

Práctica UD3: Apache Jmeter y Tomcat con Docker

Desenvolvemento de aplicacións web

MP0614. Despregamento de aplicacións web

Sumario

Instrucciones.....	3
Apache JMeter.....	4
Instalación y funcionamiento de Tomcat con Docker/Podman.....	9

Instrucciones

- Las capturas de las máquinas virtuales deben mostrar el nombre de la máquina.
- En el nombre de la máquina virtual debe contener la inicial y el apellido del alumno/a que entrega la práctica.
 - Por ejemplo, si creo una máquina virtual llamada "vsFTPd Server", debo nombrarla "jlopez vsFTPd Server".
- Las capturas deben de tener una calidad suficiente para que su contenido pueda ser legible.
- La entrega será en la tarea de la plataforma moodle mediante un fichero pdf practica_x_tu_nombre.pdf (x es número de practica y tu_nombre es tu nombre) en el que se puedan ver en las diferentes secciones lo solicitado.

Apache JMeter

Apache Jmeter es un proyecto de software libre de Apache. Se trata de una herramienta de prueba de carga para analizar y medir el rendimiento de una variedad de servicios, con énfasis en aplicaciones web.

Tu tarea será realizar una pequeña prueba de carga contra un servidor web Apache o un servidor de aplicaciones Tomcat.

Para ello, realiza los siguientes pasos.

Instala apache Jmeter

Pistas:

- [How To Use Apache JMeter To Perform Load Testing on a Web Server | DigitalOcean](#)
- [Apache JMeter - Download Apache JMeter](#)
- [¿Cómo instalar Apache JMeter en Windows? - GeeksforGeeks](#)

Descargamos la ultima version de jmeter:

wget <https://d1cdn.apache.org/jmeter/binaries/apache-jmeter-5.6.3.tgz>

```
root@examenDespliegues:~/jmeter# wget https://d1cdn.apache.org/jmeter/binaries/apache-jmeter-5.6.3.tgz
--2024-11-18 18:53:58-- https://d1cdn.apache.org/jmeter/binaries/apache-jmeter-5.6.3.tgz
Resolving d1cdn.apache.org (d1cdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to d1cdn.apache.org (d1cdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 87414762 (83M) [application/x-gzip]
Saving to: 'apache-jmeter-5.6.3.tgz'

apache-jmeter-5.6.3.tgz      100%[=====] 83.36M  8.89MB/s  in 8.7s

2024-11-18 18:54:07 (9.57 MB/s) - 'apache-jmeter-5.6.3.tgz' saved [87414762/87414762]

root@examenDespliegues:~/jmeter#
```

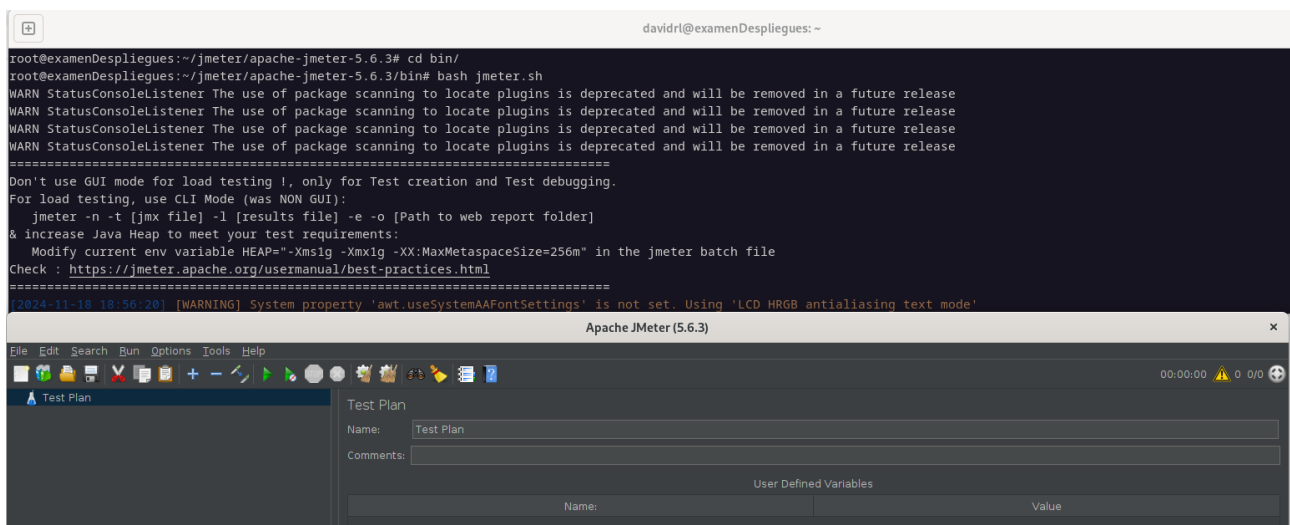
Descomprimos el paquete:

tar xf apache-jmeter-5.6.3.tgz

```
root@examenDespliegues:~/jmeter# tar xf apache-jmeter-5.6.3.tgz
root@examenDespliegues:~/jmeter# ls
apache-jmeter-5.6.3  apache-jmeter-5.6.3.tgz
```

nos movemos , a la carpeta que acabamos de descomprimir y vamos a /bin y ejecutamos el script jmeter.sh

bash jmeter.sh

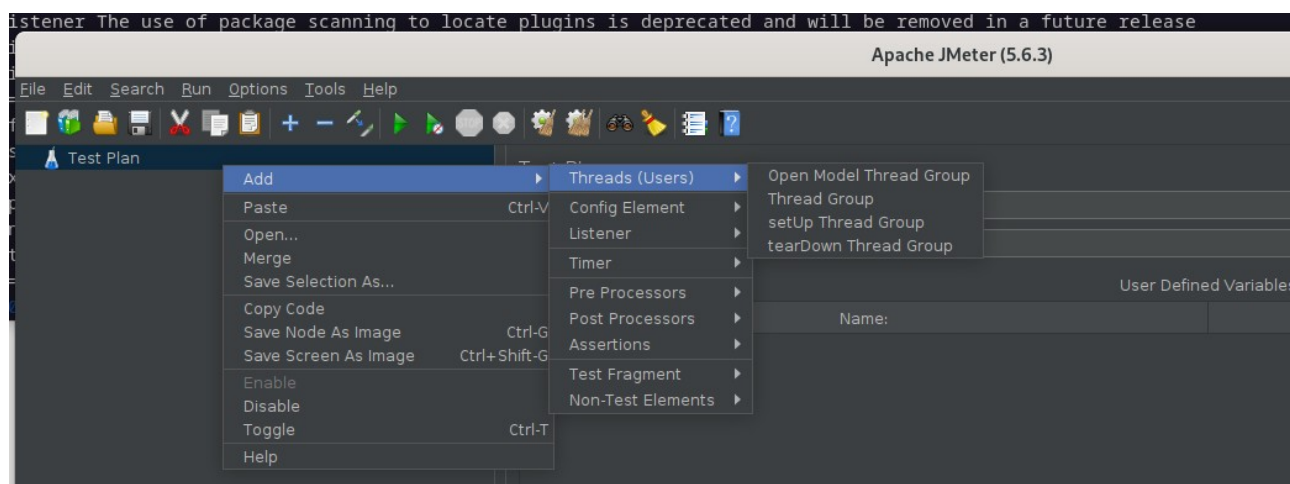


Crea un test plan básico

Pistas

- [How To Use Apache JMeter To Perform Load Testing on a Web Server | DigitalOcean](#)
- [Apache JMeter - User's Manual: Getting Started](#)
- [How to install and run Apache JMeter in 8 easy steps](#)
- [Cómo hacer pruebas de rendimiento a un servicio web con Apache JMeter](#)
- [¿Cómo hacer pruebas de rendimiento con Apache JMeter?](#)

Abrimos Jmeter y en el plan de test damos click derecho y añadimos un nuevo "Thread groups"



Se nos abra una nueva pantalla en la que principalmente debemos configurar los siguientes elementos:

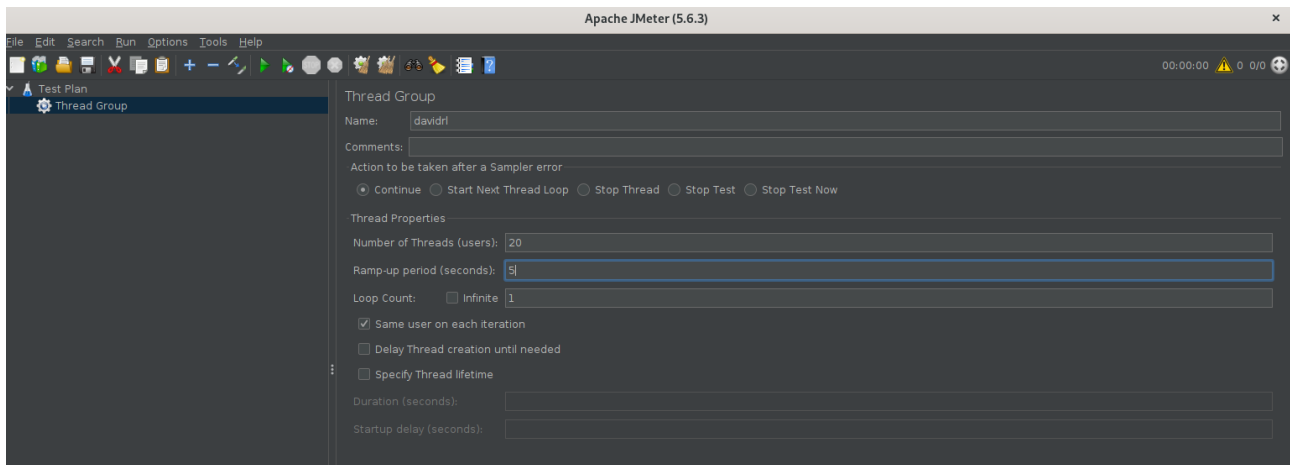
Nombre: el nombre de la prueba

Numero de amenazas(usuarios): el numero de usuarios que Jmeter intentara simular.

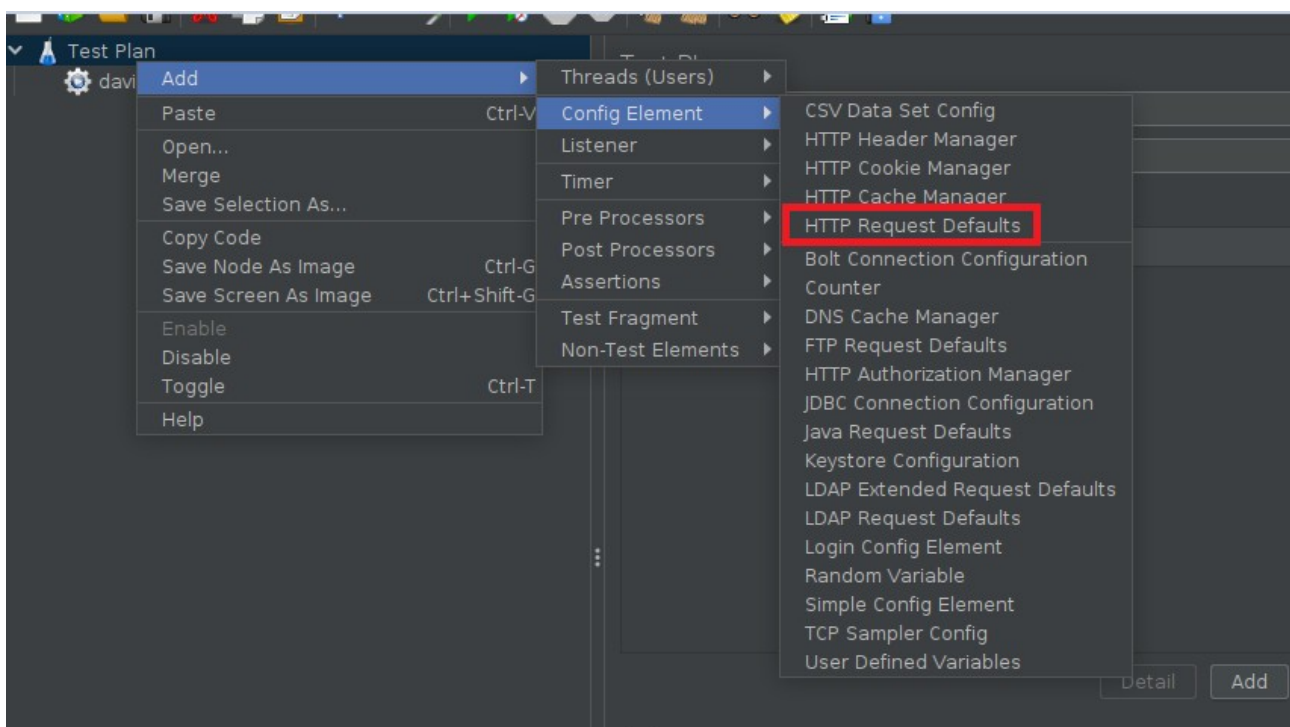
Periodo de aumento: lapso de tiempo de ejecucion de la prueba.

Cuenta en bucle: el numero de veces que Jmeter ejecutará la prueba.

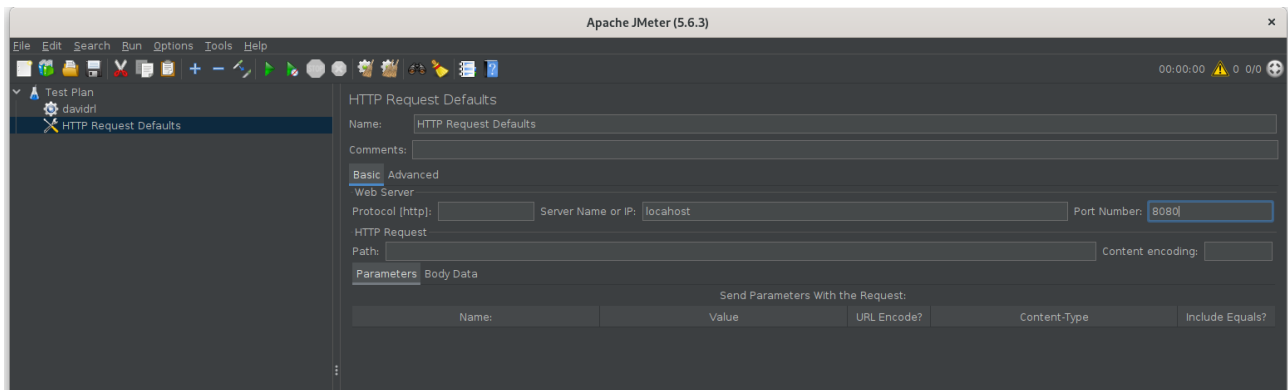
Yo lo he configurado de la siguiente manera:



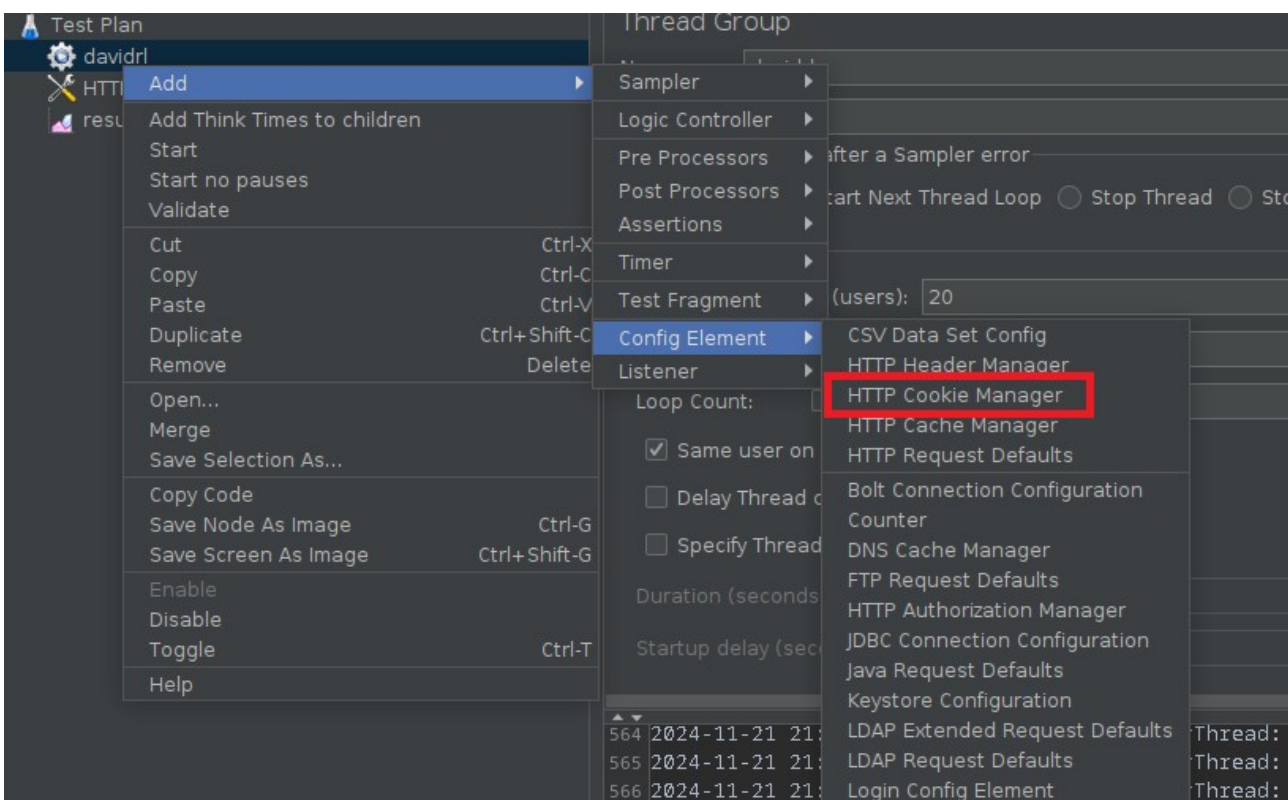
Ahora añadiremos los datos sobre donde se ejecutara el plan y el protocolo a usar, en este caso usaremos HTTP, así que damos click derecho sobre "Test plan" > add> Config Element > HTTP Request Defaults



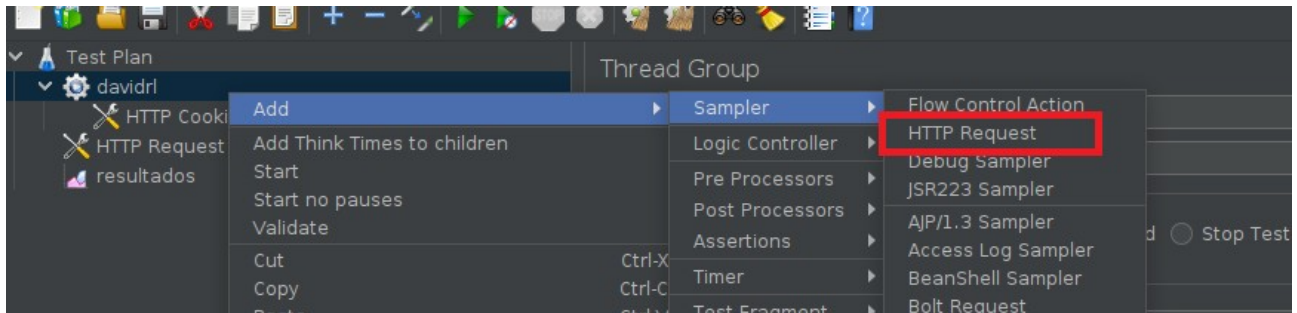
Aquí nos interesa configurar principalmente el campo que hace referencia al sitio donde ejecutaremos el plan, los campos que nos interesan son Server name y ports, yo uso localhost y puertos 8080 que es donde está mi tomcat.



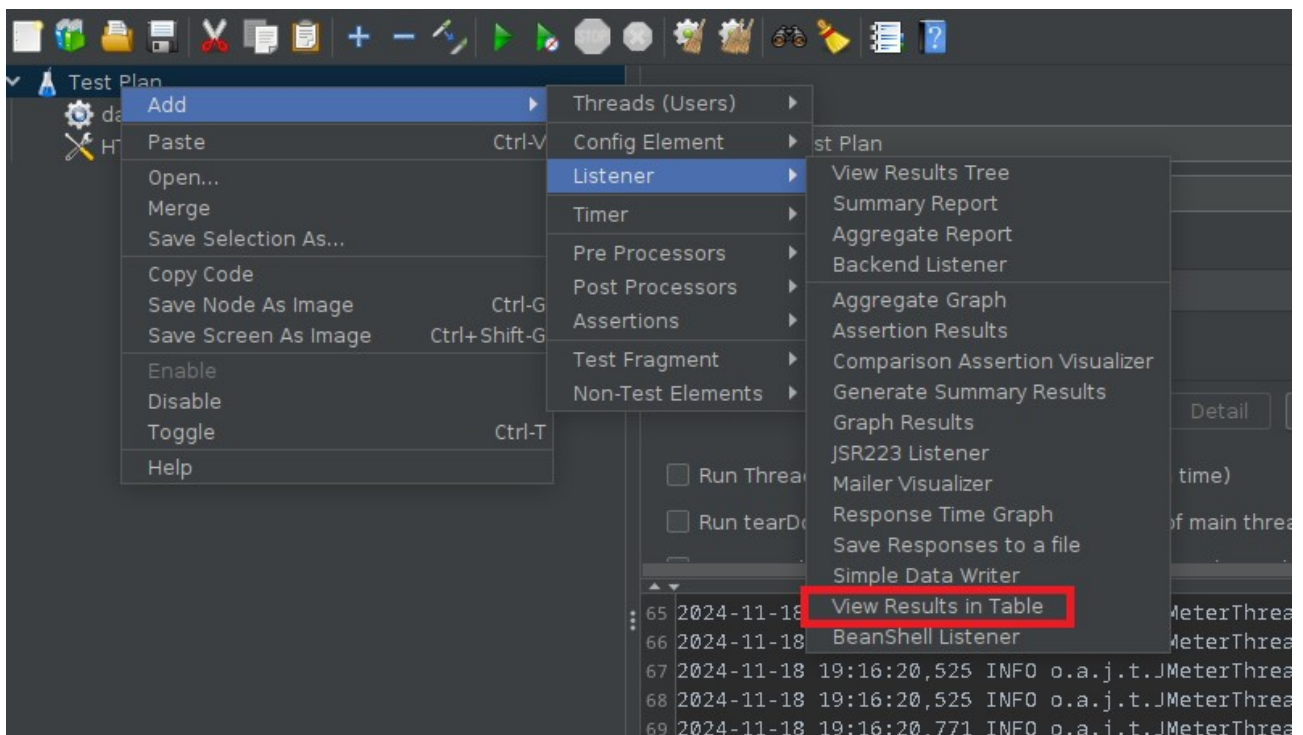
Ahora añadimos soporte para las cookies para ello, hacemos clic derecho sobre el Thread group > add > config Element > HTTP Cookie Element



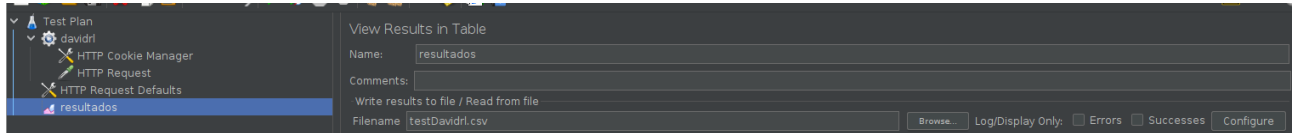
Ahora vamos a añadir un HTTP Request Sampler, para ello damos clic derecho sobre el thread group > Add > Sampler > HTTP Request



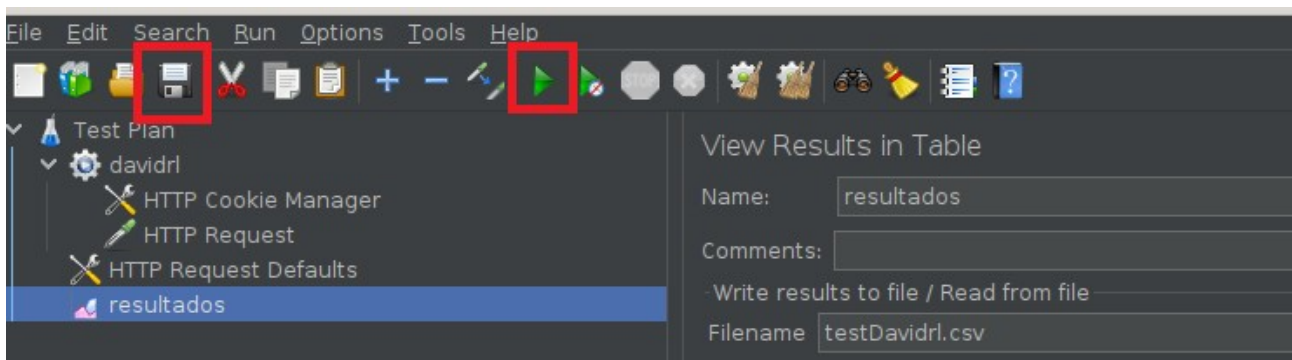
Ahora añadimos el elemento que nos permite ver los resultados del test, para ello damos nuevamente clic derecho sobre test plan > Listener > View Results in Table



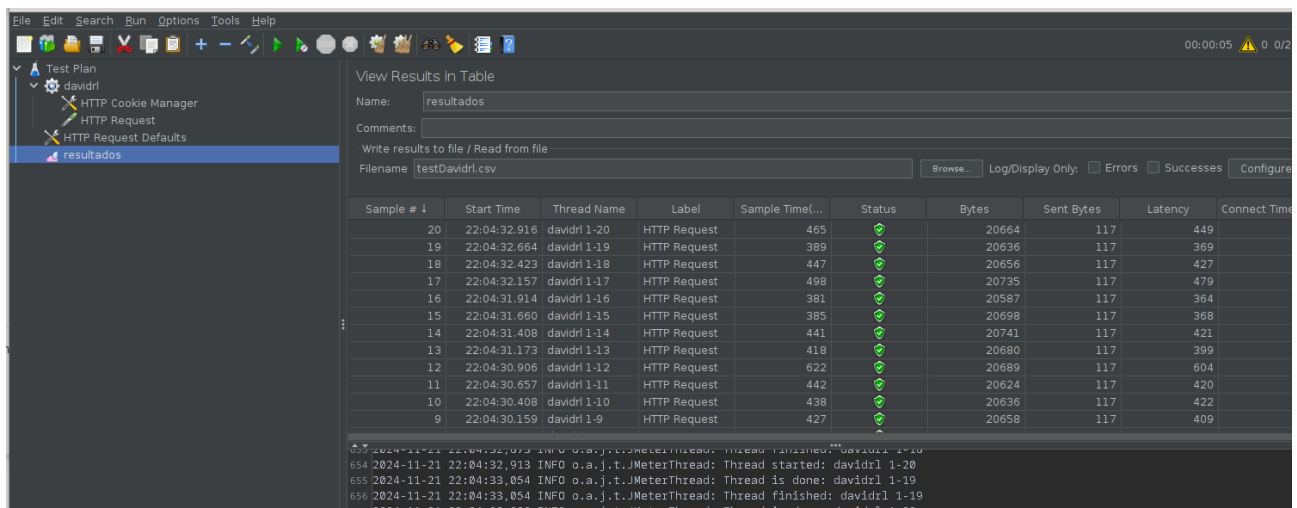
En el campo file ponemos el nombre del fichero donde se guardaran los resultados de la prueba:



Ahora damos sobre el boton de guardar y despues al boton de play para ejecutar el plan:



Una vez ejecutado, si vamos a la opcion resultados dentro de nuestro plan, podremos ver los resultados:



Ejecuta el test plan y realiza una breve interpretación de los resultados

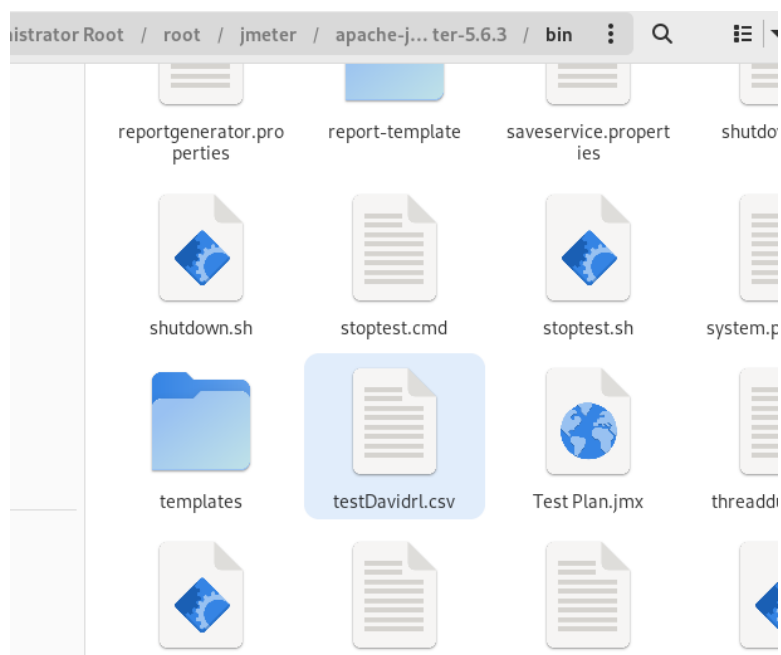
Pistas

- [How To Use Apache JMeter To Perform Load Testing on a Web Server | DigitalOcean](#)

Ahora que tenemos el plan configurado, lo ejecutamos pulsando sobre el boton de play verde.

Sample #	Start Time	Thread Name	Label	Sample Time	Status	Bytes	Sent Bytes	Latency	Connect Time
20	22:04:32.916	david1 1-20	HTTP Request	465	Success	20664	117	449	
19	22:04:32.664	david1 1-19	HTTP Request	389	Success	20636	117	369	
18	22:04:32.423	david1 1-18	HTTP Request	447	Success	20656	117	427	
17	22:04:32.157	david1 1-17	HTTP Request	498	Success	20735	117	479	
16	22:04:31.914	david1 1-16	HTTP Request	381	Success	20587	117	364	
15	22:04:31.660	david1 1-15	HTTP Request	385	Success	20698	117	368	
14	22:04:31.408	david1 1-14	HTTP Request	441	Success	20741	117	421	
13	22:04:31.173	david1 1-13	HTTP Request	418	Success	20680	117	399	
12	22:04:30.906	david1 1-12	HTTP Request	622	Success	20689	117	604	
11	22:04:30.657	david1 1-11	HTTP Request	442	Success	20624	117	420	
10	22:04:30.408	david1 1-10	HTTP Request	438	Success	20636	117	422	
9	22:04:30.159	david1 1-9	HTTP Request	427	Success	20658	117	409	

Ahora podremos ver que se ha generado el csv



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency
20	22:04:32.916	davidrl 1-20	HTTP Request	465	✓	20664	117	449
19	22:04:32.664	davidrl 1-19	HTTP Request	389	✓	20636	117	369
18	22:04:32.423	davidrl 1-18	HTTP Request	447	✓	20656	117	427
17	22:04:32.157	davidrl 1-17	HTTP Request	498	✓	20735	117	479
16	22:04:31.914	davidrl 1-16	HTTP Request	391	✓	20587	117	364
15	22:04:31.660	davidrl 1-15	HTTP Request	385	✓	20698	117	368
14	22:04:31.408	davidrl 1-14	HTTP Request	441	✓	20741	117	421
13	22:04:31.173	davidrl 1-13	HTTP Request	418	✓	20680	117	399
12	22:04:30.906	davidrl 1-12	HTTP Request	622	✓	20689	117	604
11	22:04:30.657	davidrl 1-11	HTTP Request	442	✓	20624	117	420
10	22:04:30.408	davidrl 1-10	HTTP Request	438	✓	20636	117	422

Este es el fichero del resultado del plan, tenemos los siguientes campos:

Sample# : Numero de registro de la prueba

Start Time: Fecha y hora de la prueba

Thread Name: Nombre de la prueba, contiene el nombre del plan - el loop en el que se ejecuto – el numero del usuario que ejecuto la prueba

Label: El tipo de request

Sample time(ms): La duracion de la prueba en milisegundos

Status: Nos indica si el resultado de la prueba fue satisfactorio

Bytes: Bytes recibidos del servidor

Sent Bytes: Bytes enviados al servidor

Latency: Latencia en ms de la conexión al servidor.

¿Qué tipos de pruebas se pueden hacer? ¿Contra qué servicios? ¿Qué alternativas hay a Apache Jmeter?

Pistas

- [Tutorial de pruebas de carga de JMeter para 2024 : la guía definitiva](#)
- [5 ejemplos de pruebas de carga de JMeter - LoadView](#)
- [9. Creación de un plan de prueba de servicio web Apache JMeter](#)
- [6. Creación de un plan de prueba de base de datos Apache JMeter](#)

¿Qué tipos de pruebas se pueden hacer?

-Principalmente pruebas de Rendimiento, tiempo de respuesta y escalabilidad.

¿Contra qué servicios?

Contra servicios web principalmente tales como DNS, FTP ,Apache, MySQL server...

¿Qué alternativas hay a Apache Jmeter?

Gatling, Locust, Load Runner, FunkLoad.

Instalación y funcionamiento de Tomcat con Docker/Podman

De anteriores unidades didácticas, adquirimos algunos conocimientos básicos de uso de containers. La tarea consiste en contar con un Tomcat mediante una instalación con docker (elige la versión que desees), y desplegar el fichero sample.war

Pista:

- [tomcat - Official Image | Docker Hub](#)
- [Apache Tomcat: Complete Guide to Setup and Configuration | Blog](#)

Instalamos docker:

```
apt-get install docker
```

Creamos un contenedor en docker para tomcat

```
docker pull tomcat
```

Comprobamos la imagen de docker:

```
docker images
```

```
root@examenDespliegues:/# docker images
REPOSITORY    TAG       IMAGE ID      CREATED       SIZE
tomcat        latest    f77539e7e45f  11 days ago  467MB
root@examenDespliegues:/#
```

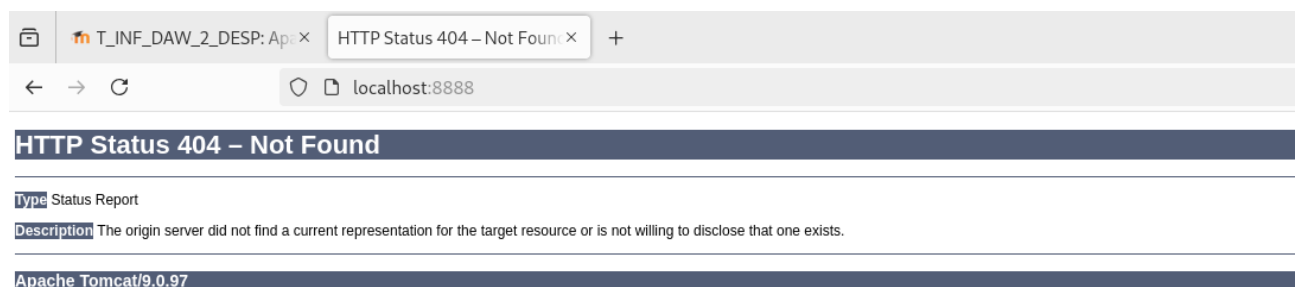
Ejecutamos la imagen que tenemos de docker:

Con el siguiente comando:

```
docker run -it --rm -p 8888:8080 tomcat:9.0
```

```
davidrl@examenDespliegues: ~  
root@examenDespliegues:/home/davidrl/Downloads# docker run -it --rm -p 8888:8080 tomcat:9.0  
Unable to find image 'tomcat:9.0' locally  
9.0: Pulling from library/tomcat  
afad30e59d72: Already exists  
918d361e6529: Already exists  
cf4dd2a7e40b: Already exists  
d152af7a0148: Already exists  
a5d7958ebd69: Already exists  
ec72214afd66: Pull complete  
4f4fb700ef54: Pull complete  
2a17c745c5af: Pull complete  
Digest: sha256:1746595b30bd1f4f9c57255f7364e6f0eb4af804db999f51c7bbdf2577f45640  
Status: Downloaded newer image for tomcat:9.0  
Using CATALINA_BASE: /usr/local/tomcat  
Using CATALINA_HOME: /usr/local/tomcat  
Using CATALINA_TMPDIR: /usr/local/tomcat/temp  
Using JRE_HOME: /opt/java/openjdk  
Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar  
Using CATALINA_OPTS:  
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
```

Ahora podremos acceder al servicio en localhost:8888



Da error 404 ya que por defecto en las webapps no tenemos nada.

Copiamos el sample.war a el contenedor:

```

root@examenDespliegues:/home/davidrl/Downloads# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
546ab0a8275e   tomcat    "catalina.sh run"       11 minutes ago Up 11 minutes  8080/tcp       crazy_payne
root@examenDespliegues:/home/davidrl/Downloads# ls
sample.war
root@examenDespliegues:/home/davidrl/Downloads# docker cp sample.war crazy_payne:/tmp/sample.war
Successfully copied 6.14kB to crazy_payne:/tmp/sample.war
root@examenDespliegues:/home/davidrl/Downloads#

```

Movemos el sample.war a la carpeta webapps dentro del contenedor:

```

root@examenDespliegues:/home/davidrl/Downloads# docker exec -it sleepy_carson sh
# ls
bin BUILDING.txt conf CONTRIBUTING.md lib LICENSE logs native-jni-lib NOTICE README.md RELEASE-NOTES RUNNING.txt temp webapps webapps.dist work
# pwd
/usr/local/tomcat
# cd /usr
# ls
bin games include lib lib64 libexec local sbin share src
# cd ..
# ls
bin boot __cacert_entrypoint.sh dev etc home lib lib64 media mnt opt proc root run sample.war sbin srv sys temp tmp usr var
# mv sample.war /usr/local/tomcat/webapps/sample.war
# cd /usr/local/tomcat/webapps/
# ls
sample sample.war
#

```

Comprobamos que se ha desplegado la aplicación:



The screenshot shows a web browser window with the title "Sample 'Hello, World' Application". The address bar displays "localhost:8888/sample/". The page content includes a cartoon cat logo and the text "Sample 'Hello, World' Application". Below this, it states: "This is the home page for a sample application used to illustrate the source directory organization of a web a". At the bottom, it says: "To prove that they work, you can execute either of the following links:" followed by two bullet points: "• To a [JSP page](#)." and "• To a [servlet](#)."