

# Práctica 4. Benchmarking y Ajuste del Sistema.

---

Pablo Olivares Martínez

## Ejercicio 1.

---

*Una vez que haya indagado sobre los benchmarks disponibles, seleccione como mínimo dos de ellos y proceda a ejecutarlos en Ubuntu y CentOS. Comente las diferencias.*

Para poder realizar los benchmarks requeridos en el ejercicio, accederemos a la página [Phoronix Test Suite](#) y descargaremos el software de Benchmarking tal y como indica su sección de descargas.

**Nota:** Podemos instalar tanto benchmarks individuales, *tests*, como herramientas con gran cantidad de tests, *suites*, que proporcionan muchas facilidades a la hora de realizar los benchmarks. Por esta razón, he decidido instalar *Phoronix Test Suite* con la intención de facilitar el proceso.

Tras dichas aclaraciones, comenzaré con las pruebas en Ubuntu Server y continuaré con las de CentOS:

### 1.1 Phoronix en Ubuntu Server

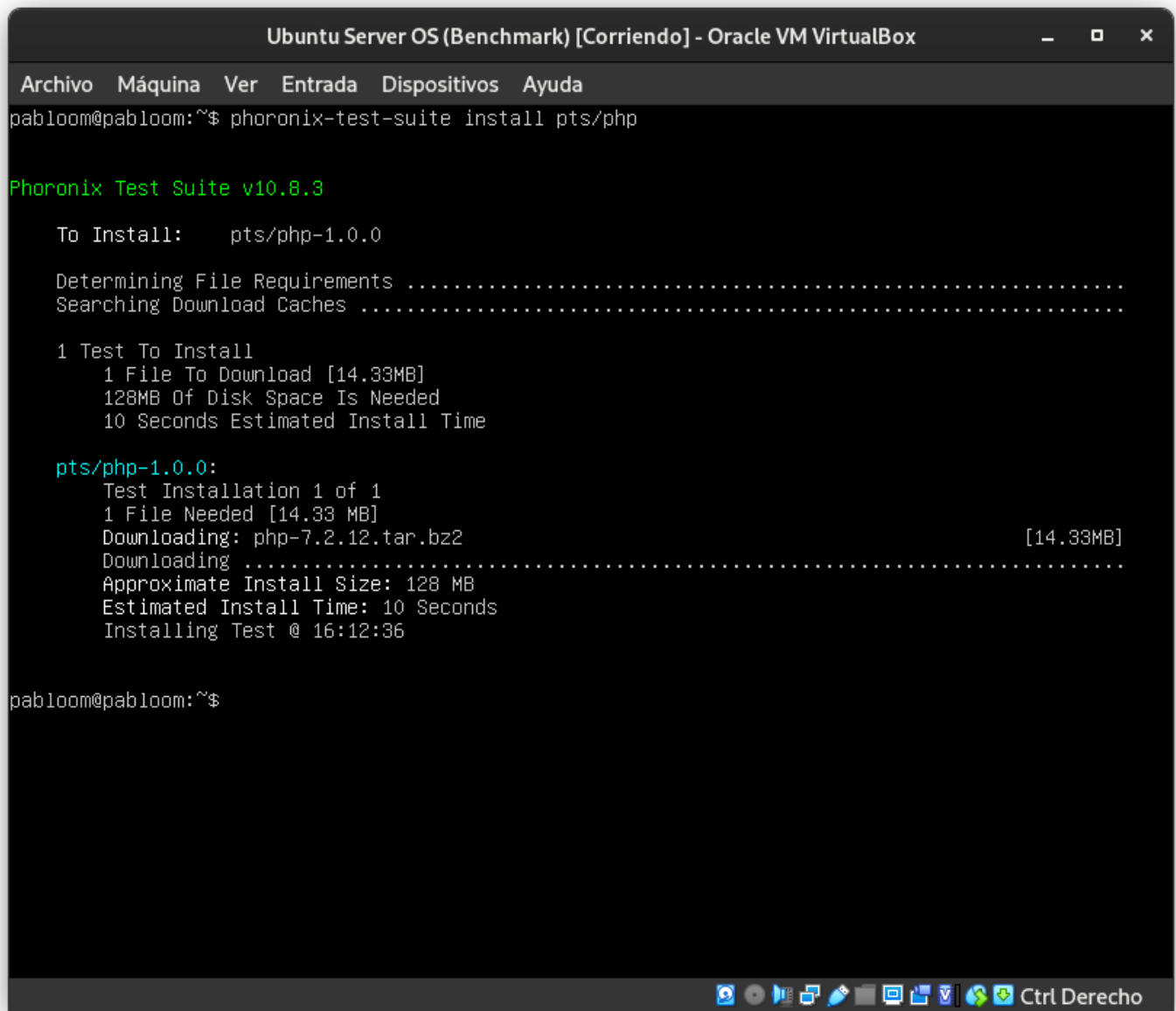
Empezaremos realizando la instalación por terminal en Ubuntu Server:

```
wget http://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-  
test-suite_10.8.3_all.deb  
sudo dpkg -i phoronix-test-suite_10.8.3_all.deb  
sudo apt -f install
```

Tras finalizar la instalación, comprobamos que el programa se ha instalado correctamente simplemente ejecutando en la terminal `phoronix-test-suite`. Dicho comando nos devolverá por pantalla información y parámetros acerca de su uso. Primero nos centraremos en los que nos proporcionan información acerca de los distintos tests disponibles, esto es, `phoronix-test-suite list-all-tests`. Este comando nos proporciona una gran lista de tests que podemos realizar, clasificados por el tipo de prueba que realizan. Estos son *Processor*, *System*, *OS*, *Disk*, *Memory*, *Network* y *Graphics*. Teniendo en cuenta los servicios de los que dispongo en la máquina Ubuntu, he decidido emplear *PHP Micro Benchmarks*, del sistema, y *PyBench*,

también del sistema. Para consultar más información acerca de estos tests, la podemos obtener a través de `phoronix-test-suite info <test>`, donde `<test>` sería en mi caso `pts/php` y `pts/pybench`. Para instalarlos:

```
phoronix-test-suite install pts/php
phoronix-test-suite install pts/pybench
```



```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
pabloom@pabloom:~$ phoronix-test-suite install pts/php

Phoronix Test Suite v10.8.3

To Install:      pts/php-1.0.0

Determining File Requirements .....
Searching Download Caches .....

1 Test To Install
  1 File To Download [14.33MB]
  128MB Of Disk Space Is Needed
  10 Seconds Estimated Install Time

pts/php-1.0.0:
  Test Installation 1 of 1
  1 File Needed [14.33 MB]
  Downloading: php-7.2.12.tar.bz2 [14.33MB]
  Downloading .....
  Approximate Install Size: 128 MB
  Estimated Install Time: 10 Seconds
  Installing Test @ 16:12:36

pabloom@pabloom:~$
```

```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Setting up libasan5:amd64 (9.4.0-1ubuntu1~20.04.1) ...
Setting up python3-pip (20.0.2-5ubuntu1.6) ...
Setting up cpp-9 (9.4.0-1ubuntu1~20.04.1) ...
Setting up binutils-x86-64-linux-gnu (2.34-6ubuntu1.3) ...
Setting up libglx-mesa0:amd64 (21.2.6-0ubuntu0.1~20.04.2) ...
Setting up libglx0:amd64 (1.3.2-1~ubuntu0.20.04.2) ...
Setting up binutils (2.34-6ubuntu1.3) ...
Setting up libgl1:amd64 (1.3.2-1~ubuntu0.20.04.2) ...
Setting up mesa-utils (8.4.0-1build1) ...
Setting up cpp (4:9.3.0-1ubuntu2) ...
Processing triggers for libc-bin (2.31-0ubuntu9.7) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for mime-support (3.64ubuntu1) ...

Phoronix Test Suite v10.8.3

To Install: pts/pybench-1.1.3

Determining File Requirements .....
Searching Download Caches .....

1 Test To Install
  1 File To Download [0.10MB]
  1MB Of Disk Space Is Needed
  2 Seconds Estimated Install Time

pts/pybench-1.1.3:
  Test Installation 1 of 1
  1 File Needed [0.1 MB / 1 Minute]
  Downloading: pybench-2018-02-16.tar.gz [0.10MB]
  Estimated Download Time: 1m .....
  Approximate Install Size: 0.5 MB
  Estimated Install Time: 2 Seconds
  Installing Test @ 16:15:41

pabloom@pabloom:~$
```

Una vez instalados comenzaré ejecutando el de PHP.

```
phoronix-test-suite run pts/php
```

Este tardará un tiempo en ejecutarse y nos preguntará si queremos almacenar los resultados y añadir una descripción. En mi caso tan solo lo he guardado en un fichero bajo el nombre de `php_benchmark` e identifico la configuración realizada en el programa bajo el nombre `phpbenchmark`. Para visualizarlos, basta con ejecutar

```
phoronix-test-suite result-file-to-text phpbenchmark
```

```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
pabloom@pabloom:~$ phoronix-test-suite result-file-to-text phpbenchmark

php_benchmark
Oracle VMware testing on Ubuntu 20.04 via the Phoronix Test Suite.

php_benchmark:

    Processor: Intel Core i5-8300H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
    440FX 82441FX PMC, Memory: 1024MB, Disk: 11GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 8280
    1AA AC 97 Audio, Network: 2 x Intel 82540EM

    OS: Ubuntu 20.04, Kernel: 5.4.0-109-generic (x86_64), Vulkan: 1.1.182, File-System: ext4, Sc
    reen Resolution: 2048x2048, System Layer: Oracle VMware

    PHP Micro Benchmarks
    Test: Zend bench
    Seconds < Lower Is Better
    php_benchmark . 0.612 |=====

    PHP Micro Benchmarks
    Test: Zend micro_bench
    Seconds < Lower Is Better
    php_benchmark . 3.194 |=====

pabloom@pabloom:~$
```

Repetimos el proceso con el benchmark de Python, llamando al archivo `python_benchmark` y a la configuración `pythonbenchmark`:

```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
pabloom@pabloom:~$ phoronix-test-suite result-file-to-text pythonbenchmark
python_benchmark
Oracle VMware testing on Ubuntu 20.04 via the Phoronix Test Suite.

pythonbenchmark:

    Processor: Intel Core i5-8300H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
    440FX 82441FX PMC, Memory: 1024MB, Disk: 11GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 8280
    1AA AC 97 Audio, Network: 2 x Intel 82540EM

    OS: Ubuntu 20.04, Kernel: 5.4.0-109-generic (x86_64), Vulkan: 1.1.182, File-System: ext4, Sc
    reen Resolution: 2048x2048, System Layer: Oracle VMware

PyBench 2018-02-16
Total For Average Test Times
Milliseconds < Lower Is Better
pythonbenchmark . 1953 |=====

pabloom@pabloom:~$ _
```

**Nota:** Cuando disponemos ya de varios resultados almacenados, puede ser complicado recordar el nombre de todos. Para consultarlos, `phoronix-test-suite list-saved-results`.

Tras haber realizado los benchmarks en Ubuntu Server, pasemos a CentOS.

## 1.2 Phoronix en CentOS

Para instalar Phoronix en CentOS podríamos hacer igual que en Ubuntu e instalar el paquete de manera manual. Sin embargo, aprovechando que debido a prácticas anteriores hizo falta incluirlo, haremos uso del repositorio EPEL (*Extra Packages for Enterprise Linux*), el cual dispone del programa deseado. Entonces para instalarlo, tan sólo ejecutaremos el siguiente comando:

```
sudo dnf install phoronix-test-suite
```

Además de proporcionar una instalación más sencilla y actualizada, el gestor de paquetes también se encarga de proporcionar las dependencias necesarias.

Dicho esto, también podemos ejecutar el test inmediatamente tras la instalación mediante el comando `phoronix-test-suite benchmark <test>`. Esto es equivalente a lo realizado anteriormente con `install` y `run`:

```
phoronix-test-suite benchmark php
phoronix-test-suite benchmark pybench
```

Hemos visto anteriormente un comando para consultar los benchmarks realizados a partir del programa. Igualmente, estos son almacenados en formato de texto en `~/.phoronix-test-suite/test-results` si hemos dicho que sí queríamos almacenarlos. De esta forma los tendríamos siempre al alcance en caso de necesitarlos en formato de texto.

Aun así, ejecutamos de nuevo `phoronix-test-suite result-file-to-text <resultadoTest>`:

Archivo Máquina Ver Entrada Dispositivos Ayuda

## Possible Suggestions:

- phpbench [Test]
- pybench [Test]
- pyopencl [Test]
- phpbenchmark [Test Result]

```
[pabloom@localhost ~]$ phoronix-test-suite result-file-to-text phpbenchmark
```

```
php_benchmark
```

```
Oracle VMware testing on CentOS Linux 8 via the Phoronix Test Suite.
```

```
phpbenchmark:
```

```
Processor: Intel Core i5-8300H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
440FX 82441FX PMC, Memory: 818MB, Disk: 9GB VBox HDD, Graphics: VMware SVGA II, Audio: Intel 82801A
A AC 97 Audio, Network: 2 x Intel 82540EM
```

```
OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: xfs, Screen Resolut
ion: 2048x2048, System Layer: Oracle VMware
```

## PHP Micro Benchmarks

```
Test: Zend bench
```

```
Seconds < Lower Is Better
```

```
phpbenchmark . 0.746 |=====
```

## PHP Micro Benchmarks

```
Test: Zend micro_bench
```

```
Seconds < Lower Is Better
```

```
phpbenchmark . 3.893 |=====
```

```
[pabloom@localhost ~]$ _
```

```
CentOS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
440FX 82441FX PMC, Memory: 818MB, Disk: 9GB VBOX HDD, Graphics: VMware SUGA II, Audio: Intel 82801A
A AC 97 Audio, Network: 2 x Intel 82540EM

OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: xfs, Screen Resolut
ion: 2048x2048, System Layer: Oracle VMware

PyBench 2018-02-16
Total For Average Test Times
Milliseconds < Lower Is Better
pythonbenchmark . 2513 |=====

Would you like to upload the results to OpenBenchmarking.org (y/n): n

[pabloom@localhost ~]$ phoronix-test-suite result-file-to-text pythonbenchmark

python_benchmark
Oracle VMware testing on CentOS Linux 8 via the Phoronix Test Suite.

pythonbenchmark:

Processor: Intel Core i5-8300H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
440FX 82441FX PMC, Memory: 818MB, Disk: 9GB VBOX HDD, Graphics: VMware SUGA II, Audio: Intel 82801A
A AC 97 Audio, Network: 2 x Intel 82540EM

OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: xfs, Screen Resolut
ion: 2048x2048, System Layer: Oracle VMware

PyBench 2018-02-16
Total For Average Test Times
Milliseconds < Lower Is Better
pythonbenchmark . 2513 |=====

[pabloom@localhost ~]$ _
```

### 1.3 Comparación de resultados

Finalmente procedamos a comparar los resultados. Para facilitar la comparación, pondré los datos resultantes en la siguiente tabla:

Test	Ubuntu Server	CentOS
PHP Zend bench	0.612 ms	0.746 ms
PHP Zend micro_bench	3.194 ms	3.893 ms
PyBench	1953 ms	2513 ms

Como podemos ver en las capturas realizadas, el resultado de un test es mejor cuanto menor sea el número de milisegundos resultante. Por tanto, se ve claramente en la tabla que mi máquina Ubuntu Server es mejor para realizar ejecuciones de programas y scripts con PHP y



Python que mi máquina CentOS. Esto no quiere decir que Ubuntu Server vaya a ser mejor siempre para dichas tareas que CentOS, si no que mi máquina virtual con Ubuntu Server es generalmente mejor que mi máquina virtual con CentOS, lo que no significa que no pueda cambiar en otros equipos con diferentes características.

## Ejercicio 2.

---

*Tras probar un test básico para una web [7], utilizaremos Jmeter para hacer un test sobre una aplicación que ejecuta sobre dos contenedores (uno para la BD y otro para la aplicación en sí). El código está disponible en el [git de David Palomar](#) donde se dan detalles sobre cómo ejecutar la aplicación en una de nuestras máquinas virtuales. El test de Jmeter debe incluir los siguientes elementos.*

### 2.1 Instalación de Docker

La idea de este ejercicio es que probemos un test de carga de diversos elementos sobre un servidor HTTP en un contenedor dentro de nuestra máquina Ubuntu Server. Para ello, comenzaré instalando docker en mi máquina virtual Ubuntu Server tal y como se especifica en el guión de prácticas:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt update

sudo apt search docker-ce
sudo apt install docker-ce
sudo usermod -aG docker pabloom
```

Después de instalarlo, lo activamos y vemos que funciona correctamente tras reiniciar sesión.

```
sudo systemctl enable docker.service
sudo systemctl start docker.service
docker info
docker run hello-world
```

```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
TriggeredBy: • docker.socket
Docs: https://docs.docker.com
pabloom@pabloom:~$ sudo systemctl start docker.service
[sudo] password for pabloom:
pabloom@pabloom:~$ sudo systemctl enable docker.service
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
pabloom@pabloom:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:80f31da1ac7b312ba29d65080fddf797dd76acfb870e677f390d5acba9741b17
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

pabloom@pabloom:~$ _
```

Una vez instalado, procederemos a instalar Docker Compose, que es una herramienta para orquestar contenedores.

```
sudo apt install docker-compose
```

Para ver que funciona:

```
docker-compose
docker-compose --version
```

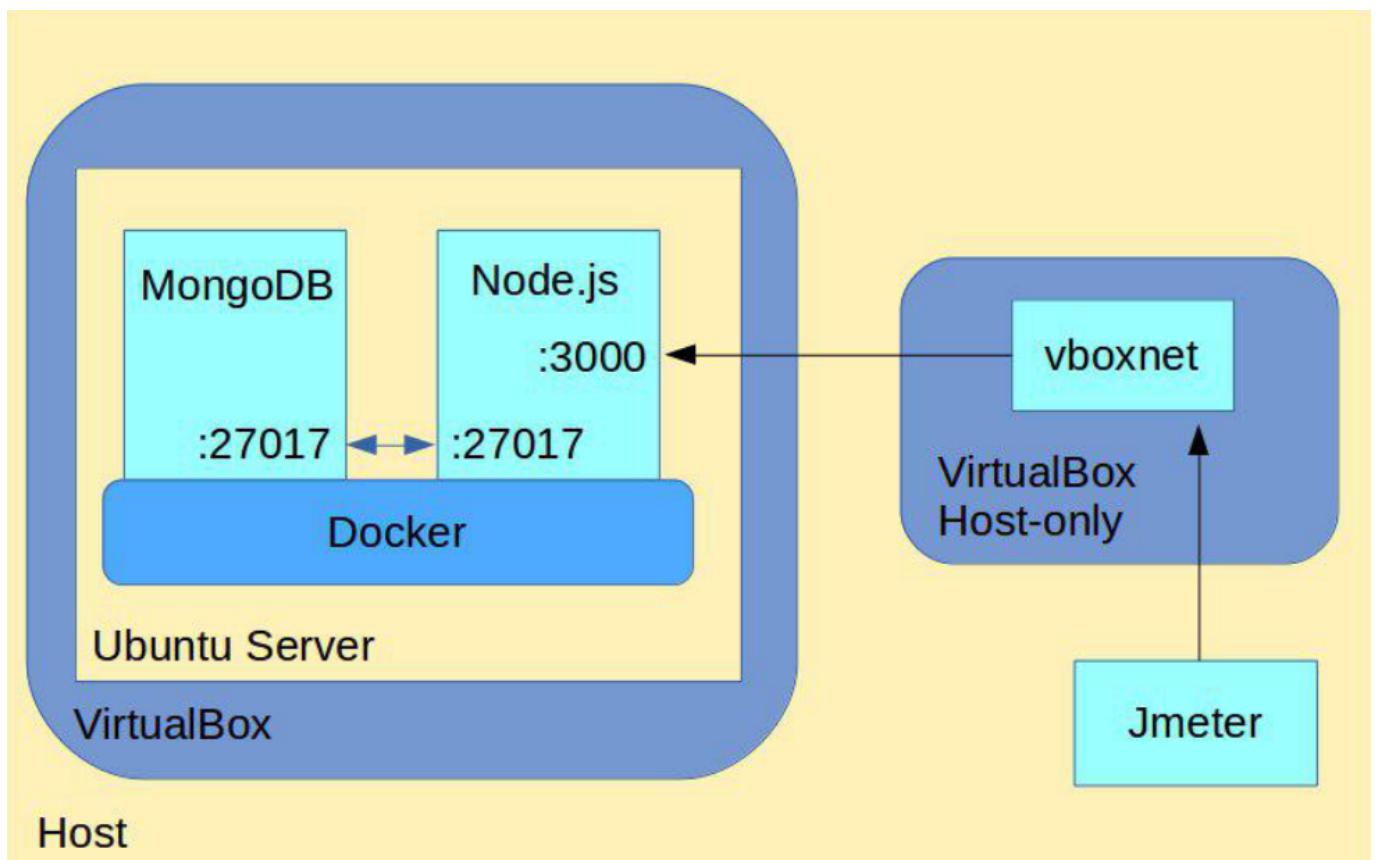
```
Ubuntu Server OS (Benchmark) [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
pabloom@pabloom:~$ docker-compose --version
docker-compose version 1.25.0, build unknown
pabloom@pabloom:~$ _
```

## 2.2 Instalación de la aplicación para el test con Jmeter

Procederemos ahora con la instalación del programa a testear con Jmeter elaborado por David Palomar. Para ello, seguiremos los pasos de instalación del [repositorio de la práctica](#).

```
git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git
cd iseP4JMeter
docker-compose up
```

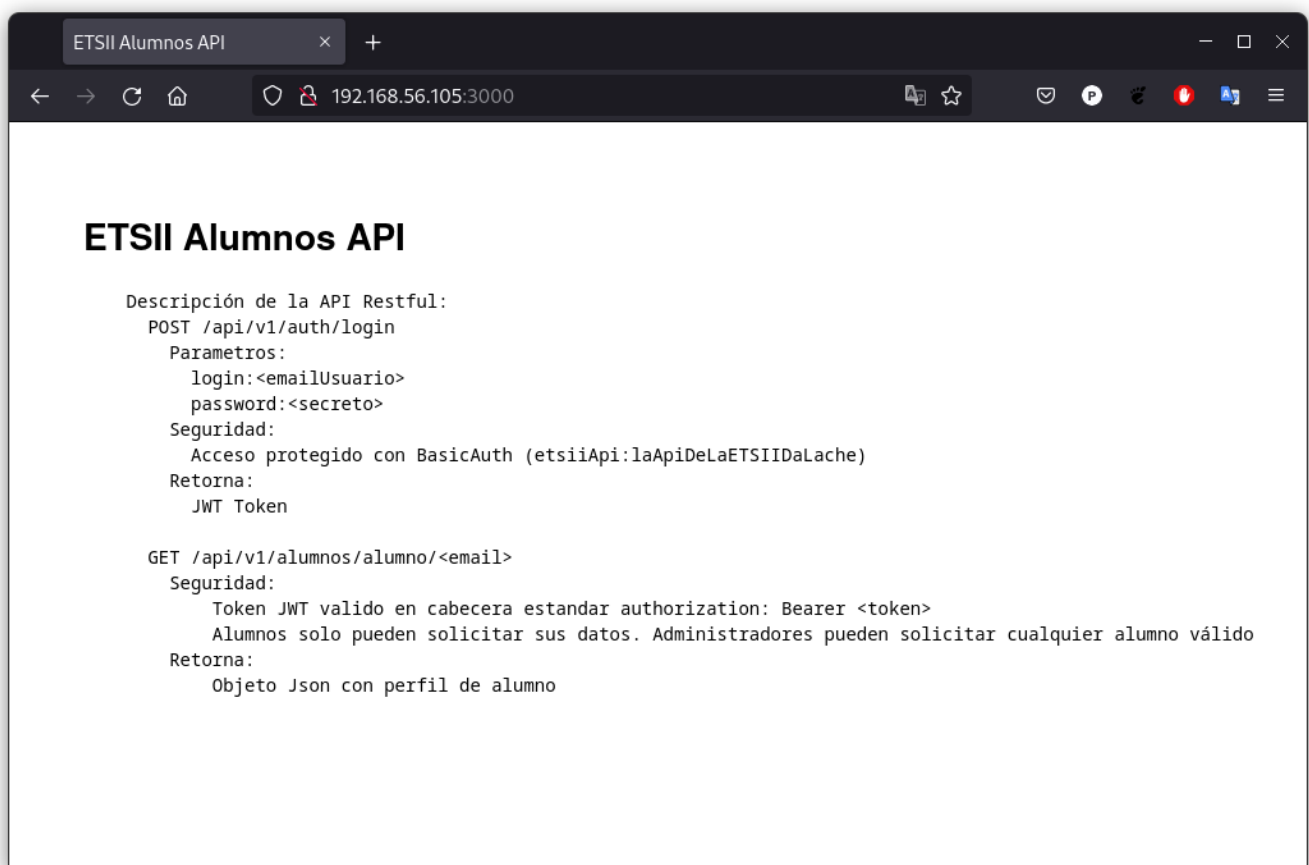
A continuación describiré qué es lo que estamos instalando. Para ello, me apoyaré en los esquemas proporcionados en el guión.



La idea es comprobar que nuestra API implementada con la arquitectura REST (*Representational State Transfer*) funciona correctamente y soportando las cargas esperadas

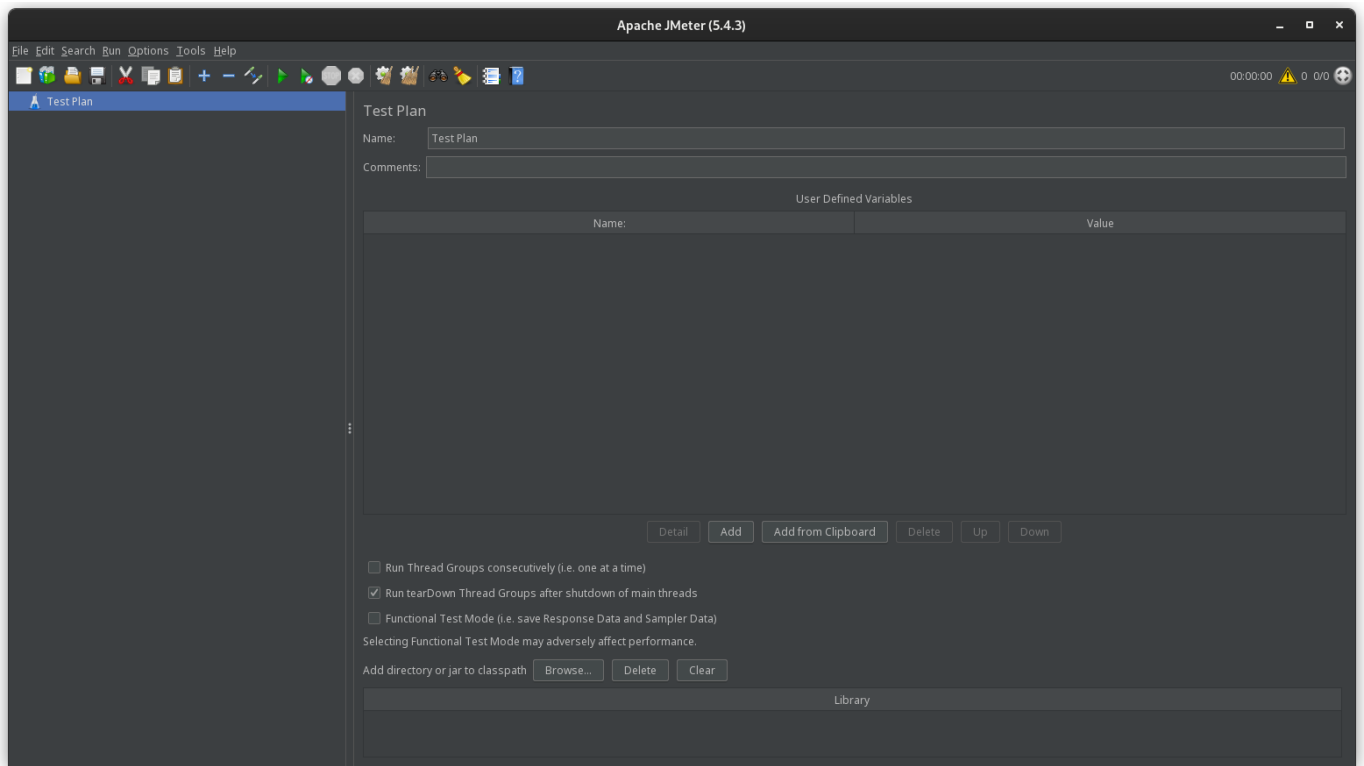
utilizando Jmeter. Dicha arquitectura es la más utilizada para la obtención y modificación de datos utilizando HTTP. La idea consiste en utilizar la API como intermediario entre el cliente y la base de datos, tal y como vemos en el esquema de arriba. Para ello hemos hecho uso de contenedores de forma que da igual la máquina sobre la que se ejecute que la implementación funcionará exactamente igual que en el entorno para el que se preparó.

Tras la instalación de los contenedores, veremos que tendremos acceso desde nuestro ordenador anfitrión a la web HTTP levantada en el entorno Node.js con enlace <http://192.168.56.105:3000> :

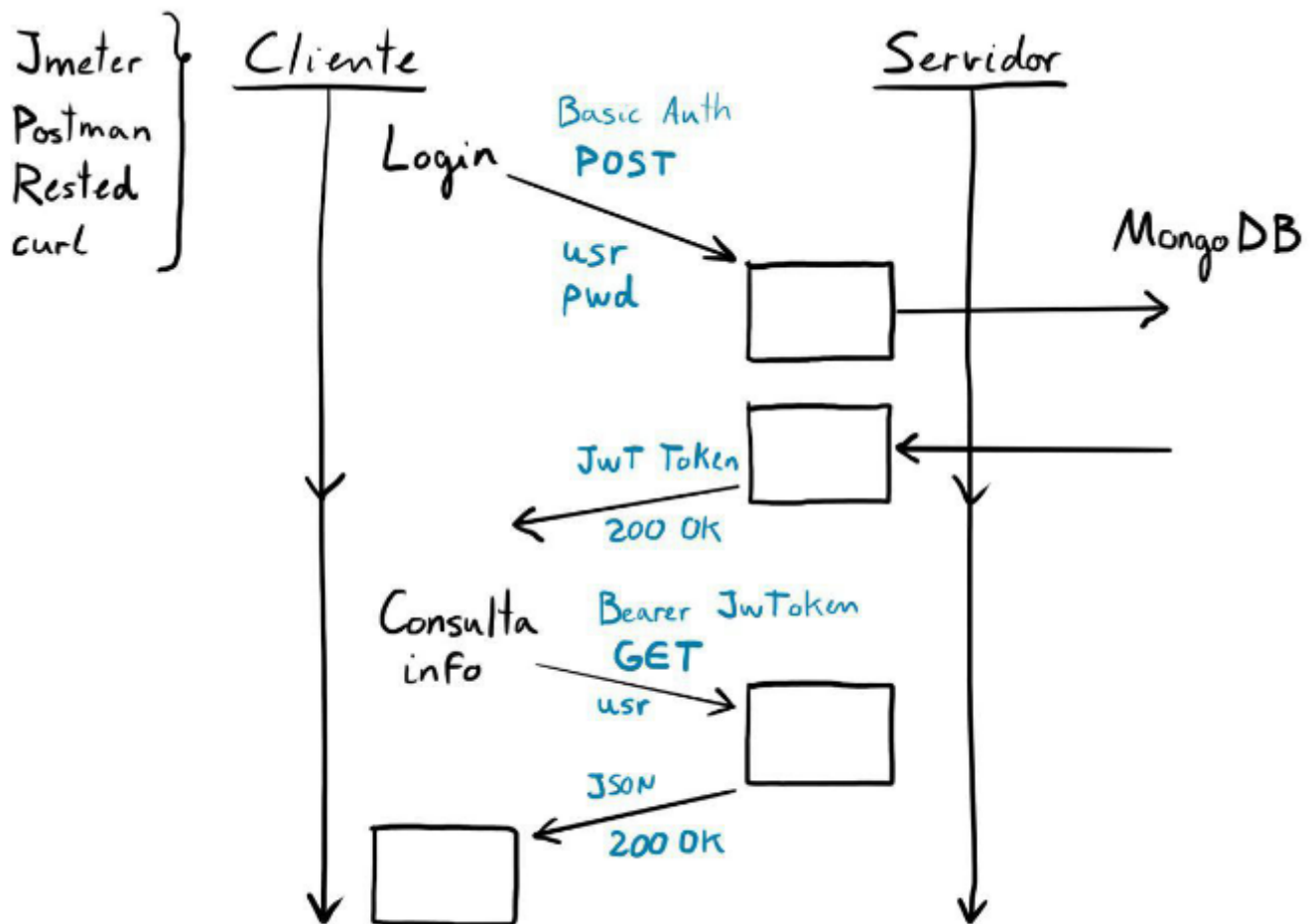


## 2.3 Creación del Test Básico de JMeter

Por último antes de comenzar con la práctica en sí, instalaremos JMeter en nuestra máquina anfitrión. Para ello descargamos el programa de la propia [página de JMeter](#) y ejecutamos el binario.



Nuestro objetivo es simular el acceso de alumnos y administradores a la aplicación. Además, esto debemos hacerlo teniendo en cuenta varios criterios que iremos viendo a continuación. El esquema que lo ilustra es el siguiente:



Empezaremos elaborando el test básico solicitado en el ejercicio siguiendo la [documentación oficial](#). Para ello comenzaré renombrando el test y parametrizando el puerto y el host.

Test Plan	
Name:	ETSII Alumnos API
Comments:	
User Defined Variables	
Name:	Value
HOST	192.168.56.105
PORT	3000

A continuación, clic derecho en el test, *Add -> Config Element -> HTTP Default Requests* para añadir una configuración que compartirán todas nuestras solicitudes HTTP. En nuestro caso, el *HOST* y el puerto, *PORT*.

**HTTP Request Defaults**

Name: Solicitudes por defecto HTTP

Comments:

**Basic** Advanced

Web Server

Protocol [http]: http Server Name or IP: \${HOST} Port Number: \${PORT}

HTTP Request

Path: Content encoding:

**Parameters** Body Data

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

### 2.3.1 Configuración del grupo de hebras Alumnos

Vamos con la configuración del grupo de hebras de Alumnos. Para ello, clic derecho en el test, *Add -> Threads(Users) -> Thread Group* y lo completamos de la siguiente forma:

**Thread Group**

Name: Alumnos

Comments:

Action to be taken after a Sampler error

☒ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☐ Stop Test Now

Thread Properties

Number of Threads (users): 70

Ramp-up period (seconds): 15

Loop Count: ☐ Infinite 5

☒ Same user on each iteration

☐ Delay Thread creation until needed

☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Acto seguido comenzaré con la creación de las solicitudes que vamos a realizar. Las podemos introducir en clic derecho sobre *Alumnos -> Add -> Sampler -> HTTP Request*.

#### 2.3.1.1 Login de Alumnos

Por un lado se encuentra el envío de credenciales: nombre de usuario (*login*) y contraseña (*password*). Además, como se puede ver en el esquema, debe ser de tipo *POST* en la dirección `http://${HOST}:${PORT}/api/v1/auth/login`. Como el puerto y el host ya los hemos introducido en la solicitud por defecto, no es necesario ponerlo aquí.

**HTTP Request**

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]:  Server Name or IP:  Port Number:

HTTP Request

POST Path:  Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
login	\${login}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	\${password}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

Sin embargo, Jmeter necesita obtener los datos que va a utilizar como usuario y contraseña de alumnos de algún lado. Nosotros emplearemos el fichero *alumnos.csv* proporcionado por David Palomar para la práctica. Para importarlo, clic derecho en *Alumnos* -> *Add* -> *Config Element* -> *CSV Data Set Config*.

**CSV Data Set Config**

Name:

Comments:

Configure the CSV Data Source

Filename:  Browse...

File encoding:

Variable Names (comma-delimited):

Ignore first line (only used if Variable Names is not empty):

Delimiter (use '\t' for tab):

Allow quoted data?:

Recycle on EOF?:

Stop thread on EOF?:

Sharing mode:

Ahora seguiremos con la implementación de la pausa aleatoria Gaussiana. La utilizaremos para establecer la frecuencia con la que el servidor va a recibir las peticiones. Para ello, clic derecho *Alumnos* -> *Add* -> *Timer* -> *Gaussian Random Timer*.

**Gaussian Random Timer**

Name:

Comments:

Thread Delay Properties

Deviation (in milliseconds):

Constant Delay Offset (in milliseconds):

Por último en el login de alumnos, veamos como obtener el token JWT que devuelve el correcto acceso de un alumno utilizando expresiones regulares. Este token nos será de gran utilidad para que el servidor identifique al usuario una vez autenticado y para que conozca qué



privilegios dispone el usuario que se ha identificado. Para obtenerlo, clic derecho en *Login Alumno* -> *Add* -> *Post Processors* -> *Regular Expression Extractor*.

The screenshot shows the 'Regular Expression Extractor' configuration window. The 'Name' field is 'Extract JWT Token'. The 'Apply to' section has 'Main sample only' selected. The 'Field to check' section has 'Body' selected. The 'Name of created variable' is 'token'. The 'Regular Expression' is '.\*+'. The 'Template' is '\$0\$'. The 'Match No.' is '0'. The 'Default Value' is empty, and the 'Use empty default value' checkbox is checked.

### 2.3.1.2 Recuperación de datos de Alumnos

Con esto ya tendríamos todo lo necesario para el inicio de sesión de alumnos. Ahora realizaremos un proceso similar a *Login Alumno* para la recuperación de datos de éste. Lo que haremos será crear una solicitud HTTP que llamaremos *Recuperar Datos Alumno*. Para ello he seguido la estructura que marca el esquema.

Antes, cuando hemos declarado la variable *login* en *Login Alumno* hemos marcado la opción *URL Encode*. Ésta se encarga de almacenar las variables en un formato que se puede transmitir por internet. Habíamos marcado esta casilla con el propósito de poder utilizar la variable en la dirección tal y como se ve en la captura:

The screenshot shows the 'HTTP Request' configuration window. The 'Name' field is 'Recuperar Datos Alumno'. The 'Basic' tab is selected. The 'Web Server' section has 'Protocol [http]:' set to 'http', 'Server Name or IP:' set to 'localhost', and 'Port Number:' set to '8080'. The 'HTTP Request' section has 'GET' selected for the method and '/api/v1/alumnos/alumno/\${\_urlencode(\${login})}' for the path. The 'Content encoding' is set to 'UTF-8'. The 'Parameters' tab is selected, and the 'Send Parameters With the Request' section is empty.

**Nota:** a la hora de implementar la función de `urlencode` he consultado [esta página](#).

Dicha solicitud HTTP está declarada tal y cómo se indica en la *Figura 2.4* del guión de prácticas. Además, en la figura, vemos que la solicitud se debe realizar utilizando el token JWT. Para ello usaremos un gestor de cabeceras HTTP asociándolo a nuestra solicitud.

**HTTP Header Manager**

Name:

Comments:

Headers Stored in the Header Manager

Name:	Value
Authorization	Bearer \${token}

Y con esto habríamos terminado con la parte de alumnos.

### 2.3.2 Configuración del grupo de hebras Administradores

Primero crearemos el grupo de hebras tal y como se ve en la captura:

**Thread Group**

Name:

Comments:

Action to be taken after a Sampler error

☒ Continue
 ☐ Start Next Thread Loop
 ☐ Stop Thread
 ☐ Stop Test
 ☐ Stop Test Now

Thread Properties

Number of Threads (users):

Ramp-up period (seconds):

Loop Count: ☐ Infinite

☒ Same user on each iteration  
☐ Delay Thread creation until needed  
☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

De momento realizaremos un proceso análogo para el login de administradores al de login de alumnos:

**HTTP Request**

Name:

Comments:

Basic Advanced

Web Server

Protocol [http]:  Server Name or IP:  Port Number:

HTTP Request

POST Path:  Content encoding:

☐ Redirect Automatically
 ☒ Follow Redirects
 ☒ Use KeepAlive
 ☐ Use multipart/form-data
 ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
login	\${login}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	\${password}	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

### CSV Data Set Config

Name:

Comments:

Configure the CSV Data Source

Filename:

File encoding:

Variable Names (comma-delimited):

Ignore first line (only used if Variable Names is not empty):

Delimiter (use '\t' for tab):

Allow quoted data?:

Recycle on EOF?:

Stop thread on EOF?:

Sharing mode:

### Access Log Sampler

Name:

Comments:

Default Test Values

Protocol:

Server:

Port:

Parse Images:

Plugin Classes

Parser:

Filter (Optional):

Log File Location

Log File:

### Gaussian Random Timer

Name:

Comments:

Thread Delay Properties

Deviation (in milliseconds):

Constant Delay Offset (in milliseconds):

### Regular Expression Extractor

Name:

Comments:

Apply to:

☐ Main sample and sub-samples ☒ Main sample only ☐ Sub-samples only ☐ JMeter Variable Name to use

Field to check

☒ Body ☐ Body (unescaped) ☐ Body as a Document ☐ Response Headers ☐ Request Headers ☐ URL ☐ Response Code ☐ Response Message

Name of created variable:

Regular Expression:

Template (\$i\$ where i is capturing group number, starts at 1):

Match No. (0 for Random):

Default Value:  ☐ Use empty default value

Hasta este punto hemos realizado el proceso análogo. Sin embargo, el ejercicio nos pide que los administradores también puedan consultar los datos de cualquier alumno válido. Para ello,

añadiremos un *Access Log Sampler* a administradores, el cual simulará el uso de la API por parte de usuarios a partir de un archivo *.log*. Para ello, clicaremos con el botón derecho sobre *Administradores* -> *Add* -> *Sampler* -> *Access Log Sampler*.

The screenshot shows the 'Access Log Sampler' configuration window. The 'Name' field is set to 'Acceso Administradores'. The 'Comments' field is empty. Under 'Default Test Values', 'Protocol' is 'http', 'Server' is '\${HOST}', and 'Port' is '\${PORT}'. 'Parse Images' is set to 'False'. Under 'Plugin Classes', the 'Parser' is set to 'org.apache.jmeter.protocol.http.util.accesslog.TCLogParser'. The 'Filter (Optional)' is 'Undefined'. The 'Log File Location' is '/home/pablo/Documentos/DGIIM3/ISE/02 - Prácticas/P4/JMeter/apiAlumnos.log', with a 'Browse...' button next to it.

Y también usaremos un gestor de cabeceras HTTP para gestionar el uso del token.

The screenshot shows the 'HTTP Header Manager' configuration window. The 'Name' field is set to 'JWT Token'. The 'Comments' field is empty. Below, there is a table titled 'Headers Stored in the Header Manager'.

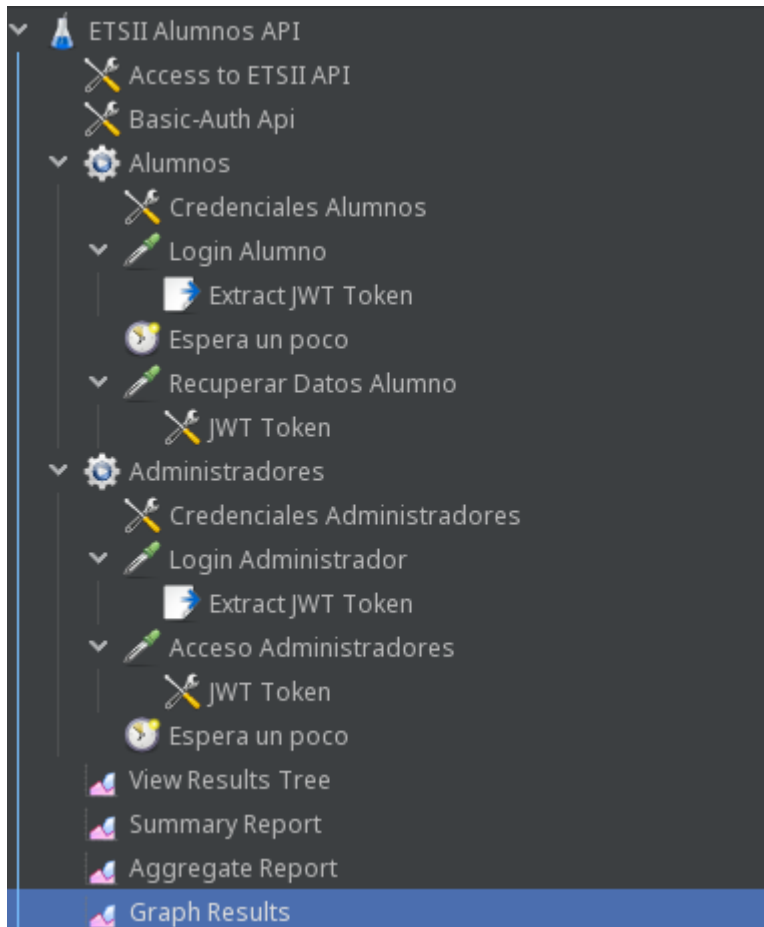
Name:	Value
Authorization	Bearer \${token}

Por último pero no menos importante, debemos darle acceso a nuestro Test a la API. Para ello, daremos clic derecho *ETSII Alumnos API* -> *Add* -> *Config Element* -> *HTTP Authorization Manager* y lo rellenamos de la siguiente forma:

The screenshot shows the 'HTTP Authorization Manager' configuration window. The 'Name' field is set to 'HTTP Authorization Manager'. The 'Comments' field is empty. Under 'Options', both 'Clear auth on each iteration?' and 'Use Thread Group configuration to control clearing' are unchecked. Below, there is a table titled 'Authorizations Stored in the Authorization Manager'.

Base URL	Username:	Password:	Domain	Realm	Mechanism
http://\${HOST}:\${PORT}/api/v1/auth/login	etsiiApi	laApiDjLaETSIIaLache			BASIC

Finalmente así debe quedar el árbol del Test:



## 2.4 Realización de la prueba de carga

Una vez hecha la configuración, tan sólo queda hacer la prueba de carga. Para poder ver los resultados de dicha prueba, añadiremos finalmente unos *Listeners* que nos permitirán ver la información según nos interese. Tras ejecutar la prueba de carga, estos son los resultados obtenidos:

**View Results Tree**

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename:   Log/Display Only: ☐ Errors ☐ Successes

Search:  ☐ Case sensitive ☐ Regular exp.

Text

- Recuperar Datos Alumno
- Login Alumno
- Login Administrador
- Recuperar Datos Alumno
- Recuperar Datos Alumno
- Login Alumno
- Login Alumno
- Login Administrador
- Login Alumno
- Login Alumno
- Recuperar Datos Alumno
- Recuperar Datos Alumno
- Login Alumno
- Login Administrador
- Login Administrador
- Login Alumno
- Login Alumno
- http://192.168.56.105:3000/api/v1/alumnos/alumno/de

**Sampler result**

Thread Name: Administradores 2-10

Sample Start: 2022-05-30 23:27:26 CEST

Load time: 4

Connect Time: 0

Latency: 4

Response header

Header	Value
HTTP/1.1 200 OK	
X-DNS-Prefetch-Control	off
X-Frame-Options	SAMEORIGIN
Strict-Transport-Security	max-age=15552000; includeSubDomains
X-Download-Options	noopen
X-Content-Type-Options	nosniff
X-XSS-Protection	1; mode=block

Additional field

Type Result: HTTPSampleResult

El *Listener View Results Tree* nos proporciona información acerca de cómo han ido cada una de las consultas realizadas y su estado.

**Summary Report**

Name: Summary Report

Comments:

Write results to file / Read from file

Filename:   Log/Display Only: ☐ Errors ☐ Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Recei...	Sent KB/sec	Avg. Bytes
Login Alumno	350	9	2	67	8.35	0.00%	20.2/sec	11.97	6.60	606.7
Login Administrador	200	8	3	47	6.87	0.00%	9.8/sec	5.91	3.20	618.0
Recuperar Datos Alumno	350	9	2	65	8.56	0.00%	20.5/sec	30.41	7.58	1515.5
http://192.168.56.105:3000/api/v1/alumnos/al...	10	11	3	30	7.98	0.00%	1.2/sec	1.36	0.00	1207.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	19	4	75	20.99	0.00%	1.2/sec	1.47	0.00	1305.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	10	4	25	6.76	0.00%	1.2/sec	1.87	0.00	1605.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	11	3	25	6.93	0.00%	1.2/sec	2.42	0.00	2044.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	10	3	19	6.10	0.00%	1.2/sec	1.33	0.00	1142.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	11	3	40	10.20	0.00%	1.2/sec	1.82	0.00	1575.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	12	3	35	10.04	0.00%	1.2/sec	1.42	0.00	1258.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	11	3	49	13.23	0.00%	1.1/sec	1.18	0.00	1082.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	6	3	11	2.70	0.00%	1.2/sec	1.73	0.00	1512.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	5	3	13	3.61	0.00%	1.2/sec	1.22	0.00	1083.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	9	3	27	8.94	0.00%	1.2/sec	1.28	0.00	1131.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	11	3	35	8.99	0.00%	1.2/sec	1.84	0.00	1631.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	15	3	53	16.61	0.00%	1.2/sec	1.42	0.00	1219.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	9	3	39	10.14	0.00%	1.1/sec	2.43	0.00	2175.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	10	3	28	7.71	0.00%	1.1/sec	1.36	0.00	1242.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	9	5	17	3.74	0.00%	1.1/sec	1.68	0.00	1512.0
http://192.168.56.105:3000/api/v1/alumnos/al...	10	10	4	22	6.33	0.00%	1.1/sec	1.17	0.00	1063.0

☐ Include group name in label?  ☒ Save Table Header

El *Listener Viewer Report* nos recoge en una tabla todos los datos parametrizados durante la prueba para poder exportarlos y analizarlos según necesitemos.

Aggregate Report

Name:

Aggregate Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

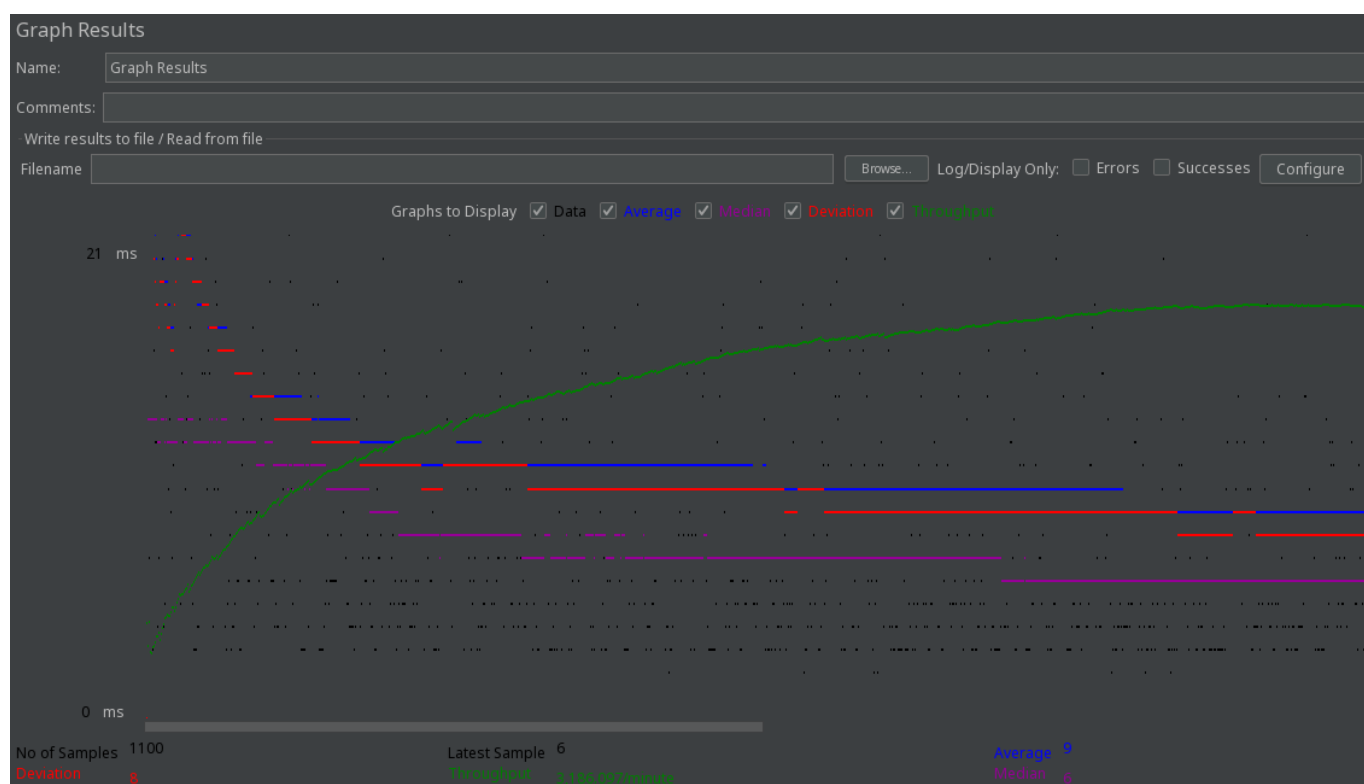
☐ Errors

☐ Successes

Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received K...	Sent KB/sec
Login Alum...	350	9	7	21	26	38	2	67	0.00%	20.2/sec	11.97	6.60
Login Admi...	200	8	6	14	22	38	3	47	0.00%	9.8/sec	5.91	3.20
Recuperar ...	350	9	6	20	26	39	2	65	0.00%	20.5/sec	30.41	7.58
http://192.1...	10	11	7	18	18	30	3	30	0.00%	1.2/sec	1.36	0.00
http://192.1...	10	19	8	32	32	75	4	75	0.00%	1.2/sec	1.47	0.00
http://192.1...	10	10	7	21	21	25	4	25	0.00%	1.2/sec	1.87	0.00
http://192.1...	10	11	9	21	21	25	3	25	0.00%	1.2/sec	2.42	0.00
http://192.1...	10	10	11	18	18	19	3	19	0.00%	1.2/sec	1.33	0.00
http://192.1...	10	11	8	15	15	40	3	40	0.00%	1.2/sec	1.82	0.00
http://192.1...	10	12	6	25	25	35	3	35	0.00%	1.2/sec	1.42	0.00
http://192.1...	10	11	6	17	17	49	3	49	0.00%	1.1/sec	1.18	0.00
http://192.1...	10	6	5	10	10	11	3	11	0.00%	1.2/sec	1.73	0.00
http://192.1...	10	5	4	12	12	13	3	13	0.00%	1.2/sec	1.22	0.00
http://192.1...	10	9	5	25	25	27	3	27	0.00%	1.2/sec	1.28	0.00
http://192.1...	10	11	10	18	18	35	3	35	0.00%	1.2/sec	1.84	0.00
http://192.1...	10	15	6	35	35	53	3	53	0.00%	1.2/sec	1.42	0.00
http://192.1...	10	9	5	12	12	39	3	39	0.00%	1.1/sec	2.43	0.00
http://192.1...	10	10	6	19	19	28	3	28	0.00%	1.1/sec	1.36	0.00
http://192.1...	10	9	10	14	14	17	5	17	0.00%	1.1/sec	1.68	0.00
http://192.1...	10	10	7	21	21	22	4	22	0.00%	1.1/sec	1.17	0.00

El *Aggregate Report* nos proporciona datos estadísticos acerca de la prueba realizada.



Por último, el *Listener Graph Results* nos dispone en una gráfica los datos estadísticos comentados anteriormente para que podamos analizarlo fácilmente.

Estos son tan sólo algunos de los *Listeners* que proporciona el programa y que he decidido implementar. Para más información sobre estos, conviene consultar la entrada en la

documentación oficial.

```
pabloom@pabloom: ~/iseP4JMeter
snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) b
ase write gen: 1"}}
mongodb_1 | {"t":{"$date":"2022-05-30T21:45:51.084+00:00"},"s":"I", "c":"S
TORAGE", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"
"message":"[1653947151:84645][1:0x7fb93c7f9700], WT_SESSION.checkpoint: [WT_VERB
_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2585, snapshot max: 2585 s
napshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) b
ase write gen: 1"}}
mongodb_1 | {"t":{"$date":"2022-05-30T21:46:51.118+00:00"},"s":"I", "c":"S
TORAGE", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"
"message":"[1653947211:118733][1:0x7fb93c7f9700], WT_SESSION.checkpoint: [WT_VER
B_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2587, snapshot max: 2587
snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0)
base write gen: 1"}}
mongodb_1 | {"t":{"$date":"2022-05-30T21:47:51.150+00:00"},"s":"I", "c":"S
TORAGE", "id":22430, "ctx":"Checkpoint", "msg":"WiredTiger message", "attr":{"
"message":"[1653947271:150939][1:0x7fb93c7f9700], WT_SESSION.checkpoint: [WT_VER
B_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2589, snapshot max: 2589
snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0)
base write gen: 1"}}
^CGracefully stopping... (press Ctrl+C again to force)
Stopping isep4jmeter_nodejs_1 ... done
Stopping isep4jmeter_mongodb_1 ... done
pabloom@pabloom:~/iseP4JMeter$
```

## Ejercicio Opcional 1.

Tras ver la parte de la práctica relacionada con contenedores, pruebe a ejecutar uno de los benchmarks de los seleccionados arriba y analice las diferencias en los resultados obtenidos en la MV directamente. Puede lanzar un contenedor con la imagen de phoronix, consulte: <https://www.phoronix.com/scan.php?page=article&item=docker-phoronix-pts&num=1> en su anfitrión (en la máquina virtual se quedará sin espacio). Otra posibilidad es ejecutar una imagen de contendedor de Ubuntu y ahí instalar phoronix.

Para realizar este ejercicio, comenzaremos instalando la imagen de Phoronix mencionada en el enunciado:

```
sudo docker run -it phoronix/pts
```

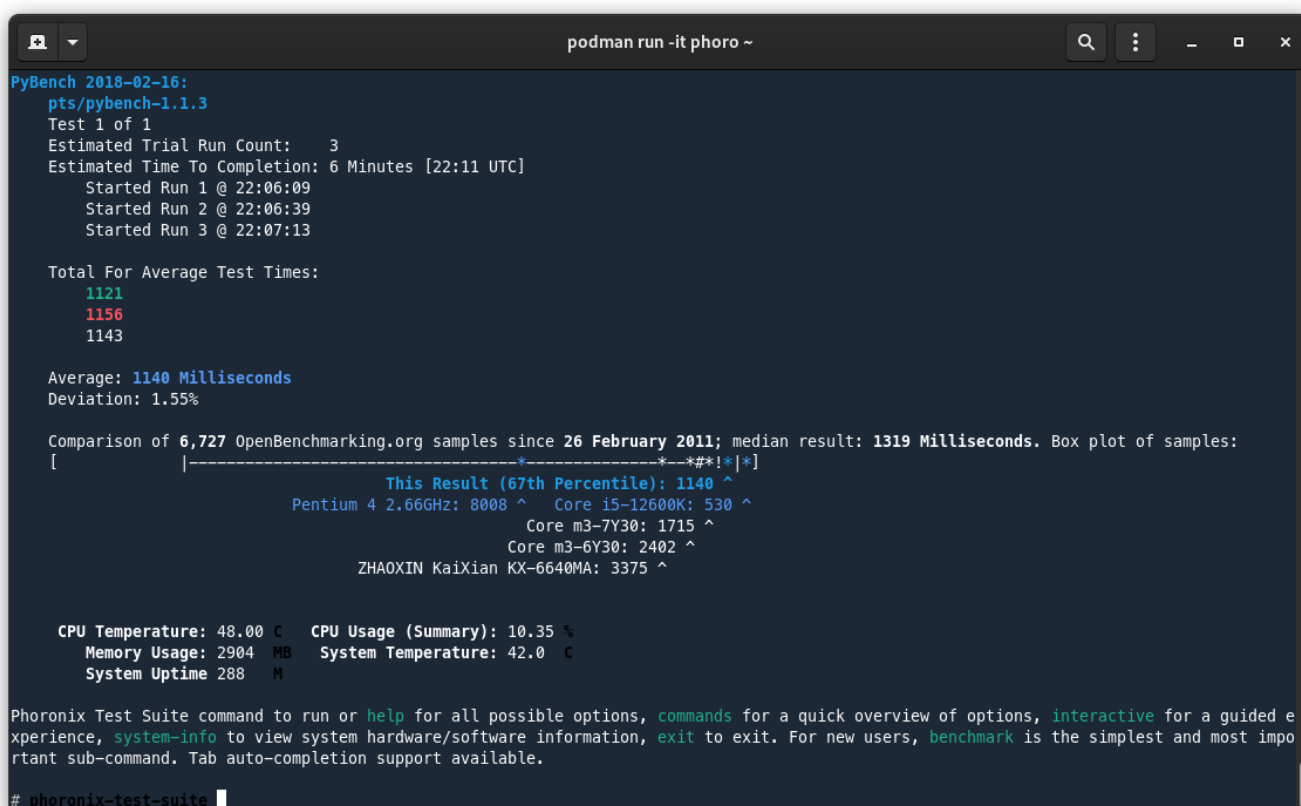
**Nota:** en mi caso usaré el comando `podman` en lugar de `docker` , pues mi dispositivo es Fedora. Sin embargo, el proceso y su funcionamiento son idénticos.



Como vamos a tener que usar el docker de manera interactiva para decidir qué test ejecutar, uso `-i` para dejar abierta la entrada estándar y `-t` para asignarle una terminal a nuestro contenedor. Además, dicho comando ejecutará el contenedor si disponemos de él y si no, lo instala y lo ejecuta.

Ahora instalaremos por ejemplo el test de python empleado en el ejercicio 1 y compararemos sus resultados con los ya obtenidos.

```
phoronix-test-suite benchmark pts/pybench
```



```
podman run -it phoro ~
PyBench 2018-02-16:
pts/pybench-1.1.3
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 6 Minutes [22:11 UTC]
Started Run 1 @ 22:06:09
Started Run 2 @ 22:06:39
Started Run 3 @ 22:07:13

Total For Average Test Times:
1121
1156
1143

Average: 1140 Milliseconds
Deviation: 1.55%

Comparison of 6,727 OpenBenchmarking.org samples since 26 February 2011; median result: 1319 Milliseconds. Box plot of samples:
[
  |-----*-----*--*!*[*]
          This Result (67th Percentile): 1140 ^
          Pentium 4 2.66GHz: 8008 ^   Core i5-12600K: 530 ^
                        Core m3-7Y30: 1715 ^
                        Core m3-6Y30: 2402 ^
          ZHAOXIN KaiXian KX-6640MA: 3375 ^

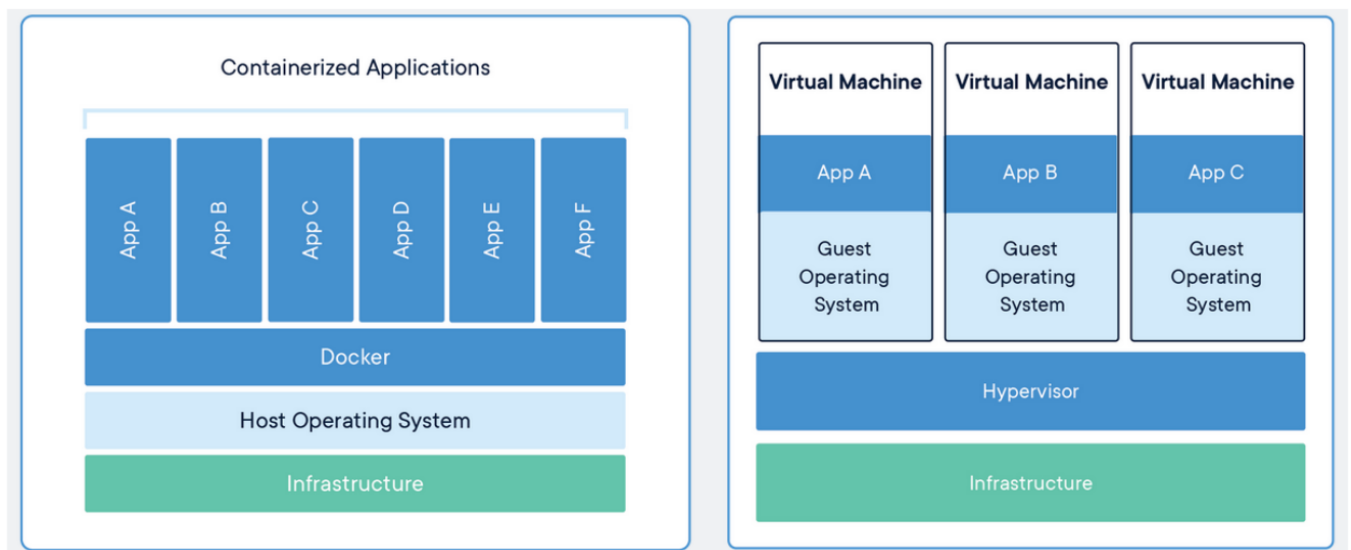
CPU Temperature: 48.00 C   CPU Usage (Summary): 10.35 %
Memory Usage: 2904 MB   System Temperature: 42.0 C
System Uptime 288 M

Phoronix Test Suite command to run or help for all possible options, commands for a quick overview of options, interactive for a guided experience, system-info to view system hardware/software information, exit to exit. For new users, benchmark is the simplest and most important sub-command. Tab auto-completion support available.

# phoronix-test-suite
```

Test	Ubuntu Server	CentOS	PTS Container
PyBench	1953 ms	2513 ms	1319 ms

Tras comparar los resultados, podemos ver que el contenedor es mucho más rápido que cualquiera de mis dos máquinas virtuales. esto es debido al tipo de virtualización que usan los contenedores frente a las VMs.



Como podemos ver en el esquema, las VMs dependen de la cantidad de recursos que el usuario y/o el *Hypervisor* deciden ofrecerle a ésta, además de tener que atravesar más capas. Por otro lado, *Docker* dispone de menos capas de abstracción, por lo que hace que el uso de recursos sea más eficiente.