

Capítulo 5

Despliegue en la nube

Hemos elegido Amazon Web Service como servicio de computación en la nube por el gran impacto que está teniendo en el mundo laboral, mayor que sus competidores directos como son Azure y Google Cloud.

Otro factor que juega a favor de AWS con respecto a sus competidores es que a la hora de desplegar una aplicación en la nube la experiencia de usuario resulta mucho mas intuitiva que la de sus competidores. También ofrece un buen soporte, donde en cualquier momento te resuelven las posibles dudas a la hora de desplegar la aplicación.

Por último y como factor de bastante importancia para aquellos pequeños emprendedores que quiera empezar a desarrollar sus ideas, es el bajo coste que supone tener una aplicación desplegada en la nube de Amazon. Por el momento, el primer año de tu cuenta en AWS es gratuita, restringida a ciertas limitaciones.

Hoy en día, Amazon Web Services proporciona una plataforma de infraestructura escalable, de confianza y de bajo costo en la nube que impulsa cientos de miles de negocios de 190 países de todo el mundo. Con centros de datos en Estados Unidos, Europa, Brasil, Singapur, Japón y Australia. Por ejemplo:

1. **Amazon.com:** Es el minorista online más grande del mundo. En 2011, Amazon.com pasó de utilizar el backup en cinta a usar Amazon S3 en la cloud para realizar copias de seguridad de la mayoría de las bases de datos de Oracle de las que se encarga. Mediante el uso de AWS, Amazon.com logró eliminar el software de backup y experimentó una mejora de desempeño 12 veces mayor, de forma que pudo reducir el tiempo de restablecimiento de 15 a 2,5 horas aproximadamente en situaciones seleccionadas.
2. **Netflix:** El referente de la televisión en streaming usa AWS para proporcionar miles de millones de horas de vídeo casa mes a sus mas de 60 millones de suscriptores. Así puede hacer uso de miles de servidores y terabites de almacenamiento en cuestión de minutos para que sus usuarios puedan ver series y películas desde cualquier parte del mundo en sus tabletas o teléfonos móviles.
3. **Dropbox:** El famoso y conocido servicio de alojamiento de archivos multiplataforma en la nube utiliza hasta el momento AWS como repositorio para almacenar todos los archivos que los usuarios de Dropbox suben a la red.
4. **FC Barcelona:** Su sitio web, que aloja más de 6 000 páginas y 12.000 fotos digitales, también usa AWS para su mantenimiento.

5. **Harvard Medical School** Desarrolla nuevos modelos de pruebas de genomas en tiempo récord.
6. **Mapfre:** Ahorró 1,3 millones de euros en infraestructura y redujo el desarrollo de semanas a días

5.1. Lanzar una instancia en AWS

Antes de subir nuestra aplicación a producción debemos de crear una instancia en AWS. Estos son los pasos a tener en cuenta a la hora de crear una instancia en AWS:

- **Paso1. Crear Key Pairs** Los *Key Pairs* se utilizan para iniciar sesión de forma segura en los servicios de AWS. Crearemos un Key Pairs para acceder a nuestra instancia de EC2.

1. Para crear nuevos pares de claves, se debe navegar hasta AWS Console y luego hacer clic en EC2.

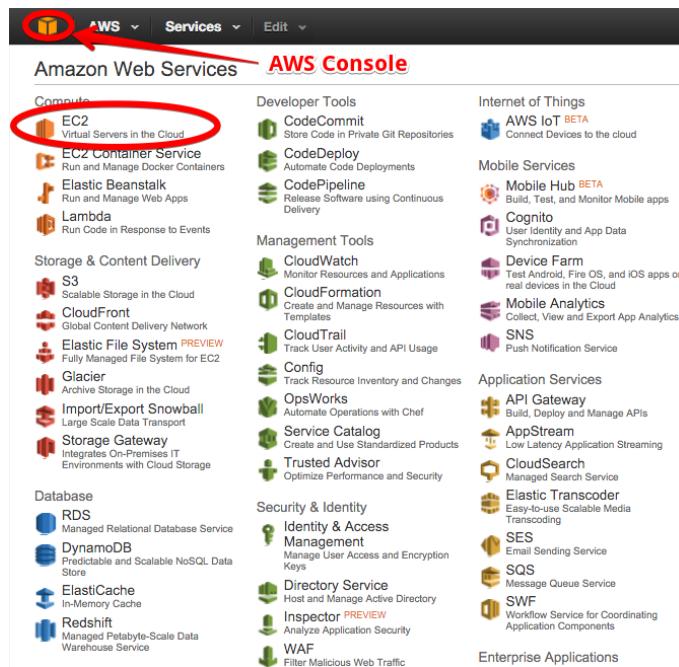


Figura 5.1: Crear Key Pairs Paso 1

2. En el panel izquierdo, se debe hacer clic en Key Pairs, luego clic en Crear Key Pairs
3. Se debe introducir un nombre para la clave, luego hacer clic en Crear Key Pairs. El Key Pairs se descargará automáticamente. Debe moverse esta clave a un directorio diferente.

Importante: Se deberá cambiar los permisos de esta clave para que sean de solo lectura, con el siguiente código:

```
chmod 400 youKeyName.pem
```

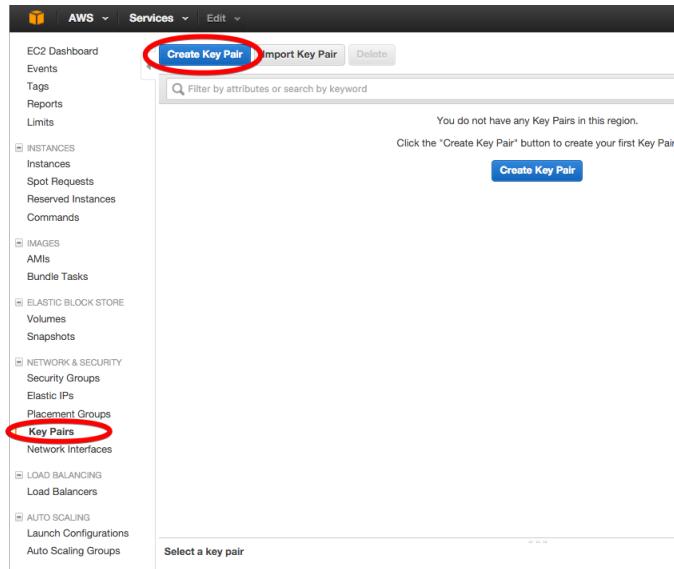


Figura 5.2: Crear Key Pairs Paso 2

- **Paso 2. Lanza una instancia de EC2 con Bitnami** En este paso, lanzaremos una instancia de EC2 desde Amazon Machine Image (AMI). Con AMI se puede activar una instancia de EC2 que esté lista para el desarrollo sin demasiada configuración. Bitnami proporciona una imagen MEAN preconfigurada que usaremos para configurarlo rápidamente.

1. Primero se debe navegar a la consola de AWS y hacer clic en AWS Marketplace.

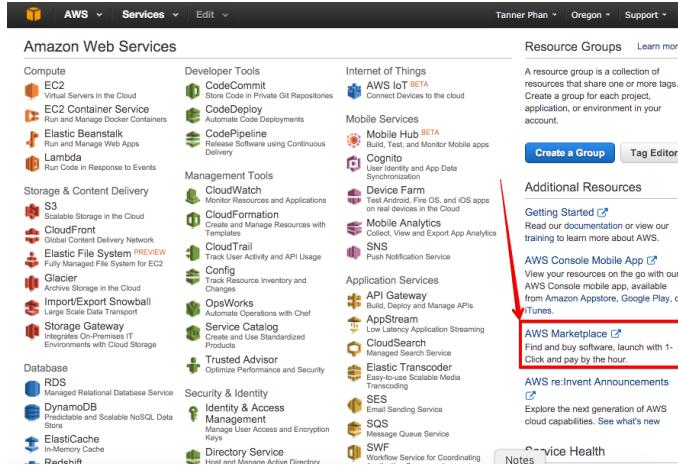


Figura 5.3: Lanza una instancia Paso 1

2. Buscar MEAN powered en Bitnami, luego seleccione 64-bit AMI para continuar. Ver figura 5.4
3. En el apartado *Pricing Details* con el fin de obtener la mejor velocidad de entrega, seleccionar la región más cercana y luego hacer clic en Continuar. Ver figura 5.5
4. Para el grupo de seguridad, se ha de elegir 'Crear nuevo' según la configuración del vendedor.

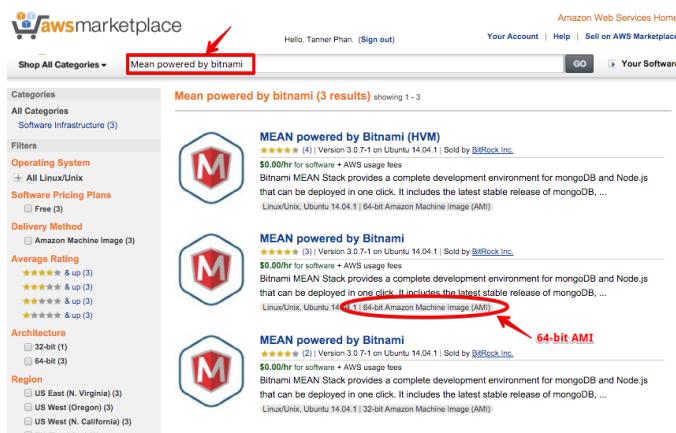


Figura 5.4: Lanza una instancia Paso 2

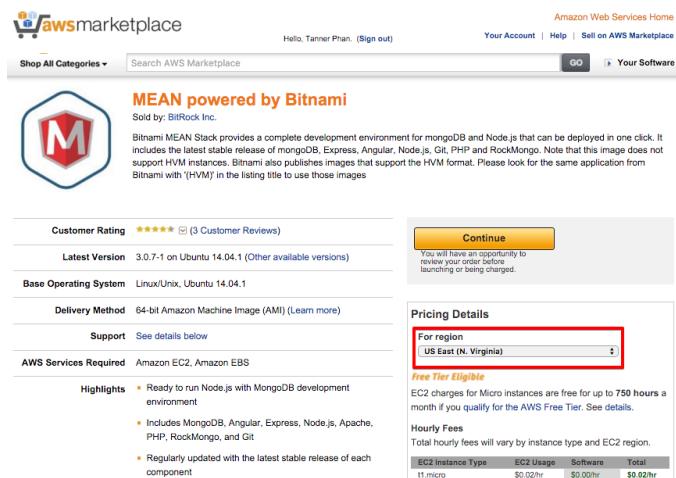


Figura 5.5: Lanza una instancia Paso 3

5. Asegurarse de tener los siguientes métodos de conexión:

1. SSH, My IP
2. HTTP, Anywhere
3. HTTPS, Anywhere

6. Seleccionar la Key Pair que creaste en el paso 1.
7. Finalmente, hacer clic en Iniciar para iniciar la instancia.

■ **Paso 3. Conéctate a tu EC2** Para conectarse por SSH a la instancia, se necesitará la IP pública de su instancia y el Key Pair que creó.

1. Dentro de EC2, hacer clic en Instancias, seleccionar la instancia recién iniciada en el paso anterior y luego hacer clic en Conectar.
2. Copiar el código que aparece marcado en rojo en la imagen [5.6]
3. En su terminal, navegar hasta el directorio donde está guardado el par de claves, luego pegar el último paso del código.

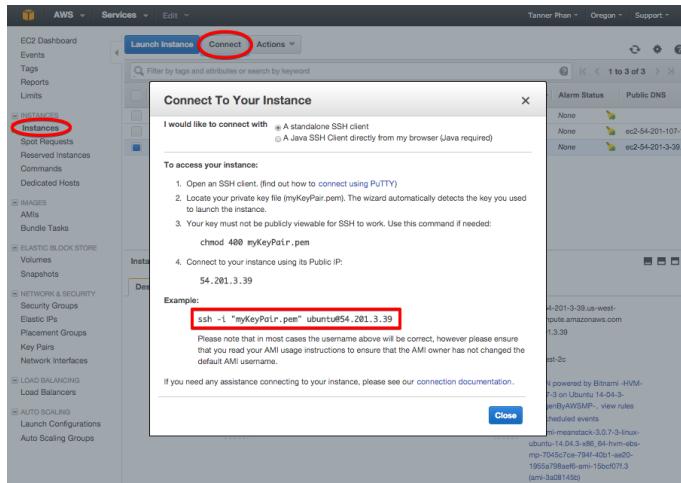


Figura 5.6: Conéctate a tu EC2 Paso 2

```
The authenticity of host '54.201.14.233 (54.201.14.233)' can't be established.
ECDSA key fingerprint is SHA256:ITSaXliGDhtY8uTh10BYnGM8EtH3B07LzR8F/f1T/xE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.201.14.233' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-71-generic x86_64)

[...]
*** Welcome to the Bitnami MEAN 3.0.7-3 ***
*** Bitnami Wiki: https://wiki.bitnami.com/ ***
*** Bitnami Forums: https://community.bitnami.com/ ***
bitnami@ip-172-31-1-86:~$
```

Figura 5.7: Conéctate a tu EC2 Paso 3

- **Paso 4. Conseguir un dominio** Una vez que ya tenemos la instancia creada y hemos sido capaces de conectarnos a ella por SSH, llega el momento de conseguir un dominio para que los usuarios se puedan conectar a la aplicación lo más fácilmente posible. El dominio que he utilizado es un dominio, proporcionado por el propio Amazon.

- **Paso 4. Obtener un certificado SSL gratis con AWS Certificate Manager**

Un certificado SSL es un fichero informático generado por una entidad de servicios de certificación que asocia unos datos de identidad a una persona física, organismo o empresa, confirmando de esta manera su identidad digital en Internet. Necesitamos un certificado SSL para que los usuarios que accedan a nuestra página lo hagan de una forma segura por el protocolo HTTPS.

Por otro lado tenemos Amazon CloudFront, es un servicio de red de entrega de contenido (CDN) global que proporciona datos, vídeos, aplicaciones y API de forma segura a sus espectadores con baja latencia y altas velocidades de transferencia.

Para conseguir el certificado SSL en AWS se necesita seguir los siguientes pasos:

1. Ir a AWS *manager certificate* desde la consola de Amazon web services y hacer click en *get started*.
2. A continuación te pedirá el dominio creado previamente y tendrás que hacer click en *Review and request*.

3. Una vez enviada la solicitud de certificado a la Autoridad certificadora, un email de confirmación será enviado a nuestra cuenta de correo vinculada con el dominio, es decir a admin@classcity.es.
4. Una vez que recibimos un correo como el que aparece en la siguiente imagen, realizamos la confirmación del certificado haciendo click en el enlace que viene en el correo.

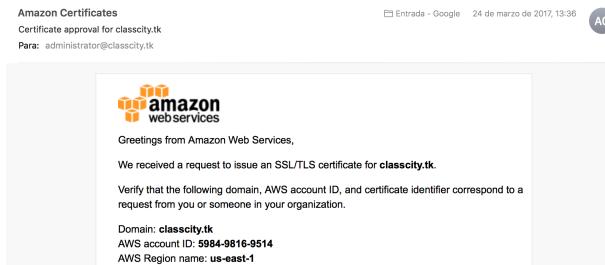


Figura 5.8: Correo de confirmación

Si ahora vamos a nuestro dominio e introducimos antes el protocolo HTTPS, es decir en nuestro caso <https://www.classcity.es>, veremos que efectivamente accedemos a nuestra aplicación de una forma segura. Ver figura 5.9



Figura 5.9: SSL

5.2. Lanzar una aplicación MEAN a producción

Una vez que dispongamos de una instancia, un dominio y un certificado SSL, tenemos todo listo para desplegar nuestra aplicación MEAN en producción. Para ello debemos seguir los siguientes pasos:

1. Conectarnos desde nuestra terminal por SSH a nuestra máquina virtual de AWS

```
sudo ssh -i /.ssh/keypair.pem root@ipinstance
```

2. Clonar en la máquina virtual de AWS nuestro repositorio git donde almacenemos la aplicación MEAN.

```
rooti@ip sudo git clone https://github.com/RoboticsURJC-students  
/2016-tfg-Mario-Fernandez.git
```

3. Instalar dependencias de la aplicación web por parte del lado cliente.

```
rooti@ip cd 2016-tfg-Mario-Fernandez
rooti@ip /2016-tfg-Mario-Fernandez sudo npm install
```

4. Correr Angular con angular-ci.

```
rooti@ip /2016-tfg-Mario-Fernandez sudo npm run build
```

5. Una nueva carpeta se crea al terminar el proceso ./dist, debemos copiarla entera en nuestra carpeta httdocs de apache.

```
rooti@ip /2016-tfg-Mario-Fernandez sudo cp -r ./dist/* ../httdocs
```

6. Ahora deberíamos poder ir a <https://www.classcity.es> y poder ver nuestra aplicación Angular corriendo en producción.



Figura 5.10: <https://www.classcity.es>

7. No olvidemos que en la pila MEAN, aún falta que configuremos el servidor en producción. Para ello lo primero es instalar las dependencias necesarias para lado del servidor.

```
rooti@ip /2016-tfg-Mario-Fernandez cd backend
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo npm install
```

8. Antes de ejecutar nada en la parte del servidor, debemos configurar ciertas rutas en el servidor apache que va ser el encargado de recibir la peticiones de entrada. Para ello debemos hacer lo siguiente:

- Editamos el archivo de configuración de apache llamado httpd.conf para que cuando llegue las solicitudes de tipo /app/* redirigir a localhost: 8080, que será donde estará escuchando nuestra aplicación de gestión de clases particulares.

```
ProxyPassMatch ^/app/(.*)$ http://127.0.0.1:8080/$1
ProxyPass /app/(.*)$ http://127.0.0.1:8080/
ProxyPassReverse /app/(.*)$ http://127.0.0.1:8080/
```

- Y cuando llega la solicitud de type /socket/* redirigir a localhost: 8000, que será el puerto donde este escuchando nuestro chat.

```
RewriteEngine On
RewriteCond %{REQUEST_URI} ^/socket.io [NC]
RewriteCond %{QUERY_STRING} transport=websocket [NC]
RewriteRule /(.*) ws://localhost:8000/$1 [P,L]
ProxyPass /socket.io http://localhost:8000/socket.io
ProxyPassReverse /socket.io http://localhost:8000/socket.io
```

9. A continuación en el lado servidor correremos la base de datos con la propiedad *screen*, para que cuando se termine la conexión por ssh el proceso siga corriendo y no se corte.

```
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo mkdir data
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo screen mongod --dbpath ./data
```

10. Y por último ejecutamos e fichero *server.js* también con la propiedad *screen*, y comprobaremos que funciona correctamente.

```
rooti@ip /2016-tfg-Mario-Fernandez/backend sudo node server.js
screen
```