

# RENDIMIENTO WEB

## Ejercicio performance



**Autor:** Albert Delgado Roda

# Instalación en máquina local

El sistema operativo con el que realizo esta práctica es: **Fedora 25 Workstation**.

Lo primero que he hecho tras ver el repositorio, ha sido realizar un fork de este a mi cuenta de [github](#).

Al clonar el proyecto en mi local e intentar lanzar un composer install, he encontrado mi **primer problema** y es que el composer.json espera que trabaje con la version **5.6** de **PHP**. En mi máquina tengo instalada la versión **7.1.4** de **PHP**

He valorado diferentes opciones:

1. Instalarme un **PHP** switcher, que me permita elegir la versión de **PHP** con la que trabajo, y que para esta práctica me permita pasar a tener **5.6** sin tener que desinstalarme la versión actual.
2. Crear una máquina virtual y instalarle directamente la versión **5.6** de **PHP**.
3. Pensar de la siguiente manera “esto esta bajo un framework **PHP**, y se espera que el cambio de **5.6** a **7** sea mero trámite porque se espera a que no hagan uso de funciones/constructores del lenguaje que hayan sido deprecadas/eliminadas en la versión **7** de **PHP**”, por tanto indicarle a composer que la versión requerida ahora sea a partir de la **7.0**

He optado por la opción 3.

Pasos realizados:

1. En el composer.json, cambiar `"php": "~5.6"` por `"php": "~7.0"`.
2. composer update

Ahora si se han instalado las dependencias. Por tanto lo siguiente que quiero hacer es ver en mi navegador la web ejecutandose. Para ello vía terminal y estando en el la raiz del proyecto lanzo el siguiente comando:

- `php -S localhost:8000 -t web/`

Al ir al navegador y poner la dirección **http://localhost:8000/** veo que me da un error 500. Por lo que me dice el servidor de php en ejecución, hay un problema con una template de twig que no encuentra.

Trás unos pocos minutos de depuración he podido ver en realidad donde está el problema. Este se encuentra en que se requiere crear una base de datos, con una serie de tablas encontradas en el fichero dataset.sql, por tanto realizo la pertinente creación de manera manual (aun siendo consciente que para cuando se necesito desplegar en un servidor, ese paso manual debe ser automatizado, pero voy partido a partido)

Tras crear la estructura, me indica el que el servidor “mysql” es incapaz de encontrarlo, con lo cual valoro dos rápidas opciones:

1. cambiar mysql por 127.0.0.1
2. crear en /etc/hosts una referencia para mysql, evitando tener que tocar el código.

Opto por la opción 2.

Tras realizar el paso elegido, la aplicación parece ya estar en funcionamiento:

**There are no articles to show**

Hago un commit con todos los cambios realizados hasta el momento. Antes de ello veo que el directorio vendor me aparecer como commiteable, para evitar commits con cosas que no tocan, creo un .gitignore indicando precisamente eso, que ignore el directorio.

Commit realizado. **¡Ya está la aplicación arrancada en local!**

# Deployar con ansistrano

Tras haber instalado la aplicación en local, ha llegado el momento de hacer el deploy de esta con ansistrano a mi servidor de amazon.

Tras haber visto el ejemplo que hay en la cuenta github, he optado por hacer una pequeña readaptacion de la estructura de directorios dentro del proyecto git, creando dos directorios padre, llamados **deployment** y **application**, toda la configuración de ansistrano ira en deployment, en application estará el aplicativo php a deployar en amazon.

Para instalar ansistrano he lanzado el siguiente comando:

```
sudo ansible-galaxy install carlosbuenosvinos.ansistrano-deploy  
carlosbuenosvinos.ansistrano-rollback
```

El deployment en amazon siguiente la estrategia git, por tanto he instalado git en el server de amazon mediante el comando

```
sudo yum install git
```

Para conectarme a amazon necesito el .pem, por tanto he creado un directorio keys, en donde se encuentra el .pem.

Al haber configurado el host de ansible, y el playbook y haber lanzado el comando:

```
ansible-playbook -i hosts deploy.yml
```

El servidor de amazon me da un error indicando que el .pem, debería tener permisos 600 con lo cual mediante el comando `chmod 600`, cambio los permisos del .pem y vuelvo a lanzar el comando `ansible`.

Lamentablemente, creyendo que todo va bien, recibo el siguiente error:

```
fatal: [ec2-54-171-123-73.eu-west-1.compute.amazonaws.com]: UNREACHABLE! =>  
{"changed": false, "msg": "Failed to connect to the host via ssh: Received message too long  
1349281121\r\n", "unreachable": true}
```

Por lo visto el problema estaba en el fichero hosts, el `ansible_ssh_user`, era incorrecto, se estaba haciendo uso de **root**, cuando se debería de haber usado **centos**.

Tras haber realizado el cambio, parece que el proceso ansible se ejecuta correctamente, ya que todo sale en letra verde. Pero de nuevo, parece que hay algún problema, si entro al servidor amazon y voy a la ruta `/var/www/html/blog`, veo que no hay absolutamente nada, con lo que el deploy no se ha realizado.

He copiado literalmente, el playbook de ejemplo que tienen en la documentacion, he probado de ejecutar y he visto que tardaba más y luego daba error, esto ya me ha dejado mejor espina, parece que hacía procesos por detras, con lo que seguramente tenía algo mal puesto, parece que en el deploy via git, no tenía había puestos comillas, con lo que apunta a que eso podría ser el problema.

Tras hacer los reemplazos con los que había copiado del ejemplo de la documentación de ansistrano he conseguido que haga el deploy en amazon.

Si entro en el lugar donde está alojado el deploy hecho por ansistrano veré que hay un error 500, igual que me daba en local, esto sucede porque no tenemos ningún mysql configurado para este deploy.

Para esta práctica procederé a instalar mysql en el mismo servidor de amazon.

Usaremos este [rol](#) de ansible galaxy para instalar mysql en amazon.

En mi máquina local he ejecutado el siguiente comando:

```
sudo ansible-galaxy install geerlingguy.mysql
```

Tras pensarlo detenidamente, he estos pasos manuales, los podriamos obviar, por tanto he procedido a crear roles en ansible, que me automatizen las siguientes tareas:

1. instalacion del repositorio epel (para poder instalarme php 7)
2. php 7.1
3. apache, incluye un reinicio de este, por si en el paso de php7.1 añado alguna extensión que no estaba de inicio
4. composer
5. git (instalo git porque como he mencionado antes, el deploy con ansistrano es mediante git)

6. La aplicacion php tiene puesto 'mysql' como host, por tanto para automatizar al máximo esto, en ansible he añadido un comando para que en el fichero /etc/host añada mysql 127.0.0.1
7. Cuando tenía deployado la aplicación en amazon, **PDO** lanzaba una excepción indicando "Permission denied" código: 1002, tras investigar, acabé en este post de [stackoverflow](#). De este post salió otro rol para ansible que se encargaba de activar mediante **setbool** el flag **httpd\_can\_network\_connect**
8. Mediante el rol de ansible galaxy **geerlingguy.mysql** instalo mysql, incluyendo la creación de la base de datos y el usuario que la aplicacion **PHP** espera.
9. Deployo en amazon centos, con **ansistrano**. Al acabar el deploy, mediante un hoot de after symlink, ejecuto composer install en el directorio current (donde se encuentra el código de la aplicación **PHP**).
10. Creo las tablas mysql que la aplicación esperaba tener de buen inicio.

En este proceso he realizado varios **commits**.

Hay una serie de procesos que voy a a realizar de manera manual, para esta práctica porque no he encontrado una manera que se me acabe de gustar y entiendo que no debo bloquearme con esto ya que no es lo mas puntuable para la práctica, los procesos manuales son los siguientes:

1. He añadido un .htaccess en el proyecto silex, para que este en apache funcionen bien las urls.
2. He cambiado el fichero httpd.conf que configura el document root para cambiar **AllowOverride None** por **AllowOverride All**, para que el htaccess creado anteriormente tenga su efecto.
3. Quitar la opción **Indexes**, para evitar que alguien pueda ver el contenido de los directorios web. El objetitvo es forzar el error **403**.

# Usar bootstrap

Proseguiré con el apartado de utilizar bootstrap para hacer más bonito el diseño del blog, para llevarlo a cabo, lo haré en el entorno local (para algo lo he integrado!), el primer escollo a solucionar ha sido como acceder a los recursos tipo css,javascript desde la plantilla twig.

Por lo visto era mucho mas simple de lo que me imaginaba, puesto que los assets estaban en el mismo directorio **web**.

He creado dos ficheros twigs mas, que he llamado **header** y **footer**, donde respectivamente he puesto los ficheros css y los ficheros javascript, debo decir que el layout lo he trabajado de una manera muy básica, he usado lo más basico de bootstrap, los formularios son quizás donde se vea más bonito. Todas las pantallas hacen uso mediante “includes” del footer y del header.

En este punto realizo un nuevo **commit**.

Tras esto realizo un nuevo deployment, para verlo en el server de amazon.

He encontrado un problema al ejecutar la web en amazon, la ruta hacia los static assets era incorrecta, con lo que he borrado **{{app.request.basepath}}**, que no aporta nada en este caso. Tras hacer el cambio y realizar el deployment, todo ha funcionado como debía.

## Configurar load balancer

Antes de proseguir con el minificado de **bootstrap** y **jquery** (la de css no porque solo uso y es directamente una versión minificada), con la subida de imagen a S3 y la instalación de Redis, paso a configurar al load balancer. El motivo de por cual adelanto este paso, es que como he automatizado el 95% de los pasos para deployar el codigo y provisionar las máquinas amazon, no me cuesta nada generar un segundo ya el segundo **frontal**.

Pasos realizados.

1. Crear otra instancia de **EC2** para generar el deployment de la web con su provisionamiento, además de los pequeños pasos manuales relacionados con el apache mencionados anteriormente.
2. Con ansible crear mas hosts, para indicar que se provisiona en un server u otro, ahora en caso de los frontales, pasamos a quitar todo lo relacionado con mysql.
3. Crear otra instancia de **EC2** que es la que contendrá mysql.
4. Entra al servidor que teniamos y quitar de /etc/hosts, la linea 127.0.0.1 mysql, dado a que como el server mysql pasa a estar en un server aislado, eso deja de ser correcto. En el siguiente provisionamiento, los frontales tendrán la ip del server de mysql en vez del 127.0.0.1, en el rol de mysql que he hecho se ve el cambio(es decir la ip)

5. Tras realizar el deployment de todo esto y los provisionamientos en cuestión, la aplicación del frontal 1 y frontal 2 no funcionan, y esto es así (y ya me temia que sería así) porque mysql no permite remote connections para el usuario root. Por tanto para suprimir esta restricción, he ido al servidor que contiene mysql, y mediante la cli he ejecutado el siguiente comando: **GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'root';** Gracias a este paso (manual, por cierto), los dos frontales ya han vuelto a comunicarse con la base de datos mysql. Cabe destacar que soy consciente que ese grant en un entorno real podría ser excesivamente peligroso, por no decir que es lo mas cercano aun suicidio tecnológico.
6. Al ver que de manera separada ya funcionaban los frontales, he configurado el load balancer de amazon con dos instancias, **frontal1 y frontal2**.

El acceso al **frontal1** es mediante:

[ec2-34-253-186-64.eu-west-1.compute.amazonaws.com/blog/current/application/web](https://ec2-34-253-186-64.eu-west-1.compute.amazonaws.com/blog/current/application/web)

El acceso al **frontal2** es mediante:

[ec2-54-194-219-31.eu-west-1.compute.amazonaws.com/blog/current/application/web](https://ec2-54-194-219-31.eu-west-1.compute.amazonaws.com/blog/current/application/web)

El acceso via **load balancer** es:

[practicaperformance-341312800.eu-west-1.elb.amazonaws.com/blog/current/application/web/](https://practicaperformance-341312800.eu-west-1.elb.amazonaws.com/blog/current/application/web/)

El **load balancer** es la que usaré a partir de ahora para realizar todas pruebas, etc.

## Generar bundle JS

Una vez ya está funcionando el load balancer y tengo formalizado cual será el acceso a la web, procedere a realizar el bundle que contendrá jquery y bootstrap, pasando de 2 files a 1. Pero antes, mediante apache Apache AB hare un test contra la web.

El comando ejecutado ha sido:

**ab -n 100 -c 10 practicaperformance-341312800.eu-west-1.elb.amazonaws.com/blog/current/application/web/**

Los resultados han sido:



**This is ApacheBench, Version 2.3 <\$Revision: 1757674 \$>**

**Copyright 1996 Adam Twiss, Zeus Technology Ltd, <http://www.zeustech.net/>**

**Licensed to The Apache Software Foundation, <http://www.apache.org/>**

**Benchmarking practica**performance-341312800.eu-west-1.elb.amazonaws.com** (be patient).....done**

**Server Software:** Apache/2.4.6

**Server Hostname:** **practica**performance-341312800.eu-west-1.elb.amazonaws.com****

**Server Port:** 80

**Document Path:** /blog/current/application/web/

**Document Length:** 476 bytes

**Concurrency Level:** 10

**Time taken for tests:** 53.641 seconds

**Complete requests:** 100

**Failed requests:** 0

**Total transferred:** 69100 bytes

**HTML transferred:** 47600 bytes

**Requests per second:** 1.86 [#/sec] (mean)

**Time per request:** 5364.131 [ms] (mean)

**Time per request:** 536.413 [ms] (mean, across all concurrent requests)

**Transfer rate:** 1.26 [Kbytes/sec] received

**Connection Times (ms)**

**min mean[+/-sd] median max**

**Connect:** 49 67 28.0 59 158

**Processing:** 5121 5282 146.0 5252 5642

**Waiting:** 113 242 107.2 224 467

**Total: 5175 5349 158.0 5301 5706**

**Percentage of the requests served within a certain time (ms)**

**50% 5301**

**66% 5363**

**75% 5393**

**80% 5512**

**90% 5698**

**95% 5699**

**98% 5706**

**99% 5706**

**100% 5706 (longest request)**

Tras tener algo con lo cual comparar, ha llegado el momento de hacer el proceso de concatenación.

He instalado lo siguiente para poder hacerlo:

1. NodeJS
2. npm
3. concat (para poder concatener jquery.min.js y bootstrap.min.js)
4. Script en el package.json denominado “concat” donde realizo el proceso

Ahora para automatizar este proceso en los frontales, debo generar mediante tareas ansible, además de ello, en el after-symlink process de ansistrano, además de lanzar el compose install, lanzare el comando npm **install** y npm **concat**

He instalalado el rol **geerlingguy.nodejs**

Ahora tras haber realizado el deploy y que todo funciona como se espera, vuelvo a lanzar el test con Apache AB.

**Copyright 1996 Adam Twiss, Zeus Technology Ltd, <http://www.zeustech.net/>**

**Licensed to The Apache Software Foundation, <http://www.apache.org/>**

**Benchmarking practicaperformance-341312800.eu-west-1.elb.amazonaws.com (be patient).....done**

**Server Software:** Apache/2.4.6  
**Server Hostname:** practicaperformance-341312800.eu-west-1.elb.amazonaws.com  
**Server Port:** 80

**Document Path:** /blog/current/application/web/  
**Document Length:** 354 bytes

**Concurrency Level:** 10  
**Time taken for tests:** 54.352 seconds  
**Complete requests:** 100  
**Failed requests:** 0  
**Total transferred:** 56900 bytes  
**HTML transferred:** 35400 bytes  
**Requests per second:** 1.84 [#/sec] (mean)  
**Time per request:** 5435.215 [ms] (mean)  
**Time per request:** 543.522 [ms] (mean, across all concurrent requests)  
**Transfer rate:** 1.02 [Kbytes/sec] received

#### Connection Times (ms)

	min	mean[+/-sd]	median	max
<b>Connect:</b>	48	56 7.5	52	78
<b>Processing:</b>	5323	5378 21.9	5376	5439
<b>Waiting:</b>	268	352 29.4	351	414
<b>Total:</b>	5391	5434 21.9	5427	5502

#### Percentage of the requests served within a certain time (ms)

50%	5427
66%	5428
75%	5430
80%	5439

90% 5490  
95% 5494  
98% 5494  
99% 5502  
100% 5502 (longest request)

## Instalar Redis

Al igual que con MySQL, voy a crear otra instancia de EC2, que contendrá exclusivamente un server de Redis, para ello voy a crear un nuevo rol que se encargue de instalar Redis. Como con MySQL ya he tenido el problema de la conexión remota, he adelantado la investigación para permitir conectarse a redis desde cualquier sitio ( un paso manual, y que no recomendaria para una aplicacion real)

La linea cambiada ha sido bind 127.0.0.1 del fichero /etc/redis.conf, por bind 0.0.0.0, para garantizar que ese cambio fuese válido, he reiniciado Redis, y desde mi máquina local y usando Redis Desktop Manager me he conectado al server que contiene Redis.

El server Redis es el siguiente: **ec2-176-34-134-152.eu-west-1.compute.amazonaws.com**

Momento en el cual hago otro commit.

# Redis Session

Para hacer que la aplicación Silex manipule las sesiones contra un servidor Redis en vez de en disco, he procedido a realizar los siguientes vendor via composer:

1. snc/redis-bundle
2. predis/predis

Al igual que con MySQL, para no tener que jugar con las configuraciones en relación a la IP donde se encuentra Redis, en mi máquina local en el fichero /etc/hosts he añadido:

127.0.0.1 redis

Via RedisDesktopManager he comprobado que guardase la session en la base de datos 0.

Tras conseguir esto, procedo a realizar otro commit.

Ahora para hacer el deploy en amazon y garantizar que esto funcione, mediante ansible, al igual que hice con MySQL, hay que crear un registro en el fichero /etc/hosts de los frontales con la ip del server de amazon que contiene Redis:

**176.34.134.152 redis**

Hago un commit con los cambios.

# Subir imagen S3

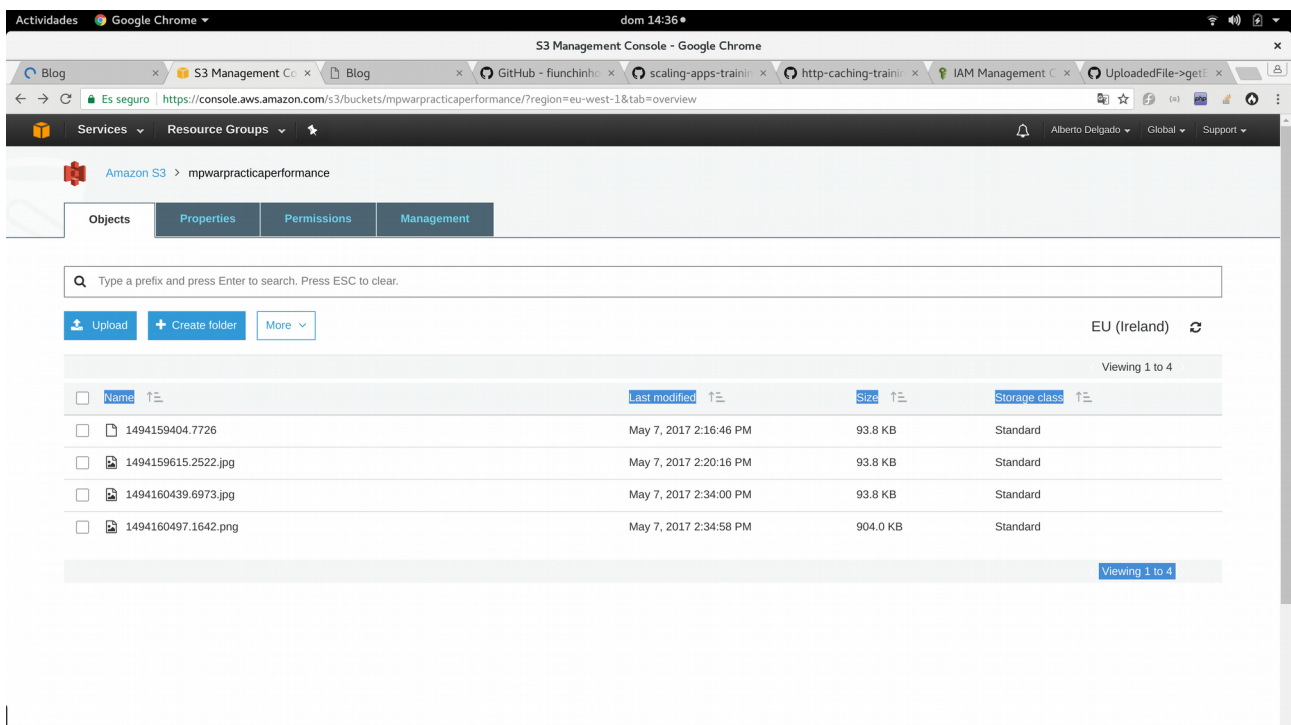
Tal y como me hicimos en clase, he creado un Buck que en este caso he llamado **mpwarpracticaperformance**.

Ahora en la aplicación Silex mediante composer he instalado las siguientes dependencias:

1. league/flysystem
2. league/flysystem-aws-s3-v3

Tras haber conseguido subir imagenes al S3 (el código es en plan guerrillero, con un cuchillo en la boca, al estilo los simpson).

Ahora voy a probar el deploy en Amazon, y ver si el código picado sigue funcionando en este nuevo entorno.



Ha habido un problema y es que en el servidor de Amazon no tenía instalado la extension php-xml, despues de instalarlo, todo ha continuado funcionando con normalidad.

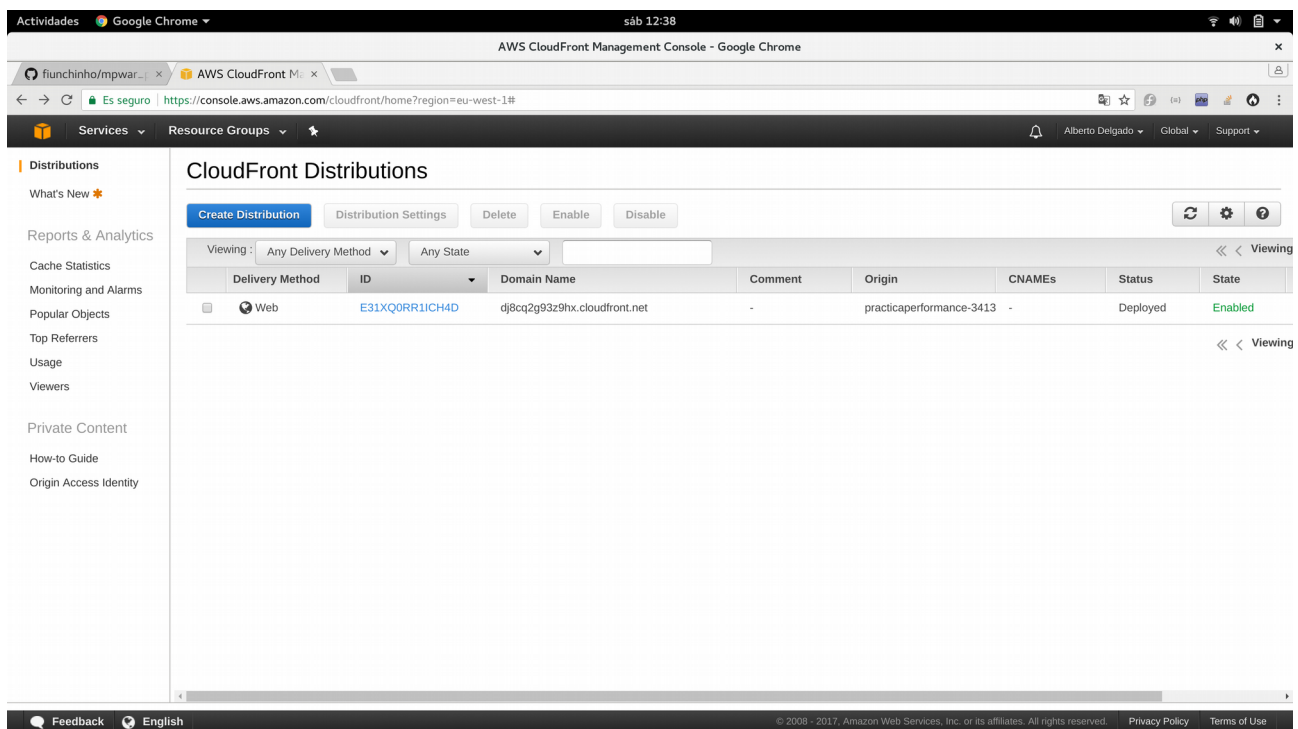
# Cloudfront

Tras configurar el cloudfront, he sido víctima de la incertidumbre, es decir ha tardado un rato en ser operativo.

He añadido una nueva variable de configuración en prod.php y dev.php, que son utilizadas en el header y footer para poner la ruta hacia los js y css correspondientes.

La dirección de la cloudfront es: **<https://dj8cq2g93z9hx.cloudfront.net>**

Una captura del panel:



En este punto donde la práctica esta tan avanzado, he empezado a probar todas las secciones en la aplicación deployada en amazon, tras ir jugueteando he visto que se generaba un error 500 que en local no sucedía. (en local funciona!). Tras mirar en el log de errores de apache del frontal1(es el que he usado para depurar), he acabado en un post de stackoverflow, he optado por implementar la solución que ofrece Dev\_Nix.

Tras hacerlo, todo ha empezado a funcionar perfectamente, con lo que la frase “en local funciona” le he añadido la coletilla “ y en prod también”

# Cachear artículos en Redis

Para realizar el cacheo de la query de obtención de un artículo he hecho uso del siguiente vendor:

**predis/service-provider**

En el controller **ReadArticle**, cacheo el objeto de manera serializada, sin ponerle ningún tiempo de expiración, invalido la cache en el momento en que es editado mediante **EditArticle**. No ha habido mucho problema, solo en decidir que vendor utilizar, etc, el código que he escrito no es precisamente digno de elogio, pero cumple con su cometido para la práctica. (o eso percibo).

Ahora haré un deploy para garantizar que en amazon funciona correctamente. Ha sido un deployment exitoso.

Es el momento de enfrentarse al ranking top 5 de los mas leídos.

He estudiado un poco el siguiente repo de [github](#) para hacer la tarea.

Hago primero un commit del ranking global

Hago otro commit con el ranking por usuario autenticado.

## Blackfire

Para la instalación de blackfire, he seguido los [docs](#) para sistemas **redhat**, en ansible he automatizado todos los pasos hasta llegar al blackfire register (donde pide key y secret), este proceso lo he hecho manualmente en los dos frontales. El proceso es el que indica en la web.

Ha sido satisfactorio tanto en local, como en amazon.

Lo que tambien voy a automatizar es la instalación de **PHP Probe**

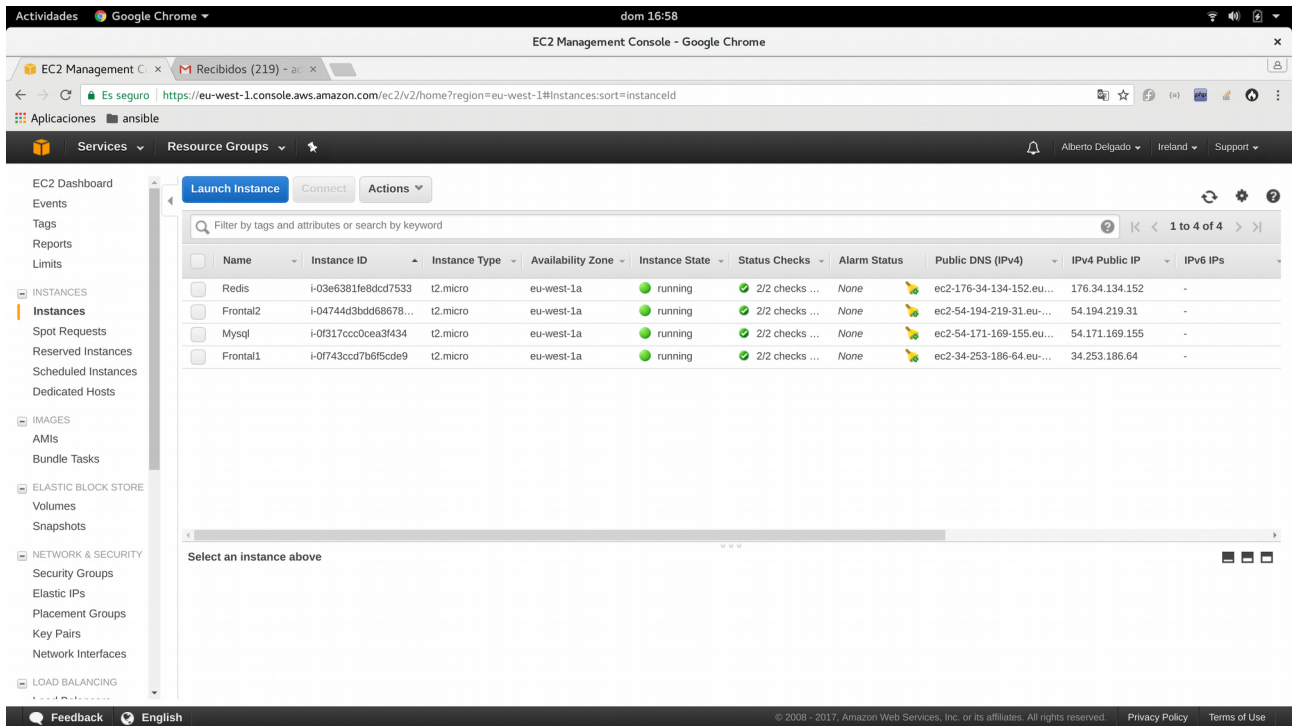
En este punto en el que todo parecía un camino de rosas, no ha resultado serlo, me he topado con varios problemas para conseguir realizar un profiling, no funcionaba ni con extension de chrome, ni utilizando el blackfire-client directamente desde el server que quería medir. Tras haber seguido varias recomendaciones por parte de la web de blackfire y ver que seguía sin ir, hablando con José Armesto, llegamos a la conclusión de que pudiese ser Centos el obstáculo, con lo que lo probé en Ubuntu, y el resultado fue completamente satisfactorio. Viendo que en Ubuntu funcionaba, recordé que selinux es algo que contiene tanto Centos como Fedora (mi sistema operativo local), por tanto había grandes posibilidades de que fuese el motivo del error. Y así fue, fue desactivarlo y empezar a funcionar, en un ejemplo real, no deshabilitaria selinux, sino que investigaria detalladamente cual es su cometido, y en cual de estos blackfire se ve impedido en realizar su función.

[Perfil público blackfire](#)

Último commit!



# Captura de pantalla de las instancias EC2 utilizadas.



The screenshot displays the AWS Management Console for EC2 instances in the eu-west-1 region. The interface includes a left-hand navigation menu with categories like INSTANCES, IMAGES, ELASTIC BLOCK STORE, NETWORK & SECURITY, and LOAD BALANCING. The main content area shows a table of four running EC2 instances: Redis, Frontal2, Mysql, and Frontal1. Each instance is a t2.micro type in the eu-west-1a availability zone. The table also shows status checks, alarm status, public DNS (IPv4), IPv4 public IP, and IPv6 IPs. Below the table, there is a section for selecting an instance above.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
Redis	i-03e6381fe8dcd7533	t2.micro	eu-west-1a	running	2/2 checks ...	None	ec2-176-34-134-152.eu...	176.34.134.152	-
Frontal2	i-04744d3bdd68678...	t2.micro	eu-west-1a	running	2/2 checks ...	None	ec2-54-194-219-31.eu...	54.194.219.31	-
Mysql	i-0f317ccc0cea3f434	t2.micro	eu-west-1a	running	2/2 checks ...	None	ec2-54-171-169-155.eu...	54.171.169.155	-
Frontal1	i-0f743ccd7b6f5cde9	t2.micro	eu-west-1a	running	2/2 checks ...	None	ec2-34-253-186-64.eu...	34.253.186.64	-