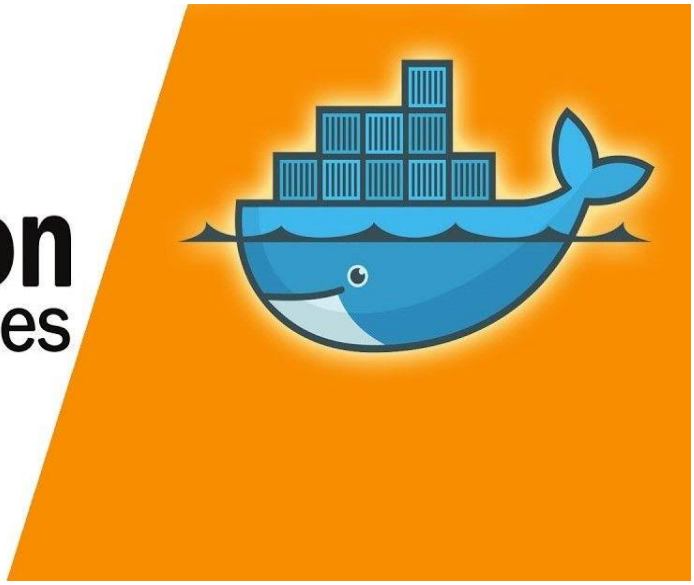
Estudios de Informática, Multimedia y Telecomunicación

Migración y optimización de un e-commerce de un servidor simple a un clúster de AWS



Alumno: Adrián Martínez Rodríguez

Año: 2022-2023

TFG - Administración de redes y sistemas operativos

Consultor: Jaume Jofre Bravo

Propuesta de plan de trabajo

Descripción del proyecto

Durante los años de pandemia por covid-19 muchos negocios dieron el salto a la implementación de comercios electrónicos para vender u ofertar sus productos en la red. Estas webs en su mayoría fueron creadas en un plazo de tiempo muy corto y para satisfacer unas necesidades que en ese momento tenían los negocios, dado que con las restricciones que se estaban empezando a aplicar veían peligrar sus ingresos con el mercado normal, y por ello decidieron crear sus tiendas online.

Hoy tras varios años de esta situación muchos de estos negocios se han dado dé cuenta de que un gran porcentaje de sus ingresos reside en el comercio electrónico y que sus sistemas actuales se están quedando obsoletos o muy pequeños de recursos para albergar el volumen de negocio que obtienen de la tienda online, o simplemente se plantean invertir más tiempo y dinero en esta sección del negocio que hasta hace un par de años era utilizada por grandes empresas mayoritariamente.

Es por ello que mi proyecto va enfocado a un simulacro de migración y optimización de un e-commerce de un servidor compartido que no logra dar el rendimiento esperado hacia un clúster de servidores den AWS donde se pueda garantizar la escalabilidad y la alta disponibilidad que se desea para un servicio como una tienda online.

Justificación del proyecto

Este proyecto lo realizo porque el volumen de ventas del comercio electrónico no para de crecer y de acaparar más mercado, como se puede ver en la progresión de los últimos 10 años hasta el 2020, ya venía creciendo, pero el cambio de consumo de los clientes desde el inicio de la pandemia hasta el día de hoy ha sido más que notable y ahora el comercio electrónico ya es una realidad, pero gracias a la situación vivida durante la pandemia los comercios ya no lo ven como un competidor, sino que puede ser una herramienta para que un comercio tradicional pueda llegar a los nuevos mercados y subsistir o incluso mejorar sus ingresos y/o popularidad. Pero existe la pequeña dificultad de que muchos comercios ven imposible mejorar sus plataformas de ventas ya sea por desconocimiento, coste, o diversas dificultades para poder obtener una infraestructura de e-commerce más actualizada.

Por eso la intención de la realización de este proyecto es la de realizar una migración de una infraestructura obsoleta a una infraestructura ágil que permita hacer crecer la infraestructura acorde a las necesidades del negocio principal.

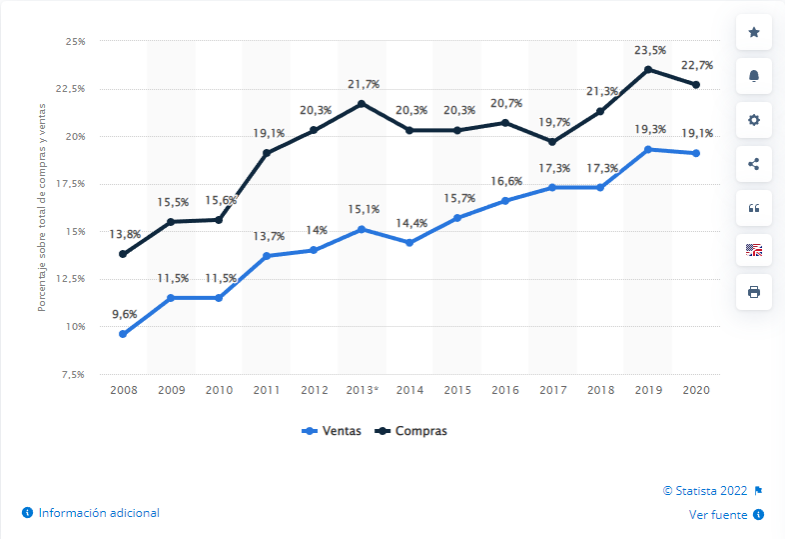


Ilustración 1- Evolución del porcentaje de compras y ventas de comercio electrónico sobre el total de compras y ventas realizadas en España de 2008 a 2020

Motivación para realizar el proyecto

El principal motivo para la realización de este proyecto es la creación de un ejemplo o caso de estudio simulado que permita demostrar cómo se puede realizar esta migración de un servicio obsoleto hacia un servicio actual y que permita a la plataforma de comercio electrónico tener un crecimiento acorde a las necesidades del negocio sin necesidad de realizar grandes inversiones o nuevos comercios pagando de nuevo por la creación de un nuevo entorno.

También creo que existe un nuevo estigma en el mundo de los comercios electrónicos y es que en un primer momento parecía que solo las grandes empresas podrían tener una página web donde vendieran sus productos, pero ahora que ese muro ha sido superado, parece que de nuevo se vuelve a levantar un muro donde un comercio habitual parece que no puede tener más que lo que ya tiene, por tema de costes, dificultad, etc...

Por eso mi proyecto se muestra desde la perspectiva de un partner tecnológico que ayuda a realizar los despliegues de las nuevas infraestructuras necesarias para los comercios electrónicos y realiza la migración y mantenimiento de la web.

Ámbito de aplicación del proyecto

En este caso este proyecto se puede aplicar para cualquier servicio web que desee escalar su infraestructura actual hacia un clúster de cualquier infraestructura cloud, este proyecto muestra solo un ejemplo de cómo se podría realizar dicha migración de manera lo más transparente posible para los usuarios y propietarios de la web.

A mayores también se tratará de aplicar nuevos servicios a dichos servicios web para tratar de mejorar su rendimiento, por lo que también se obtendrá una mejora en el servicio actual.

En resumen, este proyecto se puede aplicar a cualquier entorno web que se encuentre en un servidor compartido único y desee mejorar el escalado, fiabilidad, y rendimiento de su plataforma web.

Objetivos del proyecto

El proyecto está desglosado en 3 hitos fundamentales que coinciden con las entregas parciales y totales de la memoria. Estos son:

- Preparación y despliegue de la nueva infraestructura más la migración de los servicios a esta nueva infraestructura.
- Despliegue e implementación de servicios de mejora para el servicio web actual como pueden ser servicios de monitorización y de backups.
- Puesta en producción del proyecto más revisión y monitorización de su correcto funcionamiento.

Tareas del proyecto

- Creación de contenedor del servicio web para su despliegue en el clúster a crear.

  El objetivo de esta tarea, es analizar la infraestructura actual y crear un contenedor de Docker con los servicios necesarios para el correcto funcionamiento de la web.

- Creación del contenedor del servicio de bases de datos para su despliegue en el futuro clúster de bases de datos.

  El objetivo de esta tarea, es analizar la infraestructura actual y crear un contenedor de Docker con los servicios necesarios para el correcto funcionamiento de la o las bases de datos.

- Creación del clúster de servidor web en el cloud público de AWS más su respectivo balanceador de carga.

  Analizar las posibilidades que nos ofrece Amazon Web Services para poder crear un clúster de servicios web con los recursos necesarios y desplegar nuestro contenedor web en él.

- Creación del clúster de bases de datos en el cloud público de AWS más su respectivo balanceador de carga.

  Analizar las posibilidades que nos ofrece Amazon Web Services para poder crear un clúster de servicios de bases de datos con los recursos necesarios y desplegar nuestro contenedor de bases de datos en él.

- Despliegue de los contenedores web en el clúster web creado.

  Una vez que tengamos claro sobre cuál es la mejor forma de realizar el despliegue de nuestros contenedores, realizaremos el despliegue en el clúster web.

- Despliegue de los contenedores de base de datos en el clúster de bases de datos creado.

  Una vez que tengamos claro sobre cuál es la mejor forma de realizar el despliegue de nuestros contenedores, realizaremos el despliegue en el clúster de bases de datos.

- Verificación de correcto funcionamiento a través del balanceador de carga.

  Tras el correcto despliegue se tendrá que verificar el correcto funcionamiento de los balanceadores de carga, así como verificar que la comunicación entre la web y la base de datos se esté estableciendo correctamente.

- Despliegue de instancias para implementar los sistemas de monitorización en nuestros clústeres.

  Para esta tarea en primer lugar tendré que investigar sobre las posibles soluciones de monitorización de clúster disponibles, así como los requisitos y configuración necesaria para la implementación en nuestro proyecto, y por último realizar el despliegue de las máquinas necesarias para su funcionamiento.

- Configuración de los sistemas de monitorización y verificación de su correcto funcionamiento.

  Una vez desplegada las máquinas necesarias será necesario configurar estos sistemas de monitorización para que nos muestren las métricas de funcionamiento de los servicios deseados de nuestro clúster.

- Configuración de los sistemas de copias de seguridad.

  Investigar cual es el mejor sistema para poder realizar las copias de seguridad de nuestro servicio web desplegado en clúster y como poder implementarlo correctamente.

- Configuración de DNS y dominio para apuntar a nuestra nueva plataforma.

  Investigar la mejor forma de resolver un dominio contra nuestro clúster, configurarlo e implementarlo.

- Puesta en producción de la web en la nueva infraestructura.

  Tras la configuración del dominio, y de los servidores DNS los servicios web ya tendrán que responder correctamente desde nuestro clúster.

- Verificación de rendimiento y disponibilidad.

  Mantener en observación la monitorización de los servicios verificando que el servicio es estable y responde de la forma correcta.

Requisitos

Para poder llevar a cabo este proyecto será necesario la siguiente infraestructura:

- Ordenador con conexión a internet.
- Acceso al servidor y a los servicios actuales.
- Servicio de Docker instalado en la máquina a migrar.
- Cuenta en el clúster a utilizar, en este caso AWS.
- Cuenta en el servicio de DNS a utilizar.

Cronograma

Tarea

Nombre	Fecha de inicio	Fecha de fin
Creación contenedor Docker ficheros aplicación	17/10/22	17/10/22
Creación contenedor Docker Base de datos	24/10/22	28/10/22
Preparación contenido	29/10/22	29/10/22
Creación cluster con 2 instancias web	31/10/22	5/11/22
Creación cluster con 2 instancias BD	31/10/22	5/11/22
Balanceador de carga	31/10/22	5/11/22
Instalación docker nueva infraestructura	31/10/22	5/11/22
Creación nueva infraestructura	6/11/22	6/11/22
Despligue aplicación	7/11/22	11/11/22
Comprobación funcionamiento	14/11/22	18/11/22
Despliegue en nueva infraestructura	19/11/22	19/11/22
1ª Entrega Memoria	19/11/22	19/11/22
Despligue Monitorización	21/11/22	25/11/22
Prometheus	21/11/22	25/11/22
Grafana	21/11/22	25/11/22
Configuración y personalización de la monitorización	28/11/22	2/12/22
Monitorización	3/12/22	3/12/22
Sistemas de backups en SSM	5/12/22	11/12/22
Comprobación Backups	5/12/22	11/12/22
Configuración Backups	12/12/22	12/12/22
Configuración CDN	12/12/22	16/12/22
Modificación DNS	12/12/22	16/12/22
Revisión funcionamiento servicio.	19/12/22	29/12/22
2ª Entrega Memoria	24/12/22	24/12/22
Finalización Go live	30/12/22	30/12/22
Entrega final memoria	15/1/23	15/1/23

Ilustración 2- Tareas a realizar

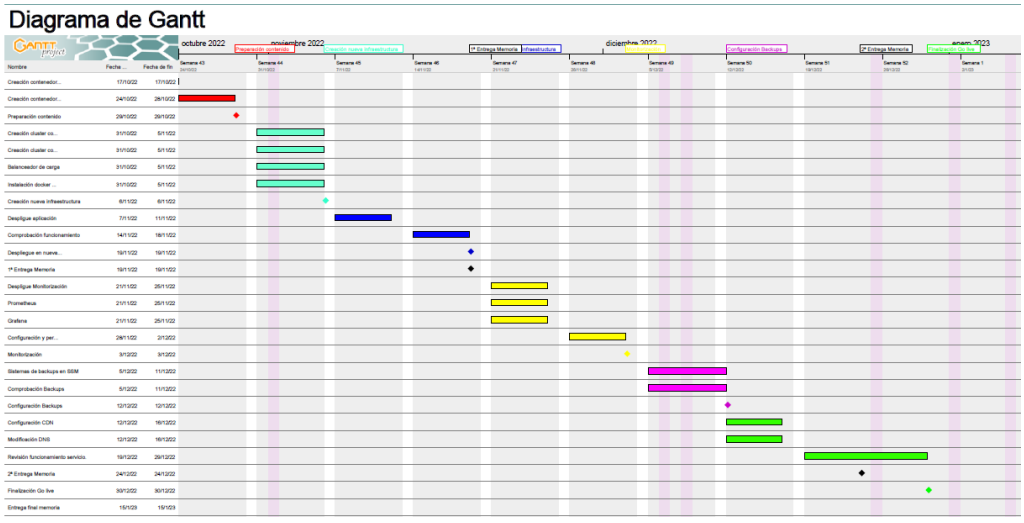


Ilustración 3- Cronograma del proyecto

1.- Análisis del servicio actual y preparación de la migración

En este apartado será necesario revisar las necesidades del servicio actual para así poder crear los contenedores de Docker con los mismos servicios, en este caso al tratarse de una aplicación simple como puede ser Prestashop y solo necesitar un servidor web, un servicio de PHP y un servidor de bases de datos, la forma más simple de hacerlo sería montar los contenedores con los servicios necesarios y realizar la migración de los ficheros del Prestashop y del SQL de la base de datos a dentro de cada contendor.

El servidor actual se encuentra bajo un sistema operativo Linux (centos7), es por ello que será necesario instalar Docker en la máquina para poder crear los contenedores e iniciar todo el proceso de migración.

Preparación de Docker.

Docker, se trata de un servicio de contenedores para la ejecución de software donde cada contenedor ya tienen todos los servicios, librerías, etc… necesarios para correr la aplicación, sin necesidad de instalarlos en la máquina anfitriona. Esto nos proporciona que nuestra aplicación pueda ser multiplataforma, y pueda correr en diversos sistemas operativos sin necesidad de adaptarla y por tanto evita también errores de versiones diferentes en diferentes máquinas, etc.

Docker tiene un repositorio de aplicaciones donde podemos encontrar las imagenes de estas aplicaciones con todo lo necesario para simplemente ejecutarlo en nuestra máquina, en primer lugar, vamos a realizar la instalación de Docker en el sistema operativo origen, tal y como nos indica la documentación de Docker, podemos instalarlo de la siguiente forma.

```
sudo yum install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Ilustración 4- Comando instalación Docker

Funcionamiento Docker.

Docker posee múltiples imágenes de contenedores ya preparados para su ejecución en sitios web como puede ser DockerHub, por ejemplo, si buscamos el servicio de MySQL en DockerHub encontraremos los diferentes servidores de MySQL que tenemos en contenedores de docker así como las explicaciones de cómo utilizarlos, por lo que solo será

necesario escoger un contenedor con la versión de MySQL que vayamos a usar y descargar a nuestra servicio, por ejemplo:

Si queremos descargar la imagen de mysql:8 podríamos realizarlo con el siguiente comando

```
docker run --name mysqldeejemplo -e MYSQL_ROOT_PASSWORD=admin -d mysql:8
```

Ilustración 5- Comando para ejecutar el contenedor de la imagen MySQL de Docker

Simplemente estamos ejecutando el comando run, añadiéndole el nombre al contenedor, más una variable de contraseña de acceso root al servidor MySQL e indicamos también la imagen que queremos descargar, en este caso de ejemplo la versión 8.

Con ese comando ya tendríamos el contenedor de mysql8 con todo lo necesario para funcionar.

```
[root@localhost ~]# docker run --name mysqldeejemplo -e MYSQL_ROOT_PASSWORD=admin -d mysql:8
Unable to find image 'mysql:8' locally
Trying to pull repository docker.io/library/mysql ...
8: Pulling from docker.io/library/mysql
feec22b5b798: Extracting [==================================>                ] 28.54 MB/40.58 MB
3b33952322b1: Download complete
8632ee03bb1c: Download complete
636ccd115361: Download complete
b07c8fac8eea: Download complete
e44c54db9c14: Download complete
cf9c45749101: Waiting
9f2fa3febc47: Waiting
44d5e1d3c311: Waiting
bb3db2c5d8ec: Waiting
e0ead729abd9: Waiting
```

Ilustración 6- Muestra de la ejecución/creación del contenedor.

Nuestro sistema será bajo MySQL por lo que eliminaremos este contenedor descargado e iniciaremos la descarga de los contenedores necesarios.
Un ejemplo de varios servicios con docker corriendo en la misma máquina puede ser la siguiente ilustración:

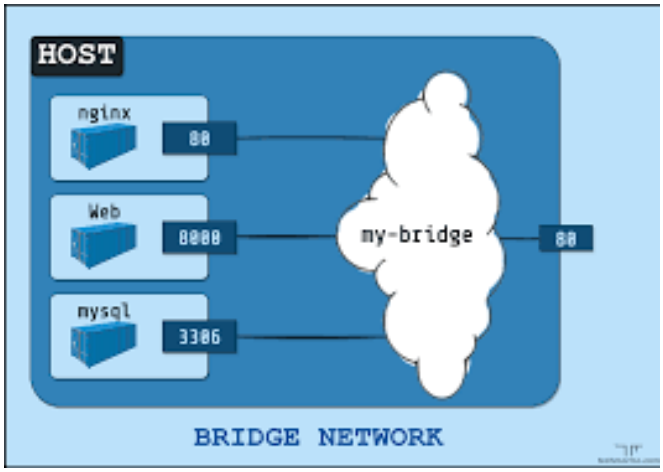


Ilustración 7- Gráfico explicativo del funcionamiento de la red de Docker

Y es que, si bien proporciona mucha facilidad para tener nuestra aplicación con todo lo necesario dentro de nuestro contenedor, este contenedor necesita tener comunicación con el exterior del mismo o incluso entre ellos, es por esto que a la hora de correr los contenedores se tienen que definir los puertos de escucha del host que aloja esos servicios y el puerto de escucha de nuestro contenedor para realizar el mapeado necesario y que las consultas que se realicen contra el servidor sean reenviadas a nuestro contenedor de docker. Además de esta situación de aislamiento de cada contenedor, no le permite comunicarse con otro contendedor por defecto, para ello docker permite crear redes internas de conectividad, donde desplegamos los contenedores para que puedan tener conectividad entre ellos. Un esquema de la distribución necesaria en el actual servidor para poder operar correctamente desde docker sería la siguiente:

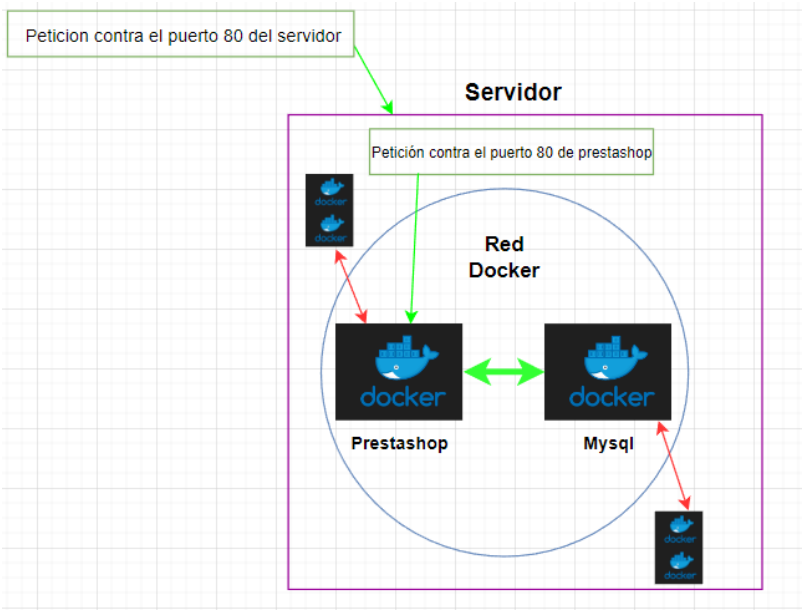


Ilustración 8- Grafico explicativo del funcionamiento de la infraestructura de docker

En el gráfico se puede observar que cada contenedor es individual y está aislado de los otros, pero se puede crear una red interna en docker y correr los contenedores dentro de esa red para que tengan conectividad entre ellos y puedan transferirse información, en nuestro caso se trata de algo necesario para el funcionamiento del e-commerce al requerir conectividad entre la base de datos y los ficheros de Prestashop.

Descarga y ejecución de contenedores.

Nuestra intención es la de tener dos contenedores, 1 para el servidor de MySQL y otro para el servidor web de Prestashop, dado que se van a montar en diferentes máquinas, es por eso que la forma más simple que se ha encontrado para realizarlo es descargar los contenedores de MySQL y de Prestashop por separado como si se trata de un Prestashop vacío, de esa forma una vez desplegados estos contenedores, realizaremos la copia del contenido de la web actual dentro de cada contenedor, y después, crearemos la imagen de ese contenedor para que esta pueda ser descargada y utilizada en cualquier otro servidor.

Antes de realizar esta tarea es necesario tener en cuenta la explicación de los contenedores de docker previa, y por ello vamos a ejecutar la descarga y ejecución de nuestros contenedores de la siguiente forma.

En primer lugar, creamos nuestra red interna de docker con el siguiente comando:

```
docker network create prestanet
```

Ilustración 9- Creación de red interna de Docker

Ahora ya tenemos una red interna en docker bajo el nombre de prestanet, por lo que vamos a desplegar nuestros contenedores, para desplegar el contenedor de MySQL, se puede desplegar con el siguiente comando.

```
docker run --name mysql --network prestanet -v mysql:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=admin -p 3307:3306 -d mysql:5.7.40
```

Ilustración 10- Ejecución y montaje del contenedor de docker de mysql

Este comando descarga y ejecuta la imagen de nuestro servidor MySQL, indicamos que corra el servicio en el rede prestanet, que guarde el contenido dentro del contenedor en la ruta */var/lib/mysql* , definimos la contraseña de root como *admin*, y realizamos la redirección de puertos indicando que las peticiones llegadas contra el servidor por el puerto 3307 se redirijan al puerto 3306 de nuestro contenedor de MySQL, por último escogemos la imagen que descargar y correr de MySQL, en este caso la versión 5.7.40.

Tras esto, realizamos la misma operación con el contenedor de Prestashop, con el siguiente comando:

```
docker run -ti --name prestashop --network prestanet -v prestashop:/var/www/html -e
DB_SERVER=mysql -p 80:80 -d prestashop/prestashop
```

Ilustración 11- Ejecución y montaje del contenedor de Docker de prestashop

Con este comando descargamos y ejecutamos el contenedor de Prestashop y lo asociamos a la red de docker llamada prestanet donde ya tenemos el servidor de MySQL corriendo, a mayores definimos también la ruta donde se encontraran los ficheros, así como el servicio de base de datos que vamos a utilizar en este caso el contenedor que hemos llamado MySQL, también realizamos la redirección de puertos para que toda petición al puerto 80 del servidor se envíe al puerto 80 de nuestro contenedor de Prestashop, por último el contenedor que hemos descargado y puesto en funcionamiento es el Prestashop/Prestashop.

Migración de ficheros al contenedor de Prestashop.

Si bien ya tenemos los contenedores desplegados, estos no tienen todavía nada de nuestra tienda, y se trata de un Prestashop vacío.

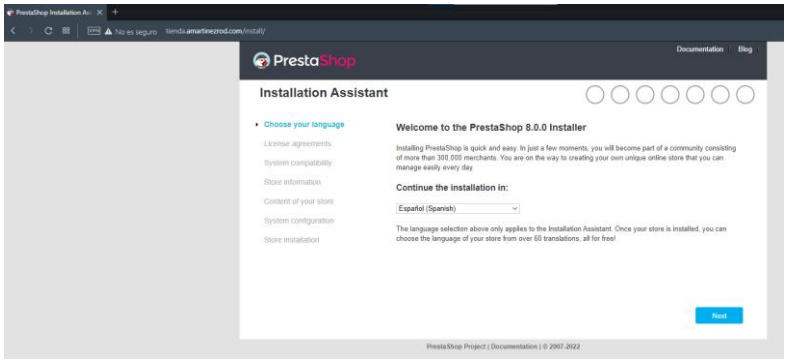


Ilustración 12- Respuesta prestashop sin desplegar ficheros ni base de datos

Por ello necesitamos realizar la migración de los ficheros y de la base de datos a los contenedores. En primer lugar, vamos a mover los ficheros de nuestro Prestashop desde nuestro servidor a nuestro contenedor Prestashop, para ello, vamos a acceder al contenedor de Prestashop para eliminar los ficheros de instalación que tiene por defecto.

```
docker exec -it prestashop /bin/bash
```

Ilustración 13- Acceso a contenedor prestashop

De esta forma entramos dentro de nuestro contenedor y podemos ver el contenido que tiene el mismo, procederemos a borrar el directorio */var/www/html/* para liberar el espacio del Prestashop sin contenido actual y copiaremos el contenido de Prestashop que tenemos en nuestra máquina a este contenedor.

Para copiar los ficheros de nuestra máquina buscaremos donde tenemos los ficheros de Prestashop y los moveremos a dentro del contenedor con el siguiente comando.

```
docker cp html/ prestashop:/var/www/html/.
```

Ilustración 14- Copiar ficheros de prestashop dentro de nuestro contenedor

Tras verificar que todo se mueva correctamente, ya tendremos finalizado nuestro contenedor de Prestashop y al tratar de acceder a nuestra web, ya no encontraremos el proceso de instalación de Prestashop, sino un error 500 dado que está realizando llamadas contra una base de datos que no encuentra actualmente.

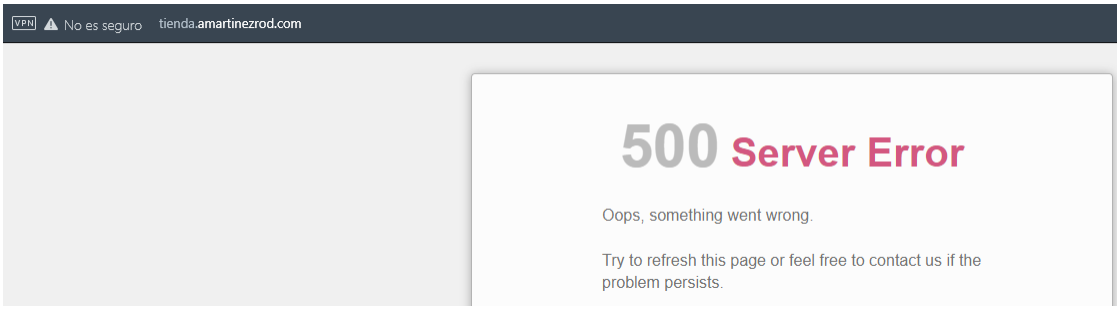


Ilustración 15- Respuesta de prestashop sin conexión contra la base de datos

Migración de base de datos al contenedor de MySQL.

Para realizar la migración de la base de datos en primer lugar es necesario crear la base de datos, y a poder ser con los mismos datos de conexión que tenemos en el Prestashop actual, por lo que creamos el usuario con la misma contraseña en la base de datos y también creamos la base de datos.

A continuación, procedemos a mover el backups en SQL de la base de datos que tenemos en nuestro servidor y lo metemos dentro del contenedor con el comando cp de docker.

```
docker cp Prestashop.sql mysql:/.
```

Ilustración 16 - Copiar los datos de la base de datos dentro del contenedor MySQL

Realmente el directorio destino nos da igual, porque después vamos a realizar la importación de ese SQL dentro de nuestra nueva base de datos.

Una vez finalizada la importación de la base de datos y tras revisar los permisos de usuario de la base de datos y propietarios de ficheros para que sean concordantes podemos comprobar que la web ya estaría respondiendo bajo nuestros contenedores de docker.

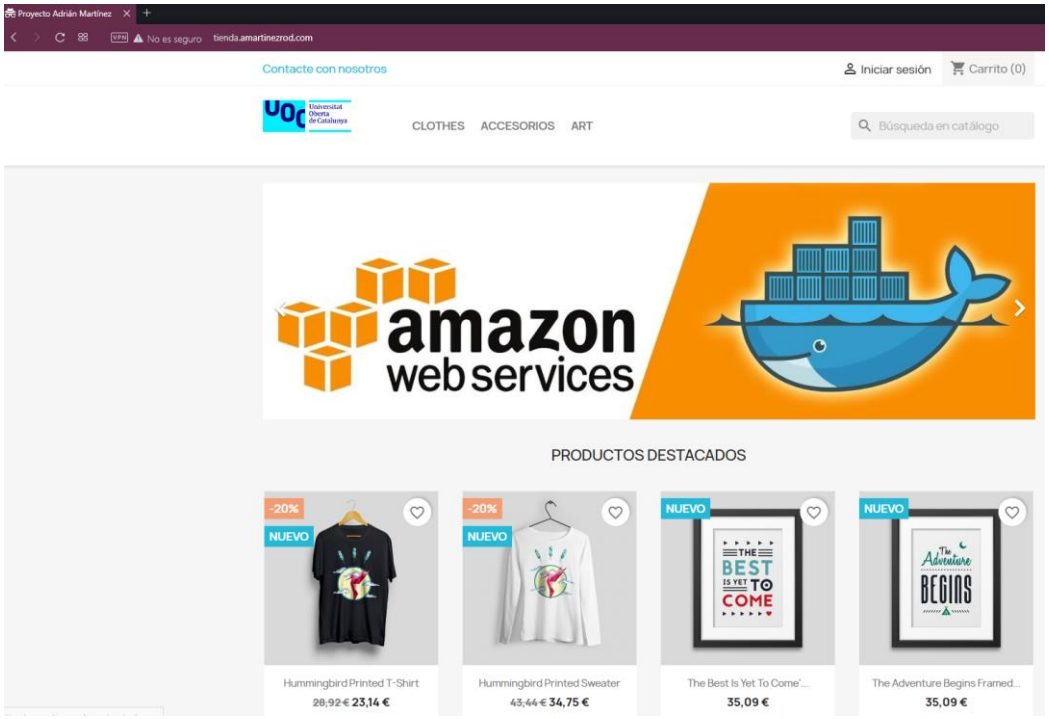


Ilustración 17- Resolución prestashop tras implementación de todos los datos.

Creación de las imágenes de Docker y subida de la imagen a repositorio de DockerHub.

Los contenedores de docker son volátiles, eso significa que si ahora borramos el contenedor no tendríamos forma de volver a recuperar el contenido de la web, para evitarlo necesitamos crear una imagen de nuestro contenedor. Para ello utilizaremos el comando commit para cada contenedor.

```
docker commit mysql tfg-mysql
docker commit prestashop tfg-prestashop
```

Ilustración 18- Realizar commit de nuestras imágenes de MySQL y Prestashop

De esta forma ya tenemos las imágenes creadas en nuestra máquina, y nos aseguramos de guardar una copia correctamente de estas imágenes en el exterior y así poder descargarla desde cualquier servidor cuando nos interese.

A continuación, será necesario cambiar el nombre de la imagen para subir a nuestro repositorio de DockerHub, y así tener nuestra imagen preparada para ser utilizada cuando queramos.

```
docker tag tfg-prestashop:latest maroansti/tfg-prestashop:migracion
docker tag tfg-mysql:latest maroansti/tfg-mysql:migracion
```

Ilustración 19- Cambiar el tag de las imágenes para poder subirlos a nuestro docker hub.

Por último, la subida de esta imagen llamada maroansti/tfg-*:migracion a nuestros repositorios de dockerhub se realiza con el siguiente comando:

```
docker push maroansti/tfg-prestashop:migracion
docker push maroansti/tfg-mysql:migracion
```

Ilustración 20- Subir nuestras imagenes a docker hub.

Si accedemos a nuestro repositorio de docker podemos ver ya nuestras imágenes subidas correctamente.

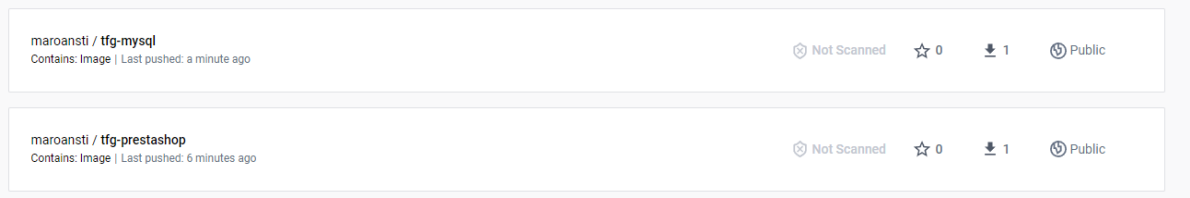


Ilustración 21 - Verificación de imagenes subidas a docker hub.

Para verificar que funciona todo correctamente, eliminamos todos los contenedores y todas las imágenes creadas con los comandos rm y rmi respectivamente de docker

```
[root@localhost lesben]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@localhost lesben]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@localhost lesben]#
```

Ilustración 22- Verificación contenedores e imágenes borradas.

Y volvemos a realizar la llamada de nuestras imágenes ya con nuestra web configurada, con los siguientes comandos.

```
docker run --name mysql --network prestanet -v mysql:/var/lib/mysql -e
    MYSQL_ROOT_PASSWORD=admin -p 3307:3306 -d maroansti/tfg-mysql:migracion

docker run -ti --name prestashop --network prestanet -v
    prestashop:/var/www/html -e DB_SERVER=mysql -p 80:80 -d maroansti/tfg-
    prestashop:migracion
```

Ilustración 23 - Ejecución y creación de los contenedores de docker de nuestras imágenes personalizadas.

Y una vez descargados y ejecutados nuestros contenedores de la imagen subida a dockerhub podemos confirmar que la web ya está *dockerizada* y respondiendo desde los servicios de docker. Por lo que ya la podremos migrar y desplegar en cualquier otro servidor que nos interese.

```
[root@localhost lesben]# docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS                               NAMES
68a5af7db38a        maroansti/tfg-prestashop:migracion   "docker-php-entryp..."   21 seconds ago      Up 19 seconds       0.0.0.0:80->80/tcp                  prestashop
63c1162d1c62        maroansti/tfg-mysql:migracion        "docker-entrypoint..."   About a minute ago  Up About a minute   33060/tcp, 0.0.0.0:3307->3306/tcp   mysql
[root@localhost lesben]#
```

Ilustración 24- Verificación de contenedores ejecutándose correctamente.

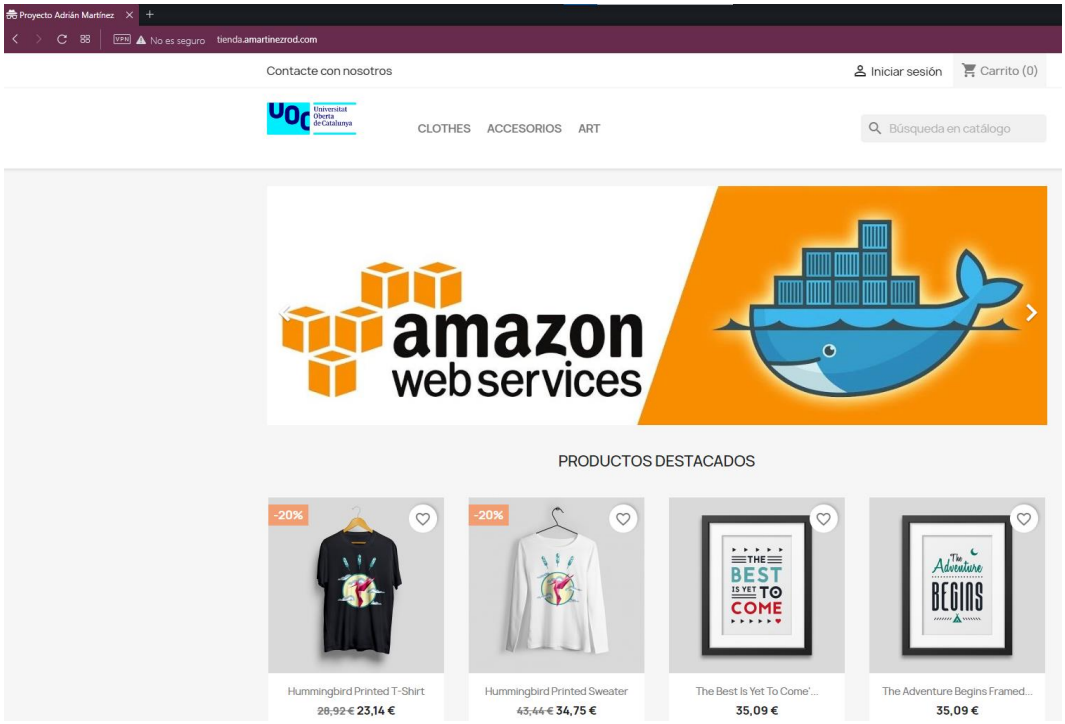


Ilustración 25 - Comprobación de respuesta de la nueva infraestructura.

2.- Análisis de la infraestructura necesaria a desplegar en aws.

Amazon Web Services, es el proveedor de alojamiento de servicios web de Amazon, el cual nos ofrece múltiples soluciones para poder desplegar cualquier infraestructura bajo sus servicios, pero antes de empezar a explicar todo el proceso de investigación y el motivo por el cual desplegamos ciertos servicios de AWS vamos a definir algunos de los servicios que nos proporcionan para así tener clara nuestra infraestructura.

Algunos servicios de AWS

- **EC2**: *Amazon Elastic Compute Cloud* son máquinas escalables de la propia nube de Amazon que nos permiten desplegar una máquina con unas características a escoger como sus recursos, sistema operativo, etc. Y desplegarla en unos pocos minutos.
- **VPC**: *Amazon Virtual Private Cloud*, se trata de una red privada virtual en la cual podemos desplegar nuestros productos y, permite definir los límites de lo que podríamos considerar red interna y red externa de nuestra infraestructura. A la hora de desplegar la mayoría de productos nos consulta bajo que red queremos desplegarlos, podemos crear diferentes VPC y configurarlas según nuestras necesidades.

- **Subredes**: Las VPC podemos dividirlas en subredes, tal y como podría estar definida una red local de una oficina donde podemos encontrar subredes departamentales las cuales se pueden configurar de diferente manera para tener unos accesos u otros, aunque todos se encuentren dentro de la misma red.
- ***Security Groups***: Los grupos de seguridad, se tratan de reglas que nos permiten definir dentro de una VPC o subred los permisos que damos al servicio para enviar/recibir, se tratan de filtros bastante configurables y se pueden crear hasta 50 reglas por cada security group, así como 500 security group por VPC. Se podría simplificar como una especie de firewall que permite darle seguridad a nuestros servicios. Cada servicio permite asociar un security Group, por lo que nos permite de una forma muy fácil fragmentar y decidir lo que permitimos y lo que prohibimos de nuestros servicios desplegados.
- ***ECR***: *Amazon Elastic Container Registry* se trata de un repositorio de imágenes en nuestro usuario de Amazon donde podemos subir tanto imágenes públicas como privadas de nuestros servicios para así ser utilizadas por los diversos gestores o administradores de contenedores que nos proporciona Amazon.
- **ECS**: *Amazon Elastic Container Service*, se trata de un servicio de administración de contenedores que nos permite desplegarlos tanto en máquinas físicas como pueden ser ec2 como mediante un modo sin servidores que simplemente corre los contenedores y facilita los servicios de los mismos. Este sistema permite desplegar con una configuración simple múltiples contenedores en múltiples servicios y permite configurar opciones como balanceadores de carga entre los servidores que ejecutan los contenedores seleccionados.
- **EFS**: *Elastic file system* se trata de un repositorio de ficheros que podemos montar como unidad o volumen de la máquina que necesitemos, este sistema permite tener de forma unificada los diversos ficheros de las diversas máquinas a desplegar dado que simplemente con montar el volumen de este repositorio de ficheros en nuestras máquinas todas tendrán los mismos ficheros, por lo que los datos serán persistentes entre todas las máquinas que utilicen este sistema de datos.
- **RDS**: *Amazon Relational Database Service* se trata de un servicio de bases de datos relacionales de Amazon que nos permite crear nuestras bases de datos bajo diversos motores de bases de datos e inclusive nos permite crear un cluster de bases de datos directamente bajo su infraestructura, estos clusters nos facilitan un endpoint o punto de conexión al cual nosotros apuntaremos nuestro servicio para tener conectividad contra la base o bases de datos, y, el sistema de administración de RDS se encarga de realizar el propio balanceo de carga y mantener los datos persistentes. Este servicio permite a mayores crear una base de datos simple, dos bases de datos en posición de maestro esclavo donde la esclava solo permite lectura y como ya he indicado un cluster de 3 o más bases de datos.
- **S3**: Amazon s3 es una simple unidad de almacenamiento en nuestro servicio de Amazon donde podemos alojar cualquier fichero que nos interese, así como las propias copias de seguridad de nuestra infraestructura.

¿Qué infraestructura montar?

Una vez que tenemos más claro algunos de los servicios que podemos desplegar, procederemos a explicar la idea principal del proyecto, así como el motivo por el que se fueron realizando los diversos cambios. En un primer momento se tenía la idea de realizar la creación de 2 máquinas EC2 con un balanceador de carga como endpoint entre los servidores web, donde se desplegaría un servicio de docker con la imagen de Prestashop que teníamos almacenada en ella trayéndola desde nuestro docker hub, estos contenedores web apuntarían a un balanceador de bases de datos del cual colgarían también dos instancias de máquinas ec2. Este proceso inicial aparte de no sacar todo el potencial de los servicios de aws, nos proporcionaba ciertas dificultades, así como carencias, por ejemplo, ¿cómo haríamos para que los datos fueran persistentes entre las diferentes máquinas? ¿Qué pasaría si existiera algún cuello de botella en los servicios que acabaran parando ambas máquinas creadas? ¿Qué costo tendría la realización de Snapshots de estas 4 máquinas? etc…

El esquema de esta infraestructura inicial podría definirse como en el siguiente gráfico.

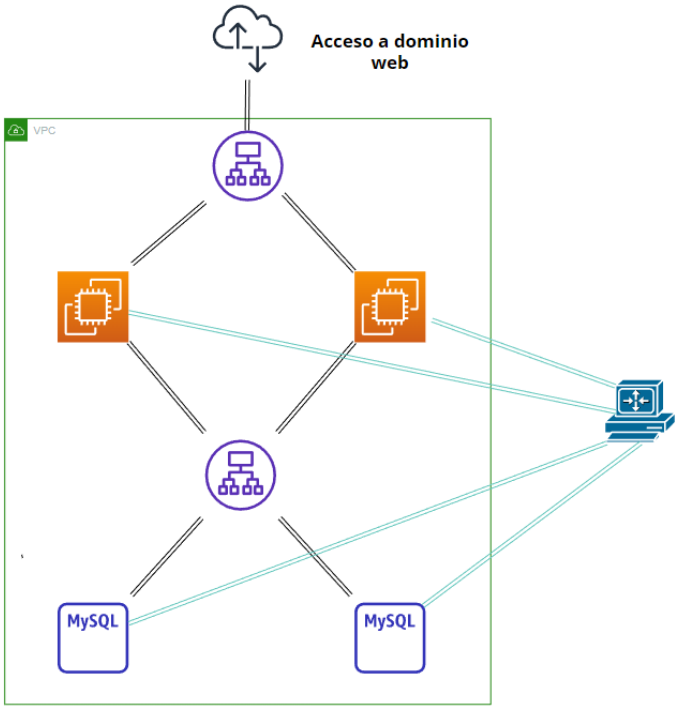


Ilustración 26 - Esquema inicial de nueva infraestructura.

Aunque en un primer momento, parecía una idea simple de ejecutar, al poco de intentar montar esta infraestructura fue desechada, dado que la redundancia y la alta disponibilidad sería una tarea más complicada de lo que a priori parece de implementar.

Es por eso que al tratar de realizar la redundancia y tras realizar las investigaciones se planteó la opción de montar simplemente las imágenes dentro de las máquinas EC2 pero los ficheros en un volumen agregado en cada máquina, esta idea también fue desechada al poco tiempo, dado que aunque las máquinas EC2 tuvieran los ficheros, sería necesario añadirlos dentro del contenedor de docker de cada máquina y seguiríamos con las complicaciones de configurar la salida/entrada de cada contenedor contra la salida/entrada de ambos balanceadores de carga, por lo que se consideró que no solventaba los problemas que se estaban observando en la infraestructura.

Un Esquema de este ejemplo podría ser el siguiente:

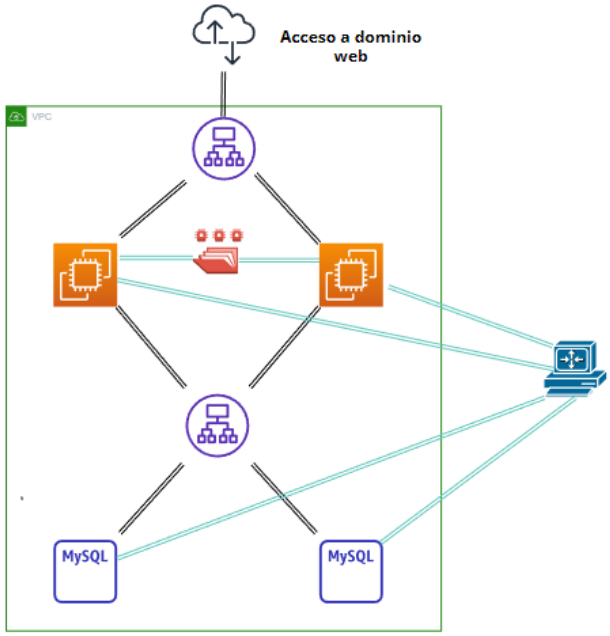


Ilustración 27- Esquema de la segunda propuesta de la nueva infraestructura.

Entonces tras investigar sobre las formas de desplegar contenedores y los servicios que proporcionaba AWS, se descubrió el servicio de ECS, donde se podría desplegar los contenedores directamente y ahí crear el cluster del mismo contenedor con su propio balanceador de carga tanto para MySQL como para servicios web, por lo que nos planteamos la opción de desplegar 2 clusters una para cada uno de los contenedores que tenía preparados, pero al intentar desplegarlos no se lograba que los cluster supieran interconectarse entre ellos e inclusive la se valoró la opción de crear un cluster de ambos contenedores MySQL y servicios web conjuntos por el número de réplicas deseadas, pero aun así parecía bastante dificultosa no solo la implantación sino la realización de la redundancia entre todos los servicios.

Ejemplos de las implementaciones descritas:

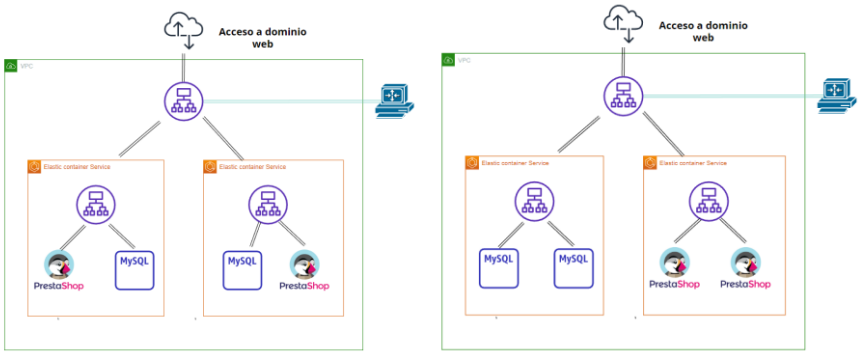


Ilustración 28- Esquemas de las dos posibles terceras nuevas infraestructuras.

En este momento, se descubrió el sistema de repositorio de imágenes de docker que se podía utilizar para almacenar las imágenes de docker dentro de AWS e incluso utilizarlas para el despliegue de los clusters sin necesidad de realizar la llamada contra un servicio externo como podría ser docker hub. Tras revisar más servicios disponibles de AWS también se descubrió el sistema RDS por lo que en ese momento se pensó en crear el cluster de Prestashop a través del sistema ECS y el cluster de bases de datos a través de RDS, dado que permitía la facilidad de crear la base de datos en este sistema y el mismo sistema se encargaría de crear el cluster y de administrarlo solo siendo necesario crear la base de datos y subir el contenido de la misma, por otro lado resultó ser muy interesante el hecho de que al tener solo la base de datos redundada, sería mucho más fácil la realización y recuperación de backups de la base de datos RDS, la cual ya permite realizar y definir las copias de seguridad en su creación, en vez de tener que realizar el Snapshots de una máquina entera. En este punto el proyecto estaba ya obteniendo una mejor valoración dado que la base de datos ya estaría redundada, con alta disponibilidad, y con simplicidad para subir nuevas versiones de la base datos, como para desplegar nuevos clusters o incluso para la realización de backups, pero se tenía que definir en los servidores web de ECS como conectar contra el cluster RDS de la base de datos. En este punto la infraestructura presentaba el siguiente esquema.

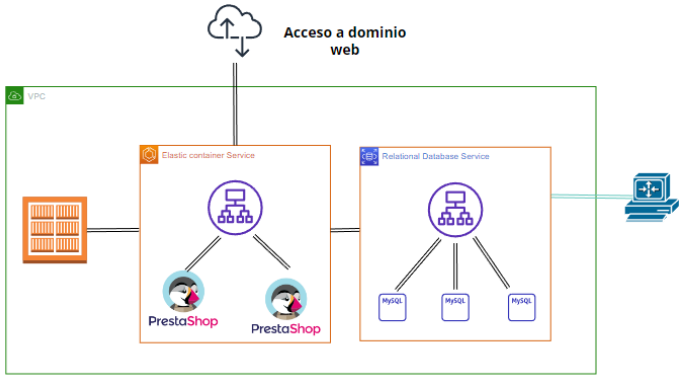


Ilustración 29- Esquema de la cuarta propuesta de la nueva infraestructura.

Aún nos encontrábamos con la dificultad de la persistencia de información en los ficheros, así como la dificultad de acceso para un usuario normal para poder actualizarlos, o, servicios de la aplicación sin necesidad de tocar la administración de Amazon como tal, en este punto se plantearon las opciones para realizar una forma cómoda donde el desarrollador o administrador de la web pudiera realizar modificaciones de su web de forma más simple y sin necesidad de tener que acceder a la administración de AWS, por ello decidimos que se podría utilizar la persistencia de un volumen de datos como habíamos creado anteriormente en vez de desplegar los contenedores individualmente con todos los datos. En ese momento, se decidió la que fue en ese momento y sigue siendo hasta ahora la infraestructura para el despliegue de esta aplicación bajo clusters de AWS.

Infraestructura decidida

Tras la explicación de cómo se ha llegado a esta decisión, voy a explicar por qué se consideró que la infraestructura finalmente planteada es acertada para esta simulación que se está realizando.

Para poner en contexto la infraestructura en primer lugar vamos a mostrar el esquema de la misma:

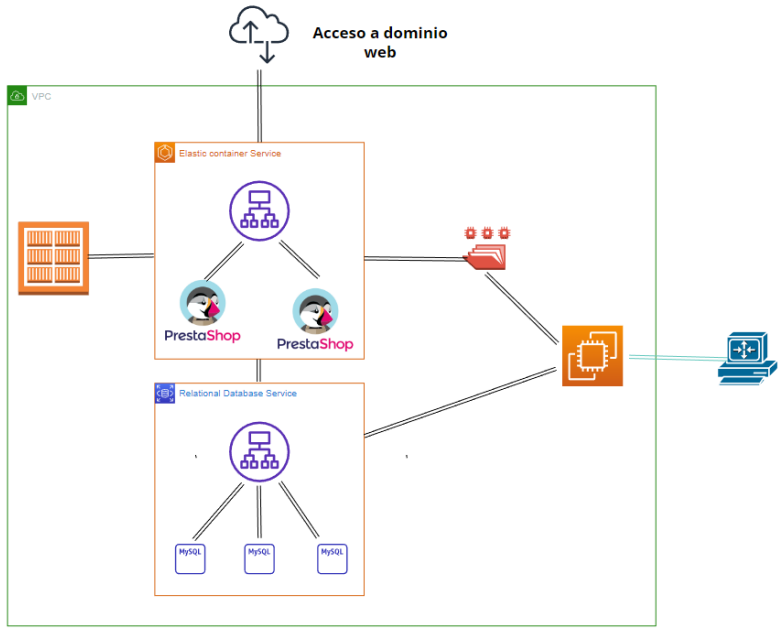


Ilustración 30- Esquema Final de la nueva infraestructura.

Pensando en no solo la plataforma si no en su seguridad y en la facilidad para que esta pueda ser modificada en un futuro, se valora la opción de tener toda la infraestructura cerrada al exterior y que desde fuera solo nos pudiéramos conectar a una máquina EC2 la cual tendrá ya conexión con toda la infraestructura y nos permitiría realizar todas las gestiones desde la misma, de esta forma un usuario final simplemente necesitaría conectarse por SSH a la

máquina ec2 y de ahí saltar a donde fuera requerido, esa máquina tendrá conectividad contra el cluster RDS, por lo que podría desde su propia terminal exportar/importar bases de datos contra el RDS sin necesidad de entrar en el panel de Amazon, del mismo modo tendrá montado el sistema de ficheros utilizado EFS por lo que podría eliminar/modificar/importar/exportar estos ficheros del mismo modo como si se tratara de un simple FTP. Por otro lado, tendremos los servicios de nuestro docker de Prestashop almacenados en un repositorio ECR y el ECS configurado para desplegar los servicios con las variables de entorno de conexión contra el RDS, el volumen de EFS y el contenedor de ECR, es por eso que, si en algún momento quisiéramos aumentar la infraestructura, simplemente con desplegar un nuevo cluster ECS o RDS con los volúmenes y bases de datos ya obtenidos se haría en cuestión de minutos, y si el usuario quiere modificar en vivo la web actual, también podría hacerlo de forma sencilla sin necesidad de tener un amplio conocimiento sobre la infraestructura que tiene montada, simplemente con tener conexión contra una máquina que está interconectada contra la base de datos y el volumen de ficheros utilizado. También esta infraestructura nos permite realizar backups contra el RDS y contra el EFS de forma sencilla y recuperar en caso de catástrofe estos backups muy fácilmente agilizando el nuevo despliegue de la nueva infraestructura con los datos correctos.

Las imágenes de docker creadas y guardadas en nuestro repositorio ECR podrán ser desplegadas para ejecutar un entorno de preproducción en una máquina local, sin necesidad de crear una nueva infraestructura para su reproducción, de esa manera nos permitirá realizar cambios sin afectar a la producción y una vez finalizados subirlos a nuestra infraestructura de clusters en producción de AWS.

3.- Creación de la infraestructura y despliegue de la aplicación.

Una vez ya tenemos nuestra infraestructura planeada empezaremos con el despliegue de la misma tratando de explicar cómo dar de alta todos los servicios, en este caso dado que se encuentra en desarrollo y por temas económicos no nos interesa realizar el gasto de la infraestructura hasta que sea completamente funcional trataremos de utilizar la capa gratuita de los servicios de Amazon para simplemente testear el correcto funcionamiento de la migración a estos servicios.

Antes de iniciar el alta de los servicios existen acciones previas en la consola de Amazon que serán necesarias realizar para la administración de los mismos, como crear un usuario con permisos de administración, configurar nuestra clave de acceso para poder conectarnos por ssh a nuestra máquina EC2, etc... Pero considero que estos puntos son acciones por defecto del servicio de AWS y que no aportan en la explicación de la creación de la infraestructura, por lo que serán reflejados aquí y en la bibliografía por si se necesitará revisar dicha información, pero no serán explicados en este documento.

Perímetro de seguridad (VPC, subredes, y security Groups)

En este caso, la idea final es limitar la entrada a la VPC a solo 1 equipo o a 1 red que pueda conectarse contra la instancia EC2 por ssh el resto de comunicaciones se realizaran mediante los protocolos http o https contra el balanceador de carga. Como la finalidad actualmente es solo probar los servicios y no montarlos en producción, abriremos todos los security groups para poder conectarnos a cualquier servicio directamente en caso de que fuese necesario mediante cualquier protocolo desde cualquier red, y antes de desplegar el servicio final en producción realizaremos los ajustes necesarios para permitir solo las conexiones obligatorias para el correcto funcionamiento del servicio, de esa forma cerraremos posibles accesos indeseados.

Es por ello que, para simplificar el desarrollo, vamos a utilizar la VPC que nos facilita nuestra cuenta de Amazon por defecto con sus 3 subredes y sus security groups. Por lo que en los servicios desplegados vamos a seleccionar siempre que sea posible las 3 subredes para tener todos los servicios en contacto dentro de la VPC y vamos a abrir todas las conexiones de los security groups. Por último, vamos a especificar que una VPC con toda la configuración de la misma, solo se encuentra disponible dentro de una misma área de AWS, en este caso para nuestro proyecto vamos a desplegar en la zona de Irlanda (eu-west-1) aunque en breves se podrá desplegar directamente en un área de España.

Para esto accederemos a nuestra sección de EC2 en la consola de AWS y de ahí accederemos a la sección de security groups, donde se nos mostraran nuestros security groups creados, los cuales podemos crear 1 por servicio, pero por el momento solo vamos a explicar que vamos a acceder con los security groups que tengamos definidos y vamos a editar su configuración tanto de entrada como de salida para permitir todo el tráfico de cualquier red.

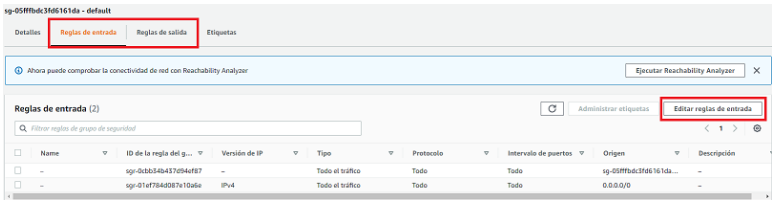


Ilustración 31- Acceso a la configuración de los Security Groups

Seleccionaremos las reglas de entrada y salida y las editaremos para permitir el acceso.

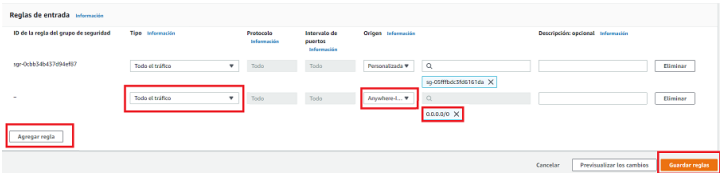


Ilustración 32 - Configuración de los security groups.

De esta forma ya tendremos toda la red con permisividad total para la comunicación y así poder testear todo correctamente.

Creación EFS

Para crear nuestro EFS, vamos a la sección de EFS y creamos nuestro volumen de datos añadiendo la red VPC escogida para la infraestructura, podríamos personalizar este volumen para crear copias de seguridad subredes, etc… pero el apartado de copias lo realizaremos posteriormente y las subredes vamos a utilizar todas las que la VPC tenga disponible por lo que no sería necesario entrar en mucho más detalle, simplemente escogeremos el almacenamiento estándar para en caso de querer realizar el cluster en diferentes zonas podamos compartir el volumen.

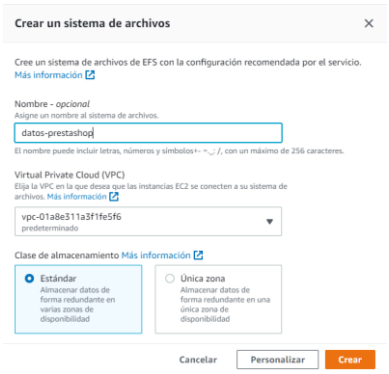


Ilustración 33- Muestra de cómo crear un EFS.

Una vez creado, si accedemos a nuestro repositorio de ficheros nos mostrará una opción llamada asociar donde nos dará la información y los comandos necesarios para conectar este repositorio de datos como un volumen en una máquina:

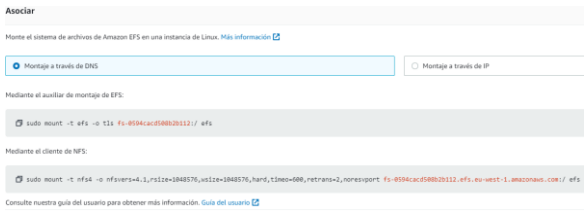


Ilustración 34- Datos de configuración de nuestro EFS.

Lo cual aplicaremos en nuestra máquina EC2 y en ECS posteriormente.

Creación RDS

Para la creación del RDS iremos a la sección con el mismo nombre dentro de AWS y le daremos a crear nueva base de datos, escogeremos un nombre, el motor de base de datos deseado, el tipo de instancia a montar para las bases de datos y los datos de conexión

maestros contra el RDS. En nuestro caso por el momento vamos a seleccionar la capa gratuita, la cual solo nos permite desplegar 1 nodo de bases de datos colgando del RDS, pero para testear la migración y el funcionamiento del servicio nos servirá y así podremos realizarlo sin costes adicionales. En el momento del despliegue final podremos ampliar el servicio con el cluster de 3 bases de datos en caso de ser necesario.

Una vez creado nuestro sistema RDS si accedemos al mismo y ya nos dará una descripción del sistema montado, así como el endpoint o host del mismo, su puerto, su VPC, su security group, y más información que unido a los datos de conexión maestros podremos utilizar para conectarnos posteriormente desde la máquina EC2 y crear nuestra base de datos e importar el SQL de la misma.

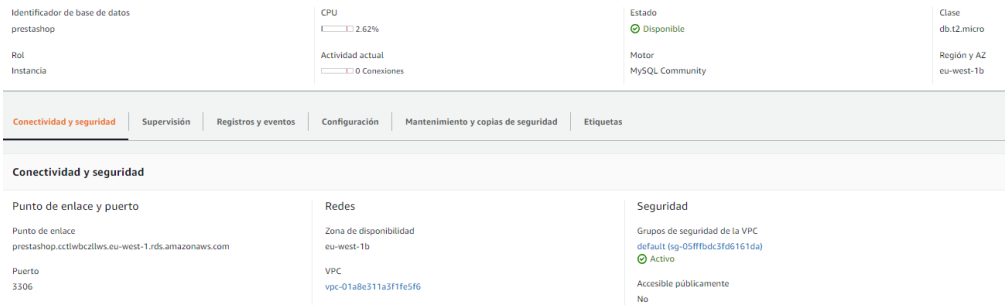


Ilustración 35- Datos de configuración y rendimiento de un RDS.

Creación ECR

Para la creación de nuestro repositorio de imágenes de docker, accederemos al servicio de ECR y crearemos un repositorio bajo el nombre que queramos y definiremos unas características del mismo así como si deseamos que sea público privado, que no se pueda modificar etiquetas, análisis de las imágenes a subir para verificar que se encuentren correctas y un cifrado de seguridad para las mismas, en este caso vamos a crear el repositorio indicando solo el nombre, aunque en un entorno de producción real sería optimo seleccionar las diversas opciones que nos proporciona para asegurarnos del correcto funcionamiento.

Una vez creado el repositorio, si accedemos a él, podremos observar un botón que nos facilita los comandos de envío necesarios para poder subir nuestra imagen a este repositorio desde nuestra máquina EC2.

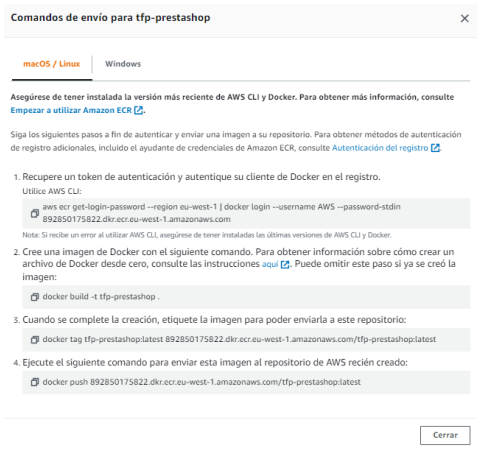


Ilustración 36 - Datos de configuración de un ECR.

Creación maquina ec2.

Como en el resto de servicios, para crear la máquina ec2 vamos a acceder a su propia sección en el panel de AWS, dentro de este servicio accederemos a la subsección de instancias y en ella pulsaremos en crear instancia, aquí vamos a crear nuestra máquina EC2 de salto donde utilizaremos la capa gratuita, con las posibilidades que nos oferta, la VPC por defecto, el security group por defecto, crearemos un par de claves RSA para conectarnos a la máquina por ssh desde nuestro ordenador y limitaremos si así lo deseamos el acceso SSH a nuestra red para limitar las conexiones de personas externas a nuestra a red a la instancia EC2. Este paso por el momento va a ser omitido, pero en el momento de entrega al cliente, se podrá desplegar de nuevo con las claves del cliente y limitar los accesos a las redes del propio cliente para evitar accesos de personas externas a nuestra red.

Una vez desplegada la máquina, podemos acceder a ella y obtendremos una descripción de la misma, así como un botón de conectar, donde al pulsar nos indica las diversas formas de conexión que tenemos contra esta instancia EC2, en nuestro caso actual como hemos creado un par de claves para conexión SSH desde nuestro ordenador vamos a utilizar ese procedimiento para conectarnos a la máquina.

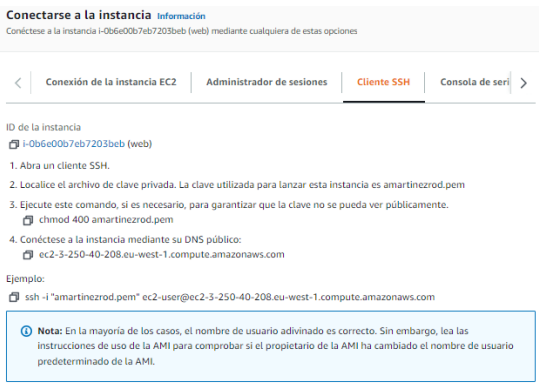


Ilustración 37- Datos de acceso ssh a la máquina ec2 creada.

Por lo que, tras descargar nuestro certificado, y seguir las instrucciones que nos proporciona el panel de AWS, ya tendremos acceso a nuestra máquina EC2 para iniciar la subida de nuestro contenedor, ficheros y base de datos.

Ilustración 38 - Confirmación de acceso a la máquina EC2.

Configuración y preparación de la maquina ec2.

En primer lugar, es necesario instalar todas las opciones que vamos a requerir tanto para la importación de nuestros ficheros, base de datos e imágenes de docker, para ello necesitaremos en primer lugar actualizar los paquetes por defecto ya instalados en la máquina para ello simplemente lo realizaremos mediante el siguiente comando:

```
sudo yum update -y
```

Ilustración 39 - Actualización de paquetes instalados en la máquina EC2.

Una vez actualizados los repositorios de la máquina instalaremos nuestra aplicación de docker, un servicio de MySQL para conectar contra el RDS y el servicio necesario para conectar contra el volumen EFS, así como su montaje.

- **MYSQL**: Para instalar el MySQL, ejecutaremos la instalación desde el repositorio por defecto, para ello ejecutaremos el siguiente comando:

```
sudo yum install mysql -y
```

Ilustración 40 - Instalación del servicio de MySQL en la máquina EC2.

- **Docker:** Para la instalación de docker será necesario realizarlo desde el repositorio de extras de Linux, por lo que será necesario realizar la siguiente serie de comandos para instalarlo, arrancarlo, configurarlo en el arranque del sistema y darle privilegios de uso del servicio docker al usuario ec2-user que es el usuario por defecto de nuestra instancia ec2.

```
sudo amazon-linux-extras install docker
sudo service docker start
sudo systemctl enable docker
sudo usermod -a -G docker ec2-user
```

Ilustración 41- Instalación y configuración de docker en la máquina EC2.

- **EFS:** Para montar el repositorio, tenemos que instalar las herramientas de efs-utils de Amazon y después solo será necesario crear una carpeta donde queramos que se encuentre el volumen EFS, y ejecutar las instrucciones descriptas anteriormente en la creación de nuestro volumen EFS.

```
sudo yum install -y amazon-efs-utils

mkdir ~/efs-mount-point
   "sudo mount -t nfs -o
    nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport
    ip-172-31-41-237.eu-west-1.compute.internal:/   ~/efs-mount-point"
```

Ilustración 42- Instalación y montaje del volumen EFS.

Ya podremos verificar que tenemos nuestro volumen montado en nuestra máquina EC2

```
[ec2-user@ip-172-31-41-237 efs-mount-point]$ df -hT
Filesystem                                      Type      Size  Used Avail Use% Mounted on
devtmpfs                                        devtmpfs  474M     0  474M   0% /dev
tmpfs                                           tmpfs     483M     0  483M   0% /dev/shm
tmpfs                                           tmpfs     483M  440K  483M   1% /run
tmpfs                                           tmpfs     483M     0  483M   0% /sys/fs/cgroup
/dev/xvda1                                      xfs       8.0G  6.2G  1.9G  77% /
fs-0594cacd508b2b112.efs.eu-west-1.amazonaws.com:/ nfs4      8.0E  508M  8.0E   1% /home/ec2-user/efs-mount-point
tmpfs                                           tmpfs      97M     0   97M   0% /run/user/1000
[ec2-user@ip-172-31-41-237 efs-mount-point]$
```

Ilustración 43- Verificación de volumen montado correctamente en la máquina EC2.

Migración de contenidos a nuestra infraestructura

En primer lugar, vamos a subir nuestra imagen de docker al repositorio ECR que hemos creado, para ello vamos a descargar de nuestro docker hub la imagen, realizar los cambios de tag y de nombre de la imagen para poder realizar el push correcto hacia nuestro servicio ECR, para ello, es necesario identificarse con los datos de nuestra cuenta en nuestra instancia de EC2 esto se realiza simplemente ejecutando el comando aws configure e introduciendo los datos de conexión contra la CLI de nuestro usuario.
Una vez registrado nuestro usuario iniciamos la descarga de nuestra imagen de docker de nuestro repositorio, modificamos los tag acorde a los necesarios para el servicio de ECR, registraremos nuestro docker con nuestro usuario para poder realizar la subida y enviaremos nuestra imagen hacia el repositorio de ECR para ser utilizado en el despliegue de ECS futuro.

```
docker pull maroansti/tfg-prestashop:migracion

aws ecr get-login-password --region eu-west-1 | docker login --username AWS --
    password-stdin 892850175822.dkr.ecr.eu-west-1.amazonaws.com

docker tag maroansti/tfg-prestashop:migracion 892850175822.dkr.ecr.eu-west-
    1.amazonaws.com/tfp-prestashop:migracion

docker push 892850175822.dkr.ecr.eu-west-1.amazonaws.com/tfp-
    prestashop:migracion
```

Ilustración 44- Descarga, registro, modificación y subida de nuestra imagen de Prestashop a ECR.

De esta forma ya tendríamos nuestra imagen subida correctamente a nuestro repositorio ECR y ya podremos utilizarla en nuestro cluster ECS cuando este sea desplegado.

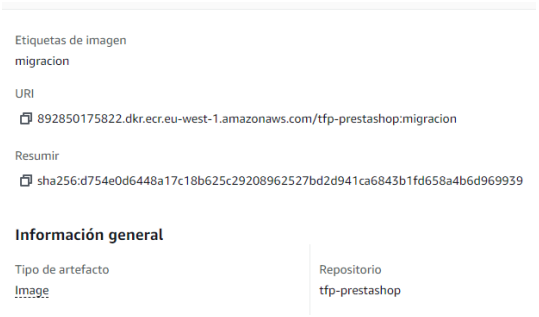


Ilustración 45- Verificación imagen subida a ECR correctamente.

Ahora realizaremos la subida de nuestros ficheros de SQL y del comprimido de Prestashop que teníamos previamente a nuestra máquina EC2 para ello Amazon nos permite realizar la

subida mediante el protocolo de transferencia de datos SFTP utilizando nuestro certificado creado para esta máquina.

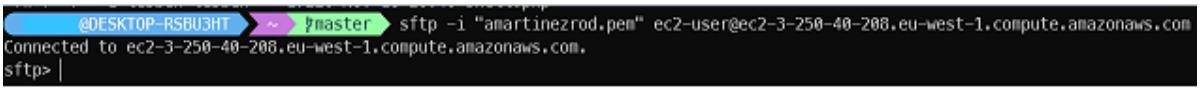


Ilustración 46- Verificación conexión por SFTP contra la máquina EC2.

Y mediante el comando put que nos proporciona el sistema podremos subir nuestros ficheros para desplegarlos en el volumen e importarlos en nuestra base de datos.

```
put prestashop.sql prestashop.tar.gz
```

Ilustración 47- Subida de ficheros de Prestashop y datos de nuestra base de datos a la máquina EC2.

De esta forma, nuestros ficheros y exportación de base de datos ya están subidos a nuestra instancia de EC2 y ya podremos importarlos ficheros en el volumen efs, así como el SQL en la base de datos, simplemente accedemos de nuevo a nuestra instancia EC2 y ejecutamos los comandos para mover, descomprimir y eliminar el comprimido para evitar consumir más espacio del necesario.

```
mv prestashop.tar.gz /home/ec2-user/efs-mount-point/.
tar -xvzf prestashop.tar.gz
rm -rf /home/ec2-user/efs-mount-point/prestashop.tar.gz
```

Ilustración 48 - Mover y descomprimir los ficheros de Prestashop en el volumen EFS y eliminar el comprimido.

Será necesario revisar que los propietarios de los ficheros sean concordantes con los que tenemos en nuestro contenedor de docker, dado que ese contenedor será el encargado de ejecutarlos y necesitará que el propietario sea el apache que tiene instalado el contenedor subido a ECR. Por otro lado, la conexión contra la base de datos ya no se realizará contra un contenedor MYSQL en local, por lo que es necesario acceder al fichero de configuración de la base de datos en Prestashop, para indicar el endpoint de nuestro RDS y así que el Prestashop sepa a donde tiene que consultar la base de datos, el resto de datos se mantendrán igual.

```
[ec2-user@ip-172-31-41-237 efs-mount-point]$ cat app/config/parameters.php
<?php return array (
  'parameters' =>
  array (
    'database_host' => 'prestashop.cctlwbczllws.eu-west-1.rds.amazonaws.com',
```

Ilustración 49- Modificar valores de conexión del Prestashop contra la base de datos.

Ahora solo nos queda acceder a nuestro motor de base de datos RDS, crear nuestra base de datos e importar nuestro SQL en ella. Para ello en primer lugar, vamos a acceder a la base de datos con los datos de usuario maestro que pusimos en la creación de la misma.

```
mysql -h prestashop.cctlwbczllws.eu-west-1.rds.amazonaws.com -u prestashopuser -p
```

Ilustración 50 - Conexión contra el sistema RDS creado.

Tras poner nuestra contraseña, estaremos dentro de nuestro RDS, por lo que utilizaremos la sintaxis normal de MySQL para crear una base de datos y asignar privilegios a nuestro usuario.

```
CREATE DATABASE prestashop;
SELECT user,host FROM mysql.user;
```

Ilustración 51- Creación de base de datos y verificación de privilegios de usuarios

En este caso para simplificar las cosas el usuario maestro y la contraseña es el mismo que el que teníamos configurado previamente para la conexión de nuestro Prestashop contra la base de datos Prestashop, por lo que no es necesario realizar la creación del usuario y darle los privilegios dado que ya los tiene.

```
MySQL [(none)]> SELECT user,host FROM mysql.user;
+----------------+-----------+
| user           | host      |
+----------------+-----------+
| prestashopuser | %         |
| mysql.session  | localhost |
| mysql.sys      | localhost |
| rdsadmin       | localhost |
+----------------+-----------+
4 rows in set (0.00 sec)

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| mysql              |
| performance_schema |
| prestashop         |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

Ilustración 52- Verificación de privilegios de la base de datos.

Ahora necesitamos volver a nuestra máquina de EC2 y realizar una importación normal de nuestro fichero SQL hacia la base de datos Prestashop que acabamos de crear mediante el siguiente comando:

```
mysql -h prestashop.cctlwbczllws.eu-west-1.rds.amazonaws.com -u prestashopuser
    -p prestashop < prestashop.sql
```

Ilustración 53- Importación de datos contra nuestra base de datos creada en RDS.

Tras poner la contraseña ya nos importará el contenido del fichero SQL en nuestra base de datos.

```
MySQL [prestashop]> show tables;
+-----------------------------------------------+
| Tables_in_prestashop                          |
+-----------------------------------------------+
| ps_access                                     |
| ps_accessory                                  |
| ps_address                                    |
| ps_address_format                             |
| ps_admin_filter                               |
| ps_advice                                     |
| ps_advice_lang                                |
```

Ilustración 54- Verificación de tablas de nuestra aplicación creadas correctamente en nuestra base de datos RDS.

Por lo tanto, ya tendremos toda la información de nuestra aplicación subida a nuestra infraestructura y solo nos faltaría lanzar los clusters de ECS utilizando la conexión contra el EFS, el RDS y el ECR para montar los contenedores de los servicios necesarios para la ejecución del Prestashop, montar la conectividad entre el cluster de ECS y las bases de datos de RDS, y montar el volumen en el directorio donde escucha el apache en nuestro cluster.

Crear y desplegar cluster ECS

En primer lugar, es necesario entender cómo funciona el cluster ECS, básicamente crearemos el cluster desde la sección del servicio de ECS en nuestro panel de AWS, donde nos pedirá un nombre para el cluster, una VPC donde crear el cluster, a que subredes tendrá acceso, el tipo de infraestructura que va a tener el cluster (máquinas ec2 o sistema Fargate "sin servidor"), por lo que ponemos el nombre del cluster, seleccionamos nuestra VPC por defecto y escogemos las 3 subredes por defecto también, por último escogeremos el sistema Fargate y crearemos el cluster.

Una vez creado el cluster, es necesario crear los servicios que queremos dentro de nuestro cluster, pero para crear un servicio primero es necesario definir una tarea del mismo, por lo

que iremos a la sección Definición de tareas, y ahí será donde definiremos los volúmenes, base de datos, etc…

Al crear una nueva tarea nos pedirá, en primer lugar, un nombre para la tarea, y una imagen para correr en un contenedor, así como el puerto en el cual va a estar escuchando el contenedor, en este caso el nombre sería irrelevante, la imagen usaremos la URI que tenemos de nuestra imagen subida a ECR y el puerto al ser un servicio web utilizaremos el 80 por el momento.

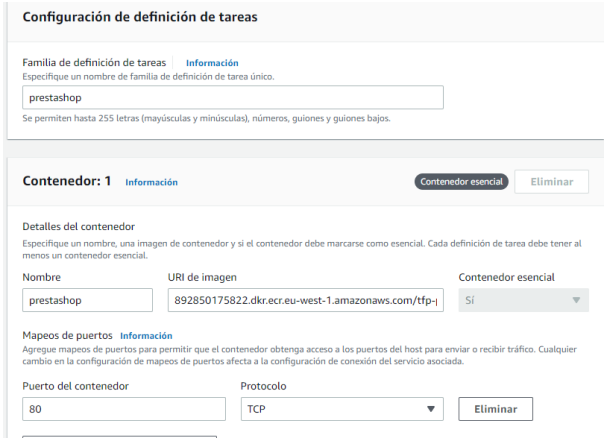


Ilustración 55- Creación de tarea ECS | Configuración de contenedores ECR.

También aprovechamos para configurar las variables de entorno para conectar nuestro contenedor contra el sistema de base de datos de RDS.

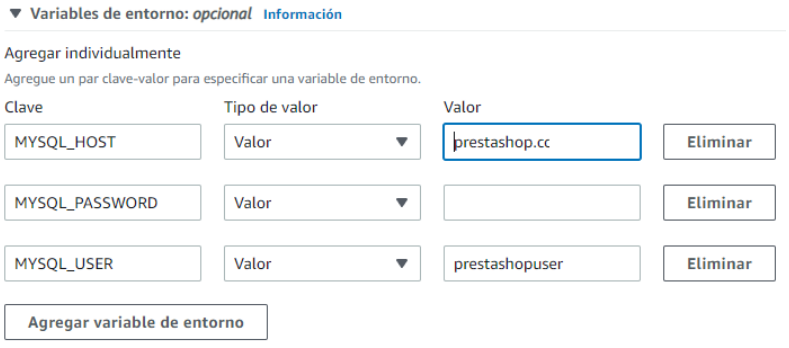


Ilustración 56 - Asignación de variables de entorno MySQL a la tarea ECS.

El resto de apartados podemos dejarlos por defecto, por lo que nos faltaría todavía indicar el volumen, parece que el nuevo asistente, aunque permita montar un volumen no contempla la opción de permitir escoger la ruta donde queremos montar el volumen, es por eso que una vez creada la tarea, la editaremos mediante Json para crear una nueva versión de la misma, pero con el volumen a montar y donde montarlo a mayores. Para ello seleccionamos nuestra tarea definida y pulsamos la opción de “Crear revisión con JSON”.

Aquí simplemente añadiremos en los campos mountPoints y volúmenes los datos de nuestro volumen EFS y donde montarlo.

```
"mountPoints": [
  {
    "sourceVolume": "datos-prestashop",
    "containerPath": "/var/www/html",
    "readOnly": false
  }
],
```

```
"volumes": [
  {
    "name": "datos-prestashop",
    "efsVolumeConfiguration": {
      "fileSystemId": "fs-0594cacd508b2b112",
      "rootDirectory": "/",
      "transitEncryption": "DISABLED"
    }
  }
],
```

Ilustración 57 - Configuración Volumen EFS en nuestra tarea ECS.

Volvemos a desplegar la revisión de nuestra tarea, por lo que ya tendremos 2 tareas como mínimo creadas, en este caso tenemos más por diversas pruebas que se habían realizado previamente. Esto se realiza de esta forma por si en un futuro se quiere cambiar algo de la arquitectura del cluster ECS, si encontráramos algo incorrecto, podríamos volver a lanzar el cluster con la imagen utilizada previamente y que sabemos que es correcta.

	Definición de tarea: revisión	Estado
○	prestashop:6	ACTIVE
○	prestashop:5	ACTIVE
○	prestashop:4	ACTIVE
○	prestashop:3	ACTIVE
○	prestashop:2	ACTIVE
○	prestashop:1	ACTIVE

Ilustración 58- Confirmación de versiones en nuestra tarea ECS.

Por último, solo quedaría por realizar el acceso de nuevo al cluster ECS e implementar la tarea actual con los valores y configuración que deseemos. En primer lugar, nos informará de en qué cluster se va a aplicar la implementación de servicios, también escogemos la plataforma de este servicio que en nuestro caso se indicó anteriormente se tratará de Fargate.

A continuación, nos preguntará sobre el tipo de aplicación que queremos desplegar y especificar la tarea que queremos utilizar para desplegarla, en este caso escogeremos una aplicación de tipo servicio, y seleccionaremos la tarea Prestashop en la revisión 6, también indicaremos que queremos que se ejecuten 2 tareas (esto definirá que se ejecuten 2 contenedores de Prestashop con sus servicios para montar el cluster de máquinas) si quisiéramos ampliar estos frontales simplemente añadiríamos un número mayor de frontales y se desplegarían.

En el apartado de redes y seguridad como hemos realizado durante todo el proceso escogeremos la VPC por defecto, todas las subredes y el security group por defecto.

A continuación, nos dará como característica opcional la implementación de un sistema de balanceo de cargas entre los 2 contenedores que vamos a desplegar, donde tendremos que definir el nombre del balanceador, el puerto de escucha del balanceador y el puerto de escucha de los contenedores, es decir en este caso sería para ambos el puerto 80, el grupo de destino, que en este caso se tratarán de los servicios Fargate a desplegar, y el resto de valores podemos dejarlos por defecto

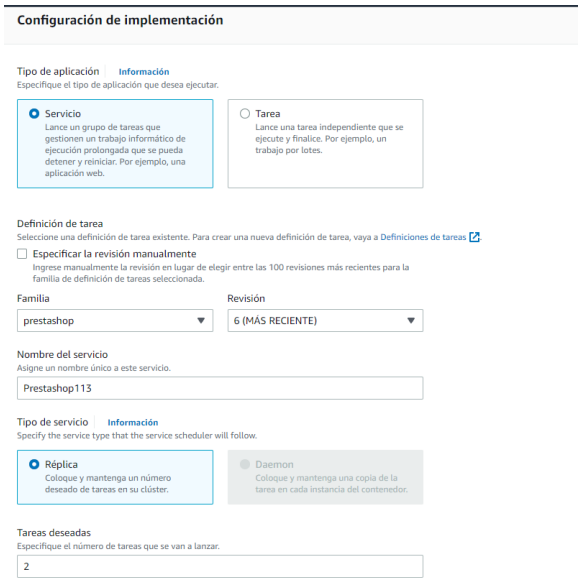


Ilustración 59- Primer paso de implementación de la tarea ECS en nuestro cluster ECS.

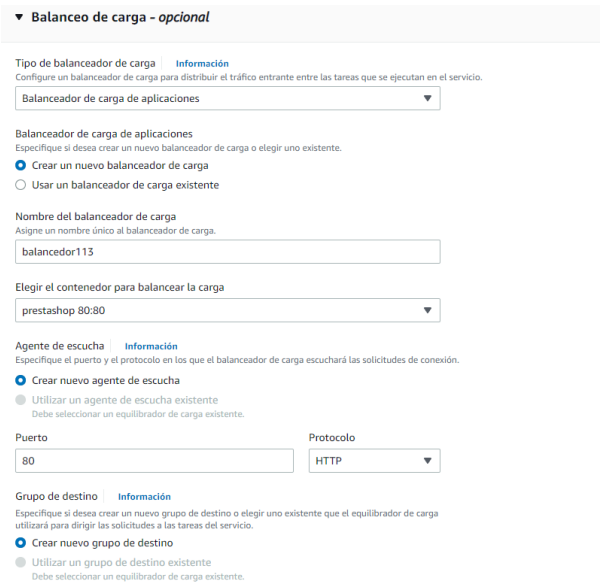


Ilustración 60- Segundo paso de implementación de la tarea ECS en nuestro cluster ECS.

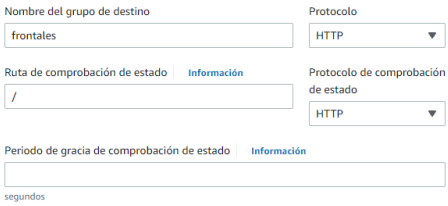


Ilustración 61- Tercer paso de implementación de la tarea ECS en nuestro cluster ECS.

Una vez configurada la implantación del servicio pulsamos en implantar y el propio servicio de ECS con todo lo proporcionado en nuestra tarea creada previamente nos levantará 2 máquinas Fargate como frontales que apuntarán a la base de datos, que montaran el volumen de EFS creado en la ruta en la que escucha por defecto nuestro apache en el contenedor subido en ECR y también a mayores nos creará un balanceador de carga que se encargará de balancear las peticiones entre nuestros frontales para no sobrecargar ninguno de ellos, este proceso es algo más lento que el resto dado que tiene que desplegar todo el sistema y puede tardar entre 5 y 10 minutos, una vez finalizado nos facilitará la url de nuestro balanceador de carga, la cual tendremos que modificar en la base de datos de Prestashop para que al hacer la consulta no nos redirecciones hacia el dominio, dado que el dominio todavía sigue apuntando al entorno anterior.

Verificación de funcionamiento

Tras configurar correctamente la tabla ps_shop_url con el valor de nuestro balanceador, podremos acceder a la dirección de nuestro balanceador y ya nos tendrá que responder nuestra web correctamente.

```
MySQL [prestashop]> select * from ps_shop_url;
+--------------+---------+-----------------------------------------------+-----------------------------------------------+--------------+-------------+------+--------+
| id_shop_url | id_shop | domain                                        | domain_ssl                                    | physical_uri | virtual_uri | main | active |
+--------------+---------+-----------------------------------------------+-----------------------------------------------+--------------+-------------+------+--------+
|           1 |       1 | balanceador113-95088977.eu-west-1.elb.amazonaws.com | balanceador113-95088977.eu-west-1.elb.amazonaws.com | /            |             |    1 |      1 |
+--------------+---------+-----------------------------------------------+-----------------------------------------------+--------------+-------------+------+--------+
1 row in set (0.00 sec)
```

Ilustración 62 - Verificación modificación de base de datos RDS con los datos del balanceador.

Ilustración 63- Verificación servicio funcionando correctamente en nueva infraestructura a través del balanceador de carga de AWS.

4.- Sistema de backups

El sistema de backups necesario para replicar nuestra web en caso de catástrofe solo sería necesario para los servicios de base de datos de RDS y para el volumen EFS.

Aunque por ejemplo RDS ya lo configuramos con sus propias copias de seguridad vamos a desplegar un nuevo sistema de backups externo al servicio de RDS. En este caso vamos a utilizar el servicio que nos proporciona AWS llamado AWS Backups. Para ello en primer lugar será necesario definir que es AWS Backups:

AWS Backups

Se trata de un servicio de AWS que nos permite configurar copias de seguridad de otros servicios que ya tenemos definidos en AWS y programar su realización, estas copias de seguridad se almacenan en un repositorio que nos proporciona el servicio por lo que también podemos configurar la retención en tiempo que queremos que estas copias se mantengan disponibles para poder restablecerlas en caso de pérdida de datos o de alguna acción incorrecta sobre los datos. Obviamente a mayor retención de las copias, mayor espacio se ocupará y por tanto mayor coste tendrá por lo que es necesario estudiar las necesidades de cada producto, en este caso actual al ser un ejemplo y no tener mucho peso el EFS o la RDS vamos a dejar la retención en 30 días inicialmente.

Creación de backups para EFS y RDS

En primer lugar, accedemos a la sección de AWS Backups, y en el menú del lateral izquierdo accedemos a la creación del plan de backups, donde iniciaremos la creación de un plan de backups para un servicio en concreto. Una vez iniciado el proceso nos dará tres opciones a utilizar: Comenzar con una plantilla, Crear un nuevo plan desde 0, Definir un nuevo plan por Json.

En este caso vamos a iniciar la creación de un nuevo plan desde 0 donde tendremos que indicar el nombre del Plan de backup, también será necesario crear una regla de configuración del backup donde definamos el nombre de la copia se seguridad, el almacén de la copia, la frecuencia de realización, la hora de realización, la retención y la zona de AWS donde queremos que se almacene. El panel nos permite crear un nuevo almacén con un cifrado por lo que lo hacemos para darle una mayor seguridad a nuestras copias y que solo puedan ser usadas por un usuario autorizado. Una vez definidos estos valores creamos nuestro plan.

En este caso vamos a crear un plan individual para los servicios de RDS y EFS por lo que una vez creado cada plan al acceder de nuevo a la sección vamos a ver nuestros planes, así como un resumen informativo de la creación y última ejecución de los mismos.

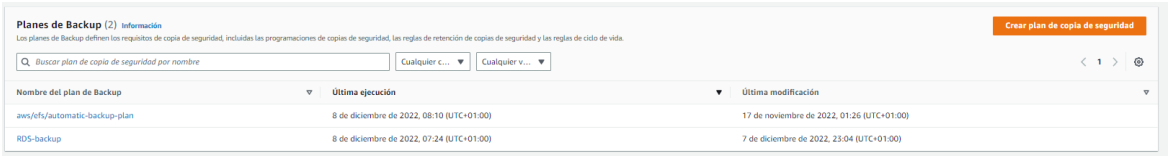


Ilustración 64 - Planes de backup

Pero estos planes ahora mismo no tienen información sobre de que tienen que hacer la copia, no dejan de ser planes explicando cómo, cuándo y por cuanto tiempo hacer y/o almacenar la copia, para indicarle de que recursos hacer copia, es necesario acceder a cada uno de los planes creados y definirlos. Una vez dentro del plan nos mostrará un resumen del plan de backups, así como las reglas definidas, los recursos y etiquetas o más recursos, solo es necesario acceder a la sección recursos y asignarle un recurso como se muestra en la siguiente captura.

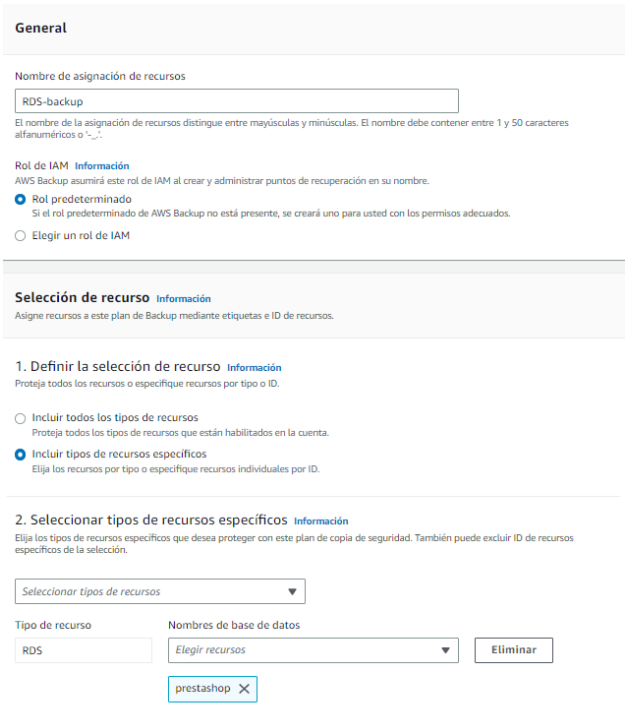


Ilustración 65- Configuración backups

De esta forma se le pone un nombre a la asignación de estos recursos, y se selecciona dentro del recurso de cual queremos realizar el backup, en el caso mostrado en la captura escogemos dentro de los servicios de RDS sobre la base de datos llamada Prestashop.

Una vez seleccionado los recursos, ya estarán programados los backups según el periodo y hora escogido y sobre los servicios escogidos también. Por lo ya se irán realizando y almacenando los backups según la configuración decidida.

Para poder confirmar la correcta realización, necesitaremos acceder a la sección de almacenes de copias de seguridad, donde se nos mostrarán las copias que tenemos configuradas, así como el número de copias que tenemos para cada una.

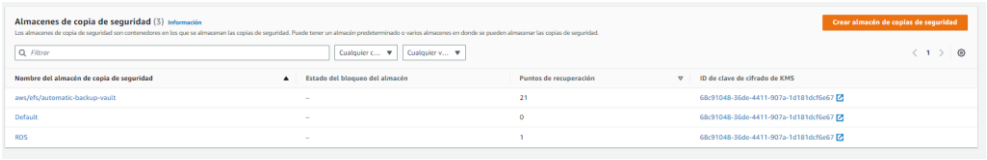


Ilustración 66 - Verificación Backup

Restauración de backups

Ya tenemos las copias de seguridad configuradas, ahora vamos a explicar cómo realizar la restauración, dado que uno de los motivos es recuperar la información en caso de una catástrofe, pero también es importante la rapidez y facilidad para recuperarlo, en este caso vamos a ver lo fácil que sería recuperar una copia de seguridad de nuestro sistema EFS.

Accedemos a la sección de copias de seguridad, accedemos el almacén del cual queremos restaurar alguna de las copias realizadas, y una vez dentro nos mostrará todas las copias de seguridad que tenemos almacenadas según la retención escogida durante la creación.

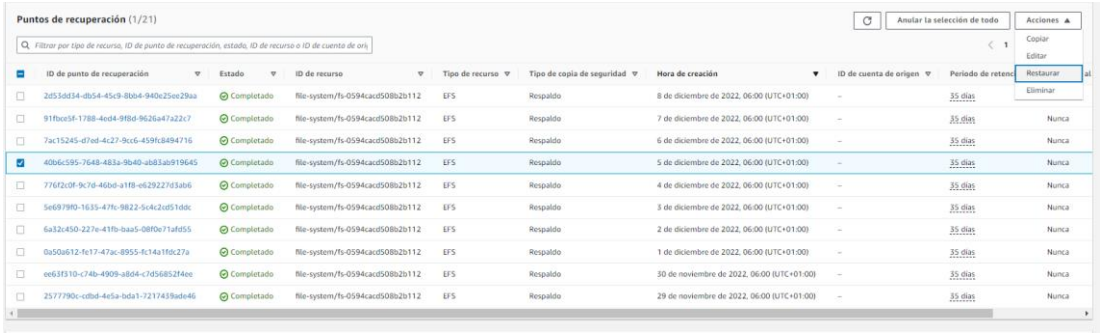


Ilustración 67- Restauración Backup

Esto nos abrirá un panel donde nos pregunta ciertas opciones sobre qué y cómo queremos restaurar la copia de seguridad, por ejemplo, en EFS nos pregunta si queremos la restauración completa o de algún directorio en concreto, y, si queremos restaurar en un nuevo servicio o en el servicio del cual se hizo el backup. Tras escoger las opciones deseadas el sistema de AWS backup ya se encarga de realizar todo y en unos minutos ya tendríamos nuestra copia de seguridad recuperada en nuestro servicio.

5.- Configuración de Cloudflare y configuración de dominio.

Que es Cloudflare, creación y configuración de dominio

Para explicar lo que es **Cloudflare**, primero es necesario explicar lo que es una CDN, una CDN se trata de una red de contenidos distribuida que copia la información de un servidor con la finalidad de evitar saturar el servidor final con peticiones, por ejemplo, si tenemos nuestro servidor en Irlanda, y no tenemos una CDN todas las peticiones tendrán que ir hasta Irlanda, lo cual si se trata de un usuario que se encuentra lejos de Irlanda, el tiempo de respuesta del servidor y por lo tanto de la web será bastante alto, y eso es algo que en una tienda online no se puede permitir.

Pues bien, Cloudflare se trata de un CDN que a mayores permite implementar servicios como certificados SSL/TLS, WAF, AntiDDos y múltiples opciones más según el plan que se desee escoger.

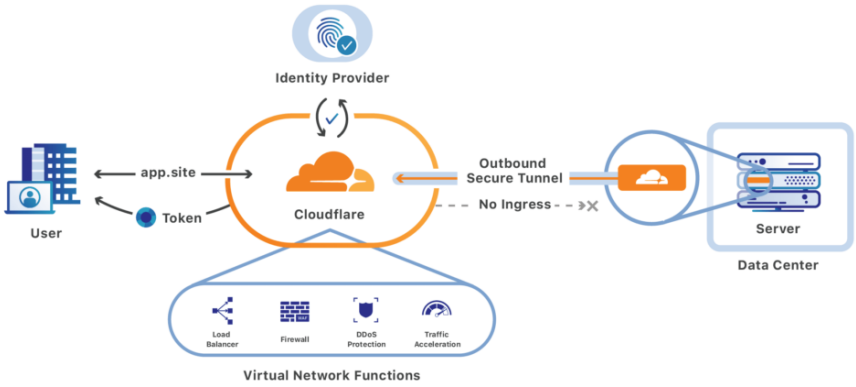


Ilustración 68 - Esquema funcionamiento Cloudflare

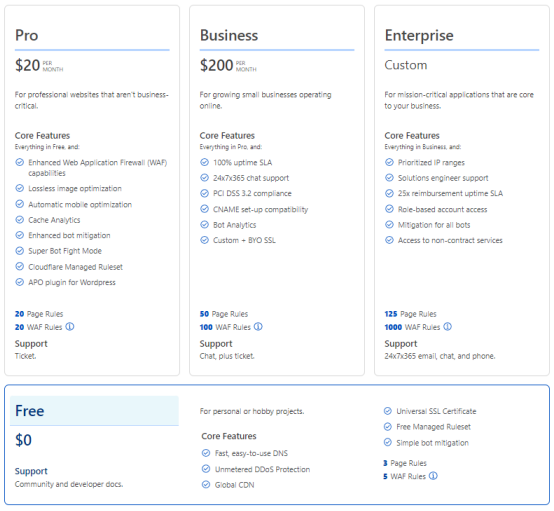


Ilustración 69 - Planes Cloudflare

Para este proyecto para evitar costes, vamos a escoger el plan Free, dado que solo queremos configurar los registros DNS de nuestro dominio a través de Cloudflare para que nuestra web

sea cacheada por la red distribuida de Cloudflare y a mayores que el dominio resuelva bajo un certificado SSL que también nos ofrece gratis.

Tras crear la cuenta es necesario vincular nuestro dominio, por lo que pulsaremos en *addsite* y pondremos el nombre de nuestro dominio.

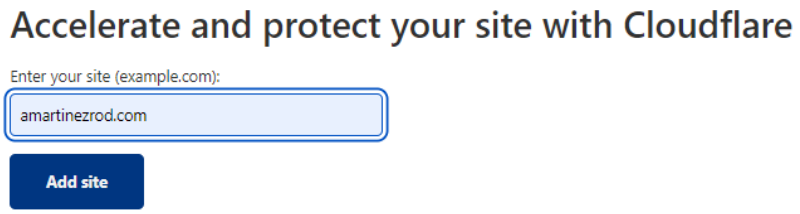
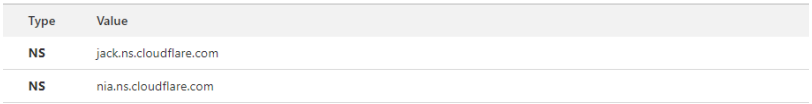


Ilustración 70- Agregación de dominio a Cloudflare

Ahora que ya tenemos el dominio, podemos acceder a Cloudflare y nos mostrará las opciones que nos permite realizar en este dominio, así como los servidores DNS que es necesario configurar en el dominio para que sea administrado por Cloudflare. Para ello copiamos los servidores DNS y los configuramos en nuestro dominio



Type	Value
NS	jack.ns.cloudflare.com
NS	nia.ns.cloudflare.com

Ilustración 71- Verificación de DNS

Realizando un *whois* al dominio podemos verificar que ya se han cambiado

```
Name Server: jack.ns.cloudflare.com
Name Server: nia.ns.cloudflare.com
```

Ilustración 72- Verificación DNS dominio

Ahora será necesario configurar en Cloudflare los registros de nuestras DNS hacia el balanceador de carga de nuestro ECS, en este caso iremos a la sección DNS de Cloudflare y añadiremos un registro de tipo Cname, desde el subdominio tienda hacia el endpoint facilitado del balanceador, este registro por el momento lo vamos a configurar en modo bypass, para que Cloudflare por el momento solo apunte el dominio al balanceador.

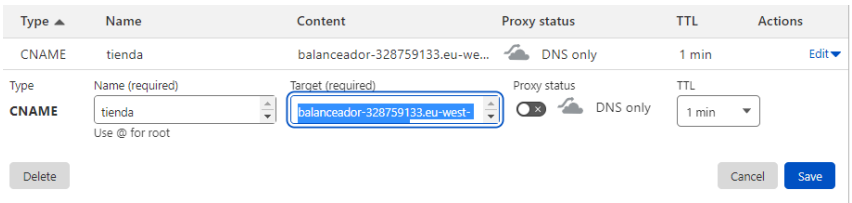


Ilustración 73- Creación Registro Cname

Si ahora realizamos un ping hacia nuestro subdominio, podemos observar que según el momento que sea nos responderá una ip u otra, es decir los servidores web de nuestro cluster que son apuntados por nuestro balanceador que tenemos configurado en los registros DNS.

```
--- PING balanceador-328759133.eu-west-1.elb.amazonaws.com (52.16.105.146) 56(84) bytes of data. ---
64 bytes from 52.16.105.146: icmp_seq=1 ttl=233 time=28.9 ms
64 bytes from 52.16.105.146: icmp_seq=2 ttl=233 time=28.8 ms
64 bytes from 52.16.105.146: icmp_seq=3 ttl=233 time=28.9 ms
64 bytes from 52.16.105.146: icmp_seq=4 ttl=233 time=28.9 ms

--- balanceador-328759133.eu-west-1.elb.amazonaws.com ping statistics ---

--- PING balanceador-328759133.eu-west-1.elb.amazonaws.com (54.76.141.97) 56(84) bytes of data. ---
64 bytes from 54.76.141.97: icmp_seq=1 ttl=233 time=29.5 ms
64 bytes from 54.76.141.97: icmp_seq=2 ttl=233 time=29.5 ms
64 bytes from 54.76.141.97: icmp_seq=3 ttl=233 time=29.5 ms
64 bytes from 54.76.141.97: icmp_seq=4 ttl=233 time=29.5 ms

--- balanceador-328759133.eu-west-1.elb.amazonaws.com ping statistics ---
```

Ilustración 74- Verificación respuesta ping

Si ponemos el dominio en un navegador ya podemos ver que nuestra web ya está resolviendo correctamente con el subdominio tienda.amartinezrod.com a través de Cloudflare.

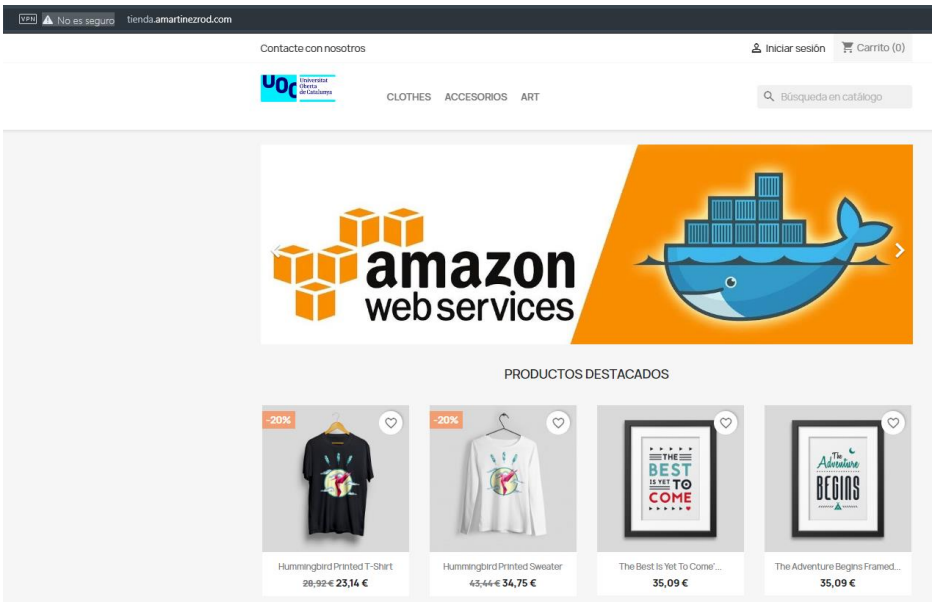


Ilustración 75- Resolución web tras balanceador

Configurar CDN y resolución con certificado SSL

Ahora ya que tenemos la resolución de nuestra web vamos a activar la opción de CDN para que las peticiones pasen a través de Cloudflare y este sea el que responda si tiene el contenido en la cache del mismo sin necesidad de hacer la consulta al servidor principal.

Para ello, volvemos a la configuración de los registros DNS y activamos el modo proxy

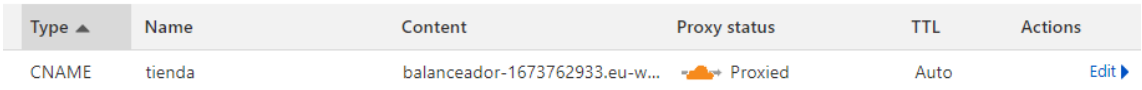


Ilustración 76- Activación proxy Cloudflare

Este modo tarda algo de tiempo en responder mientras Cloudflare se encarga de obtener la información del servidor web y empieza a cachearlo, una vez que finalice si realizamos por ejemplo el comando dig para saber la ip que está respondiendo detrás del subdominio tienda.amartinezrod.com podremos observar que está resolviendo unas ips de Cloudflare.

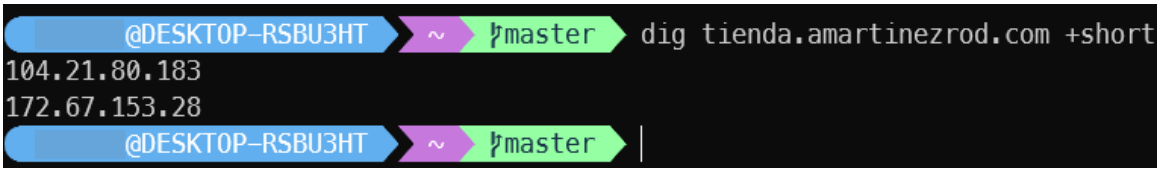


Ilustración 77- Resolución DNS del dominio

Esto también nos da también algo de protección a nuestro servidor web dado que si una persona intenta saber dónde está alojado nuestro servidor no será tan fácil como revisar la ip que responde, dado que esta ip estará registrada a nombre de Cloudflare y no se tratará de nuestro servidor.

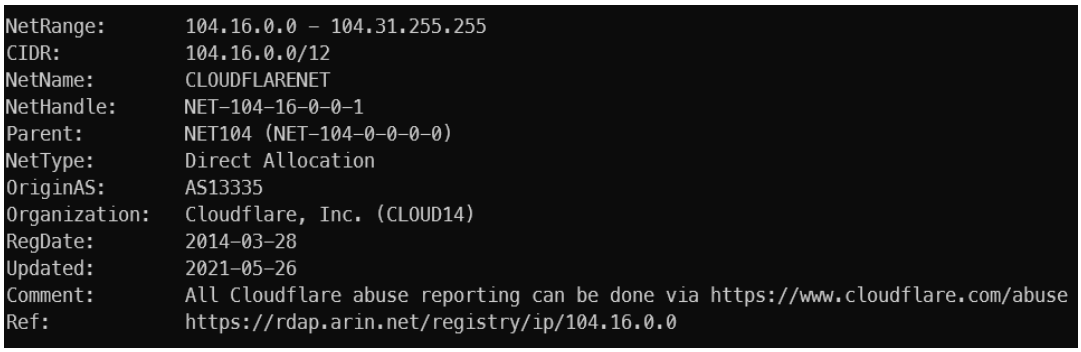


Ilustración 78- Verificación IP de Cloudflare

Si quisiéramos obtener mayor protección en nuestro servidor, también podremos activar los servicios WAF o AntiDDos para evitar acciones maliciosa o intentos de denegación de servicio contra nuestro servidor web en el apartado de seguridad, pero en este proyecto al no tener un trasfondo comercial final no vamos a realizar esta configuración para no extenderlo más de lo necesario.

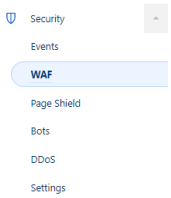


Ilustración 79- Opciones proporcionadas por Cloudflare

A continuación, nos faltaría activar y crear los certificados SSL/TLS para que la resolución de la web funcione a través del protocolo HTTPS y por tanto cifre la conexión entre el servidor y el cliente, para garantizar que los datos enviados o introducidos por un cliente en nuestro e-commerce no puedan ser escuchados fácilmente por un tercero. Existen varios tipos de certificados, para un e-commerce el más valioso es el certificado de validación de identidad, que a mayores de cifrar la conexión cliente-servidor también verifica la entidad que se encuentra detrás de esa web, por lo que si tuviésemos una marca comercial daría una mayor seguridad al saber que la conexión es cifrada y a mayores confirmar que la web pertenece a esa empresa.

Pero en nuestro caso para evitar costes durante el proyecto vamos a utilizar el certificado gratuito Let´s encrypt que nos proporciona el propio Cloudflare, para ello accedemos a la sección SSL/TLS, y vamos a indicarle que el cifrado sea flexible, es decir que cifre la conexión del navegador al CDN, existen otras opciones, pero ya sería necesario crear certificados en el ECS y para agilizar la resolución del ejemplo vamos a activar el flexible.

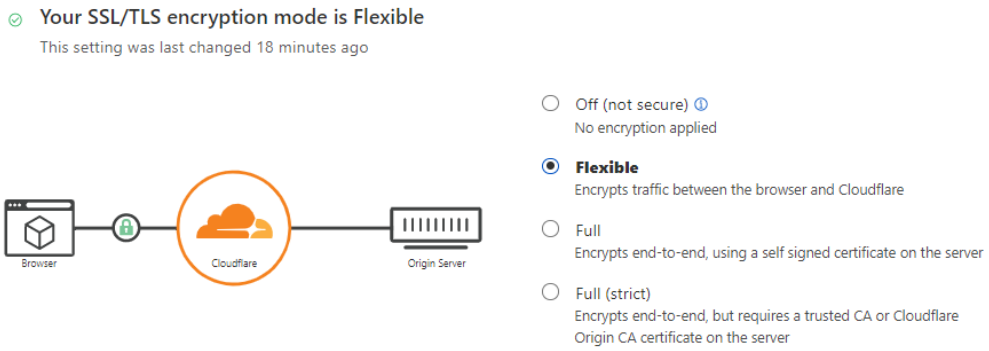


Ilustración 80- Activación TLS Cloudflare

En el menú lateral derecho, vamos también a las secciones de *client certificates* y *Origin server* y creamos los certificados gratuitos, dándole a crear certificado. Una vez finalizada la solicitud del certificado a la entidad certificadora, ya podremos acceder a nuestra web y ya nos responderá correctamente por https con nuestro certificado Let's encrypt.

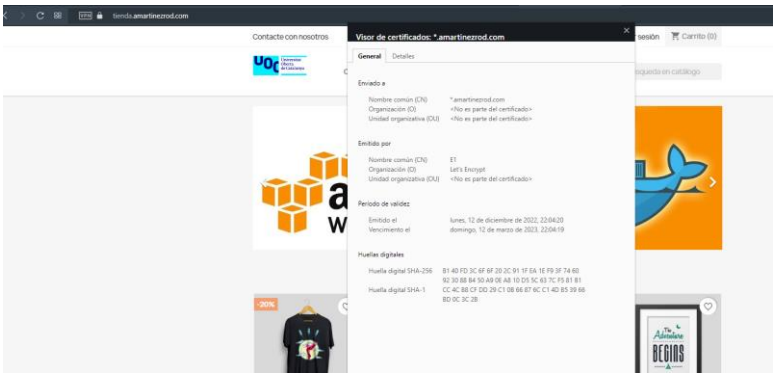


Ilustración 81- Verificación resolución https

6.- Prometheus y Grafana

Prometheus

Prometheus se trata de un servicio de recopilación de métricas de servicios con la finalidad de realizar la monitorización de los mismos, Prometheus también permite generar alertas en base a las métricas obtenidas para alertar a los propietarios o administradores de sistemas de que algo está sobrecargado o funcionando incorrectamente en los servicios monitorizados.

Prometheus consta de un servidor central que recopila las métricas, y de exportadores que se tienen que configurar en cada uno de los servidores, nodos o instancias que utilizamos en nuestra infraestructura para que pueda recopilar todos estos datos de los mismos y enviarlos al servidor central, a continuación, se muestra un ejemplo de la arquitectura de uso de Prometheus.

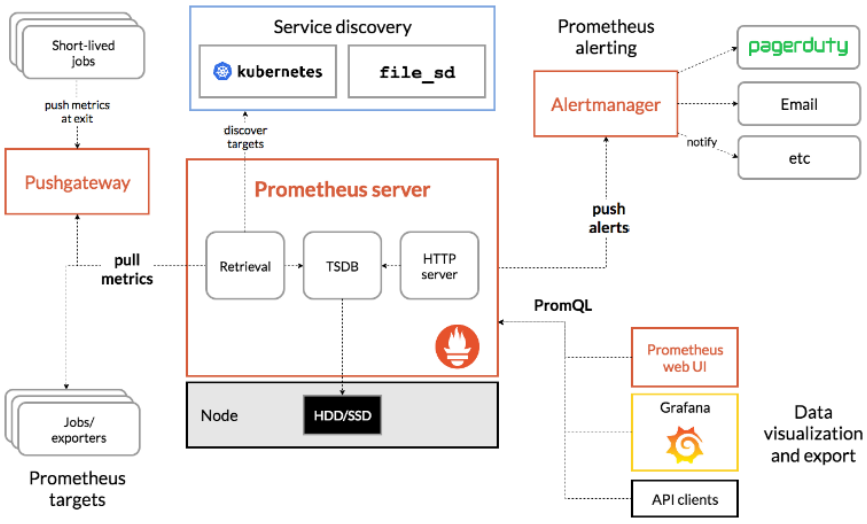


Ilustración 82- Esquema funcionamiento prometheus

En nuestro caso vamos a desplegar una instancia EC2 donde vamos a instalar un contenedor de docker de Prometheus y que será donde tendremos el servicio central de Prometheus, después instalaremos los exporters en cada uno de los servicios que queremos monitorizar y configuraremos nuestro Prometheus para obtener esas métricas y así poder monitorizarlas.

Para ello, vamos a desplegar de nuevo una instancia EC2 en la capa gratuita, y tras realizar las actualizaciones del sistema pertinentes, vamos a realizar la instalación del servicio de docker como se ha realizado ya anteriormente. Vamos a descargar la imagen del contenedor de Prometheus oficial y vamos a correrla en nuestra instancia EC2.

```
docker run -d --name=prometheus -p 9090:9090 -v
    /etc/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
    prom/prometheus
```

Ilustración 83- Instalación docker Prometheus

Realizamos el comando docker run para correr la imagen prom/Prometheus, definimos el nombre del Container como Prometheus, el puerto de escucha el 9090 y el volumen de datos en /etc/Prometheus/Prometheus.yml, este será el fichero de configuración donde indicaremos los servicios a monitorizar y como poder llegar a ellos.

Con esto ya tendríamos el servidor de Prometheus inicializado, vamos ahora a configurar los exporters para cada uno de los servicios.

Exporter EFS

En primer lugar, vamos a configurar el sistema de exportación de datos en el volumen de archivos EFS, esto podríamos configurarlo en la máquina EC2 dado que ya teníamos el volumen montado, pero para simplificarlo vamos a montar de nuevo el volumen en esta nueva máquina EC2 y realizaremos la instalación y configuración del exportador node_exporter para esta máquina ec2.

Tenemos dos opciones o descargarnos el exportador del repositorio de GitHub, o montarnos un nuevo contenedor con el exportador, como también vamos a querer obtener las métricas en nuestro servicio ECS vamos a realizar la instalación y configuración manual del exporter para nuestro servicio de EFS.

Procedemos a la descarga del repositorio de GitHub.

```
wget
    https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_e
    xporter-0.18.1.linux-amd64.tar.gz
```

Ilustración 84- Comando descarga node_exporter

Descomprimimos el fichero comprimido, y movemos los ficheros a la carpeta donde se alojan los binarios de nuestro servidor /usr/local/bin

```
tar xvfz node_exporter-*.*-amd64.tar.gz
sudo mv node_exporter-*.*-amd64/node_exporter /usr/local/bin/
```

Ilustración 85- Descompresión y creación de binario

A continuación, creamos un usuario para poder correr el servicio node_exporter.

```
sudo useradd -rs /bin/false node_exporter
```

Ilustración 86- Creación de usuario en sistema EC2

El siguiente paso será la creación del servicio node_exporter, para ello crearemos el fichero node_exporter.service con el siguiente contenido

```
sudo nano /etc/systemd/system/node_exporter.service

[Unit]
Description=Node Exporter
After=network.target
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
```

Ilustración 87- Configuración node_exporter

Y ya por último reiniciaremos los demonios configuraremos el servicio de node_exporter para que se arranque por defecto con el arranque del sistema y lo arrancaremos.

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
```

Ilustración 88- Reinicio, activación y fijación del demonio del servicio.

Por lo que, tras realizar esta configuración si accedemos a la ip pública de nuestra máquina EC2 con el puerto 9100/metrics podremos ver las métricas que está recogiendo nuestro sistema exportador.

Exporter ECS

En este caso vamos a utilizar la imagen de docker oficial de node_exporter, por lo que tendremos que editar la tarea ECS que teníamos creada anteriormente y añadir en ella un nuevo contenedor, por lo que accedemos a nuestro panel ECS y vamos a la sección de tareas,

seleccionamos la tarea creada anteriormente para nuestro Prestashop y ejecutamos una nueva revisión.

En ella vamos a crear un nuevo contenedor a mayores del de Prestashop ya creado donde vamos a utilizar el contenedor de docker hub oficial de node_exporter, indicando el puerto de escucha del node_exporter también en el puerto 9100.

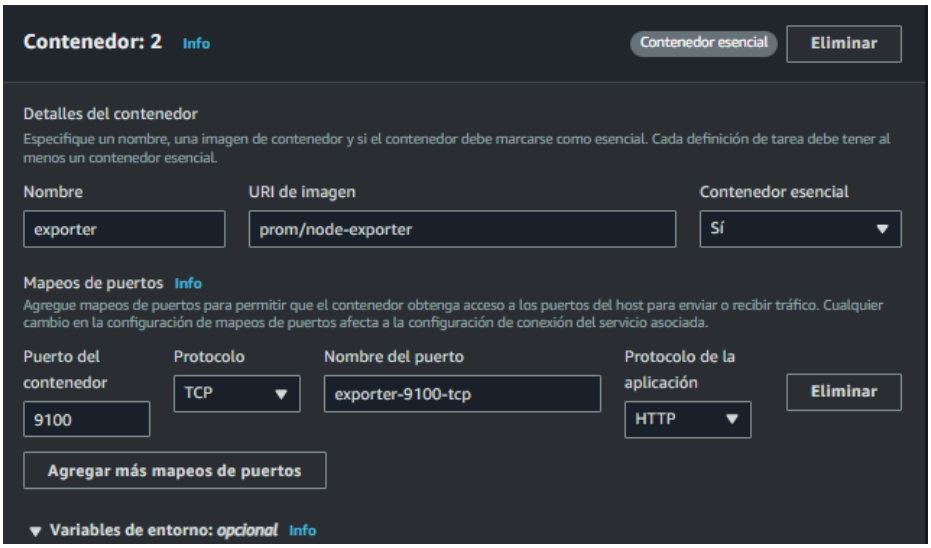


Ilustración 89- Creación nuevo contenedor en ECS

Por lo que si volvemos a ejecutar el servicio ECS con las 2 tareas para que nos cree las 2 máquinas con los dos contenedores, ya podremos acceder del mismo modo a la ip: puerto/metrics para ver las métricas que está recogiendo el node_exporter. De esta forma podremos monitorizar los 2 servicios levantados por nuestro cluster ECS, y en caso de que obtuviéramos algún error en alguno de ellos podríamos saber cuál sería necesario revisar para obtener la información del porqué de esos errores.

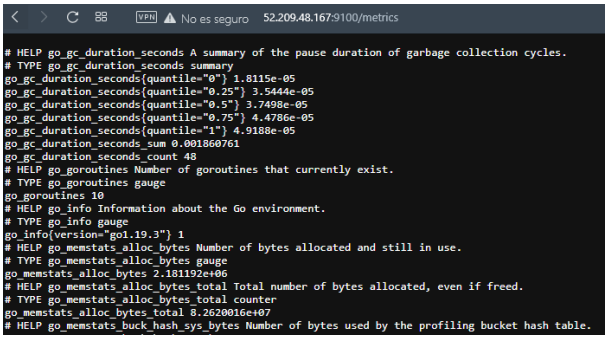


Ilustración 90- Métricas obtenidas node_exporter

Exporter RDS

Para el servicio de RDS, Amazon no permite acceso por SSH actualmente, la única opción que da para poder sincronizar los datos es a través de los servicios de CloudWatch, pero eso se sale del plan gratuito, es por lo que no vamos a configurarlo, simplemente vamos a revisar las métricas a través de Amazon CloudWatch. Para ello accedemos desde nuestro panel de AWS a los servicios de RDS, accederemos a nuestra instancia de base de datos Prestashop y en ella podremos ver las métricas en la sección de Supervisión.

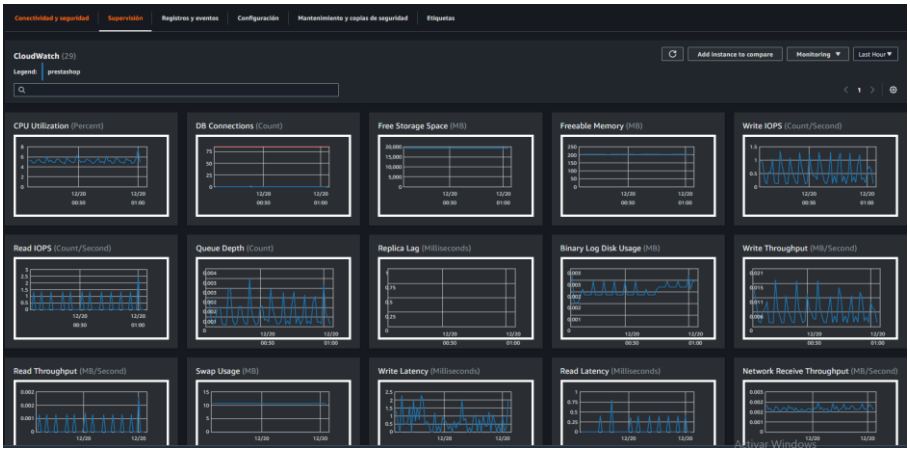


Ilustración 91- Métricas CloudWatch de RDS

Configuración Prometheus.

Ahora que ya tenemos los exporters del ECS y del EFS operativos, es momento de configurar el servidor de Prometheus para que coja las métricas que el node_exporter se encarga de recopilar de los diferentes servicios. Para ello dado que hemos configurado el volumen de Prometheus dentro de /etc/Prometheus/Prometheus.yml, vamos a acceder a esa ruta e indicarle al servidor de Prometheus donde se encuentran las métricas y de qué servicio son.

Necesitamos por tanto editar el fichero yml con las ips de los servicios y los puertos configurados en el node_exporter para que Prometheus se pueda conectar con los exportadores y obtener las métricas. Lo haríamos de la siguiente forma:

```
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'ecs1'
    static_configs:
      - targets: ['52.209.48.167:9100'] # dirección IP de ECS task1.
  - job_name: 'ecs2'
    static_configs:
      - targets: ['54.229.155.158:9100'] # dirección IP de ECS task2.
  - job_name: 'efs'
    static_configs:
      - targets: ['34.246.183.153:9100'] # dirección IP de EFS
```

Ilustración 92- Configuración Prometheus

Definiríamos el nombre del job con el nombre a mostrar del servicio y después el target con la ip y el puerto.

Tras finalizar esta configuración ya podremos acceder a la interfaz gráfica de Prometheus para verificar que ya reconoce estos servicios y que tiene conectividad con ellos, por lo que es necesario poner la ip de la instancia EC2 con el puerto definido en el contenedor de docker en nuestro navegador y ya podremos acceder a la interfaz web de Prometheus.

En nuestro caso si accedemos a la url : http://34.246.183.153:9090/ y ya podremos acceder a la interfaz web de Prometheus.

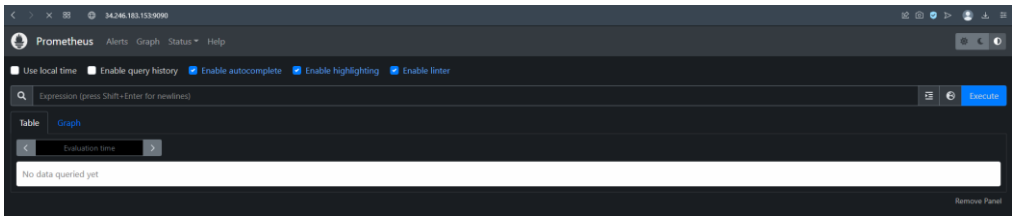


Ilustración 93- Interfaz web Prometheus

Una vez dentro, podremos configurar alertas, etc… pero en nuestro caso para no alargar más el proyecto de lo necesario, simplemente vamos a acceder a la opción de status>targets para verificar que ya tenemos acceso correcto a las métricas de nuestros servicios las cuales vamos a enviar a nuestro servicio de Grafana para graficar.

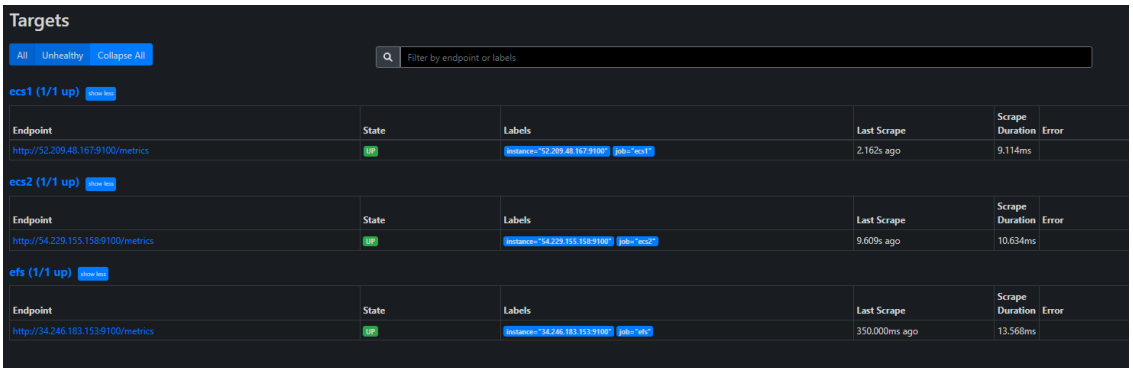


Ilustración 94- Targets operativos de Prometheus

Al acceder a esta sección comprobamos que el servidor de Prometheus ya tiene las métricas de los servicios definidos por lo que damos por concluida la configuración de Prometheus y ahora vamos a configurar nuestro Grafana para graficar estos datos y que se puedan observar de manera más simple.

Grafana

Grafana es un software open source que realiza las gráficas de métricas obtenidas, en este caso a través de Prometheus. Este sistema permite no solo generar las gráficas con un grado de personalización casi completo, sino que también nos proporciona ciertos dashboards ya predefinidos los cuales podremos utilizar para generar el panel de gráficos de nuestros recursos.

Grafana se puede instalar desde su repositorio Github y también tiene una imagen de docker que podemos montar un contenedor, en este caso lo que vamos a realizar es montar la imagen de docker en un contenedor en nuestra nueva instancia EC2 para que esa nueva instancia aloje tanto el servidor de Prometheus como de Grafana, para ello vamos a ejecutar el siguiente comando.

```
docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

Ilustración 95- Instalación contenedor Grafana

De esta forma corremos la imagen oficial de docker de Grafana bajo el puerto 3000 de nuestra máquina y solo con esto ya finalizaríamos la instalación esencial, se podría también indicarle la instalación de plugins o similares, pero en nuestro caso vamos a dejar la instalación por defecto y acceder a la interfaz web para realizar las acciones necesarias.

Para ello, vamos a acceder a la siguiente url http://34.246.183.153:3000/login donde, nos pedirá un usuario y una contraseña, Grafana por defecto tiene como usuario admin y contraseña admin, por lo que una vez pongamos esos datos nos preguntará si queremos definir una nueva contraseña para el usuario admin.

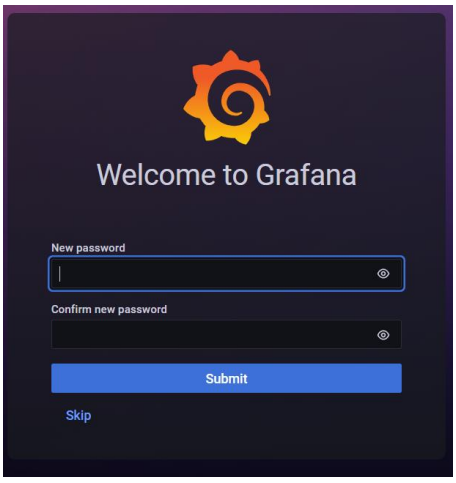


Ilustración 96- Acceso Interfaz web Grafana

Tras la configuración de la nueva contraseña accederemos a la interfaz web de nuestro Grafana donde tendremos que indicarle de donde solicitar las métricas para graficar, así como crear nuestro panel.

Configuración Data Source

Para configurar el origen de los datos como Prometheus es necesario tener instalado el plugin de Prometheus en primer lugar, en nuestra imagen oficial el plugin ya está instalado por defecto, pero igualmente podemos realizar la instalación de cualquiera de los plugins gratuitos que Grafana facilita.

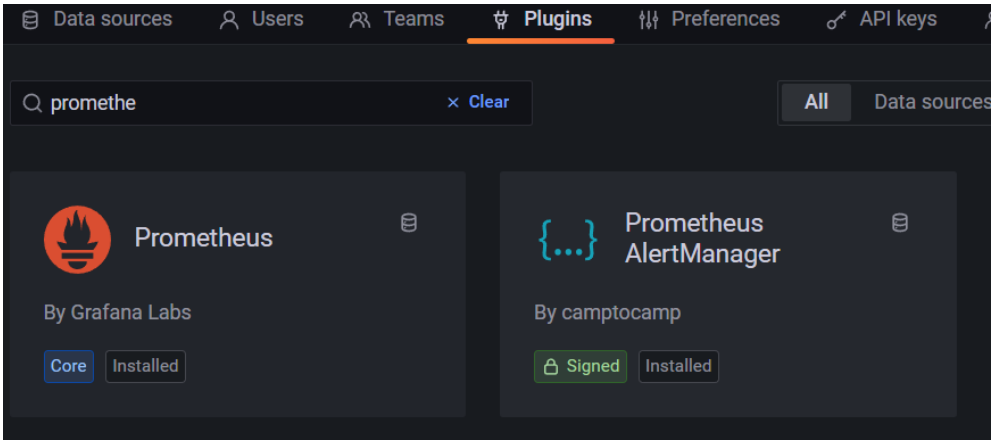


Ilustración 97- Instalación plugin Prometheus.

A continuación, será necesario definir el Data Source, para que Grafana obtenga las métricas almacenadas en el servidor de Prometheus y podamos crear la configuración, para ello dentro de la sección de configuración acedemos al apartado Data sources, y ahí seleccionamos Prometheus y ya nos saldrá un formulario a completar para poder conectar nuestro sistema Grafana con nuestro Prometheus, definiremos la ruta donde se encuentra nuestro Prometheus y ya estaría, como no tenemos configurado usuario/contraseña para el acceso a nuestro Prometheus podemos dejarlo así y después ya limitáremos el acceso en los security groups de AWS.

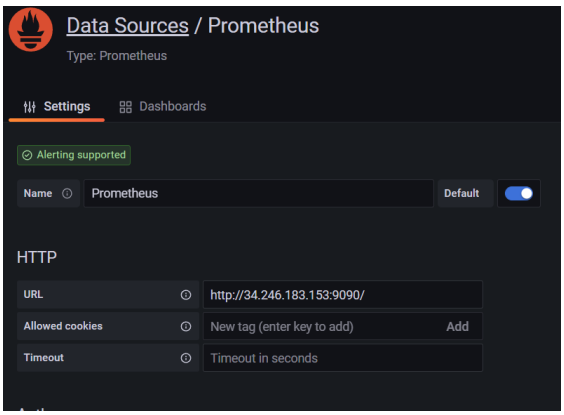


Ilustración 98 - Configuración Data Source

Creación dashboards

Los paneles que genera Grafana se llaman dashboards, que es donde se mostraran las gráficas que nosotros creemos de nuestros servicios, para ello accederemos en el menú lateral a la sección de dashboards y crearemos uno nuevo, al realizar esta acción nos abrirá una ventana donde nos mostrará en primer lugar, en la parte superior, una visualización de la gráfica configurada, en la parte inferior los datos que queremos utilizar para realizar la gráfica y en lateral derecho nos mostrará las opciones administrativas y de configuración visual de la gráfica.

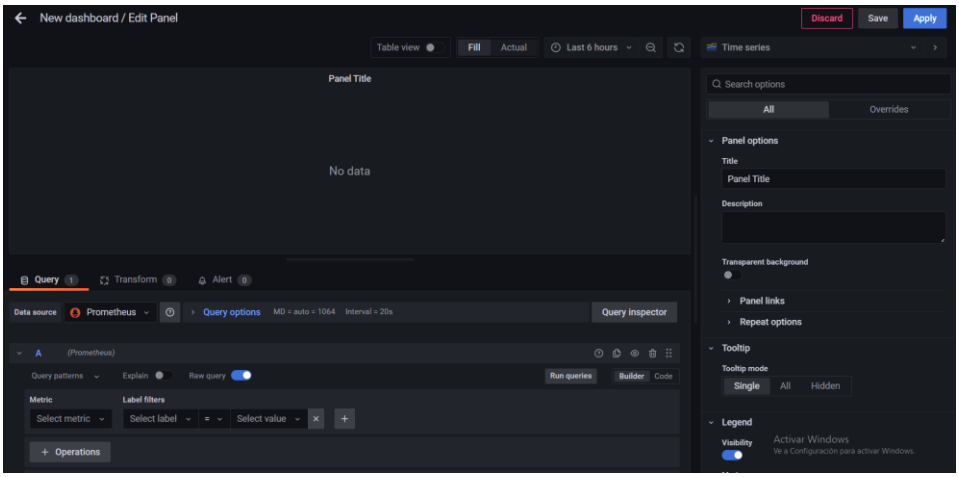


Ilustración 99- Creación dashboard

De esta forma, podremos seleccionar las métricas que necesitemos de cualquiera de los Jobs de métricas que tenemos configurado en nuestro Prometheus para que Grafana se encargue de graficarlo y mostrarlo.

Solo será necesario seleccionar la métrica y el job y una vez lo tengamos ya podremos utilizar la sección de *run query* para que nos muestre una pre visualización de la gráfica obtenida con los datos proporcionados.

Ilustración 100 - Creación gráfica en Grafana

Configuraremos un nombre para esa gráfica, y le daremos a aplicar, esto nos mostrará ya la gráfica en nuestro panel de los datos obtenidos con la configuración seleccionada y ya podremos darle a guardar al panel. Nos pedirá un nombre para nuestro dashboard.

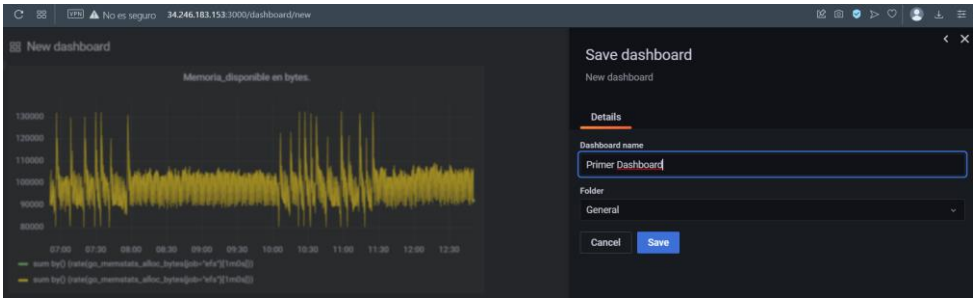


Ilustración 101 - Vista de primera grafica en el dashboard

Ahora que ya tenemos nuestra gráfica y nuestro dashboard guardados podemos acceder a él cuándo sea necesario para revisar nuestras gráficas, para acceder desde el menú principal solo es necesario seleccionar el dashboard al que queremos acceder y ya podremos acceder a él para administrarlo o revisarlo.

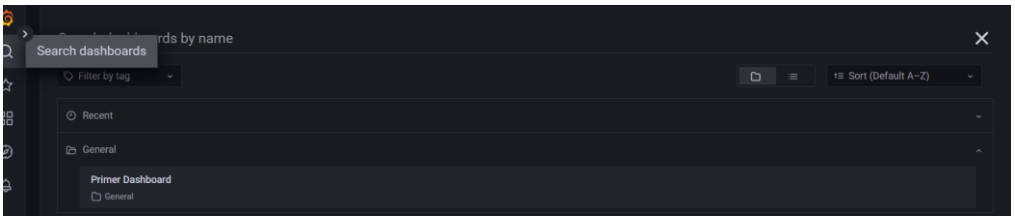


Ilustración 102 - Acceso a Dashboard

Estos gráficos se pueden modificar de colores, esquemas, tamaños, métricas, etc… y crear el panel al gusto del usuario final o del administrador que lo gestione, para crear un pequeño ejemplo vamos a crear unas cuantas gráficas más y así ver como podría ser un ejemplo de un *dashboard* final, en el siguiente ejemplo podemos observar como podría ser una creación de un panel personalizado.

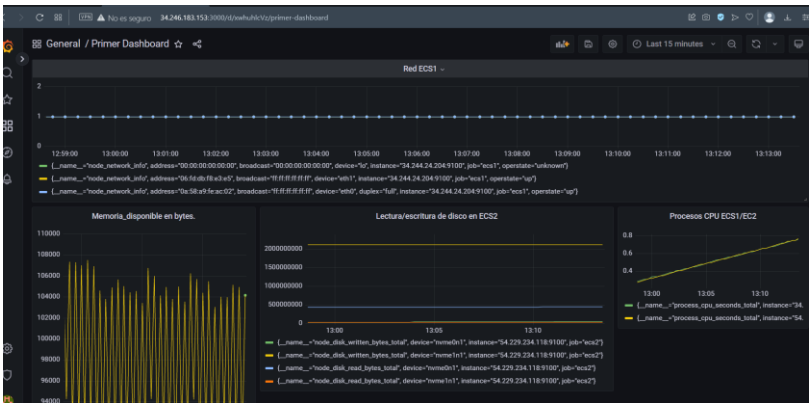


Ilustración 103 - Dashboard inicial con varias gráficas.

Si bien la personalización de las gráficas que ofrece Grafana es muy amplia, también significa que la inversión de tiempo también será muy elevada, para poder obtener los gráficos exactos que queramos, es por eso que Grafana también facilita unos dashboards pre configurados que nos facilitará la creación y adaptación de nuestro dashboard a nuestras necesidades.

Importación dashboards

Para importar un dashboard ya pre configurado, solo es necesario acceder a la página web de Grafana y en la sección dashboard ya existen múltiples dashboards según los data source y según lo que queramos graficar, en este caso vamos a escoger un dashboard de node_Exporter donde solo necesitaremos copiarnos el ID del dashboard e importarlo en nuestro Grafana. Para ello accedemos en el menú izquierdo a la sección de dashboard y le daremos a importar, solo será necesario pegar el ID que teníamos copiado y darle a cargar.

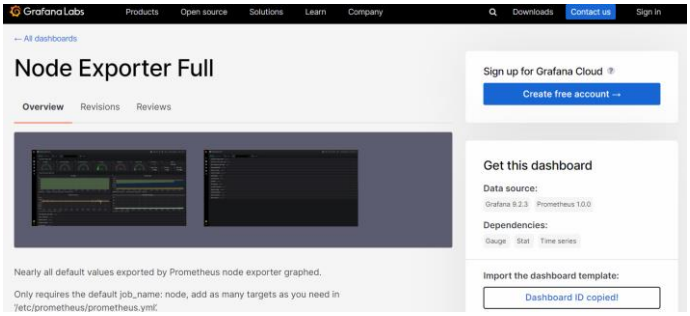


Ilustración 104 - Obtención plantilla de ejemplo dashboard

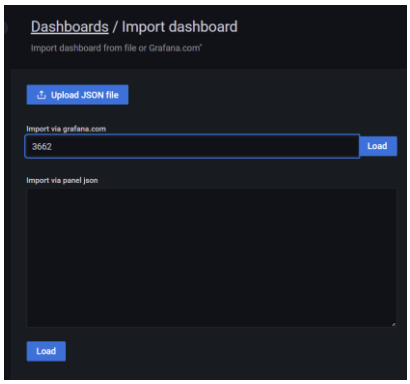


Ilustración 105- Importación plantilla.

Tras realizar esto ya podremos configurar el nombre de nuestro dashboard y acceder a él para ver si todas las métricas por defecto configuradas se están mostrando correctamente.

Tras esto, ya tendremos nuestro panel de Grafana configurado con todas las métricas que en este caso que nos facilita el node_Exporter de cada uno de nuestros servicios desglosadas y ordenadas por servicio.

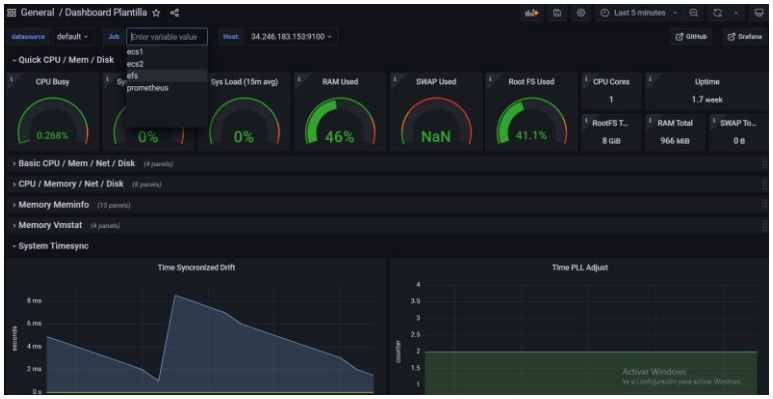


Ilustración 106- Visualización dashboard plantilla.

7.- Resumen y valoración del proyecto.

Ya para finalizar vamos a mostrar el esquema final de nuestro proyecto, donde vamos a cerrar los security groups y la VPC para que solo se pueda acceder por el puerto 80 y el 443 para la resolución web en nuestro cluster ECS, por el puerto 22 a la máquina de salto y por el puerto 3000 al Grafana desde una IP especifica externa, de esta forma aislaremos nuestro proyecto para darle una mayor seguridad.

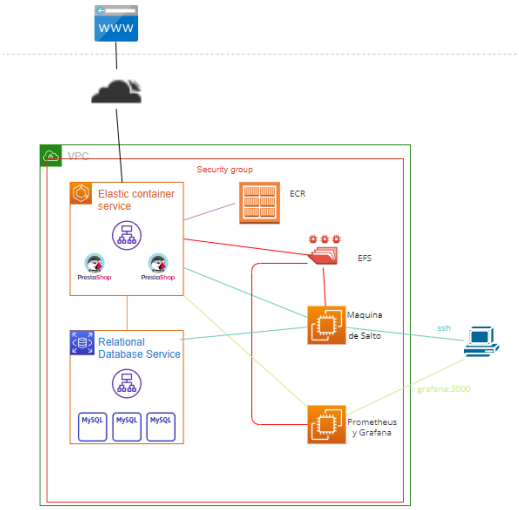


Ilustración 107- Arquitectura final.

El proyecto me ha resultado muy satisfactorio dado he trabajado con tecnologías o servicios muy demandados actualmente como son Docker, AWS, Prometheus, Grafana, Clusters, etc… y he comprendido su funcionamiento en mayor o menor medida logrando no solo explicarlo en esta memoria sino realizarlo y testearlo bajo un dominio propio. Si bien me he encontrado más dificultades de las esperadas para tratar de establecer el proyecto bajo la capa gratuita aun así he tenido que desembolsar unos pocos euros por errores o simplemente por sobredimensionar ciertas pruebas realizadas, considero que he aprendido mucho sobre dichos servicios durante todo este proceso y me siento satisfecho por lo aprendido y por lo realizado.

Bibliografía

¿Qué es Amazon EC2? *docs.aws.amazon.com.* [En línea] [Citado el: 29 de 10 de 2022.] https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html.
¿Qué es Amazon Elastic Container Registry? *docs.aws.amazon.com.* [En línea] [Citado el: 05 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonECR/latest/userguide/what-is-ecr.html.
¿Qué es Amazon Elastic Container Service? *docs.aws.amazon.com.* [En línea] [Citado el: 04 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonECS/latest/developerguide/Welcome.html.
¿Qué es Amazon Elastic File System? *docs.aws.amazon.com.* [En línea] [Citado el: 06 de 11 de 2022.] https://docs.aws.amazon.com/es_es/efs/latest/ug/whatisefs.html.
¿Qué es Amazon Relational Database Service (Amazon RDS)? *docs.aws.amazon.com.* [En línea] [Citado el: 08 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/Welcome.html.
¿Qué es Amazon VPC? *docs.aws.amazon.com.* [En línea] [Citado el: 30 de 10 de 2022.] https://docs.aws.amazon.com/es_es/vpc/latest/userguide/what-is-amazon-vpc.html.
2022. Amazon RDS Backup & Restore Using AWS Backup. *aws.amazon.com.* [En línea] 07 de 12 de 2022. https://aws.amazon.com/es/getting-started/hands-on/amazon-rds-backup-restore-using-aws-backup/.
AWS | Almacenamiento de datos seguros en la nube (S3). *aws.amazon.com.* [En línea] [Citado el: 15 de 11 de 2022.] https://aws.amazon.com/es/s3/.
2022. AWS Backup administra y automatiza de forma centralizada las copias de seguridad en los servicios de AWS. *aws.amazon.com.* [En línea] 17 de 11 de 2022. https://aws.amazon.com/es/backup/.
Conexión a una instancia de base de datos de Amazon RDS. *docs.aws.amazon.com.* [En línea] [Citado el: 08 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/CHAP_CommonTasks.Connect.html.
Controlar el tráfico hacia los recursos mediante grupos de seguridad. *docs.aws.amazon.com.* [En línea] [Citado el: 30 de 10 de 2022.] https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_SecurityGroups.html.
Creación de una instancia Amazon EC2 y copia de la base de datos comprimida. *docs.aws.amazon.com.* [En línea] [Citado el: 09 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/MySQL.Procedural.Importing.NonRDSRepl.html#MySQL.Procedural.Importing.Import.Database.
Creación de una instancia de base de datos de Amazon RDS. *docs.aws.amazon.com.* [En línea] [Citado el: 08 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html.
2022. Dashboards. *grafana.com.* [En línea] 20 de 12 de 2022. https://grafana.com/grafana/dashboards/.
Docker Documentation | Docker Documentation. *docs.docker.com.* [En línea] [Citado el: 17 de 10 de 2022.] https://docs.docker.com/.
2022. DockerHub - node_exporter. *hub.docker.com.* [En línea] 17 de 12 de 2022. https://hub.docker.com/r/prom/node-exporter.
2022. DockerHub - Prometheus. *hub.docker.com.* [En línea] 17 de 12 de 2022. https://hub.docker.com/r/prom/prometheus.
2022. Entonces, ¿qué es Cloudflare? *www.cloudflare.com.* [En línea] 12 de 12 de 2022. https://www.cloudflare.com/es-es/learning/what-is-cloudflare/.

Evolución del porcentaje de compras y ventas de comercio electrónico sobre el total de compras y ventas realizadas en España de 2008 a 2020. *ecommercerentable.es.* [En línea] [Citado el: 13 de Octubre de 2022.] https://ecommercerentable.es/ecommerce-espana-2021/.
Instalación de Docker en Amazon Linux 2. *docs.aws.amazon.com.* [En línea] [Citado el: 29 de 10 de 2022.] https://docs.aws.amazon.com/es_es/AmazonECS/latest/developerguide/create-container-image.html#:~:text=CLI%20de%20Docker.-,Instalaci%C3%B3n%20de%20Docker%20en%20Amazon%20Linux%202,-Creaci%C3%B3n%20de%20una.
Instalación o actualización de la versión más reciente de AWS CLI. *docs.aws.amazon.com.* [En línea] [Citado el: 29 de 10 de 2022.] https://docs.aws.amazon.com/es_es/cli/latest/userguide/getting-started-install.html.
Instalar el cliente de Amazon EFS. *docs.aws.amazon.com.* [En línea] [Citado el: 06 de 11 de 2022.] https://docs.aws.amazon.com/es_es/efs/latest/ug/installing-amazon-efs-utils.html.
2022. Install Prometheus Node Exporter on Linux [2 Steps]. *fosstechnix.com.* [En línea] 17 de 12 de 2022. https://www.fosstechnix.com/install-prometheus-node-exporter-on-linux/ instalación node_exporter.
2022. Let's encrypt. *letsencrypt.org.* [En línea] 12 de 12 de 2022. https://letsencrypt.org/es/.
Montaje de sistemas de archivos EFS. *docs.aws.amazon.com.* [En línea] [Citado el: 06 de 11 de 2022.] https://docs.aws.amazon.com/es_es/efs/latest/ug/mounting-fs.html.
Mysql - Official Image | Docker Hub. *hub.docker.com/_/mysql.* [En línea] [Citado el: 22 de 10 de 2022.] https://hub.docker.com/_/mysql.
MySQL :: MySQL 5.7 Reference Manual :: 4.9 Environment Variables. *dev.mysql.com.* [En línea] [Citado el: 12 de 11 de 2022.] https://dev.mysql.com/doc/refman/5.7/en/environment-variables.html.
Networking overview | Docker Documentation. *docs.docker.com/network/.* [En línea] [Citado el: 17 de 10 de 2022.] https://docs.docker.com/network/.
Paso 3: Monte el sistema de archivos en la instancia EC2 y pruebe. *docs.aws.amazon.com.* [En línea] [Citado el: 06 de 11 de 2022.] https://docs.aws.amazon.com/es_es/efs/latest/ug/wt1-test.html.
prestashop / prestashop - Docker Image | Docker hub. *hub.docker.com/r/prestashop/prestashop.* [En línea] [Citado el: 22 de 10 de 2022.] https://hub.docker.com/r/prestashop/prestashop.
2022. Red de distribución de contenidos. *es.wikipedia.org.* [En línea] 12 de 12 de 2022. https://es.wikipedia.org/wiki/Red_de_distribuci%C3%B3n_de_contenidos.
2022. Run Grafana Docker image. *grafana.com.* [En línea] 19 de 12 de 2022. https://grafana.com/docs/grafana/v8.5/installation/docker/.
2022. Trabajo con copias de seguridad. *docs.aws.amazon.com.* [En línea] 07 de 12 de 2022. https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/USER_WorkingWithAutomatedBackups.html.
Transferencia de archivos mediante un cliente. *docs.aws.amazon.com.* [En línea] [Citado el: 29 de 10 de 2022.] https://docs.aws.amazon.com/es_es/transfer/latest/userguide/transfer-file.html.
Tutorial: Utilización de sistemas de archivos de Amazon EFS en Amazon ECS con la consola clásica. *docs.aws.amazon.com.* [En línea] [Citado el: 13 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonECS/latest/developerguide/tutorial-efs-volumes.html.
Volúmenes de Amazon EFS. *docs.aws.amazon.com.* [En línea] [Citado el: 13 de 11 de 2022.] https://docs.aws.amazon.com/es_es/AmazonECS/latest/bestpracticesguide/storage-efs.html.
2022. Web application firewall. *es.wikipedia.org.* [En línea] 12 de 12 de 2022. https://es.wikipedia.org/wiki/Web_application_firewall.
2022. What is Grafana? *grafana.com.* [En línea] 19 de 12 de 2022. https://grafana.com/.
2022. What is Prometheus? *prometheus.io.* [En línea] 17 de 12 de 2022. https://prometheus.io/docs/introduction/overview/.