configurar la misma para funcionar con la herramienta Node_exporter.

Para esta sección, se utilizó como base la siguiente guía: <https://medium.com/@prashant.vats/how-to-install-node-exporter-on-ubuntu-16-04-a087496767ed>

1. Configurar los puertos de la máquina virtual en la sección port forwarding de la configuración de red de la siguiente manera:

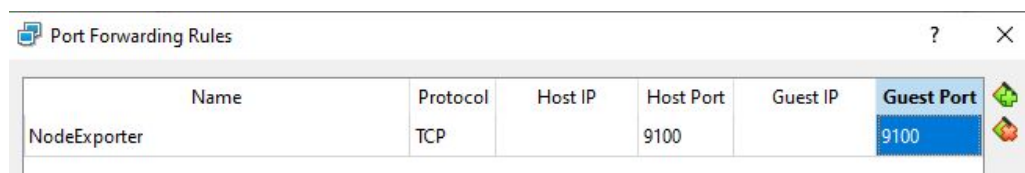


Figura 2. configuración de puertos de la Máquina Virtual

Así el puerto 9100 de la máquina virtual estará vinculado con el puerto 9100 de la máquina real, esto nos permitirá acceder al servicio desde el servidor con Prometheus.

2. Descargar node exporter en su última versión para intel 386 desde la página oficial (https://github.com/prometheus/node_exporter/releases)

3. Extraer el tarball con el siguiente comando: `tar -xvf nombreadarchivo.tar.gz`

4. Mover el ejecutable al directorio `/usr/local/bin`:

```
sudo mv node_exporter-0.18.1.linux-amd64/node_exporter
/usr/local/bin/
```

5. Crear un servicio de node_exporter usando systemd:

```
sudo vi /etc/systemd/system/node_exporter.service
```

Dentro del archivo generado se debe copiar el siguiente código, cambiando el usuario y grupo por el que corresponda:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=pi
Group=pi
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

6. Iniciar el servicio node_exporter con los siguientes comandos:

```
sudo systemctl daemon-reload
sudo systemctl start node_exporter
```

7. Chequear el estado del servicio para ver si funciona correctamente:

```
sudo systemctl status node_exporter
```

8. Habilitar el servicio node_exporter para que inicie automáticamente al prender la máquina virtual:

```
sudo systemctl enable node_exporter
```


Si se quiere tener más de una RPi simulada, simplemente se debe clonar la máquina virtual con `node_exporter` funcionando, y en la configuración de red de VirtualBox cambiar el puerto del host por otro que esté disponible (el puerto 9100 ya estará ocupado por la primer RPi por ejemplo), para que la herramienta Prometheus pueda acceder a cada una de las RPi simuladas correctamente. También se debe modificar la sección de targets del archivo *prometheus.yml* para que tome todas las RPi simuladas.

Para la realización de este informe, se utilizaron dos RPi, llamadas “Raspi-1” y “Raspi-2”, las cuales utilizan los puertos 9100 y 9101, respectivamente.

Servidor

El servidor será el encargado de contar con las herramientas de monitoreo necesarias para la visualización de las métricas, estas serán instaladas en un contenedor de Docker.

Antes de comenzar con la instalación, se recomienda aprender las nociones básicas de Docker y Docker-compose, así como también el manejo de imágenes y contenedores, para poder comprender el funcionamiento del sistema y la configuración del mismo. Los pasos a seguir para su instalación serán utilizando el SO Linux:

1. Instalación de Docker-compose desde la página oficial (<https://docs.docker.com/compose/install/>)
2. Instalación de las herramientas de monitoreo:
 - a. Crear un archivo, en el directorio que se desee, con el nombre *docker-compose.yml* con el contenido del archivo mostrado en la sección “Guía de instalación/Ambiente de desarrollo”
 - b. Crear un archivo en la ruta relativa al directorio anterior *./data/prometheus/config/* con el nombre *prometheus.yml* con el contenido del archivo también mostrado en la sección “Guía de instalación/Ambiente de desarrollo”, cambiando la IP y el puerto de cada RPi por lo que corresponda.
Nota: la ruta mencionada debe proveer de permisos de ejecución.
 - c. Ubicarse en el directorio donde se creó el archivo *docker-compose.yml* y levantar todos los servicios ejecutando `docker-compose up -d`

Luego de realizados estos pasos, ya se puede acceder a la plataforma de Grafana a través de cualquier navegador en `[IP_del_servidor]:3000`, como así también a la herramienta de Prometheus en `[IP_del_servidor]:9090` y comenzar a trabajar con la configuración y monitoreo de las métricas.

En este caso, debido al contexto de distanciamiento social, la realización del proyecto se llevó a cabo a distancia entre ambos integrantes del grupo. Por lo que fue necesario abrir los puertos del router donde se encuentran instaladas las herramientas de monitoreo, de manera que sea posible trabajar ambos integrantes en la configuración de las métricas y tableros de Grafana.

5. Problemas y soluciones

Conexión por SSH

Uno de los problemas encontrados fue a la hora de establecer la comunicación mediante el protocolo SSH. Hubo que investigar de qué manera abrir puertos en el software VirtualBox, y luego de tener los puertos de las máquinas virtuales asociados a puertos de la máquina host, se tuvieron que abrir los puertos del router para que se pueda acceder a las terminales SSH desde fuera de la red. Ya que, sin esto, no habría manera de ingresar a las máquinas virtuales ambos integrantes del grupo. *(Ver inciso 6, figuras 8, 9 y 10).*

Creación de contenedores Docker

Otro problema menor que se presentó fue al momento de crear los contenedores de Docker, con Docker-compose, ya que no se podían descargar las imágenes de Grafana ni Prometheus. Luego de investigar un poco y tratar de hacer un `docker pull prom/prometheus` se descubrió que el problema era que había que tener una cuenta en Docker hub para poder descargar las imágenes. Luego de crear la cuenta y usar el comando `docker login` no se presentaron más problemas.

Creación del .yaml

Al buscar en internet un archivo .yaml a fin de levantar los servicios de Grafana y Prometheus, la gran mayoría tienen incluidos varios servicios más a modo de ejemplo práctico. Por lo que fue necesario estudiar la estructura y funciones de este tipo de archivos, ya que al tratarse de necesidades y configuraciones tan específicas para cada proyecto, resulta inevitable analizarlo y adaptarlo a las necesidades de cada aplicación. De esta manera, se podrá obtener una instalación limpia de las herramientas, sin problemas de vinculación, compatibilidad, comunicación, etc.

Instalación de Node_exporter

Para la instalación de Node_exporter hubo varios problemas, el primero surgió al investigar la herramienta, ya que en la misma página oficial recomiendan no utilizar docker porque el programa necesita acceso al sistema host.

Más información en: https://github.com/prometheus/node_exporter

Por este motivo se decidió crear un servicio de Node_exporter, el cual arrancaría junto con el inicio del sistema, evitando así utilizar docker.

El segundo problema que se presentó fue a la hora de conectar Prometheus con Node_exporter, ya que al analizar los puertos abiertos, parecía que Node_exporter utilizaba IPv6 para comunicarse, y por este motivo se utilizó la IPv6 de la máquina virtual en la configuración de Prometheus. Luego de hacer algunas pruebas se llegó a la conclusión de que había que usar IPv4 y Prometheus logró obtener las métricas normalmente.

Configuración de alertas y notificaciones

Investigando como hacer las alertas y notificaciones en Grafana, descubrimos que se puede utilizar Slack como canal de notificaciones, y es por eso que decidimos hacerlo de esta manera, creando un servidor de Slack aparte dedicado únicamente a las notificaciones de Grafana. Siguiendo las instrucciones de la siguiente página <https://blog.knoldus.com/setup-alert-notification-channels-in-grafana/>, logramos configurar las notificaciones a través de Slack y confirmar que funcionan viendo que el mensaje de prueba llega correctamente, como se puede ver a continuación:

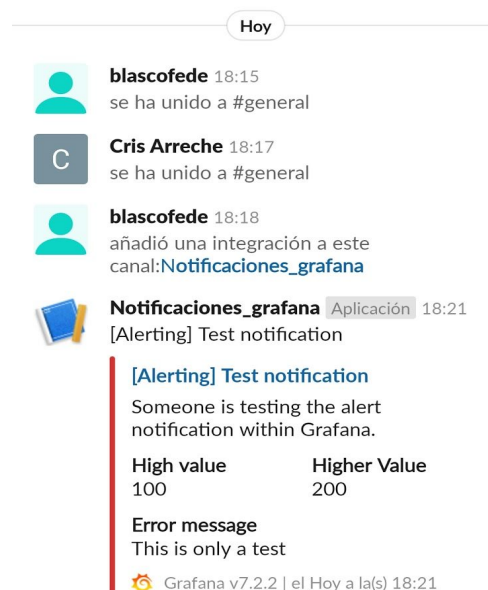


Figura 3. Alerta de prueba en el canal #general de Slack

Al tratar de configurar las alertas nos dimos cuenta que la única manera de hacerlas en Grafana es con los paneles de tipo gráfico, y no pueden utilizar una variable en sus query, mientras que nosotros tuvimos que usar la variable en todas las query, ya que cada una está relacionada a una RPi en específico. Es por este motivo que no le encontramos sentido a utilizar alertas para este proyecto, ya que no se podría configurar ninguna que sea lo suficientemente útil y sería necesario reestructurar toda la organización de los paneles.

No obstante, el canal de alertas y notificaciones de Slack funciona perfectamente, y se podría utilizar en un futuro.

Trabajo futuro

Seguridad

La seguridad del sistema es algo que se podría mejorar mucho, ya que actualmente tanto Grafana como Prometheus funcionan a través del protocolo HTTP, el cual expone toda la información del sistema, incluido el usuario y contraseña utilizado para acceder al mismo.

Esto es una falla de seguridad grave, y no recomendamos dejar los puertos del router abiertos con los servicios funcionando por un período prolongado de tiempo. Se hizo de esta manera para que ambos integrantes puedan acceder al sistema de manera remota en un contexto de aislamiento social por la pandemia.

Implementación en hardware real

Uno de los cambios que hubo que hacer, debido al aislamiento social, fue el de utilizar máquinas virtuales para simular las RPi. Esta es una aproximación muy buena, ya que la instalación del sistema es exactamente la misma, pero sería interesante probar el sistema en el hardware real, para ver la variación de las métricas y su comportamiento al utilizarlo con diferentes programas o servicios.

Agregar más métricas útiles

Node_exporter tiene muchas métricas, con las cuales se pueden configurar gráficos y paneles útiles para el monitoreo de las RPi. En nuestro caso, algunas de estas no pudieron ser utilizadas debido a que al tener virtualizado el sistema de monitoreo, Node_exporter no brinda algunas métricas como la frecuencia del CPU y la temperatura de algunos recursos de hardware. Si se lleva a cabo la implementación en hardware real, sería interesante agregar monitoreo de métricas de este estilo.

Configuración de alertas y notificaciones

Se podrían configurar alertas con Grafana que notifiquen cada vez que haya un problema potencial, ya sea por falta de recursos como poco espacio de disco, o utilización excesiva de la CPU, por ejemplo. Estas alertas o notificaciones se pueden hacer tanto por mail como por Slack, Discord, o más aplicaciones soportadas por la herramienta.

Como mencionamos anteriormente, esta configuración finalmente no se llevó a cabo debido a que Grafana solo permite el uso de estas para las métricas de tipo gráfico que no utilicen variables en sus Querys.

Se podría buscar una alternativa para la configuración de estas notificaciones o modificar la organización de los paneles para poder implementarlas y que tengan un funcionamiento útil.

6. Documentación en formato gráfico y video

El video con la demostración final del proyecto está disponible para descargar en formato .mp4 en:

https://github.com/tpll/2020-ps-4.2-prometheus-grafana/blob/master/3.Informe%20final/Video_informe_final.mp4

A continuación, se encuentran disponibles algunas capturas de pantalla de diferentes tareas realizadas:

- Inciso 3: Simulación de las RPi

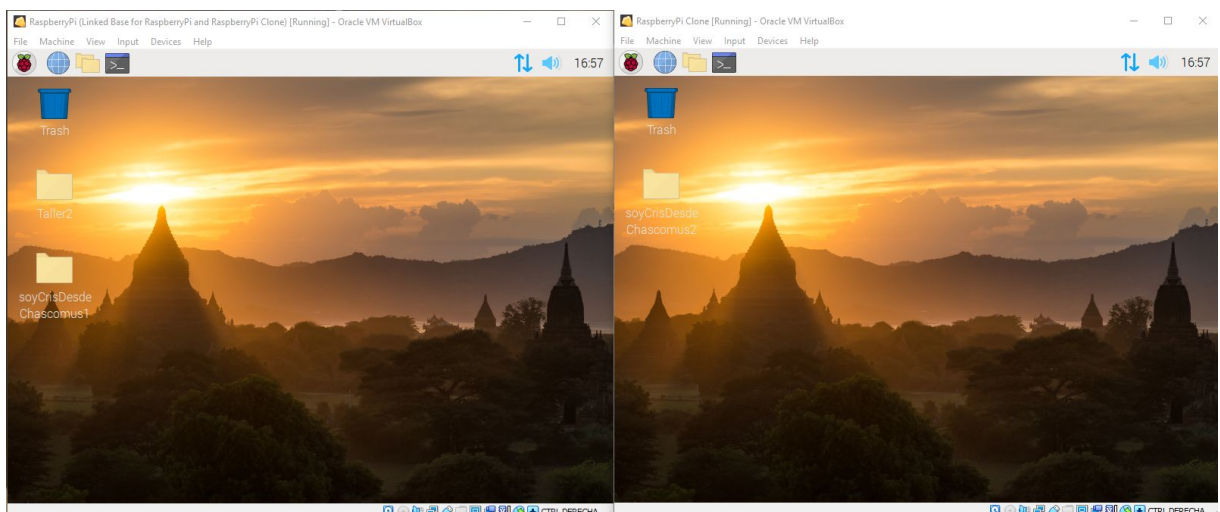


Figura 4. RPi simuladas a través de Máquinas Virtuales

- Inciso 4: Instalación de Docker-Compose

```

/bin/bash
/bin/bash 80x24
federico@UbuntuPeasant ~ $ docker-compose -v
docker-compose version 1.17.1, build unknown
federico@UbuntuPeasant ~ $

```

Figura 5. Instalación correcta de Docker-Compose

- Inciso 4: Instalación de Grafana y Prometheus

```

/bin/bash
/bin/bash 150x38
federico@UbuntuPeasant ~ $ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                    NAMES
bf58b8e650b8   grafana/grafana:latest "/run.sh"              3 days ago    Up 4 hours    0.0.0.0:3000->3000/tcp    monitoring_graf
c880995f073f   prom/prometheus:latest "/bin/prometheus --c..." 3 days ago    Up 4 hours    0.0.0.0:9090->9090/tcp    monitoring_prom
etheus

```

Figura 6. Instalación de los servicios de monitoreo

- Inciso 4: Tableros de Grafana con las métricas de las RPi

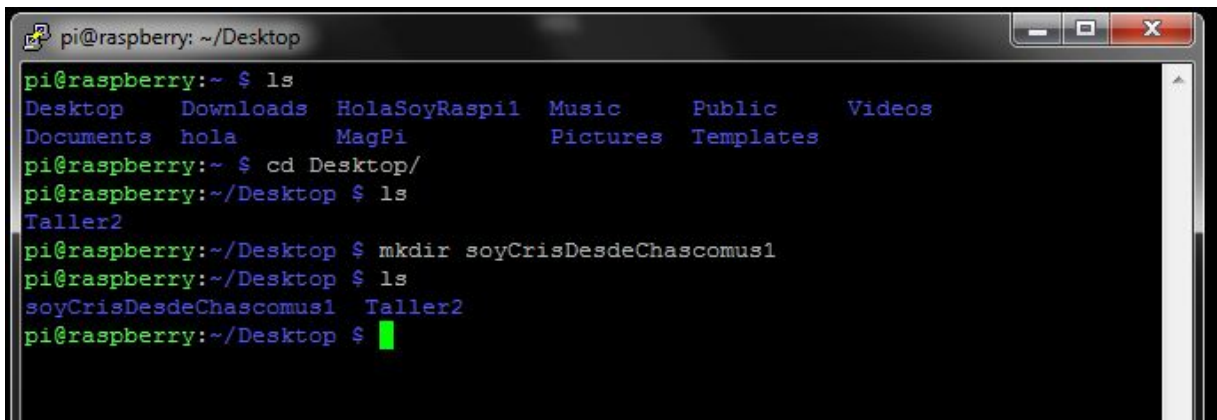


Figura 7. Algunas métricas monitoreadas en grafana

- Inciso 5: Conexión por SSH

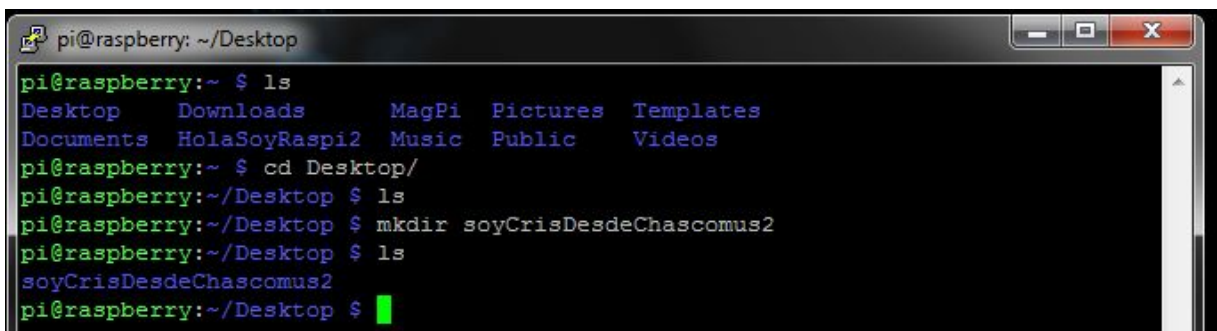
SSH Raspi1	2222	2222		TCP/UDP	2222	2222	192.168.0.6	wanbridge	<input type="checkbox"/>	<input checked="" type="checkbox"/>
SSH Raspi2	2223	2223		TCP/UDP	2223	2223	192.168.0.6	wanbridge	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 8. Puertos abiertos en el router



```
pi@raspberrypi: ~/Desktop
pi@raspberrypi:~$ ls
Desktop  Downloads  HolaSoyRaspi1  Music  Public  Videos
Documents  hola  MagPi  Pictures  Templates
pi@raspberrypi:~$ cd Desktop/
pi@raspberrypi:~/Desktop$ ls
Taller2
pi@raspberrypi:~/Desktop$ mkdir soyCrisDesdeChascomus1
pi@raspberrypi:~/Desktop$ ls
soyCrisDesdeChascomus1  Taller2
pi@raspberrypi:~/Desktop$
```

Figura 9. Conexión por SSH a la máquina virtual 1



```
pi@raspberrypi: ~/Desktop
pi@raspberrypi:~$ ls
Desktop  Downloads  MagPi  Pictures  Templates
Documents  HolaSoyRaspi2  Music  Public  Videos
pi@raspberrypi:~$ cd Desktop/
pi@raspberrypi:~/Desktop$ ls
pi@raspberrypi:~/Desktop$ mkdir soyCrisDesdeChascomus2
pi@raspberrypi:~/Desktop$ ls
soyCrisDesdeChascomus2
pi@raspberrypi:~/Desktop$
```

Figura 10. Conexión por SSH a la máquina virtual 2

7. Conclusiones

El monitoreo de sistemas con las herramientas vistas en este proyecto es relativamente simple de realizar, siempre y cuando se tenga en claro la estructura esquemática planteada, con las conexiones y comunicaciones establecidas, y los archivos de configuración brindados para instalar y configurar correctamente todos los servicios necesarios para el funcionamiento total del sistema. Reuniendo estas características, el potencial obtenido es muy grande, ya que se podría utilizar para monitorear todo tipo de computadoras, como servidores Linux o RPi utilizadas en algún proyecto de IoT.

La utilización de docker-compose para la creación del servidor web y su base de datos hace que crear un servidor desde cero sea muy simple, evitando así problemas de compatibilidad entre sistemas operativos o algún problema de dependencias. Además de esto, una vez que el proyecto está en funcionamiento, las herramientas Prometheus y Grafana permiten exportar su configuración, por lo que tampoco es necesario perder tiempo en configurar el sistema, salvo algunas excepciones como las IP y los puertos utilizados, que varían dependiendo de la red en la que se esté trabajando.

Como los servidores web funcionan en un servidor dedicado, el único programa funcionando en el sistema a monitorear es el de Node_exporter, encargado de exportar las

métricas del mismo, que representa una carga mínima para el sistema. Es por este motivo que esta es una muy buena aproximación al monitoreo de sistemas reales.

Finalmente, por todas las características mencionadas, creemos que la utilización de las herramientas Prometheus y Grafana para el monitoreo de sistemas es una opción muy interesante, ya que su complejidad no es muy grande, brinda portabilidad, una interfaz configurable según las necesidades y gustos del usuario, brindando paneles de control autoexplicativos y fáciles de utilizar.