

# EC2+ RDS + Cloudfront CDN + ROUTE 53 + Roles IAM + Test Stress JMeter + Wordpress

El objetivo de esta práctica es desplegar Wordpress en AWS utilizando los servicios de Amazon que nos permiten mejorar la implementación de Wordpress. En la siguiente práctica le agregaremos a esta solución escalabilidad y disponibilidad mediante el uso de balanceadores de carga y grupos de escalabilidad. En esta práctica migraremos la base de Wordpress a AWS RDS, guardaremos el contenido estático de la web en AWS S3, utilizaremos el AWS Cloudfront para mejorar la distribución del contenido estático de nuestra aplicación, mediante el servicio de DNS Route 53 además de publicar la aplicación mediante un nombre de dominio propio mejoraremos la disponibilidad al ofrecer un servidor de respaldo.

Es importante recalcar que si debido al tiempo no se pueden implementar todas las prestaciones requeridas es posible aprobar la práctica. La nota irá en función de las características implementadas. Por ejemplo , en caso de no disponer a tiempo de un dominio DNS podréis no realizar esta parte (pudiendolo presentar más adelante en caso de necesitarlo).

En esta práctica utilizareis el AMI de Wordpress generado en la primera práctica como base. Y desplegaréis el wordpress en una subred pública. Si no disponéis de esta imagen he puesto mi AMI accesible en AWS, este es el nombre del AMI de la primera práctica con wordpress y mysql (**froga-wordpress - ami-0d6aa44a208c79287**) .

Si quereis podeis adelantar la parte del servicio Route 53 y los dominios DNS y hacerlo desde el principio , así trabajareis con un nombre de dominio en lugar de con IP en el transcurso de la práctica. Los pasos sobre Route 53 son a partir del número 35.

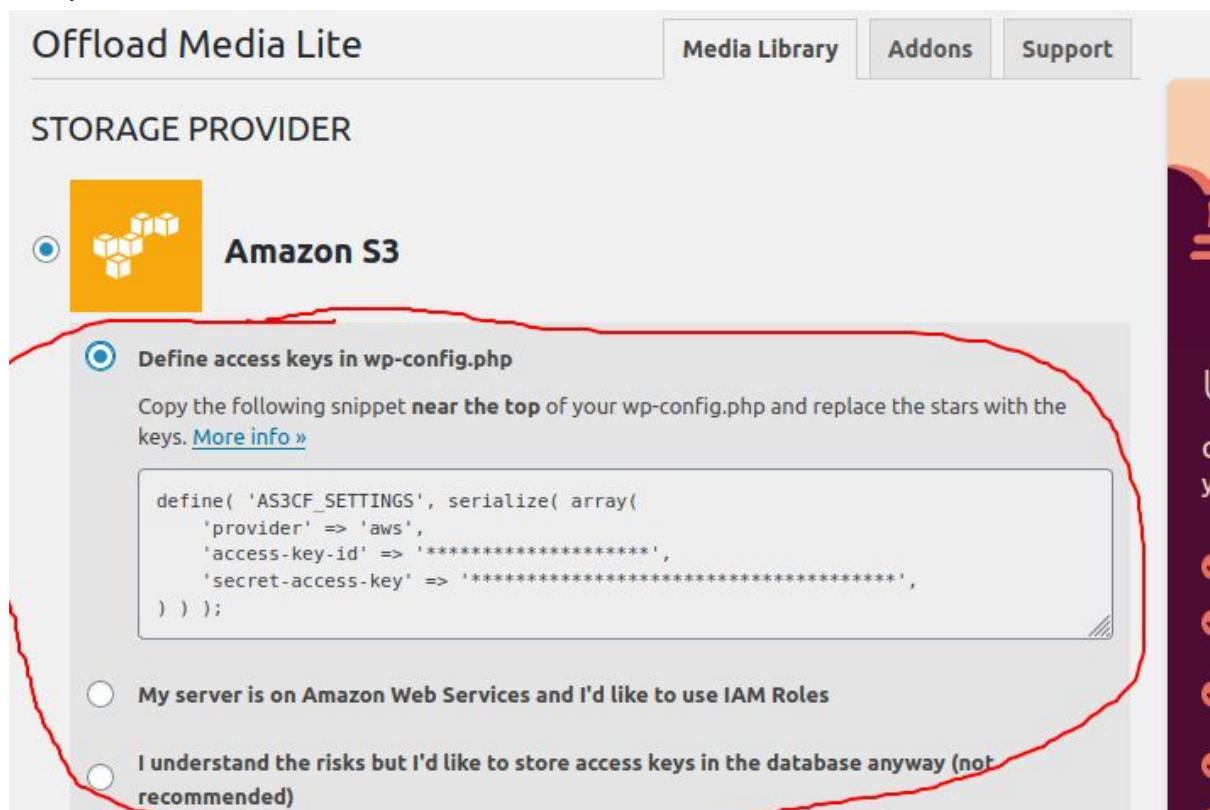
## Albergando la información del Wordpress en S3

1- En este apartado configuraremos Wordpress para que albergar las imágenes y los videos en un bucket S3 de AWS. Primero hay que instalar el Plugin “*WP Offload Media Lite for Amazon S3, DigitalOcean Spaces, and Google Cloud Storage*” desde la consola de administración de wordpress. Si al intentar instalar el plugin os pide acceso ftp es que la carpeta de wordpress ‘/var/www/html’ no tiene los permisos adecuados. Chequear los permisos y ofrecer acceso de escritura al usuario que ejecuta la aplicación web via apache , por ejemplo en mi caso es el usuario www-data (Si vais con el AMI de la primera práctica será ese usuario ;-).

```
$ sudo chown www-data:www-data -R /var/www/html
```

Ahora el wordpress puede instalar los plugins. Instalar el plugin.

2- Una vez instalado el plugin pasaremos a configurar que todas las imágenes y multimedia de los post los guarde en un bucket de S3. Para ello el plugin primero nos pedirá las credenciales a usar. En nuestro caso solo tenéis credenciales con Token (el de vuestro usuario de AWS Educate, el mismo que utilizais en .aws/credentials) que en este caso no sirven, ya que no es duradero y el plugin no contempla esta situación (solo contempla Access-key-id junto con su secret-key). Por lo tanto, para poder acceder desde el servidor a S3 crearemos un Role en el servicio de IAM de AWS (servicio de gestión de usuarios) que permita utilizar S3 y después asignaremos el Role a la instancia Ec2 donde está el wordpress.



The screenshot shows the 'Offload Media Lite' plugin interface. At the top, there are tabs for 'Media Library', 'Addons', and 'Support'. Below this is the 'STORAGE PROVIDER' section, where 'Amazon S3' is selected with a radio button. A red circle highlights the configuration instructions and the code snippet. The instructions say to copy the snippet near the top of wp-config.php and replace the stars with the keys. The code snippet is as follows:

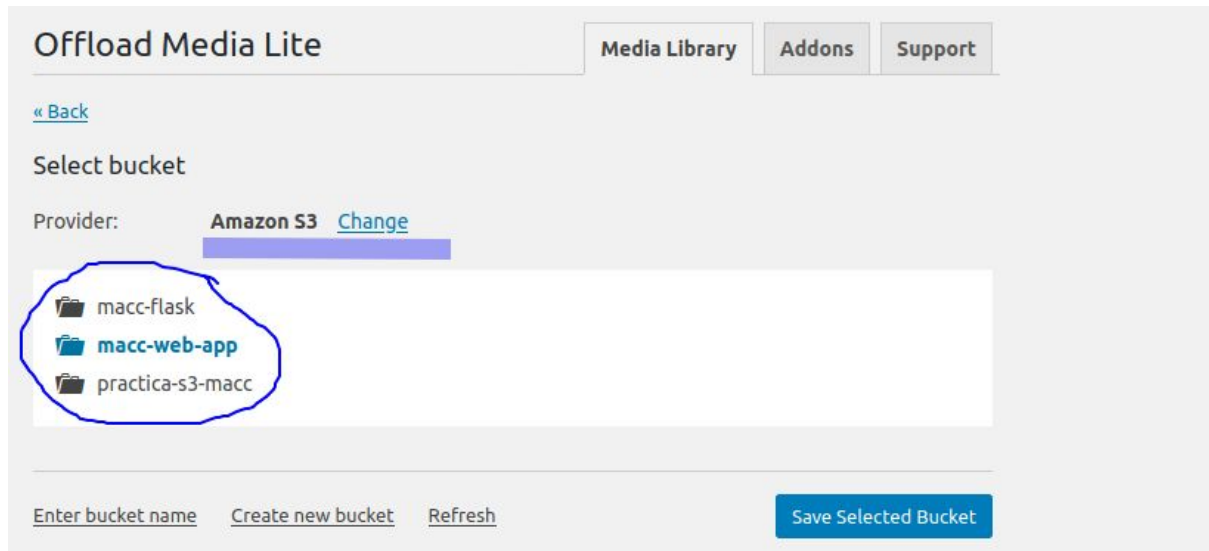
```
define( 'AS3CF_SETTINGS', serialize( array(
    'provider' => 'aws',
    'access-key-id' => '*****',
    'secret-access-key' => '*****',
) ) );
```

Below the code, there are two radio button options: 'My server is on Amazon Web Services and I'd like to use IAM Roles' (which is selected) and 'I understand the risks but I'd like to store access keys in the database anyway (not recommended)'.

Por lo tanto antes de configurar el plugin iremos a IAM y crearemos el Role. En Educate no podemos crear usuarios dentro de una cuenta y crearle acceso vía un access-key (como la primera opción) pero sí Roles que luego asignaremos a recursos, en este caso al servidor EC2 permitiéndole acceder a S3. Ir al servicio IAM y crear un role para elementos EC2 que de acceso total a S3 (De esta forma también evitamos el escribir en los ficheros de credenciales la información del token y acceso). Llamar al role creado 'ec2-s3'.

3- Ahora asignarle a la instancia EC2 que alberga el wordpress el role. Para ello una vez seleccionado la instancia ir a a 'Acciones-> Seguridad' y elegir modificar IAM Role y seleccionar el role creado

4- Ahora si volvemos a la configuración del plugin S3 de Wordpress y seleccionamos que utilizaremos rol en instancia EC2 podemos comprobar que ya tenemos acceso a los buckets. Y seleccionamos el que queremos utilizar en wordpress para albergar los elementos multimedia.



Y ahora ya podemos utilizar el bucket para albergar elementos multimedia del wordpress. Fijaros en las opciones que da el plugin.

5- Probemos a crear elementos multimedia en la librería de wordpress. En algunas versiones de wordpress el menú de drag de la librería de multimedia no funciona bien, utilizar el formato en modo lista y agregar una imagen y comprobar que se ha guardado en S3

Storage Provider:  
Amazon S3

Bucket:  
macc-web-app

Path:  
wp-content/uploads/2020/11/23161924/Habitacion.jpg

Region: US East (N. Virginia)

Access: Public

---

**Save** ^ v ▲

Uploaded on: Nov 23, 2020 at 16:19

File URL:  
<http://macc-web-app.s3.amazonaws.com/wp-content/uploads/2020/11/23161924/Habitacion.jpg>

Copy URL

File name: Habitacion.jpg

File type: JPG

File size: 67 KB

Ahora también crear un post con algún recurso de S3.

## Acelerando la distribución del contenido estático mediante Cloudfront

6-Una vez que alojamos las imágenes en S3 creamos un CDN en AWS Cloudfront para distribuciones de contenido via cache y acelerar así la distribución de la páginas.

7- Primero creamos en AWS en el servicio Cloudfront una 'Distribución' de tipo WEB. Ahora seleccionamos como nombre de dominio origen el Bucket que hemos configurado en el wordpress, por ejemplo 'macc-web-app' y seleccionar comprensión de objetos automático y dejar las demás opciones por defecto. La creación del punto de distribución no es inmediata. Una vez lista copiar la URL del dominio del punto de distribución.

8- Una vez creado el punto de distribución en las propiedades del plugin S3 de wordpress tenemos que configurar el modo de distribución via el cloudfront creado. Primero seleccionamos modificar y elegimos el tipo Cloudfront.

**DELIVERY**

**Provider:** **Amazon S3** [Change](#)

☒ **ON**

**Rewrite Media URLs**  
For Media Library files that have been copied to your bucket, they are served from Amazon S3 instead of your server. [More info »](#)

☐ **OFF**

**Force HTTPS**  
By default we use HTTPS when the request is HTTPS and regular HTTP when the request is HTTP, but you may want to force the use of HTTPS always, regardless of the request. [More info »](#)

9- Ahora en Wordpress elegimos el dominio que utilizaremos para el CDN para lo que tenemos que activar el CNAME. Rellenamos con la dirección del punto de distribución creado.

**DELIVERY**

**Provider:** **Amazon CloudFront** [Change](#)

☒ **ON**

**Rewrite Media URLs**  
For Media Library files that have been copied to your bucket, rewrite the URLs so that they are served from Amazon CloudFront instead of your server. [More info »](#)

☒ **ON**

**Custom Domain (CNAME)**  
We strongly recommend you configure a subdomain of to point at your Amazon CloudFront distribution. If you don't enter a subdomain of your site's domain in the field below it will negatively impact your site's SEO. [More info »](#)

☐ **OFF**

**Force HTTPS**  
By default we use HTTPS when the request is HTTPS and regular HTTP when the request is HTTP, but you may want to force the use of HTTPS always, regardless of the request. [More info »](#)

Ahora comprobamos como se han modificado las url de las imagenes y ahora contiene el nombre del CDN. Para mejorar el SEO podríamos crear un subdominio CDN de nuestro dominio y hacer que apunte al CDN.

## Offload



Storage Provider:

Amazon S3

Bucket:

macc-web-app

Path:

wp-content/uploads/2020/11/23161924/Habitacion.jpg

Region:

US East (N. Virginia)

Access:

Public

## Save



Uploaded on: Nov 23, 2020 at 16:19

File URL:

<http://d2xji93z9vieo0.cloudfront.net/wp-content/uploads/2020/11/23161924/Habitacion.jpg>

Copy URL

File name: **Habitacion.jpg**

## Select or Upload Media





Upload files

Media Library


Filter media

All dates

Search

ATTACHMENT DETAILS



Habitacion.jpg

November 23, 2020

67 KB

626 by 352 pixels

Edit Image

Delete permanently

Alt Text

Describe the purpose of the image. Leave empty if the image is purely decorative.

Title

Habitación

Caption

Description

File URL:

http://d2xji93z9viero0.clou

Copy URL

10- Ahora creamos un post con imágenes y desde 'websitepulse' mediremos los tiempos de acceso a la página desde New York por ejemplo. Realizamos la petición varias veces desde new york y vemos que en la segunda la imagen se ha descargado más rápido mejorando los tiempos, lo mismo sucede con los demás gracias a que en Cloudfront se dispone transparentemente de una red de servidores de caché distribuido por todas las regiones de AWS ;-).

See All Free Tools

<div> <div>Host:</div> <div>http://wordpress.macc-practicas.com/index.php/2020/11/23/froga-s3/</div> </div> <div> <div>Monitoring Location:</div> <div>New York, NY</div> </div> <div> <div>Date:</div> <div>2020-11-23 17:01:35 (GMT +0000)</div> </div>			
URL	Status	Size	Timings
GET froga-s3	UP	28.7 KB	0.093 sec
GET castle-1.jpeg	UP	8.4 KB	0.057 sec
2 Requests		37.1 KB	0.093 sec



Monitoring Location: New York, NY			
Date: 2020-11-23 17:02:13 (GMT +0000)			
URL	Status	Size	Timings
GET frog-a-s3	UP	28.7 KB	0.068 sec
GET castle-1.jpeg	UP	8.4 KB	0.004 sec
2 Requests		37.1 KB	0.068 sec

11- Si probamos con el mismo post pero con la misma imagen sin CDN comprobareis que siempre da tiempos parecidos , no reduce el tiempo en nuevas peticiones;-)

## Testeando el rendimiento de la aplicación

12- Una vez mejorado el rendimiento de acceso a nuestras páginas de Wordpress, ahora vamos a comprobar el rendimiento de computación que presenta nuestro servidor , en este caso desplegado en una máquina *t2.micro*.

13- Para realizar tests de stress utilizaremos la herramienta Apache JMeter. Por lo tanto lo primero que haremos será instalar la herramienta JMeter en el ordenador que utilizaremos para realizar las pruebas de test, por ejemplo vuestro pc ( Cuando se quieren realizar test de muy alta carga se suele crear un cluster de pc que despues lanzarán sincronizados los test a los servidores a Testear). Esta herramienta esta implementado en Java por lo que tendréis que tener instalado el JDK en el sistema (Por ejemplo podéis instalar openjdk. Si no vais a desarrollar nada en Java os sirve con el Jre , yo instalare el JDK)

```
$ sudo apt-get install openjdk-14-jdk
$ java --version
```

Y ahora instalamos JMeter. Para ello descargamos de la página oficial el binario , lo descomprimos en una carpeta y ejecutamos el binario `./jmeter` que se ubica en la carpeta `'bin/'` de JMeter.

14- Y ahora escribamos el primer test de rendimiento. Será un test que simulará 50 peticiones HTTP GET en 10 segundos de arranque. Para ello primero creamos un 'Test Plan' y en este Test Plan creáis un grupo de hilos (Thread Group) y lo configurais con 50 hilos y una rampa de 10 segundos (podéis utilizar el botón derecho del ratón para crearlo). Después en el Thread group añadir un Sampler de HTTP (en el que configuraréis la petición que vais a realizar) y después también añadiréis un 'Listener' del tipo 'View Result in Table'. En el sampler HTTP configurar la ip del servidor worpdress , el puerto y el path que quereis testear (puede ser '/'). y ahora ejecutais y veis cómo ha respondido y con que tiempos , y vemos que el servidor ha respondido a todas las peticiones correctamente.



15- Ahora crear los siguientes Thread group y ver como responde el servidor.

Test1- 50 hilos 10 segundos

Test2- 200 hilos 10 segundos

Test3- 500 hilos 10 segundos

Test4- 700 hilos 10 segundos

¿Que ha ocurrido? ( Si vemos que el Jmeter no termina la ejecución de algún test pararlo ;-)

Seleccionar la instancia en EC2 y en la sección Monitoring podéis ver diferentes métricas.

16- Para volver a poner el sistema activo otra vez deberéis de reiniciar el servicio Apache2.

```
$ sudo service apache2 restart
```

17- Para poder ver qué ha ocurrido AWS nos ofrece vía el servicio Cloudwatch de monitorización con métricas por defecto de rendimiento de una instancia EC2. Para verlas simplemente seleccionar en EC2 la instancia y en el menú Monitoring las tenemos . Pero si queremos realizar algo más complejo como definir alarmas, o ver otras métricas , o configurar un Dashboard personalizado o incluso crear métricas nuevas relacionadas con mensajes logs de nuestras aplicaciones podemos utilizar el Servicio Cloudwatch de Amazon.

18- A continuación activaremos el servicio Cloudwatch y crearemos una alarma para que cuando el uso de la CPU supere el 40% nos envíe un mensaje de email de aviso. Para ello primero activaremos en la instancia EC2 que queramos la monitorización. Esto lo hacemos en EC2 seleccionando la instancia y en Acciones en el menú monitorización seleccionamos gestión de la monitorización.

19- Una vez activada la monitorización (ahora disponemos de métricas cada minuto). Vamos al Servicio Cloudwatch y seleccionamos Alarmas donde crearemos la alarma. Es importante mencionar que al crear alarmas se puede utilizar el servicio de Mensajería de AWS SNS el cual permite enviar mensajes de email relacionados con un tema (topic). Al crear la alarma nos permitirá crear desde el mismo menu el topic SNS, pero también podemos crear el topic desde el servicio SNS. Por lo tanto hagámoslo en dos pasos.

20- Vamos al servicio SNS y crearemos un topic/tema denominado 'Alarma-Uso-CPU' y asignaremos un email al topic. Desde el email habrá que confirmar la suscripción al topic/tema. Al crear temas elegimos el tipo 'Standard' y le damos un nombre, y con todas propiedades por defecto le damos a crear. Ahora nos permite crear una suscripción , clickamos el boton y le damos una dirección de email.

21- Tras crear el tema SNS ya podemos crear una alarma Cloudwatch que nos permitirá enviar mensajes de email dependiendo de unas métricas en concreto. En nuestro caso crearemos una alarma de métrica EC2 de la instancia de wordpress sobre la métrica 'CPU Utilisation'. Al crear la regla de la alarma indicaremos que la alarma se lanzará cuando se supere el 40% de uso y también le relacionamos el tema SNS previamente creado , para que envíe un email en caso de alarma. Ahora no lo haremos pero incluso podemos definir que hacer en caso de alarma , por ejemplo escalar el número de instancias (Esto lo haremos en la siguiente práctica).

22- Una vez creada la alarma volver a ejecutar el test de stress y comprobar que os llega el email.

23- Por otro lado podéis probar a escalar verticalmente la instancia EC2 y modificar el tipo de CPU y volver a ejecutar una prueba de stress. Para cambiar el tipo de CPU seleccionar la instancia EC2, parar la máquina y en acciones seleccionar modificar tipo de instancia (por ejemplo un t2-large , t3-large, ...). ¿Hemos mejorado el rendimiento?

24- También podemos crear un tipo de alarma en base al estado de la instancia.

## Migrando la base de datos a RDS

25- En la práctica anterior hemos escalado verticalmente, pero puede que tanto por rendimiento como por disponibilidad queramos escalar horizontalmente, es decir , crear réplicas de un servicio. En estos casos si nuestra aplicación utiliza una m base de datos también tendremos que crear un cluster de la base de datos y gestionarlo nosotros. Con el servicio RDS tanto la creación de la base de datos, su gestión y su escalado se facilitan muchísimo, convirtiéndose hoy en día en uno de los servicios PaaS más utilizados de AWS. Este servicio permite crear bases de datos relacionales de diferentes tipos: Mysql , Aurora(mysql adaptado ) , PostgreSQL,... A nivel de desarrollo el utilizar RDS o no modifica nada, simplemente utilizaremos el endpoint que nos de AWS para acceder a la base de datos y sustituiremos eso por la dirección que actualmente utilizais en vuestra aplicación.

26- En este apartado vamos a migrar la base de datos Mysql que se ubica en la misma máquina a un cluster RDS de Mysql.

27- Lo primero que haremos será conectarnos vía a la instancia EC2 que alberga el wordpress y la base de datos.

28- Ahora haremos un backup/dump de la base de datos de wordpress.

```
$ mysqldump -u root -p --databases wordpressdb > wpdump.sql
```

29- Ahora creamos la base de datos Mysql en AWS RDS. Para ello ir al servicio RDS y seleccionar base de datos. Cuando os pregunte por el tipo seleccionar Aurora con compatibilidad con Mysql (Las base de datos Aurora está ligeramente modificada para optimizar su ejecución en AWS). En las propiedades de réplica seleccionaremos un master

con opción de varios lectores , que es la opción por defecto. Más abajo tenéis más propiedades como el nombre , llamarle 'wordpress-db' por ejemplo y seleccionar un password. Después se selecciona el tipo de instancia sobre la que se ejecutará la base de datos, aquí elegimos la de por defecto pero podríamos seleccionar otra. En disponibilidad podemos indicarle que cree una réplica en otra zona de disponibilidad (de momento seleccionaremos que si , esto aumenta el precio ;-). Finalmente seleccionamos en que VPC queremos que se lance , en este caso en la VPC del wordpress y podeis indicar si dareis acceso público o privado. Si habéis seleccionado que se despliegue con réplicas y en diferentes zonas de disponibilidad en el VPC tenéis que tener dos subredes como mínimo y cada una de ellas en zonas de disponibilidad diferentes. En nuestro caso será privado. Y le agregamos un security group nuevo (podríamos asociarlo uno previo también). En las demás opciones que dejaremos por defecto se indica que haga un backup diario. Analizar un poco todo lo que ofrece y crear la base de datos.

Una vez generada la base de datos analizar lo generado. En mi caso he generado un cluster con un master/writer y un slave/reader, y todo esto replicado en dos zonas de disponibilidad (az).

DB identifier	Role	Engine	Region & AZ	Size
<input checked="" type="radio"/> wordpress-db	Regional	Aurora MySQL	us-east-1	2 Instances
<input type="radio"/> wordpress-db-Instance-1-us-east-1a	Reader	Aurora MySQL	us-east-1a	db.r5.large
<input type="radio"/> wordpress-db-Instance-1	Writer	Aurora MySQL	us-east-1b	db.r5.large

Size	Status	CPU	Current activity	Maintenance	VPC	Multi-AZ
2 Instances	Modifying	-		none	-	-
db.r5.large	Available	4.00%	0 Sessions	none	vpc-0eddc648add104681	2 Zones
db.r5.large	Available	4.00%	0 Sessions	none	vpc-0eddc648add104681	2 Zones

Endpoint name	Status	Type	Port
database-1.cluster-ro-cezwvvugw89l.us-east-1.rds.amazonaws.com	Creating	Reader	3306
database-1.cluster-cezwvvugw89l.us-east-1.rds.amazonaws.com	Creating	Writer	3306

30- Lo siguiente que haremos será conectarnos a la base de datos. En mi caso he configurado la base de datos con acceso solo privado, por lo cual solo podremos acceder desde una máquina ubicada en la misma VPC. Pondremos como endpoint el cluster , ya que queremos modificar la base de datos . El endpoint ro solo sirve para realizar operaciones de lectura.

```
ubuntu@ip-10-0-0-113:~$ mysql -u admin -p --host
wordpress-db.cluster-cezwvvugw89l.us-east-1.rds.amazonaws.com
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

31- Una vez conectados a la base de datos lo primero que haremos será crear un usuario para wordpress y darle permisos.. Si al crear la base de datos no habéis creado una base de datos para wordpress, podéis crearla ahora. Finalmente le dareis permisos al usuario creado en la base de datos correspondiente.

```
mysql>CREATE USER 'wpuser'@'%' IDENTIFIED BY 'mypassword';
mysql>CREATE DATABASE wordpressdb;
mysql>GRANT SELECT ON wordpressdb.* TO 'wpuser'@'%'
mysql> flush privileges;
```

32- Ahora solo nos queda migrar la base de datos

```
$mysql -u admin -p -h
wordpress-db.cluster-cezwvvugw89l.us-east-1.rds.amazonaws.com < wpdump.sql
```

Para comprobar que sea copiado entrar en la base de datos y mirar las tablas.

33- Y por último solo nos queda decirle al wordpress donde esta la base de datos. Para ello ir al fichero '/var/www/html/wp-config.php' .

```
define( 'DB_PASSWORD', 'mypassword' );

/** MySQL hostname */
define( 'DB_HOST', 'wordpress-db.cluster-cezwvvugw89l.us-east-1.rds.amazonaws.com' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

/**#@+
 * Authentication Unique Keys and Salts.
 */
```

34- Y para comprobar que funciona crear un nuevo post utilizando la base de datos de RDS y luego cambiar el endpoint al mysql local y comprobar que no está el nuevo post ;-)

35- Finalmente volver a poner el endpoint de RDS y comprobar que está el nuevo post ;-)

## Mejorando la disponibilidad mediante Route 53

35- En esta práctica utilizaremos el servicio Route 53 de Amazon para darle un nombre de dominio a nuestro servidor Wordpress y también para mejorar la disponibilidad de nuestro servidor con una réplica para situaciones de fallo. estos dos objetivos los realizaremos mediante el servicio Route 53.

36- Si no disponemos de un nombre de dominio no podremos finalizar/realizar esta práctica.

37- Una vez obtenido un nombre de dominio lo primero que haremos en Route 53 será crear un 'Hosted Zone' mediante el cual podremos gestionar nuestro nombre de dominio. Si por ejemplo tenemos nuestro nombre de dominio actualmente gestionado por 'Dynahosting' tendremos que indicarle una vez creado el Hosted Zone en Route 53 que servidores de nombre DNS utilizaremos para nuestro dominio. Bueno comencemos creando el 'Hosted Zone'. para ello ir al servicio Route53 de AWS y clickar en crear hosted zone. En las opciones simplemente indicar vuestro nombre de dominio y si quereis una descripción y creáis los servidores públicos.

Zonas alojadas (1)

El modo Automatic es el comportamiento actual de búsqueda que se ha optimizado para obtener los mejores resultados del filtrado. [Para cambiar los modos, vaya a la configuración.](#)

Ver los detalles

Editar

Eliminar

Crear una zona alojada

Filtrar zonas alojadas por propiedad o valor

<

1

>

	Nombre de dominio	Tipo	Creado por	Recuento de registros	Descripción	ID de zona alojada
<div><div></div><div>macc-practicas.com</div></div>	Pública	Route 53	3	-	Z06543952FS46B578THC0	

38- Una vez que teneis la zona creada clicar en ella y podremos ver los nombres de los 4 servidores DNS que tenemos asociados. Es importante recalcar que el servicio gestionado de PaaS de DNS que nos ofrece AWS es de alta disponibilidad a nivel de regiones, lo que quiere decir que si cayera toda una región nuestro servicio de DNS seguiría activo. Pues indiquemos le a 'dynahosting' que servidores utilizaremos a partir de ahora para resolver las ips de nuestro dominio.

<input type="checkbox"/>	Nombre del registro ▾	Tipo ▾	Política de direccionamiento ▾	Diferenciador ▾	Alias ▾	Valor/Dirigir el tráfico a ▾
<input type="checkbox"/>	macc-practicas.com	NS	Simple	-	No	ns-1611.awsdns-09.co.uk. ns-191.awsdns-23.com. ns-587.awsdns-09.net. <del>ns-1388.awsdns-45.org</del>
<input type="checkbox"/>	macc-practicas.com	SOA	Simple	-	No	ns-1611.awsdns-09.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
<input type="checkbox"/>	wordpress.macc-practicas.com	A	Simple	-	No	34.225.90.217

39- Ir a dinahosting y en dominios indicarle que no utilizareis los servidores de dynahosting e introducir los nombres de los servidores de vuestra zonas de hospedaje. Es posible que esta migración tarde un tiempo.

**DOMINIO**  
macc-practicas.com
 

- Inicio
- Contactos
- Servidores DNS**
- Zonas DNS
- Redirecciones
- Cuenta de email
- Parking
- Panel de acceso
- Acceso independiente
- Logs

Establecer DNS

**DNS por defecto**  
☐ Quiero utilizar los DNS de dinahosting

**DNS propios**  

DNS 1  
 ns-1611.awsdns-09.co.uk

DNS 2  
 ns-191.awsdns-23.com

DNS 3  
 ns-587.awsdns-09.net

DNS 4  
 ns-1388.awsdns-45.org

DNS 5

DNS 6

Guardar

40-Ahora crearemos un registro , en mi caso ha sido *wordpress.macc-practicas.com* el cual relacionaremos con vuestra instancia de wordpress. Antes de hacer esto en el servicio EC2 pedir una IP elastica y asignarsela a la instancia EC2 del wordpress (recordar que el wordpress en este práctica está en una red pública)

41- Una vez que tenemos la ip elástica asignada apuntamos la ip y volvemos al servicio Route53 a nuestro Hosted Zone y clickais en crear registro. Os apareceran diferentes opciones/wizards de direccionamiento para cuando tenemos mas de un servidor ofreciendo el mismo servicios: 1) Ponderado un porcentaje del tráfico a una ip y otro porcentaje a otra ip 2) geolocalizado en base a la cercanía de la petición nos dará un aip u otra 3) en base a la latencia nos dara una ip u otra 4) direccionamiento de disponibilidad . De momento elegiremos la estandard , donde relacionamos una IP con un registro de dominio. En las opciones del registro del tipo A indicareis el subdominio a utilizar y la ip que utilizareis.

### Definir un registro simple

**Nombre del registro**

Para dirigir el tráfico a un subdominio, introduzca el nombre del subdominio. Por ejemplo, para dirigir el tráfico a blog.ejemplo.com, introduzca *blog*. Si deja este campo en blanco, el nombre de registro predeterminado será el nombre del dominio.

blog

.macc-practicas.com

Caracteres válidos: a-z, 0-9, ! " # \$ % & ' ( ) \* + , - / : ; < = > ? @ [ \ ] ^ \_ ` { | } . ~

**Valor/Dirigir el tráfico a**

La opción que elija determina cómo Route 53 responde a las consultas de DNS. Para la mayoría de las opciones, debe especificar adónde desea dirigir el tráfico de Internet.

Dirección IP u otro valor en función del tipo de registro

192.0.2.235

Introduzca varios valores en líneas separadas.

**Tipo de registro**

El tipo de DNS del registro determina el formato del valor que Route 53 devuelve en respuesta a las consultas de DNS.

A: dirige el tráfico a una dirección IPv4 y a algunos recursos de AWS

Elija cuándo dirigir el tráfico a recursos de AWS para EC2, API Gateway, Amazon VPC, CloudFront, Elastic Beanstalk, ELB o S3. Por ejemplo: 192.0.2.44.

Cancelar

Definir un registro simple



<input type="checkbox"/>	Nombre del registro ▾	Tipo ▾	Política de direccionamiento ▾	Diferenciador ▾	Alias ▾	Valor/Dirigir el tráfico a ▾	TTL (segundos) ▾
<input type="checkbox"/>	macc-practicas.com	NS	Simple	-	No	ns-1611.awsdns-09.co.uk. ns-191.awsdns-23.com. ns-587.awsdns-09.net. ns-1388.awsdns-45.org.	172800
<input type="checkbox"/>	macc-practicas.com	SOA	Simple	-	No	ns-1611.awsdns-09.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400	900
<input type="checkbox"/>	wordpress.macc-practicas.com	A	Simple	-	No	34.225.90.217	300

42- Y ya podéis probar a poner en el navegador en lugar de la ip el nombre del dominio y tendrías que ver vuestro wordpress.



**Froga** Just another WordPress site



43- Ahora crearemos un servidor de respaldo y haremos que Route53 realice un healthcheck y en caso de que un servidor caiga nos responda con la ip del servidor secundario. Para esto crearemos una política de enrutamiento.

44- Antes de comenzar con esto crear una nueva instancia con el wordpress, asignarle una ip elastica y comprobar que funciona.



**Froga** Just another WordPress site



44- Primero crearemos un servicio de healthcheck. Esto lo hacemos en el servicio Route53. Si teneis el navegador en castellano en el menu de Route53 os aparecera como 'Comprobaciones de estado'. Le dais un nombre y luego indicais que es una comprobación de punto de enlace , el protocolo será HTTP y le dais la IP del servidor principal. Y después si queréis crear también una alarma para que se os envíe un email. una vez creada la comprobación esta tardara unos minutos en recopilar los primero datos. Tambien teneis que crear una segunda comprobación de estado para el servidor de respaldo

<input type="checkbox"/>	Servidor wordpress	<div><div></div></div> <div>hace 32 minutos</div>	<div><div></div></div> <div>hace 2 minutos</div>	Incorrecto	http://34.225.90.217:80/	<div><div></div></div> <div>1 de 1 en ALARMA</div>
<input type="checkbox"/>	servidro respaldo	<div><div></div></div> <div>hace 17 minutos</div>	<div><div></div></div> <div>hace 2 minutos</div>	Buen estado	http://54.204.61.113:80/	No hay alarmas configur...

45- Ahora ya estamos listos para crear un registro secundario para fallos para nuestro dominio de wordpress. para ello ir la zona de hospedaje creada , seleccionar el registro Tipo A creado y clickar editar . Ahora podéis cambiar el tipo de política del registro (abajo en la página) y seleccionaremos conmutación por error. Al seleccionar nos saldran mas opciones, en este caso seleccionaremos que este registro es el principal. Y luego tendremos que crear el secundario.

La cantidad de tiempo, en segundos, que los solucionadores de DNS y los navegadores web almacenan en cache la configuración en este registro. ("11L" significa "tiempo de vida").

300

Valores recomendados: de 60 a 172 800 (dos días)

**Política de direccionamiento** [Info](#)  
La política de direccionamiento determina la forma en que Amazon Route 53 responde a las consultas.

Comutación por error

**Tipo de registro de conmutación por error**  
Elija **Principal** para dirigir el tráfico al recurso especificado de forma predeterminada o **Secundario** para dirigir el tráfico al recurso especificado cuando el recurso principal no esté disponible. Solo puede crear un registro de conmutación por error de cada tipo.

Principal

**Comprobación de estado** [Info](#)  
Elija la comprobación de estado que desea que Route 53 utilice para determinar si este conjunto de registros está en buen estado. Puede crear una comprobación de estado en la [consola de comprobación de estado](#).

Servidor wordpress

**ID de registro** [Info](#)  
Escriba una descripción única que diferencie a este registro de otros con el mismo nombre y tipo.

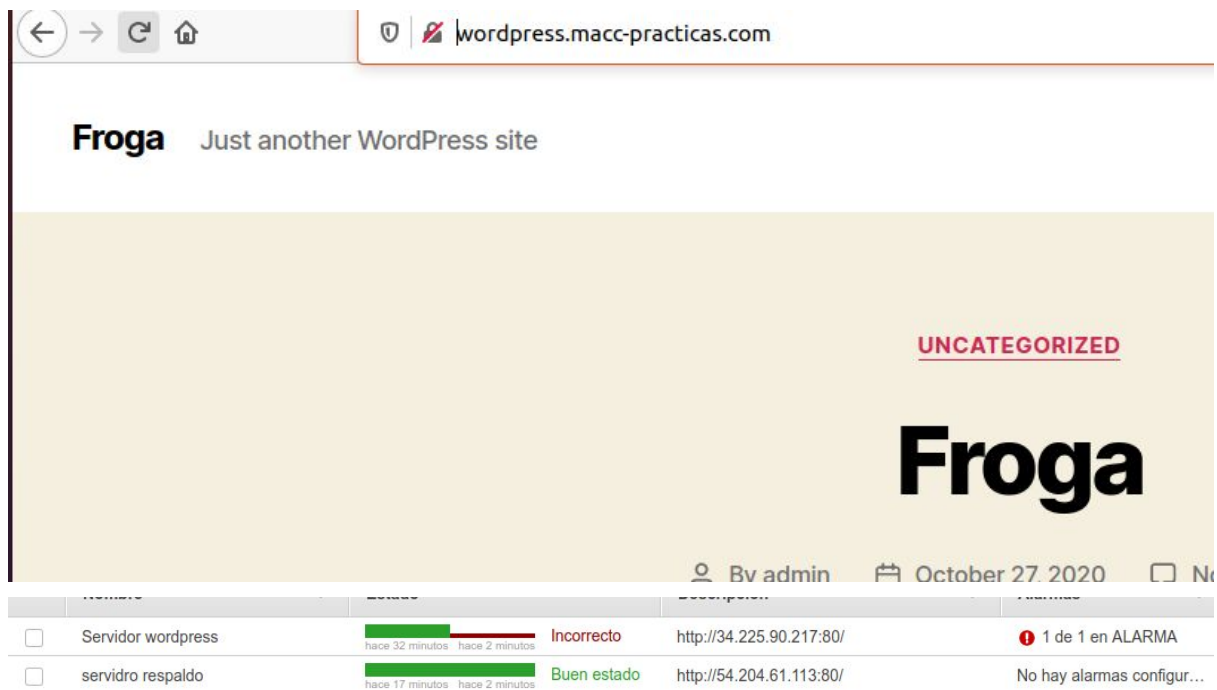
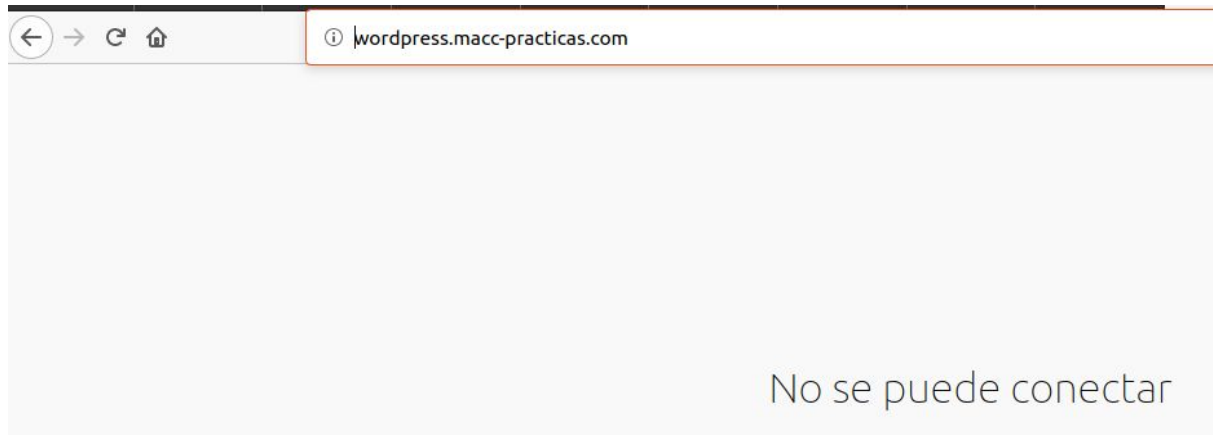
Wordpress principal

46- Ahora crearemos el registro secundario.

7200 900 1209600 86400						
<input type="checkbox"/>	wordpress. macc- practicas.co m	A	Conmutación por error	Principal	No	34.225.90.217 300
<input type="checkbox"/>	wordpress. macc- practicas.co m	A	Conmutación por error	Secundario	No	54.204.61.113 300

47- Ahora solo nos queda apagar el servidor primario y comprobar que sigue funcionando. La conmutación no es inmediata ya que el healthcheck lo tenemos puesto que sea para cuando se dé la alarma a partir de los tres fallos .Apagar el principal y comprobar el

redireccionamiento. Vereis que de inmediato la página ya no responde ir al healthcheck y esperar a que salte la alarma, una vez que salte la alarma veréis como ha entrado en juego el segundo servidor.



48- Ahora si queréis podeis seguir jugando con otras políticas de direccionamiento.

## Costes aproximados de la práctica

Una vez que hemos utilizado diferentes servicios de AWS quiero que reflexiones sobre los costos. En la siguiente imagen os dejo lo gastado por mi cuenta de AWS Educate en la clase de la asignatura. he realizado las mismas prácticas que vosotros. Ahora quiero que analicéis los diferentes tipos de gatos y precios por servicio que tiene AWS. Para ayudaros aquí os dejo tres links , dos de ellos para calcular gastos. Rellenar la tabla con un resumen de precios y gastos.

[https://aws.amazon.com/es/pricing/?nc2=h\\_gl\\_pr\\_ln](https://aws.amazon.com/es/pricing/?nc2=h_gl_pr_ln)

<https://calculator.aws/#/>

<https://calculator.s3.amazonaws.com/index.html>

/ aagirre@mondragon.edu		\$9.695885 / \$75 / \$0
2020-10	Item Description	Cost
	DataTransfer-Out-Bytes	\$0.00074
	EBS:VolumeUsage.gp2	\$0.010781
	USE1-Event-64K-Chunks	\$1.4E-5
	BoxUsage:t2.micro	\$0.107584
	EBS:SnapshotUsage	\$0.056452
	DataTransfer-Regional-Bytes	\$0.000542
2020-10 subtotal		\$0.176113
2020-11	Item Description	Cost
	InstanceUsage:db.r5.large	\$4.110748
	CW:AlarmMonitorUsage	\$0.000359
	DataTransfer-Regional-Bytes	\$0.002236
	USE1-DataProcessing-Bytes	\$0.004015
	USE1-Event-64K-Chunks	\$0.000113
	Requests-Tier2	\$0.000295
	EBS:SnapshotUsage	\$0.306667
	USE1-USW1-AWS-Out-Bytes	\$1.5E-5
	BoxUsage:t2.micro	\$1.719851
	ElasticIP:IdleAddress	\$0.591
	Requests-Tier1	\$0.00092
	HostedZone	\$0.5
	DataTransfer-Out-Bytes	\$0.021373
	DNS-Queries	\$0.000588
	CW:MetricMonitorUsage	\$0.044167
	USE1-USW2-AWS-Out-Bytes	\$5.0E-6
	USE1-USE2-AWS-Out-Bytes	\$1.0E-6
	EBS:VolumeUsage.gp2	\$2.167472
	Aurora:BackupUsage	\$1.0E-5
	Aurora:StorageUsage	\$8.5E-5
	Aurora:StorageIOUsage	\$0.032796
	BoxUsage:t3.large	\$0.017056
2020-11 subtotal		\$9.519772
Grand total		\$9.695885

**A rellenar con lista de precios por servicios**

=====