Práctica 4

Phoronix

Enunciado: 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.

Instalación de Phoronix Test Suite

- En Ubuntu
    1. *$ sudo apt search phoronix* (búsqueda del paquete)
    2. *$ sudo apt install phoronix-test-suite/xenial* (instalación)
- En CentOS
    - Logueados como superusuario
    1. *# yum install wget php-cli php-xml bzip2* (instalación de dependencias)
    2. *# wget https://phoronix-test-suite.com/releases/phoronix-test-suite-8.4.1.tar.gz*
    3. *# tar xvfz phoronix-test-suite-8.4.1.tar.gz*
    4. *# cd phoronix-test-suite*
    5. *# ./install-sh*

Para saber como funciona phoronix consultamos al manual

$ man phoronix-test-suite

Benchmarks seleccionados

- CacheBench: https://openbenchmarking.org/test/pts/cachebench
- RAMSpeed: https://openbenchmarking.org/test/pts/ramspeed

Instalación de los benchmark mencionados

 # phoronix-test-suite install pts/ramspeed

 # phoronix-test-suite install pts/cachebench

Ejecución para cada benchmark

 # phoronix-test-suit run pts/ramspeed

 # phoronix-test-suit run pts/cachebench

Verificado de finalización

 # phoronix-test-suite finish-run pts/ramspeed

 # phoronix-test-suite finish-run pts/cachebench

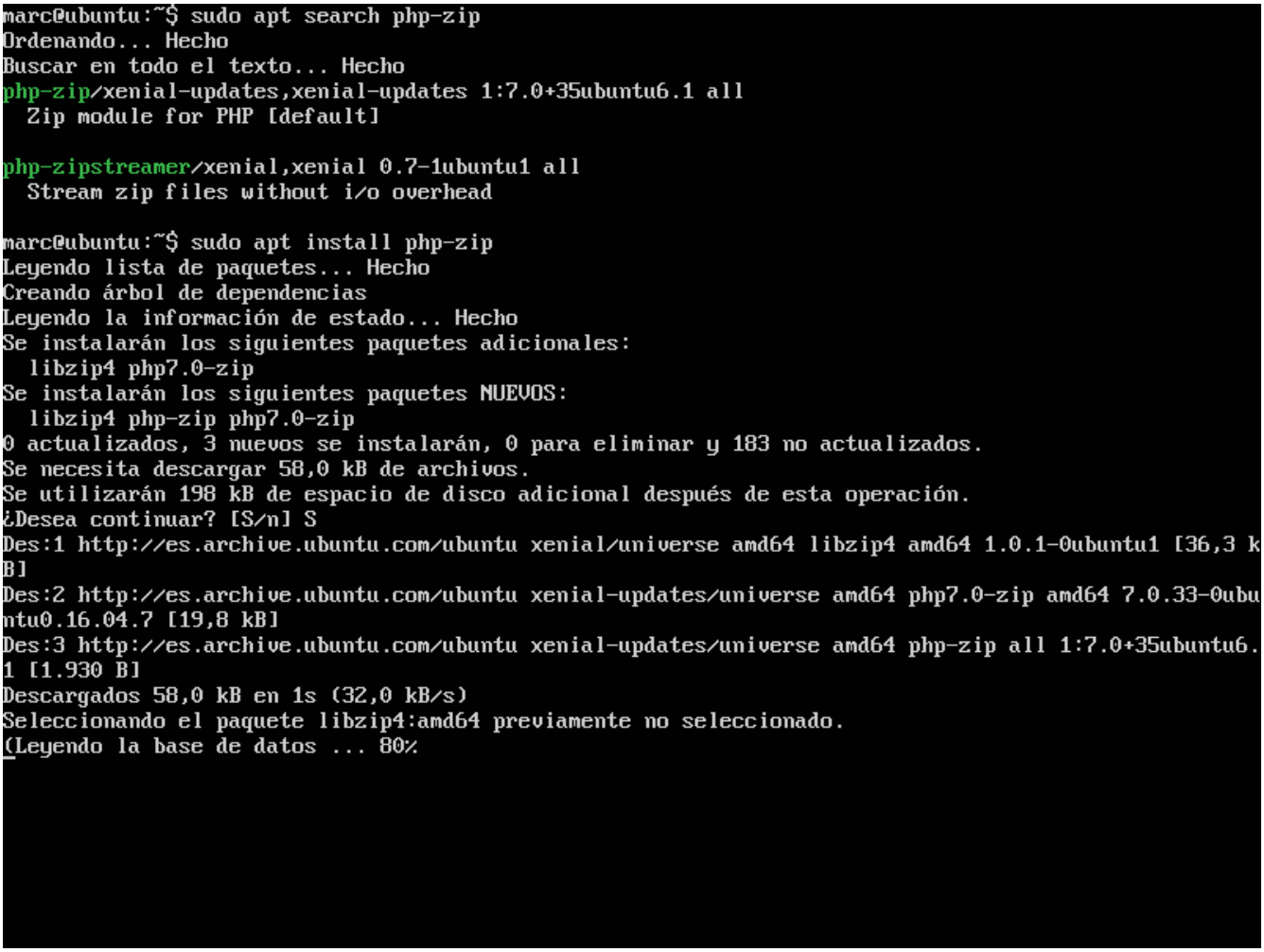
El verificado de finalización puede ser útil cuando ejecutamos los tests en segundo plano (background).

Instalación y ejecución en UbuntuServer:

Cuando se trata de instalar un test, nos encontramos que hace falta el soporte ZIP para extraer el test.

```
marc@ubuntu:~$ phoronix-test-suite install pts/ramspeed

[PROBLEM] Failed to find ZIP support for extracting file: /home/marc/.phoronix-test-suite/openbenchm
arking.org/pts/ramspeed-1.4.2.zip. Install PHP ZIP support or the unzip utility.

[PROBLEM] Invalid Argument: pts/ramspeed

CORRECT SYNTAX:
phoronix-test-suite install [Test | Suite | OpenBenchmarking.org ID | Test Result]  ...

See available tests to run by visiting OpenBenchmarking.org or running:

    phoronix-test-suite list-tests

Tests can be installed by running:

    phoronix-test-suite install <test-name>
```

Buscamos el paquete y lo instalamos.

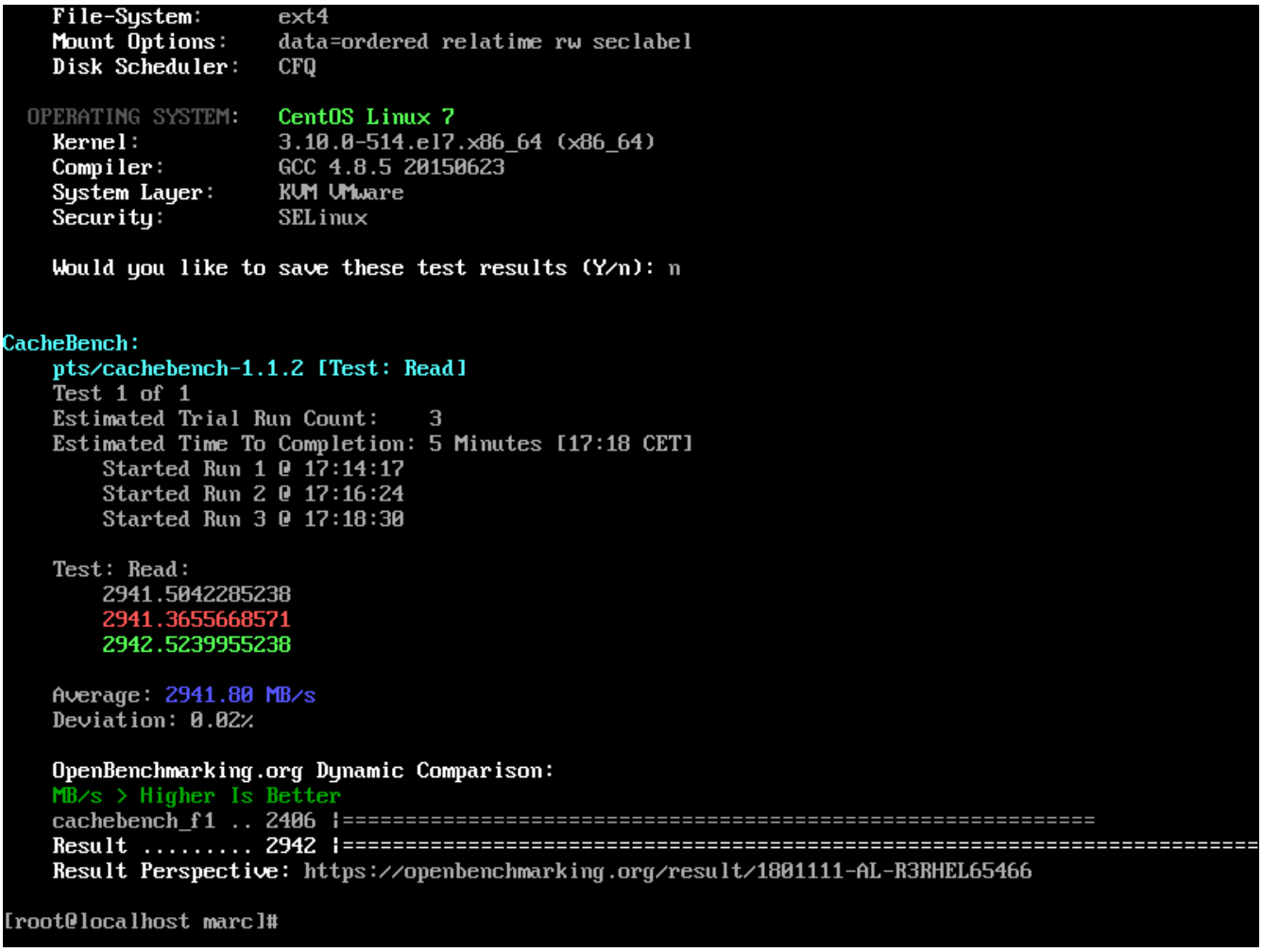


Una vez se ha solucionado este error podemos instalar los tests. Primero se instalan las dependencias y, una vez están instaladas, se instala el test.

Instalación y ejecución en CentOS:

Primero instalamos los paquetes necesarios. Phoronix no ofrece soporte para yum, por lo que para instalarlo hay que descargar el repositorio directamente desde la página oficial. A continuación se descomprime y se ejecuta el script de instalación.

Para ejecutarlo, se usan los mismos pasos que usamos para UbuntuServer.



Prueba de ejecución de CacheBench en CentOS

Comparativa de rendimiento de ambos sistemas

Ramspeed

Integer

	Copy	Scale	Add	Triad	Average
UbuntuServer	5654.06 MB/s	5597.96 MB/s	7804.23 MB/s	7931.86 MB/s	5934.87 MB/s
CentOS	8566.34 MB/s	3844.83 MB/s	4324.24 MB/s	5076.36 MB/s	3735.44 MB/s

Floating Point

	Copy	Scale	Add	Triad	Average
UbuntuServer	6004.12 MB/s	6214.48 MB/s	6639.91 MB/s	9519.16 MB/s	6343.26 MB/s
CentOS	8556.02 MB/s	7015.02 MB/s	8433.45 MB/s	7756.12 MB/s	8288.28 MB/s

Resultados

Este benchmark testea el rendimiento de la memoria RAM con las operaciones de copia, escala, suma, triada, y media; tanto para operaciones con enteros como para operaciones con punto flotante.

En términos generales, UbuntuServer funciona mejor que CentOS para operaciones con enteros pero ocurre al contrario para las operaciones con punto flotante.

Para UbuntuServer las operaciones de punto flotante son 1.07 veces más rápidas que las operaciones de enteros, mientras que para CentOS son 2.22 veces más rápidas.

Por tanto, si se buscase un sistema más equilibrado que pueda ejecutar operaciones de ambos tipos escogeríamos UbuntuServer. Si el número de operaciones con enteros no representa una gran carga del sistema es preferible escoger CentOS porque con operaciones de números flotantes es más eficiente.

Cachebench

	Read	Write	Read/Modify/Write
UbuntuServer	2903.86 MB/s	10900.00 MB/s	38880.68 MB/s
CentOS	2894.01 MB/s	22303.00 MB/s	39297.21 MB/s

Resultados

Este benchmark testea el rendimiento de la memoria y el ancho de banda de la cache del procesador.

Podemos dividr la comparativa en operaciones de lectura, escritura y lectura/modificación/escritura.

- Lectura: UbuntuServer es 1.003 veces más rápido que CentOS.
- Escritura: UbuntuServer es 0.49 veces más rápido que CentOS.
- Lectura/Modificación/Escritura: UbuntuServer es 0.99 veces más rápido que CentOS.

Como vemos no existe una gran diferencia entre las operaciones de lectura y las operaciones de lectura/modificación/escritura para ambos sistemas. Para las operaciones de escritura encontramos que UbuntuServer no es ni un 50% de lo rápido que es CentOS.

En general ambos sistemas tienen el mismo rendimiento, aunque es preferible CentOS porque tiene un ancho de banda bastante mayor al de UbuntuServer.

Apache Benchmarking

En ambos sistemas operativos tenemos instalado AB debido a que está incorporado en el paquete de apache2, utilizado en una práctica anterior para obtener el LAMP.

```
marc@ubuntu:~$ sudo apt search apache benchmark
Ordenando... Hecho
Buscar en todo el texto... Hecho
apache2-utils/xenial-updates 2.4.18-2ubuntu3.14 amd64 [actualizable desde: 2.4.18-2ubuntu3.9]
  Servidor HTTP Apache (programas utilitarios para servidores web)

marc@ubuntu:~$
```

Comprobación en Ubuntu Server

Ejecución de AB

Apache Benchmarking se encarga de generar carga. No tiene sentido generar cargar desde el propio sistema, y tampoco podríamos hacerlo solo con Ubuntu y CentOS porque la máquina más lenta tarda más en responder pero también en lanzar peticiones, por lo que no notaríamos la diferencia entre ambas máquinas. Por tanto, las peticiones se ejecutan desde una tercera máquina. En mi caso, utilizo mi host con el sistema Ubuntu 19.10.

Para UbuntuServer

 # ab -k -n 100 -c 10 http://192.168.56.105/

Para CentOS

 # ab -k -n 100 -c 10 http://192.168.56.110/

La opción -k activa la herramienta KeepAlive que sirve para realizar simultáneas peticiones en una única sesión HTTP.

La opción -n especifica el número de peticiones a lanzar.

La opción -c especifica cuántas hebras se ejecutan concurrentemente.

Comparativa de rendimiento

	UbuntuServer	CentOS
Tiempo para los tests (segundos)	0.034	0.078
Peticiones por segundo (media)	2944.47	1278.31
Tiempo por petición (media)(ms)	3.396	7.823
Tiempo por petición (media, entre todas las peticiones concurrentes)(ms)	0.340	0.782
Ratio de transferencia (Kbytes/segundo)	33444.72	325.95

Para todas las comparativas obtenemos que UbuntuServer funciona mejor que CentOS.

JMeter

Para las pruebas con Apache JMeter solo utilizamos uno de los dos sistemas. En mi caso he utilizado UbuntuServer por la sencillez de uso y por su eficiencia al realizar peticiones HTTP.
Los pasos que seguimos son:

Desde UbuntuServer:

1. Añadimos llave GPG para validar el repositorio
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
2. Añadimos el repositorio (todo en la misma línea)
   sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu$ (lsb_release -cs) stable"
3. Actualizamos la lista de repositorios:
   sudo apt update
4. Buscamos el repositorio de docker (community edition) y lo instalamos:
   apt search docker-ce
   sudo apt install docker-ce

5. Añadimos el usuario al grupo docker
   sudo usermod -aG docker marc
6. Reiniciamos el sistema y probamos alguno de los dos comandos siguientes
   docker info; docker run hello-world
7. Instalamos docker compose
   sudo apt install docker-compose
8. Probamos su funcionamiento
   docker-compose - -version
9. Clonamos el repositorio de David Palomar
   git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git
10. Levantamos docker compose
    cd iseP4JMeter ; docker-compose up

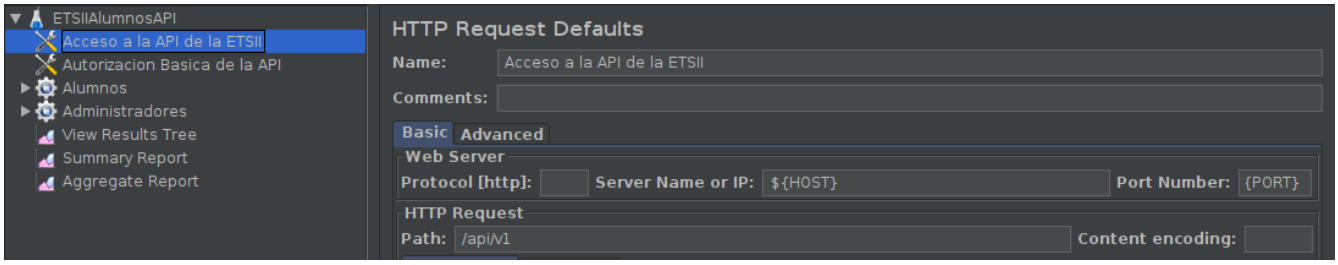
Desde nuestro host

1. Clonamos el repositorio de David Palomar (mostrado como hacerlo anteriormente).
2. Ejecutamos el guión de prueba
   chmod u+x pruebaEjecucion.sh ; ./pruebaEjecucion

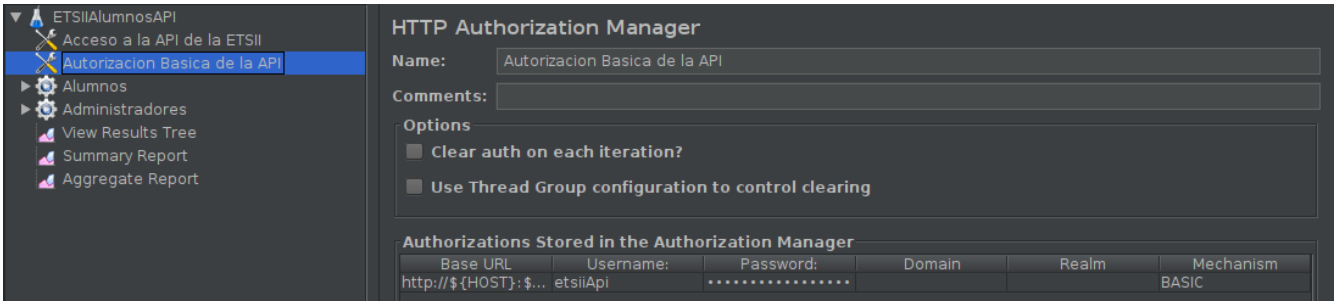
Creación del guión de Jmeter

La aplicación instalada presenta una API Rest para obtener información sobre los alumnos tras autenticarse. Para ejecutar esta aplicación utilizaremos un guión de prueba que creamos en Apache JMeter.

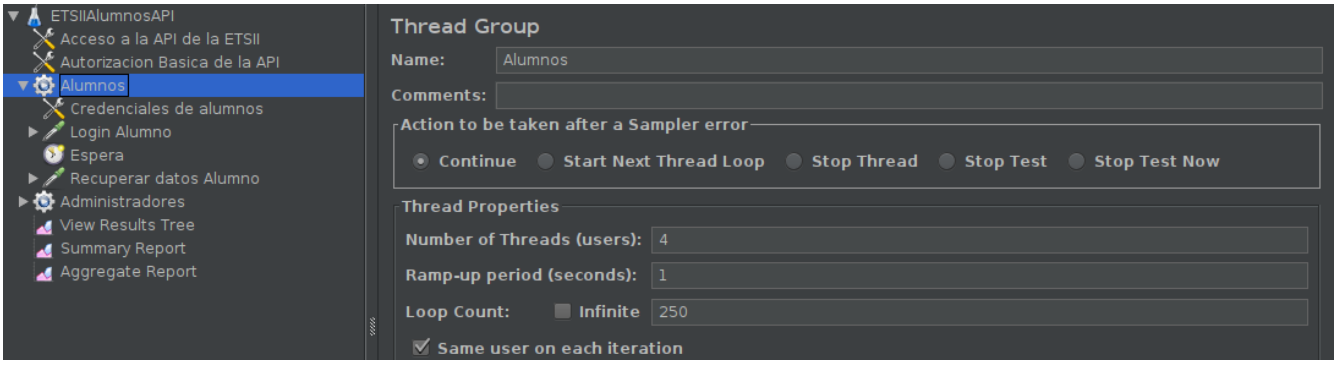
Añadimos HOST (IP de UbuntuServer) y PORT (puerto de conexión a docker)



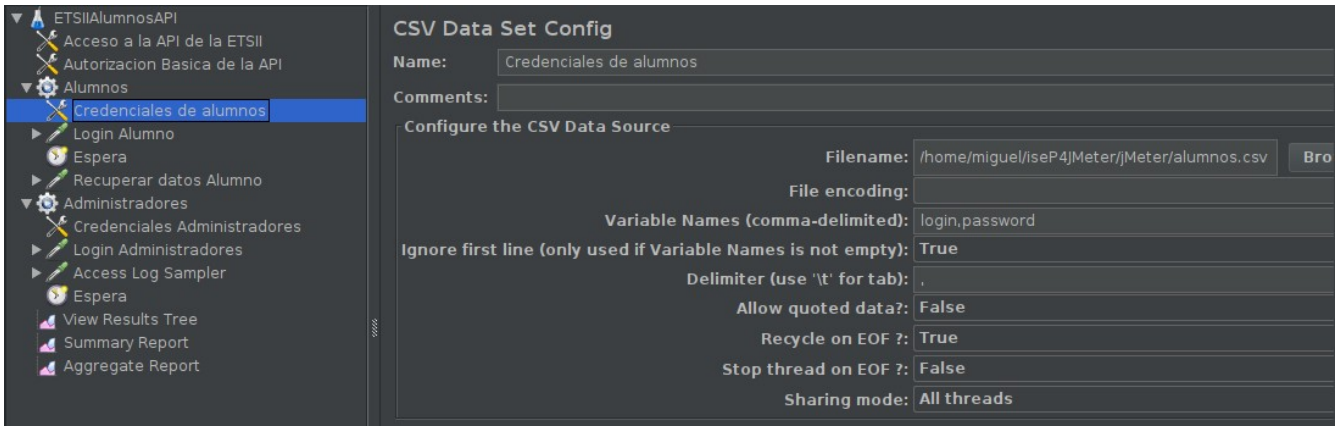
Añadimos los valores por defecto para peticiones HTTP. Insertamos el host, el puerto y la ruta para la petición HTTP



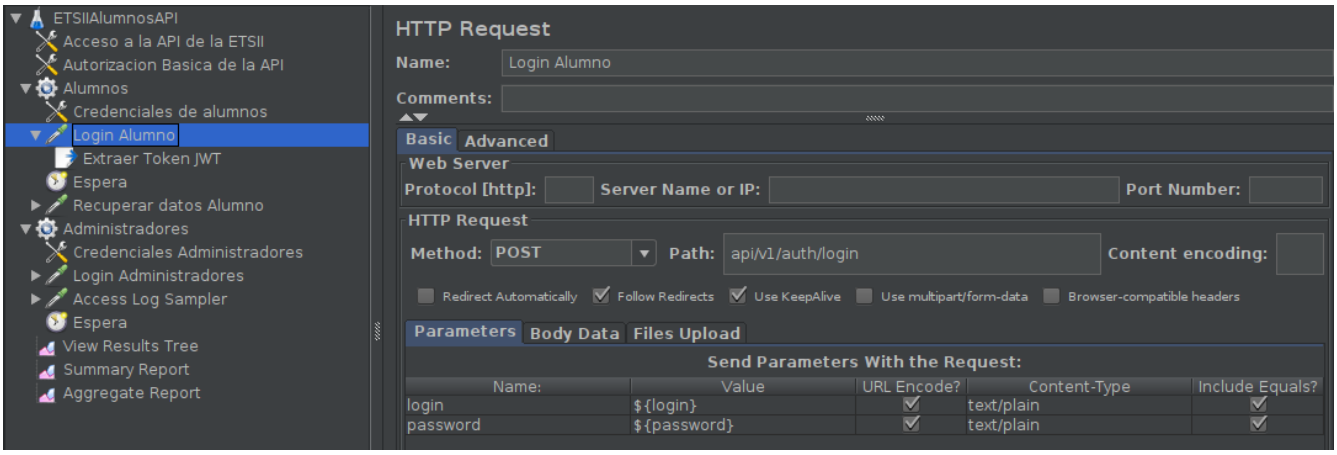
Añadimos autorización



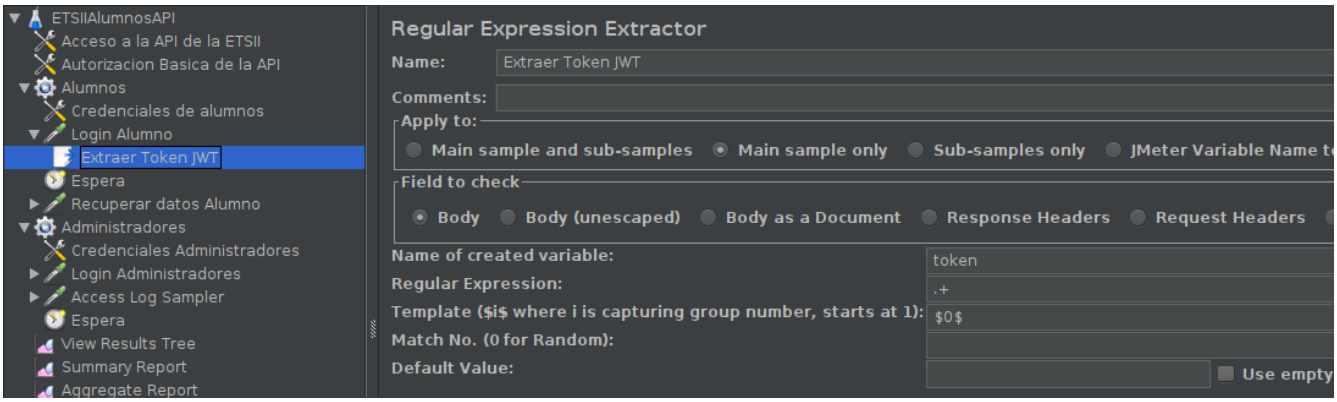
Añadimos dos grupos de usuarios: Alumnos y Administradores



Configuramos las credenciales

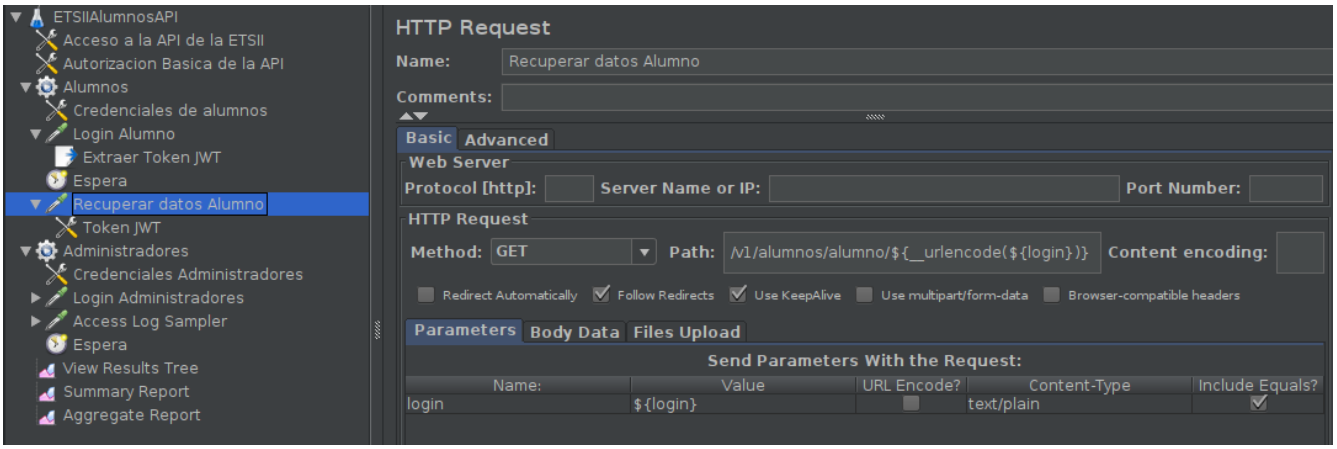


Login de los alumnos. Cambiamos el método a POST, añadimos la ruta y configuramos los parámetros de entrada

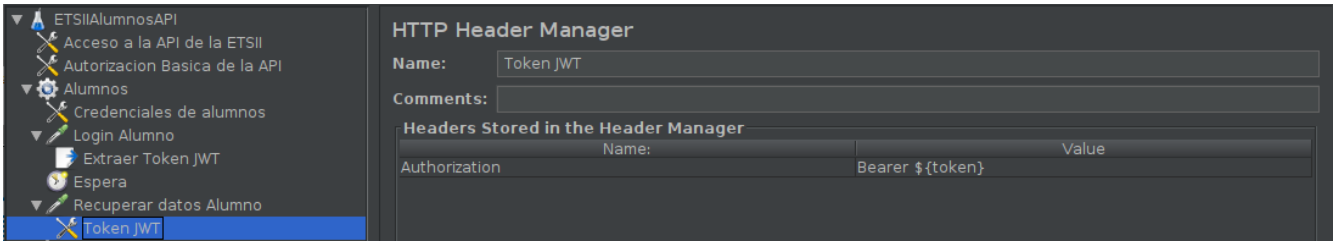


Extraemos el token JWT

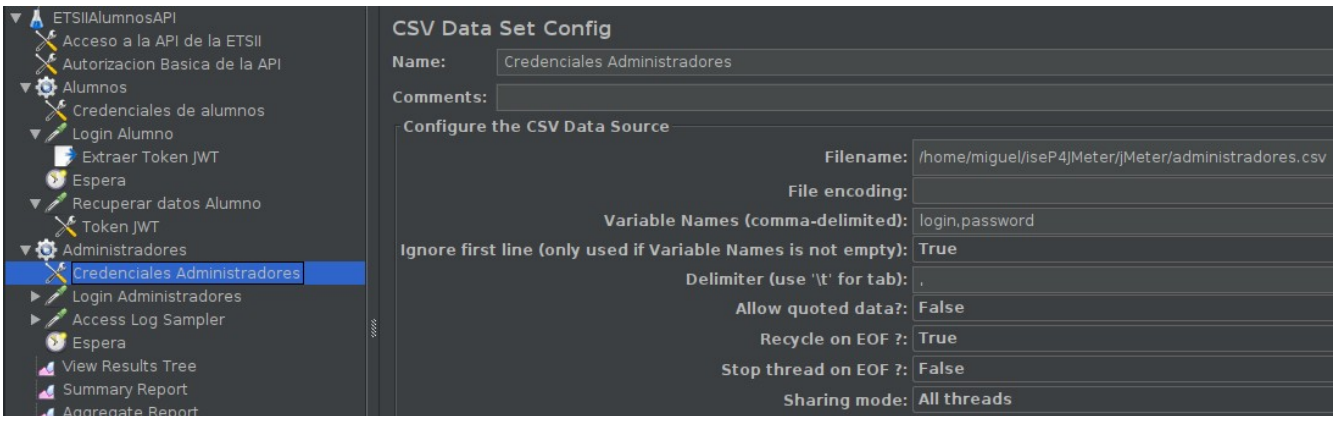
Añadimos un temporizador entre operaciones para simular que el usuario hace algo en la página

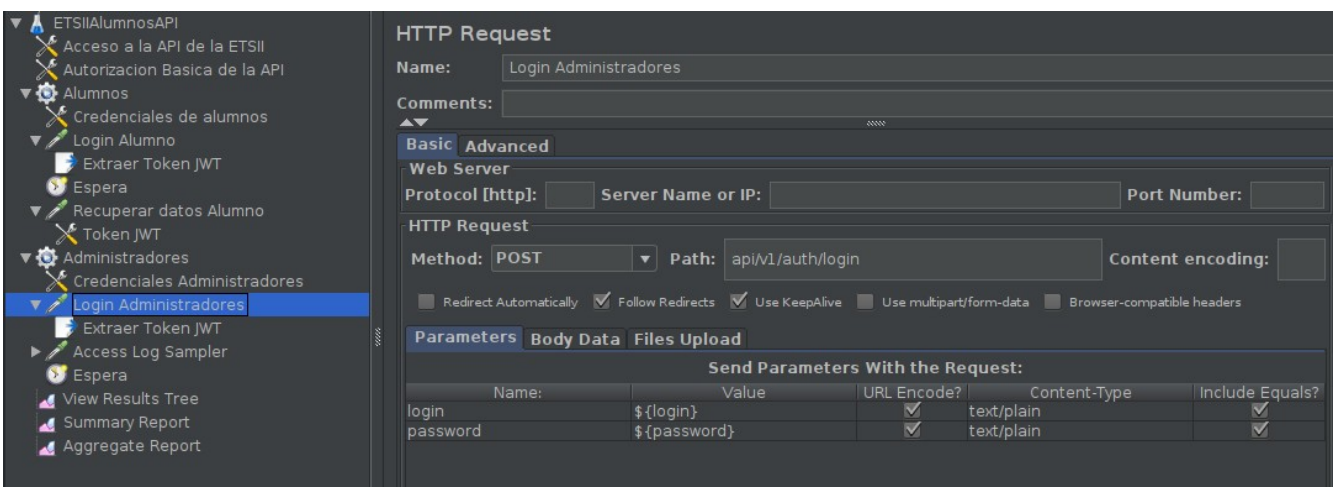


Recuperamos los datos del alumno. Añadimos la ruta y el parámetro login

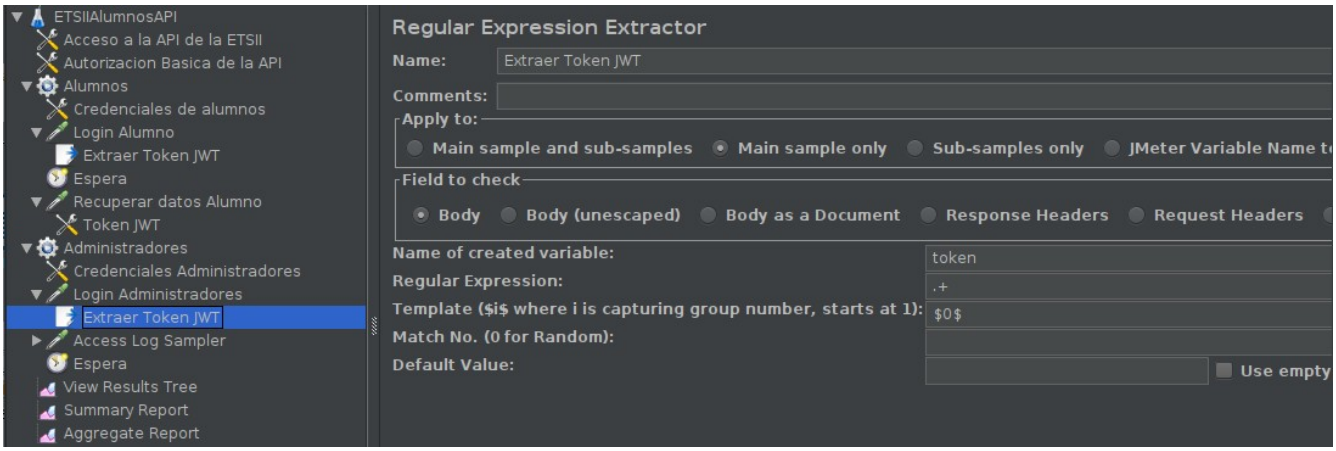


Al autenticarnos con un nombre de dominio, el token no nos serviría si la aplicación nos redirige a otro nombre de dominio. El gestor de cabecera HTTP resuelve este problema

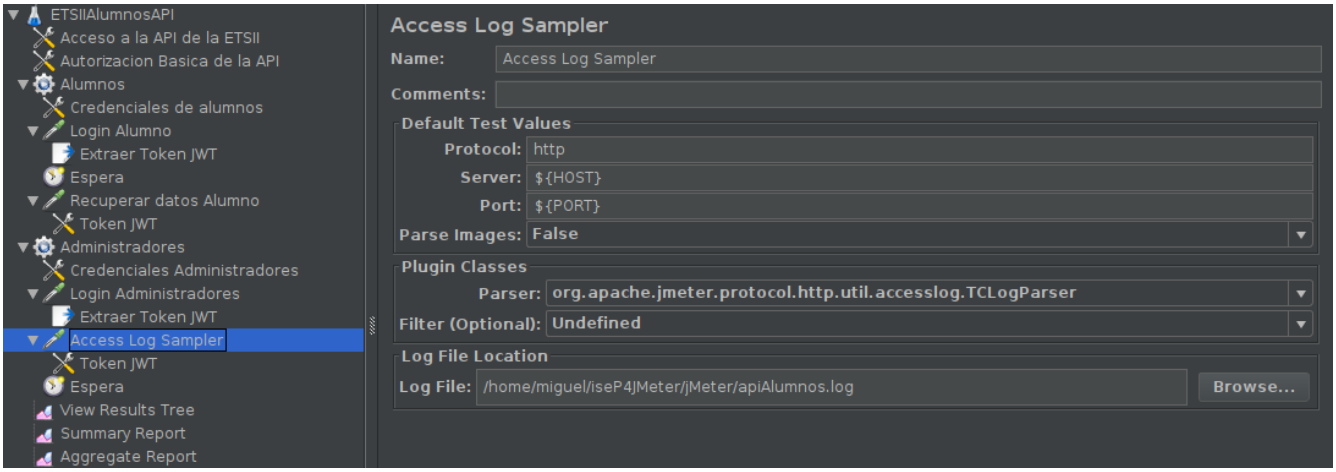




Login de Administrador. Cambiamos el método a POST, añadimos la ruta y configuramos los parámetros de entrada

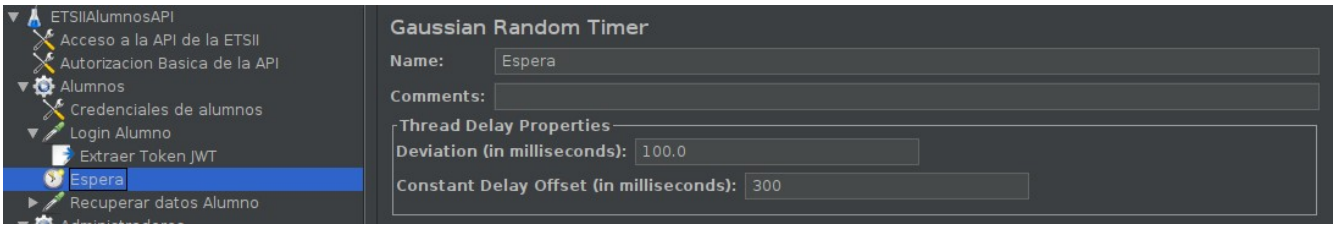


Extraemos el token JWT del login del Administrador

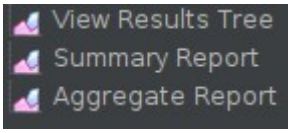


Muestreamos el archivo .log de accesos al sistema. Añadimos host y puerto y la ubicación del archivo log de la carpeta JMeter

Pasa igual que con el gestor de cabecera HTTP para alumnos

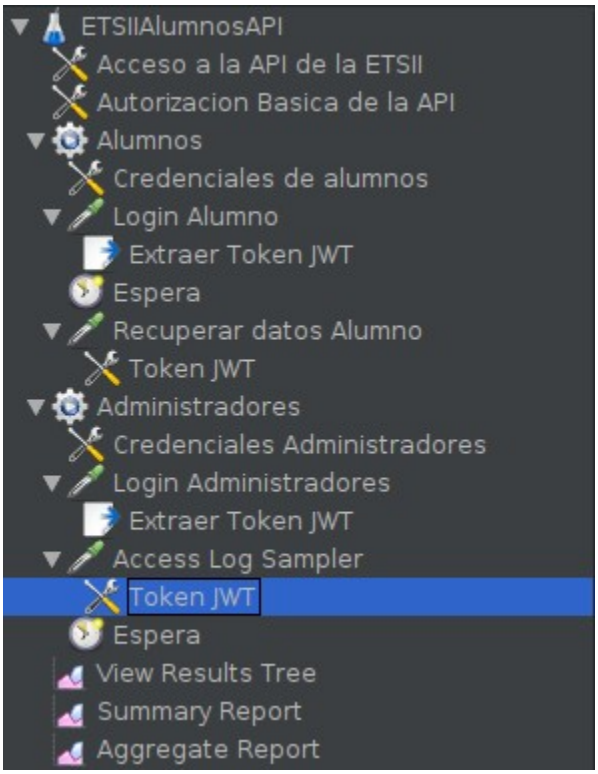


Añadimos un temporizador antes de acabar



Añadimos informes para ver los resultados.

El árbol final del archivo jmx nos debe quedar de la siguiente forma:



En el summary report la columna de errores debe tener en todas sus filas el valor 0,00%. Del mismo modo para el View Result Tree todos los símbolos deben estar en verde para indicar que el estado es correcto.