

Benchmarking

Benchmarking a nivel de sistema (phoronix) y a nivel de servicio (ab y jmeter)[gatling, locust]

A nivel de sistema

- Phoronix: ssh con -X
 - [Pasamos a root]
 - apt install phoronix-test-suite
 - apt install firefox
 - phoronix-test-suite gui
 - Leemos mensaje y ejecutamos script (añadimos permisos de ejecución antes...)
- /root/.phoronix-test-suite/web-server-launcher

A nivel de servicio

- ab (apache benchmark)
 - Conurrencia
 - ¿Qué devuelve?
- Jmeter
 - Conurrencia
 - Veamos la carga de aplicación basada en microservicios con detalle...

Instalación de Docker en Ubuntu

- Guía de digital ocean o manual de docker:
<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>
- Añadimos llave GPG para validar el repositorio:
 - curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
- Añadimos repositorio:
 - sudo add-apt-repository "deb [arch=amd64]
[https://download.docker.com/linux/ubuntu \\$\(lsb_release -cs\) stable](https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable)"

Instalación de Docker en Ubuntu

- Actualizamos lista de repositorios:
 - sudo apt update
- Buscamos el repositorio de docker (community edition) y lo instalamos:
 - apt search docker-ce
 - apt install docker-ce
- Comprobar estado del servicio (ver que está iniciado y activado...)

Instalación de Docker en Ubuntu (III)

- Añadimos el usuario al grupo docker:
 - sudo usermod -aG docker su_nombre_de_usu
 - (Volver a loguearse o llamar a bash)
- Podemos probar los comandos:
 - docker info y docker run hello-world

```
alberto@ubuntu:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adb7777e9aacf18357296e799f81cabc9fde470971e499788
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.
```

Instalación de Docker Compose en Ubuntu

- Ansible orquesta máquinas virtuales, para orquestar contenedores tenemos compose
- Instalamos (ya tenemos el repo configurado...)
 - apt install docker-compose
- Y probamos...
 - docker-compose
 - docker-compose --version

```
alberto@ubuntu:~$ docker-compose --version
docker-compose version 1.8.0, build unknown
alberto@ubuntu:~$ _
```

Instalación de la aplicación

- Nos descargamos el código
 - Podemos clonar el repo:
 - git clone <https://github.com/davidPalomar-ugr/iseP4JMeter.git>
- Accedemos al directorio con el y lanzamos la aplicación:
 - cd iseP4JMeter
 - docker-compose up

Descripción del dockerfile y compose file

- Es el archivo donde indicamos todos los comandos necesarios para crear una imagen
- Define qué incluir dentro del entorno del contenedor (red y disco virtuales)
 - Nos proporciona portabilidad del contenedor
- El compose file nos permite especificar cómo interconectar los contenedores
- Refs:
 - <https://docs.docker.com/get-started/part2/#dockerfile>
 - <https://docs.docker.com/engine/reference/builder/>
 - <https://docs.docker.com/compose/compose-file>

Sobre autenticación

- HTTP posee un mecanismo básico de autenticación introduciendo credenciales en la cabecera de la petición
 - Reducimos ruido de Internet
- La aplicación incorpora otro mecanismo: JSON Web Tokens (JWT)
 - Tiene varias ventajas frente a otros sistemas (claridad de JSON vs xml, tamaño, forma de cifrado)
 - Resuelve el problema de cross-domain

Refs:

<https://jwt.io/introduction>

<https://auth0.com/learn/json-web-tokens/>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication>

Probamos la aplicación

- Accedemos a IP:3000



ETSII Alumnos API

Descripción de la API Restful:

POST /api/v1/auth/login

Parametros:

login:<emailUsuario>
password:<secreto>

Seguridad:

Acceso protegido con BasicAuth (etsiiApi:laApiDeLaETSIIDaLache)

Retorna:

JWT Token

GET /api/v1/alumnos/alumno/<email>

Seguridad:

Token JWT valido en cabecera estandar authorization: Bearer <token>

Alumnos solo pueden solicitar sus datos. Administradores pueden solicitar cualquier alumno válido

Retorna:

Objeto Json con perfil de alumno

```
INDEX [conn6] build index on: etsii.u
{ id: 1.0 }, name: "id_1", ns: "etsii.u
INDEX [conn6] building index
to 500 megabytes of RAM
INDEX [conn6] build index done. sca
tically" : false,
mongodinit_1 | "numIndexesBefore" : 1,
mongodinit_1 | "numIndexesAfter" : 2,
mongodinit_1 | "ok" : 1
mongodb_1 | 2018-11-02T12:41:23.703+0000 I NETWORK [conn6] end connection 172.18.
0.4:49852 (1 connection now open)
mongodb_1 | 2018-11-02T12:41:23.711+0000 I NETWORK [listener] connection accepted
from 172.18.0.4:49854 #7 (2 connections now open)
mongodinit_1 | 2018-11-02T12:41:23.712+0000 connected to: mongodb
mongodinit_1 | 2018-11-02T12:41:23.713+0000 imported 20 documents
mongodb_1 | 2018-11-02T12:41:23.714+0000 I NETWORK [conn7] end connection 172.18.
0.4:49854 (1 connection now open)
lsep4jmeter_mongodinit_1 exited with code 0
nodejs_1 | GET / 200 12.754 ms - 843
nodejs_1 | GET /stylesheets/style.css 200 4.155 ms - 111
nodejs_1 | GET /favicon.ico 404 2.199 ms - 41
```

Probamos aplicación (II)

- Ejecutamos pruebaEntorno.sh

```
#!/bin/bash
SERVER=localhost
TOKEN=$(curl -s \
-u etsiiApi:laApiDeLaETSIIDaLache \
-d "login=mariweiss@tropoli.com&password=anim" \
-H "Content-Type: application/x-www-form-urlencoded" \
-X POST http://$SERVER:3000/api/v1/auth/login)
resultado=$?
if test "$resultado" != "0"; then
echo "ERROR: Curl fallo con resultado: $resultado"
fi
curl \
-H "Authorization: Bearer $TOKEN" \
http://$SERVER:3000/api/v1/alumnos/alumno/mariweiss%40tropoli.com
```

Definimos la IP (o hostname) del servidor

Hace un POST al servidor para recibir un token jwt
Haciendo un Basic Auth (opción -u) y pasando credenciales en el cuerpo con -d

Comprueba que ha habido un resultado

Realiza la petición GET incluyendo el token recibido en la llamada POST anterior y muestra el resultado

```
alberto@localhost:~/BORRAME$ ./pruebaEntorno.sh
{
  "id": "5bdc8adff751a68cb999128f2",
  "nombre": "Mari",
  "apellidos": "Fletcher Weiss",
  "sexo": "female",
  "email": "mariweiss@tropoli.com",
  "fechaNacimiento": "1992-04-04T00:00:00.000Z",
  "comentarios": "Aliquip dolor laboris ullamco id ex labore. Ipsum eiusmod ut aliquip non cillum deserunt sunt commodo anim ad nisi excepteur eu deserunt. Sit sunt proident Lorem irure irure minim adipisicing cillum. Nostrud officia in proident velit in elit sit fugiat pariatur quis ad laboris minim dolor elit. Sint velit pariatur commodo sint veniam exercitation. Duis proident minim consequat consectetur sint et tempor labore culpa esse. Exercitation laborum non esse mollit tempor ea dolor minim adipisicing molit in aliqua.\r\nUllamco adipisicing excepteur commodo sunt nulla quis sunt velit Lorem pariatur sunt ad do incididunt. In eu nostrud ullamco laboris eu minim veniam. Consequat sit et eiusmod officia ex sit minim sit laborum quis laborum labore non. Dolor nulla ut pariatur reprehenderit minim dolore consequat sunt aliquip ipsum esse. Excepteur consequat fugiat elit et nisi dolore aute minim nostrud et.\r\n",
  "cursos": [
    {"curso": 1, "media": 5.2},
    {"curso": 2, "media": 9.1}
  ],
  "usuario": 10
}alberto@localhost:~/BORRAME$
```

Probamos aplicación (III)

- Ejecutamos pruebaEntorno.sh (II) (paso a paso)

```
alberto@localhost:~/BORRAME$ SERVER=192.168.56.105
alberto@localhost:~/BORRAME$ TOKEN=$(curl -s \
> -u etsiiApi:laApiDeLaETSIIDaLache \
> -d "login=mariweiss@tropoli.com&password=anim" \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -X POST http://$SERVER:3000/api/v1/auth/login)
alberto@localhost:~/BORRAME$ echo $TOKEN
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIo... .ytf-tvfVLbJJbmthzICyqTQy5H8ows-U-S9ptkjbaA
```

The screenshot shows the jwt.io website interface. A black arrow points from the terminal output above to the 'Encoded' text input field on the left. The 'Encoded' field contains the long JWT string shown in the terminal. The 'Decoded' section on the right shows the token's structure:

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

PAYOUT: DATA

```
{  
  "sub": "10",  
  "iat": 1541243775,  
  "nbf": 1541243715,  
  "exp": 1541247435,  
  "role": "Alumno"  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

Probamos la aplicación (IV)

Request

POST 192.168.56.105:3000/api/v1/auth/login Send request

Headers ▾

Content-Type application/x-www-form-urlencoded

+Add header

Basic auth ▾

etsiiApi laApiDeLaETSIIDaLache Show password?

Request body ▾

Type URLencoded form data

login mariweiss@tropoli.com
password anim

+Add parameter

Response (0.081s) - http://192.168.56.105:3000/api/v1/auth/login

200 OK

Headers ▾

```
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
Content-Type: text/html; charset=utf-8
Content-Length: 184
ETag: W/"b8-VED0H5fCek/kPNXEkw9TrzAfHxE"
Date: Mon, 19 Nov 2018 23:04:02 GMT
Connection: keep-alive
```

Preview ▾

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ev]zdWIiOiEwLC]pYXOjOjE1NDI2Niq2NDIsIm5iZiI6MTU0MjY2ODU4MiwiZXhwIjoxNTQyNjcvMzAvLC]vb2xIjoiOWx1bW
```

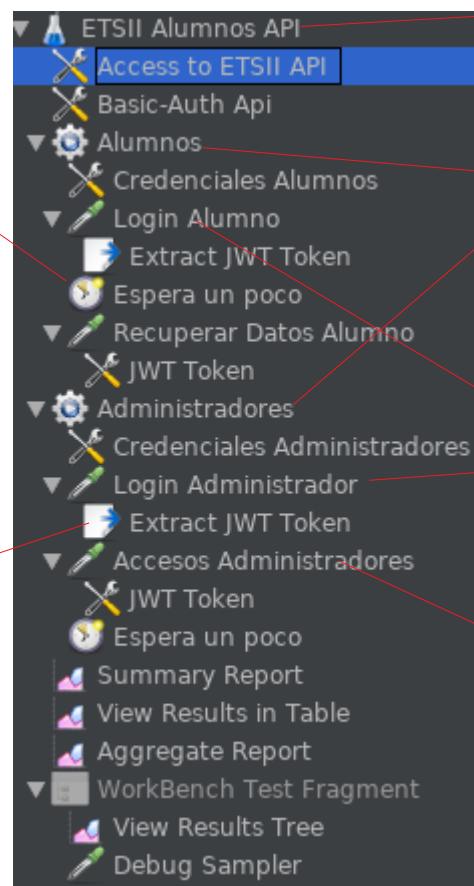
Jmeter

- Descarga:
<http://apache.rediris.es/jmeter/binaries/>
- Hacer test básico
 - <http://jmeter.apache.org/usermanual/build-web-test-plan.html>
- Grabar una sesión
 - usando la máquina local como proxy (opcional)
 - http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Proxy_Server

Trabajo autónomo

- Diseñe un test con los elementos mostrados en la imagen que estrese la aplicación instalada

Añadimos esperas aleatorias (Gaussian Random Timer)



Use una expresión regular (Regular Expression Extractor) para extraer el token JWT que hay que añadir a la cabecera de las peticiones (usando HTTP Header Manager)

Parametrice el Host y el Puerto en el Test Plan (puede hacer referencia usando \${param})

Debe hacer dos grupos de hebras distintos para simular el acceso de los alumnos y los administradores. Las credenciales de alumno y administrador se cogen de los archivos: alumnos.csv y administrador.csv respectivamente.

El login de alumno, su consulta de datos (recuperar datos alumno) y login del administrador son peticiones HTTP.

El muestreo para simular el acceso de los administradores lo debe coger el archivo apiAlumnos.log (usando un Acces Log Sampler)

