ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Máster

Comparativa de la implementación y despliegue de una aplicación Web con distintos proveedores de servicio Cloud y no Cloud

AUTOR: Félix Reverte Hernández

DIRECTOR: Fernando Losilla López

Junio 2016

Autor	Félix Reverte Hernández
E-mail del autor	felix@felixreverte.com
Director(es)	Fernando Losilla López
Título del TFG	Comparativa de la implementación y despliegue de una aplicación Web con distintos proveedores de servicio Cloud y no Cloud

Resumen:

En este trabajo se van a estudiar las diferentes opciones disponibles de hosting para una aplicación Web.

A lo largo de esta memoria se describirán y estudiarán las múltiples opciones para alojar y ejecutar una aplicación Web, bien sean Cloud (Amazon Web Services, Google Cloud Platform y Microsoft Azure) o No Cloud (VPS, Hosting y Housing). Es difícil decantarse por la mejor elección para una determinada aplicación. Con este trabajo se pretende ayudar en la elección de la solución más óptima en escalabilidad, coste y rendimiento.

Se compararán las distintas soluciones teniendo en cuenta la escalabilidad, el tiempo de puesta en marcha, la usabilidad de la plataforma y el coste. Además, se proporcionarán las configuraciones utilizadas en los distintos servicios.

Se pretende ofrecer como conclusión final una solución genérica que ayude a elegir la plataforma IaaS (Infrastructure as a Service) óptima para cada aplicación.

Titulación	Máster en Ingeniería de Telecomunicación
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	1 de Junio de 2016

Índice

1. Introducción

Es bien sabido que a fecha de 2016, la forma en que se almacena y sirve la información ha cambiado, ya sea música, fotos, video o documentos. Se ha pasado de guardar todo esto en discos duros o memorias de modo local a guardarlo todo en "la nube", gracias a la aparición de servicios como el famoso Dropbox. Ahora se puede acceder a los datos desde cualquier lugar y dispositivo, pero también se pueden desplegar servicios sobre "la nube", habiendo cambiado también la forma de desplegar aplicaciones y servicios.

Empresas como Google, Microsoft y Amazon tienen sus propios servicios en "la nube", estos servicios son conocidos como "Cloud Computing", permitiendo alojar, desplegar y ejecutar aplicaciones de cualquier tipo.

Por esto, en este trabajo se pretende dar una visión global de los principales servicios Cloud y No Cloud disponibles, comparando sus características, pero centrándose en servicios Cloud Infrastructure as a Service (IaaS).

En este primer capítulo se van a presentar los motivos para la realización de este trabajo, así como los objetivos que se pretenden alcanzar. Finalmente se describirá sucintamente el resto de los capítulos que componen esta memoria.

1.1. Motivación

La realización de este trabajo busca aportar respuestas a cuál es el servicio de alojamiento idóneo para una aplicación Web. Para ello será necesario, en primer lugar, conocer los diferentes servicios existentes, ya sean Cloud (Amazon Web Services, Google Cloud Platform y Microsoft Azure) o No Cloud (VPS, Hosting y Housing), tratando de buscar sus principales características y costes.

Teniendo en cuenta que, para una empresa tecnológica, que necesita sus servicios o aplicaciones cuanto antes en el mercado, que sean escalables, de fácil mantenimiento, accesibles mundialmente y hospedarlas en un almacenamiento de coste comedido, las opciones comentadas anteriormente son muy amplias y es difícil elegir.

Los servicios No Cloud a priori son más económicos, pero menos escalables y difíciles de mantener, mientras que los servicios Cloud son configurables de modo que permiten escalar en función de la demanda.
Con el desarrollo de este trabajo se obtendrá una comparativa de los distintos servicios mencionados, aportando una idea de cuál escoger, pero siempre haciendo uso de la capa gratuita ofrecida por los diferentes servicios Cloud. Para probar las diferentes implementaciones posibles sobre cada proveedor de una misma aplicación, se hará uso de una de ejemplo dada por Amazon que se comenta más adelante.

1.2. Objetivos

Los objetivos globales del trabajo son la búsqueda, prueba y comparativa final de los diferentes servicios de alojamiento, tanto Cloud como No Cloud, y la ejecución de una aplicación Web sobre dichos servicios.
Sobre todas las opciones que se prueben se tendrá un entorno para la aplicación Web de ejemplo que cuente con:

- Servidor Web.
- Base de datos NoSQL.
- Envío de correo electrónico.

1.3. Descripción del trabajo

En esta sección se va a proceder a enumerar el contenido de cada uno de los siguientes capítulos.

- En el capítulo 2 se realiza una explicación detallada sobre Cloud Computing, los diferentes servicios que proporcionan los proveedores y los múltiples servicios No Cloud disponibles.
- En el capítulo 3 se explica la aplicación que se va a utilizar, su funcionamiento y sus componentes.
- En el capítulo 4 se definen diferentes soluciones para la ejecución y hospedaje de la aplicación, aportando las configuraciones realizadas en los distintos servicios.
- Por último, en el capítulo 5 se exponen las conclusiones extraídas en el desarrollo del trabajo y las posibles líneas futuras.

2.Servicios Cloud y No Cloud

En este capítulo se va a realizar una descripción detallada y conceptual de los servicios Cloud y No Cloud, aportando las definiciones, herramientas y servicios que se utilizarán en el desarrollo de este trabajo.

2.1. Cloud Computing

Según el NIST (National Institute of Standards and Technology) Cloud Computing se define como "el modelo tecnológico que permite el acceso ubicuo, adaptado y bajo demanda en red a un conjunto compartido de recursos de computación configurables (por ejemplo: redes, servidores, equipos de almacenamiento, aplicaciones y servicios) que pueden ser desplegados rápidamente, con mínimo esfuerzo de gestión o interacción con el proveedor del servicio."

2.1.1. Modelo Cloud

Para que este modelo sea considerado como Modelo Cloud debe componerse de 5 características esenciales, 3 modelos de servicio y 4 modelos de despliegue según el NIST.

Estas son las 5 características esenciales que debe tener un servicio Cloud:

- **Accesibilidad**: Los usuarios pueden acceder a los servicios que se encuentran en la nube desde cualquier ordenador, Smartphone, Tablet, etc. con conexión a Internet.

- **Autoservicio de servicios bajo demanda**: Un consumidor puede acceder a una serie de recursos de computación disponibles en la nube, sin interacción humana por parte del proveedor, además de poder aumentar o disminuir los recursos en función de las necesidades.

- **Compartición de recursos**: Todos los usuarios tienen acceso a los servicios disponibles en la nube, y los recursos, ubicados por centros de datos, países, provincias, etc., son asignados conforme la demanda del usuario en cada momento.

- **Elasticidad**: Según las necesidades del cliente, la cantidad de recursos y servicios pueden incrementarse o decrementarse, pudiendo automatizarse en función de la demanda. Los recursos del servidor son transparentes para el cliente, y podrá acceder a cualquier servicio cuando lo desee.

- **Servicio medido**: Se controla y se mide el uso de recursos computacionales por el proveedor, que debe notificar al cliente, evitando posibles colapsos de algún servicio o recurso. Los costes del servicio son predecibles.

Modelos de Servicio:

- **IaaS (Infrastructure as a Service)**
  El proveedor ofrece su propia infraestructura como servicio, como son recursos de computación, almacenamiento, red, etc. El cliente puede desplegar cualquier tipo de software en las instalaciones del proveedor (sistema operativo y aplicaciones), pero no gestiona la infraestructura del proveedor, sólo tiene el control sobre el software que ha desplegado.

- **PaaS (Platform as a Service)**
  El proveedor ofrece servicios creados sobre su infraestructura añadiendo una capa de abstracción sobre los recursos, haciéndolos accesibles normalmente a través de una API. La plataforma se encarga de desplegar la aplicación del cliente sobre su infraestructura o sobre la de un proveedor de IaaS. El cliente no tiene que preocuparse por aspectos complejos como el escalado de la aplicación que está desarrollando.

- **SaaS (Software as a Service)**
  El servicio ofrecido a los clientes es el resultado de una aplicación que se ejecuta sobre una infraestructura Cloud, pudiendo contratarse recursos de un proveedor de IaaS, PaaS o recursos propios.

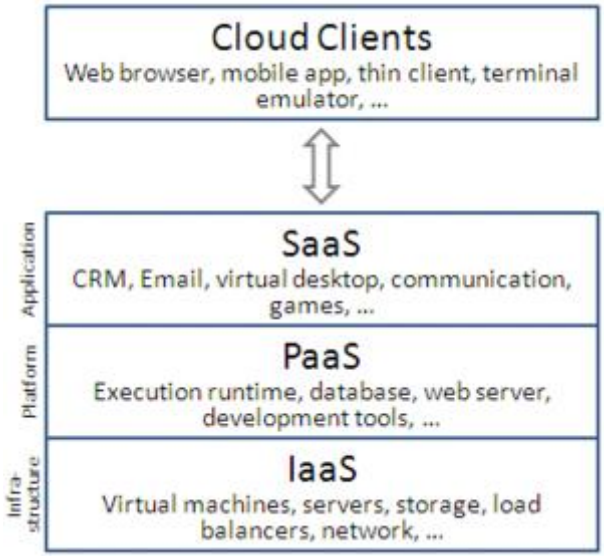


Figura 1: Modelos de servicios de Cloud Computing.

Modelos de Despliegue:

- **Nube privada:**
  Sólo disponible para los miembros de una determinada organización y suele encontrarse en las instalaciones de dichas organizaciones, por tanto, ofrece un mayor nivel de seguridad y control sobre los datos e información. La propia organización decide qué, quién y dónde se pueden ejecutar los procesos, lo que provoca que no se ofrezcan servicios a personas fuera de la organización.

- **Nube pública:**
  El acceso a estos servicios se puede hacer de forma gratuita o mediante el pago de un alquiler por uso y cualquiera puede contratarlos. La información almacenada por los usuarios se encuentra en los mismos equipos, sistemas de almacenamiento, servidores e infraestructuras en la nube.
  Posee alta capacidad de procesamiento y de almacenamiento transparente al usuario, ya que éste no paga por el mantenimiento, sólo por el uso.

- **Nube comunitaria:**
  Posee una infraestructura compartida entre distintas organizaciones con intereses comunes. Pueden ser controladas y administradas tanto internamente como por terceros, pudiéndose encontrar dentro o fuera de dichas organizaciones.

- **Nube híbrida:**
  Combinación de nubes privadas y públicas, permitiendo que el cliente sea propietario de algunas partes y otras sean compartidas, manteniendo el control de la estructura. Requiere menor inversión inicial que una nube privada, pero es más compleja al distribuir las aplicaciones entre los tipos de nube citados.

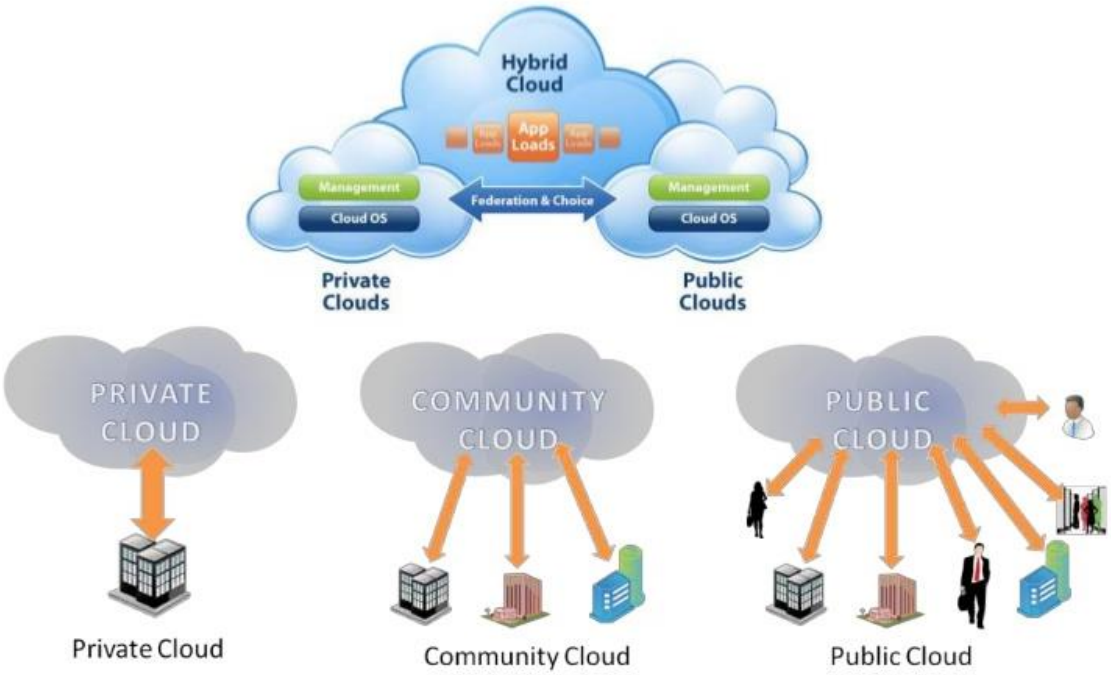


Figura 2: Modelos de despliegue de Cloud Computing.

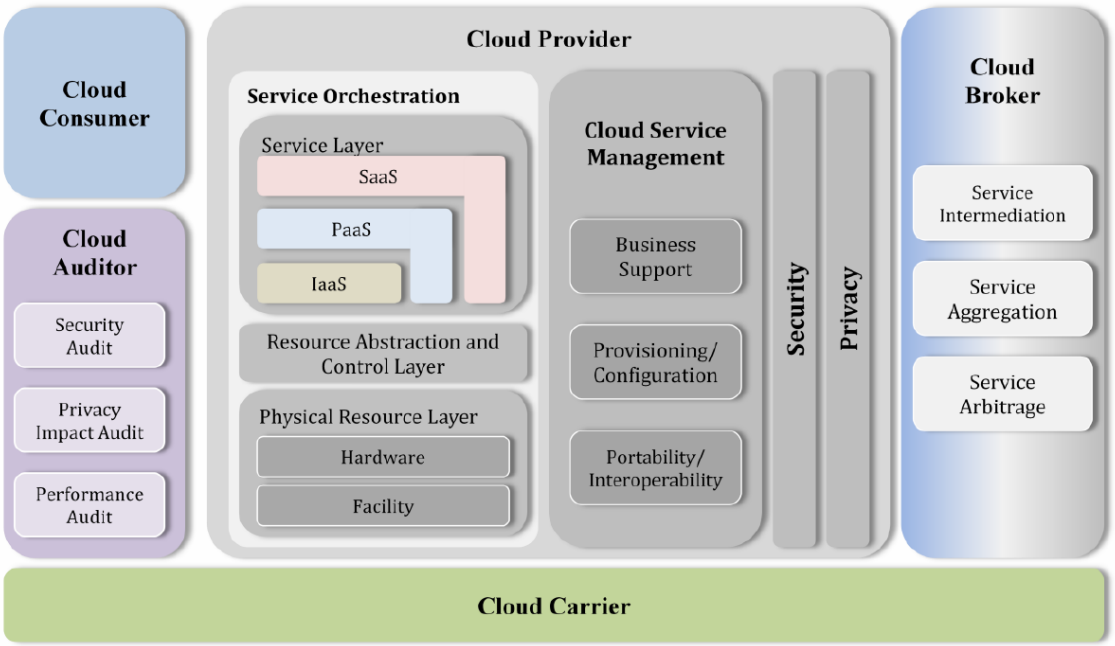


Figura 3. Modelo conceptual de referencia.

En la figura 3 se desglosan esquemáticamente las diversas partes que intervienen en la utilización del Cloud, mientras que en la tabla siguiente se explican cada una de ellas, denominadas como actuadores.

Actuadores	Definición
Cloud Consumer	Persona u organización que mantiene una relación comercial con, y utiliza el servicio de, los *Cloud Providers* (proveedores de nube).
Cloud Provider	Persona, organización o entidad responsable de hacer que un servicio esté disponible para las partes interesadas.
Cloud Auditor	Un grupo que puede realizar una evaluación independiente de los servicios Cloud, información de las operaciones del sistema, rendimiento y seguridad de la implementación de la nube.
Cloud Broker	Entidad que gestiona el uso, rendimiento y entrega de los servicios Cloud, y negocia relaciones entre *Cloud Providers* y *Cloud Consumers*.
Cloud Carrier	Intermediario que proporciona conectividad y transporte de los servicios Cloud desde los *Cloud Providers* hasta los *Cloud Consumers*.

Tabla 1: Actuadores en Cloud Computing.

2.1.2. Ventajas e inconvenientes

El modelo tecnológico Cloud Computing ha revolucionado el mundo tecnológico y proporcionado nuevas formas de utilizar la tecnología, en esta sección se describen sus ventajas e inconvenientes.

Ventajas:

- Simplificación del proceso de desarrollo.
  Los proveedores ofrecen a los desarrolladores y a las empresas servicios que permiten desarrollar, desplegar y mantener las aplicaciones más fácilmente. Servicios como balanceo de carga, almacenamiento de datos, envío de emails, despliegues automáticos, etc.

- Costes.
  Debido a que no se necesita una inversión inicial elevada para iniciar un nuevo proyecto de desarrollo se reducen los costes de la infraestructura, mantenimiento y energía. Por otro lado, sólo se paga por el uso de los recursos y servicios utilizados.

- Escalado y aprovisionamiento instantáneo de recursos.
  Se pueden contratar o liberar recursos en un reducido periodo de tiempo según las necesidades.

- Alta disponibilidad.
  Los servicios se encuentran redundados geográficamente haciendo que sean accesibles desde cualquier lugar y en cualquier momento, teniendo además un alto grado de disponibilidad operacional.

Inconvenientes:

- Dificultad para portar aplicaciones.
  Aplicaciones existentes que no se encuentren desplegadas en la nube pueden requerir un rediseño importante para poder desplegarlas en un proveedor Cloud. Además, una aplicación desarrollada para un determinado proveedor puede ser difícil de portar a otro.

- Uso de bases de datos.
  Las bases de datos relacionales son difíciles de escalar además de presentar precios elevados en Cloud. En la medida de lo posible se usan bases de datos no relacionales para bases de datos grandes.

- Las aplicaciones deben ser tolerantes a fallos.
  Las máquinas que alojan las aplicaciones en un entorno Cloud pueden fallar y, según lo contratado, se crean y destruyen instancias (servidores) bajo demanda.

- Caídas del servicio.
  Aunque es muy poco habitual, el proveedor Cloud puede tener problemas como corte del suministro eléctrico o cortes de red, incluso quiebra del proveedor.

- Seguridad.
  Generalmente los proveedores Cloud son seguros, pero puede tener vulnerabilidades.

- Algunas aplicaciones no requieren realmente un servicio Cloud, porque no se requiera escalabilidad, se van a ofrecer a un número reducido de clientes, se infrautilizan recursos, etc., aun así, podrían ser más económicas con proveedores Cloud.

2.1.3. Proveedores

En este apartado se explican los proveedores de servicios Cloud más relevantes, dando a conocer sus zonas o regiones de funcionamiento, servicios ofrecidos y tarificación.

2.1.3.1. Amazon Web Services

El más conocido y más extendido a nivel global, AWS es el conjunto de servicios de computación en la nube de Amazon.com. Los servicios ofrecidos se clasifican en distintos grupos, los más importantes son: almacenamiento, bases de datos, entrega de contenido, mensajería y procesamiento de datos.

Figura 4. Amazon Web Services.

Con el objetivo de ofrecer cobertura global y una latencia reducida, AWS dispone de centros de datos distribuidos por todo el mundo, permitiendo a los usuarios elegir en que región desplegar sus servicios. Además de minimizar la latencia, los costes varían según la región de despliegue y la de uso, ya que el tráfico de datos se tarifica.

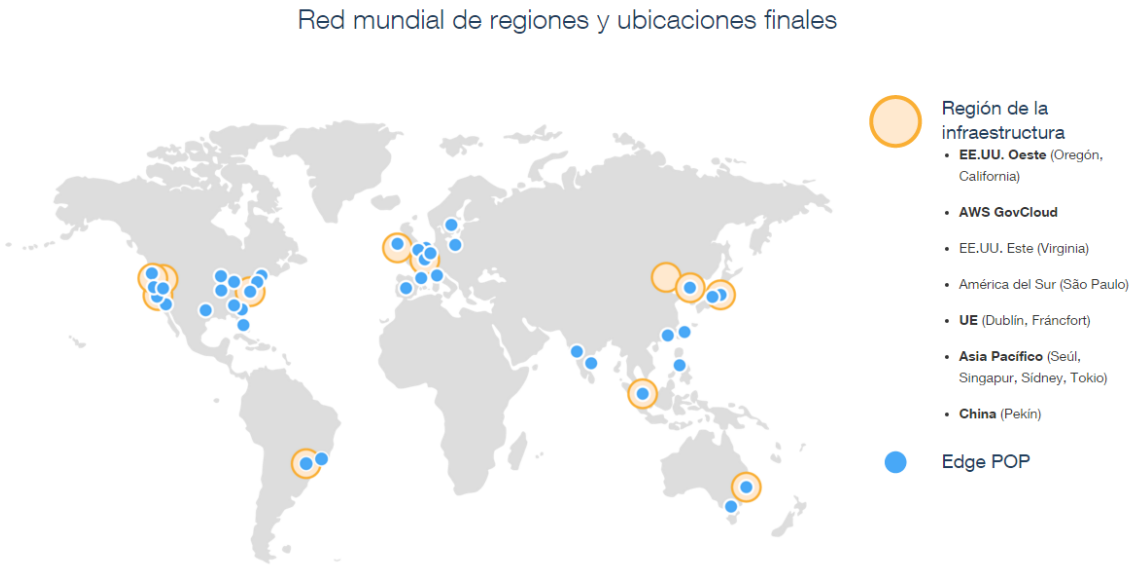


Figura 5. Regiones de AWS.

Las regiones están aisladas entre sí, facturando la trasferencia de datos entre regiones por entrada/salida de datos, además, estas regiones están divididas en varias zonas de disponibilidad. Por otro lado, los clientes podrán replicar y realizar copias de seguridad de sus aplicaciones y/o datos en las regiones que lo deseen.

Amazon ofrece una calculadora de costes que permite calcular el precio de los servicios que se contraten en función de las tecnologías utilizadas, el tiempo de uso, número de instancias, lecturas/escrituras, etc.

Para el desarrollo de este trabajo se van a utilizar los servicios que se explican a continuación:

- Elastic Beanstalk.
  Es un servicio de implementación y escalado de servicios y aplicaciones web desarrollados en Java, .NET, PHP, NodeJS, Python, Ruby, Go y Docker en servidores como Apache, Nginx, Passenger e IIS.
  Solo se tiene que cargar el código y este servicio administra automáticamente la implementación, el aprovisionamiento de la capacidad, el equilibrio de la carga, el escalado automático y la monitorización del estado de la aplicación.

- EC2.
  Es un servicio web que proporciona máquinas virtuales en la nube controladas por el cliente y de tamaño modificable, diseñado para facilitar a los desarrolladores el Cloud Computing escalable basado en web.

- DynamoDB.
  Es un servicio de base de datos NoSQL rápido y flexible para las aplicaciones que requieren latencias constantes y de escasos milisegundos a cualquier escala. Se trata de una base de datos totalmente administrada en la nube, compatible con modelos de almacenamiento de datos de valor de clave y de documentos.

- SNS.
  Servicio de publicación-suscripción de mensajería, permite el envío de mensajes push, es escalable, multiplataforma y entrega entre distintos protocolos, Apple iOS, Android, colas, etc.

2.1.3.2. Google Cloud Platform

Google Cloud Platform es un conjunto de servicios modulares basados en la nube que permiten crear desde simples sitios web hasta aplicaciones complejas.

El funcionamiento y los servicios que ofrece son similares a los de AWS, prometiendo menores costes y ofreciendo una calculadora en la que compara el uso de su plataforma frente a AWS. Ofrece también una calculadora de costes al estilo de la de Amazon pudiendo calcular los costes de funcionamiento de la misma manera.

Figura 6. Google Cloud Platform.

Figura 7. Servicios Google Cloud Platform.

Al igual que AWS, se encuentra dividido en regiones, que a su vez se sub-dividen en zonas, siendo importantes a la hora de ofrecer servicio con baja latencia. Existen límites de cuota para el proyecto y la zona para recursos como imágenes, direcciones IP estáticas y reglas de firewall.

Region	Location	Available zones	Features
Eastern US	Berkeley County, South Carolina	us-east1-b us-east1-c us-east1-d	• Haswell processors • 32-core machine types • Local SSDs
Central US	Council Bluffs, Iowa	us-central1-a	• Sandy Bridge processors • Local SSDs
		us-central1-b us-central1-c	• Haswell processors • 32-core machine types • Local SSDs
		us-central1-f	• Ivy Bridge processors • 32-core machine types • Local SSDs
Western Europe	St. Ghislain, Belgium	europe-west1-b	• Sandy Bridge processors • Local SSDs
		europe-west1-c	• Ivy Bridge processors • 32-core machine types • Local SSDs
		europe-west1-d	• Haswell processors • 32-core machine types • Local SSDs
East Asia	Changhua County, Taiwan	asia-east1-a asia-east1-b asia-east1-c	• Ivy Bridge processors • 32-core machine types • Local SSDs

Figura 8. Regiones Google Cloud Platform.

Para el desarrollo de este trabajo de este proveedor se van a utilizar los servicios que se explican a continuación:

- Compute Engine.
  Servicio de máquinas virtuales de alto rendimiento escalables controladas por el cliente.

- Datastore.
  Servicio de base de datos NoSQL altamente escalable para aplicaciones, maneja automáticamente la replicación y los datos compartidos de forma muy rápida.

2.1.3.3. Microsoft Azure

Del mismo modo que los dos servicios anteriores, Microsoft cuenta con su propia plataforma Cloud, proporcionando los mismos servicios, pago por uso, distribución geográfica, elasticidad, alta disponibilidad, seguridad, escalabilidad.

Figura 9. Microsoft Azure.

Azure tiene disponibilidad general en 22 regiones de todo el mundo y ha anunciado planes para 5 regiones adicionales.

Figura 10. Zonas Microsoft Azure.

Por último, en este proveedor los servicios utilizados en la realización de este trabajo son los siguientes:

- Máquina virtual.
  Servicio de máquinas virtuales escalables en la nube con Windows Server o Linux controladas por el cliente.

- DocumentDB.
  Es un servicio de bases de datos NoSQL de baja latencia.

2.2. Entornos No Cloud

También conocido como modelo tradicional, el modelo no Cloud se caracteriza por estar formado por una infraestructura fija, es decir, se cuenta con un equipamiento físico difícilmente ampliable.

2.2.1. Modelo No Cloud

Este modelo se basa en las arquitecturas monolíticas, en las que se tiene una infraestructura rígida, en la que los aumentos de demanda de servicio son lentamente satisfechos y, por otro lado, los excesos de infraestructura no son amortizables ni retroactivos. De esta forma, se hace costoso escalar la infraestructura, teniendo que planificar adecuadamente los sistemas a modo de prever aumentos de demanda.

2.2.2. Ventajas e inconvenientes

Este modelo tecnológico tradicional se sigue utilizando, en esta sección se describen sus ventajas e inconvenientes.

Ventajas:

- Conocimiento de la infraestructura, ya que ésta es fija y cuenta con un diseño personalizado a cada aplicación.

Inconvenientes:

- Se puede sobredimensionar el sistema para evitar la falta de recursos en determinados momentos.

- El tiempo de aprovisionamiento de nuevos recursos es lento. Por otro lado, si en una hora puntual del día nuestra aplicación tiene gran demanda, pero el resto del día no, esos recursos se desaprovechan.

- Alto coste para pequeñas empresas.

2.2.3. Servicios

Este modelo tradicional de computación se encuentra presente de múltiples formas, ya sea desde la posesión en la empresa de un CPD propio (Centro de Procesamiento de Datos) hasta el alquiler de servidores a terceros. En esta sección se describen varias de estas opciones.

2.2.3.1. VPS

Las siglas VPS, Servidor Virtual Privado (Virtual Private Server, en inglés) denominan al método de particionar un servidor físico en varios servidores virtualizados, a modo de que cada servidor virtual pueda contar con sus propios permisos y sistema operativo. Este modelo sería similar al Infrastructure as a Service (IaaS) del modelo Cloud.

De esta forma algunas empresas como OVH, Hetzner o KIO Networks ofrecen sobre un mismo servidor físico varios servidores virtuales que alquilan a los usuarios, siendo el usuario el que puede usar a su antojo la partición virtualizada de la que dispone.

Los servidores físicos se encuentran localizados en una zona, sin disponer de redundancia regional como los servicios Cloud. Tampoco ofrecen escalado de potencia de procesado, solo ofrecen escalado de almacenamiento, es decir, si se dispone de un servidor virtual con un CPU y una RAM solo se puede aumentar el tamaño del disco duro.

Por último, si un servidor físico se rompe depende la recuperación de los datos de que el usuario disponga de copias de seguridad, ya que normalmente, el proveedor no se hace responsable de pérdidas.

2.2.3.2. Hosting

En este tipo de infraestructura el proveedor ofrece dos opciones al cliente, por un lado, se dispone de un directorio dentro del equipo físico, permitiendo al cliente una funcionalidad limitada, tanto de espacio de almacenamiento como de capacidad de computo, además de limitar el tipo de aplicaciones que puede instalar o la forma de hacerlo, por ejemplo, no se permite el uso de SSH.

Por otro lado, el proveedor ofrece un equipo completo para el cliente, llamado servidor dedicado, que cuenta con mayores prestaciones y almacenamiento, incrementando el precio mensual por supuesto, pero permitiendo al usuario utilizar una maquina más potente sin prácticamente limitaciones.

2.2.3.3. Housing

El Housing o co-location, se proporciona por pocos proveedores, en él se ofrece al cliente emplazar sus equipos en las instalaciones del proveedor de modo que se aumenta la seguridad física y ante incendios, el suministro eléctrico, refrigeración y la conexión a internet, pero siendo el cliente el que se encarga de las copias de seguridad y el mantenimiento de sus sistemas.

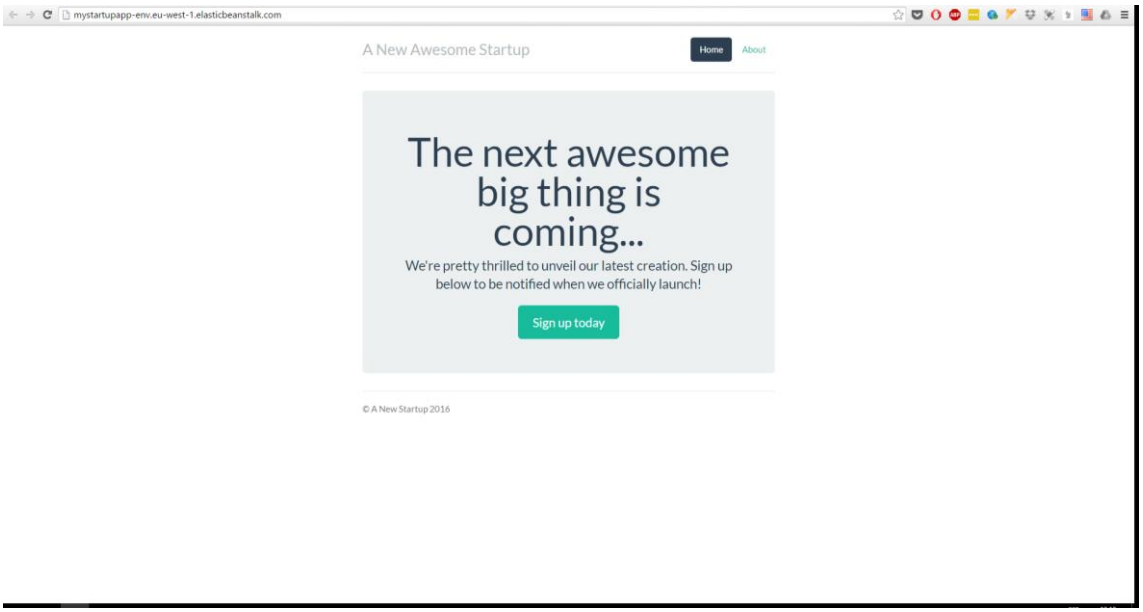
3.Aplicación desplegada

En este capítulo se describe la aplicación que se va a utilizar en el despliegue sobre los distintos servicios, las tecnologías que utiliza y su funcionalidad.

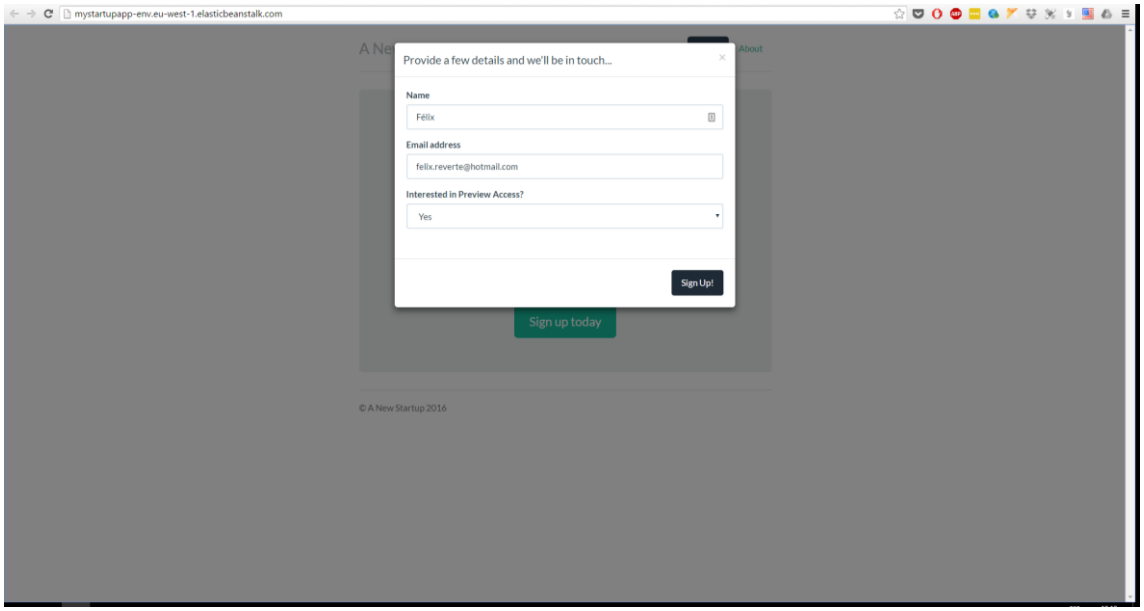
3.1. Aplicación

La aplicación utilizada como ejemplo permite a los clientes suscribirse a una web que prevé su lanzamiento. Esto se realiza mediante el envío del nombre, el email y el interés o no del usuario por recibir acceso anticipado la plataforma.

En las siguientes capturas se puede ver la aplicación en funcionamiento. En primer lugar, la vista principal que obtiene el usuario al ingresar en la plataforma.



En segundo lugar, se puede ver el formulario que puede rellenar el usuario con el que se suscribe a las notificaciones. Tras el registro, se envía un correo electrónico al administrador indicando una nueva suscripción.



Respecto a las tecnologías utilizadas en el funcionamiento de la aplicación cabe destacar que está construida sobre NodeJS, una plataforma que usa JavaScript para formar el lado del servidor de la aplicación. NodeJS consta de una librería y una rutina de ejecución, proporcionada ésta última por el motor V8 de JavaScript.

NodeJS está diseñado sobre un modelo de entrada/salida no bloqueante conducido por eventos, que lo hace útil para la creación de servidores web altamente escalables. La aplicación utiliza dos módulos externos de NodeJS: Express (un framework para aplicaciones web) y Jade (un motor de plantillas usado para crear documentos HTML dinámicamente).

Para que la aplicación se vea bonita y sea adaptable a móviles se utiliza Bootstrap, un framework iniciado como proyecto de Twitter.

En el fichero *package.json* se define el script que ejecuta la aplicación y las dependencias necesarias.

La aplicación original es libre y puede descargarse en la siguiente dirección:

https://github.com/awslabs/eb-node-express-signup

Además, las distintas versiones desarrolladas en la creación de este trabajo para las diferentes implementaciones en cada servicio y proveedor se encuentran también disponibles en mi Github personal y se comentan en la descripción de cada caso.

4.Desarrollo y despliegue

En este capítulo se explican las diferentes soluciones, tanto Cloud como No Cloud, que se han desarrollado utilizando los distintos servicios y tecnologías, aportando las configuraciones necesarias para la ejecución de la aplicación ejemplo.

4.1. Cloud

4.1.1. Amazon Web Services

En Amazon se dispone de dos opciones posibles para el despliegue de esta aplicación, una utilizando instancias EC2 y otra utilizando el servicio Elastic Beanstalk. A continuación, se explican ambas opciones:

1. **Utilización de Elastic Beanstalk con SNS y DynamoDB:**

   Los servicios utilizados en este caso son, en primer lugar, Elastic Beanstalk, un servicio que permite desplegar y configurar rápidamente aplicaciones en la nube de AWS eliminando la preocupación por la infraestructura sobre la que se ejecutan las aplicaciones, simplemente se sube la aplicación, y el servicio se encarga automáticamente de manejar los detalles de capacidad de aprovisionamiento, balanceo de carga, escalado y monitorización, aunque posteriormente se pueden configurar.
   En segundo lugar, SNS (Simple Notification Service), servicio de notificaciones push instantáneas.
   Y, por último, DynamoDB, servicio de base de datos NoSQL escalable en la que se creará una tabla para nuestra aplicación.

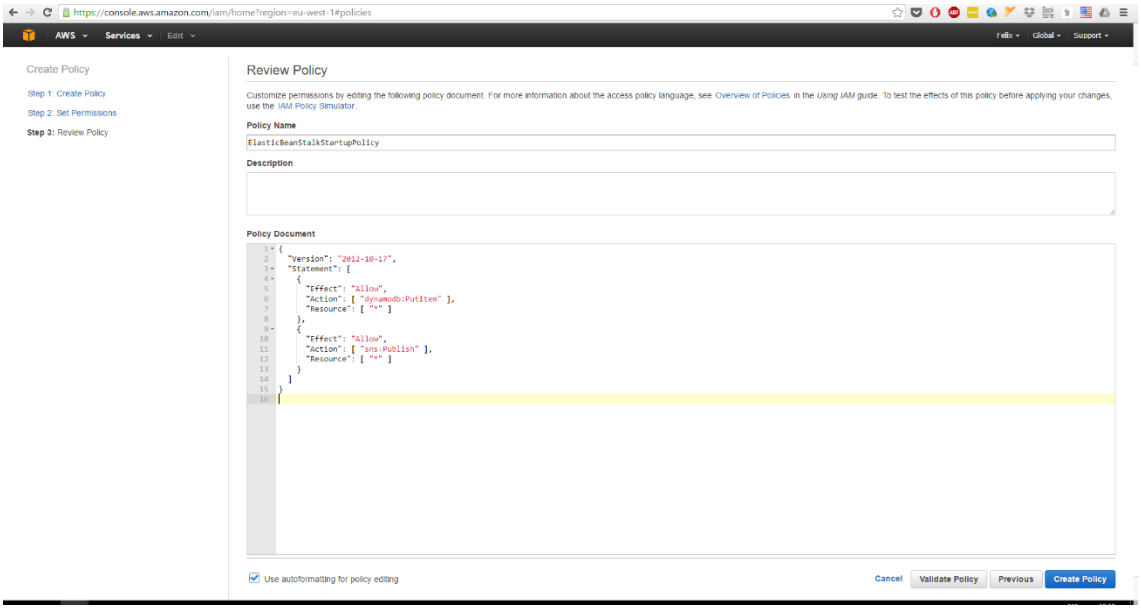
   El coste de ejecutar un sitio web mediante Elastic Beanstalk puede variar según distintos factores, como el número de instancias EC2 necesarias para administrar el tráfico del sitio web, el ancho de banda consumido por la aplicación y qué base de datos u opciones de almacenamiento usa la aplicación. El principal coste para una aplicación web será para la instancia EC2 y para el Elastic Load Balancing que distribuye el tráfico entre las instancias que ejecutan la aplicación.

   Elastic Beanstalk permite utilizar distintos lenguajes de programación preinstalados, en este caso se elige NodeJS.

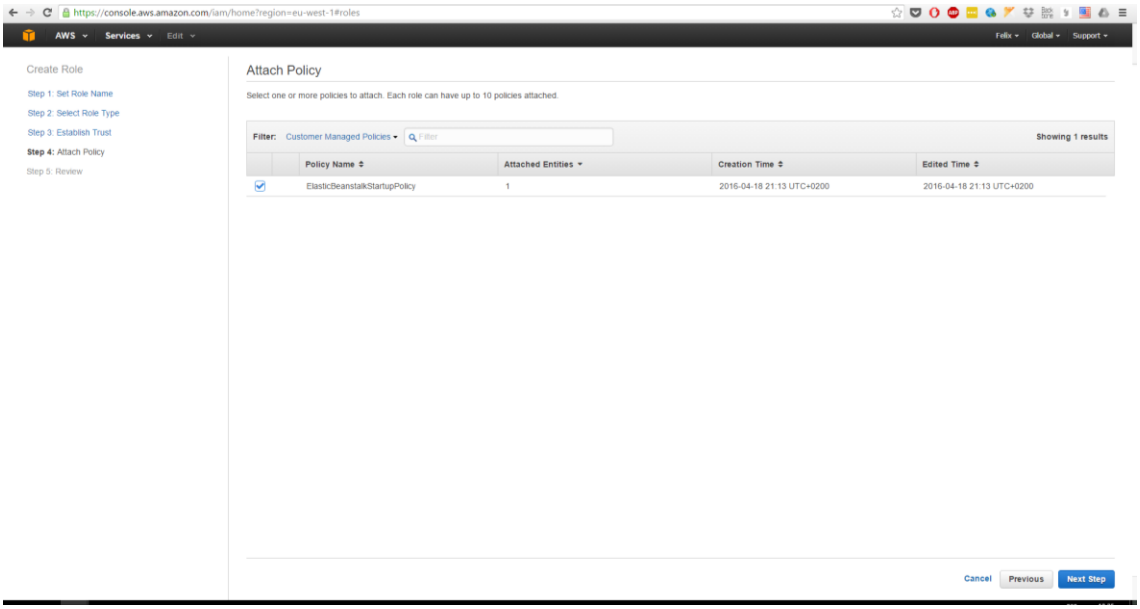
A continuación, se detallan los pasos seguidos para el despliegue de la aplicación:

- Creación de un rol IAM:
  Se debe crear un rol con el objetivo de garantizar permisos a la aplicación para publicar notificaciones SNS y añadir elementos en la tabla DynamoDB.
  Una vez realizado este paso se utilizará también en la siguiente implementación.

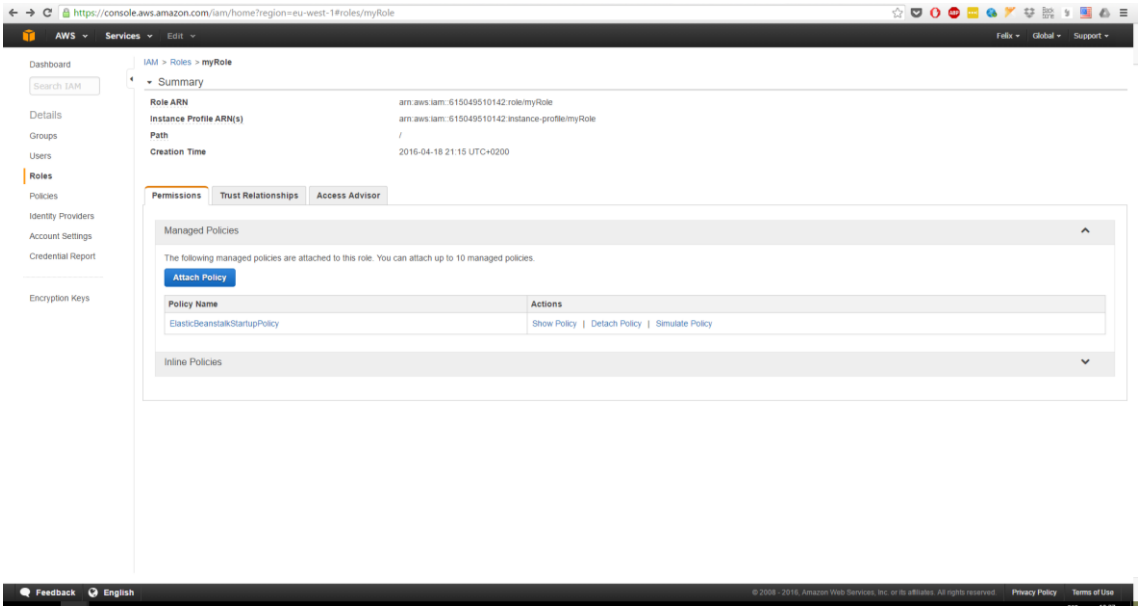
1. Abrir la consola IAM en https://console.aws.amazon.com/iam.
2. En el panel de navegación, seleccionar **Policies**.
3. Elegir **Create Policy** para lanzar el ayudante.
4. En la página de **Create Policy**, en **Create Your Own Policy**, seleccionar **Select**.
5. En **Review Policy**, introducir un nombre de política, en este caso: *ElasticBeanstalkStartupPolicy.*
6. Pegar el contenido del fichero *iam_policy.json* en la caja de texto **Policy Document**. Y clicar en **Create Policy**.



7. En el panel de navegación, seleccionar **Roles**.
8. Seleccionamos **Create New Role**.
9. En la página **Set Role Name**, en la caja **Role Name**, introducir un nombre y seleccionar **Next Step**.
10. En la página **Select Role Type**, en la fila de **Amazon EC2**, clicar en **Select** para permitir a las instancias EC2 realizar llamadas a los distintos servicios de AWS en nombre del usuario.
11. En la página **Attach Policy**, junto a **Filter**, seleccionar **Policy Type**, y elegir **Customer Managed Policies**. Seleccionar **Next Step**.
12. Seleccionar la política *ElasticBeanstalkStartupPolicy* y pulsar **Next Step**.

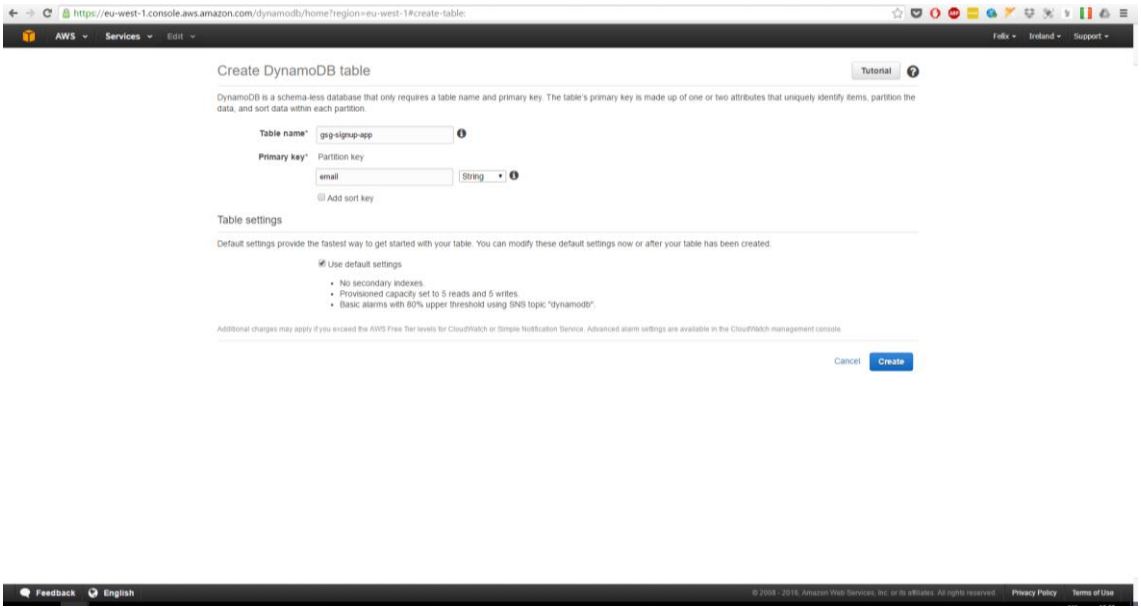


13. En la página **Review**, verificar la información y elegir **Create Role**.



En la imagen superior se puede ver el resumen de los roles creados.

- Creación de una tabla DynamoDB:
  1. Abrir la consola de DynamoDB en https://console.aws.amazon.com/dynamodb/home.
  2. En la barra de menú, se debe tener seleccionada la región en la que se quiere desplegar la aplicación, en este caso, **EU(Ireland)**.
  3. Seleccionar **Create Table**.
  4. En **Table Name**, escribir *gsg-signup-app*.
  5. En **Primary Key**, escribir *email*. Seleccionar **Create**.

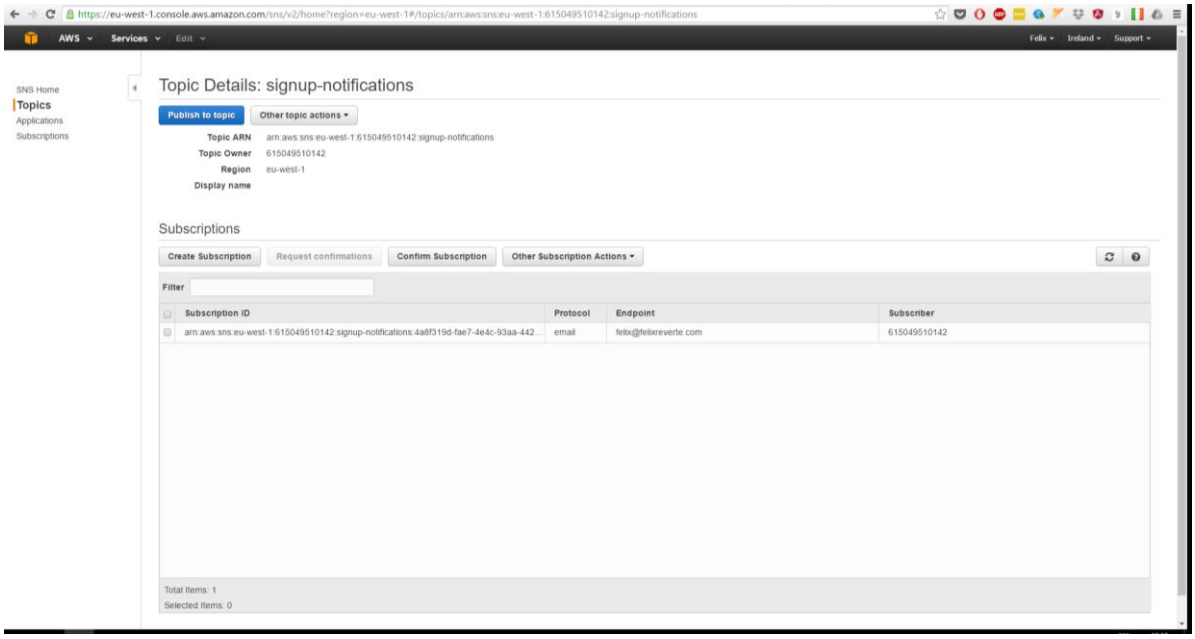


6. Añadir el nombre de la tabla y la región al fichero de configuración de la app (fichero *app_config.json*).

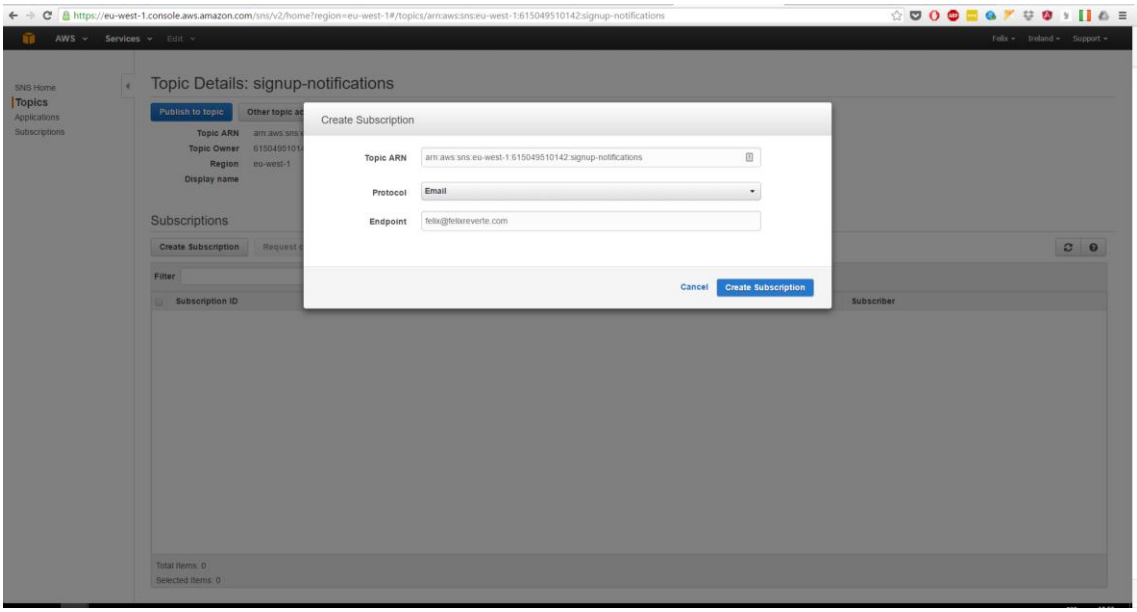
```
{
  "AWS_REGION": "eu-west-1",
  "STARTUP_SIGNUP_TABLE": "gsg-signup-app",
  "NEW_SIGNUP_TOPIC": ""
}
```

- Crear tema SNS.
  1. Abrir la consola SNS en https://console.aws.amazon.com/sns/.
  2. Seleccionar **Create Topic**.
  3. En **Topic Name**, escribir *signup-notifications*. Seleccionar **Create Topic**.

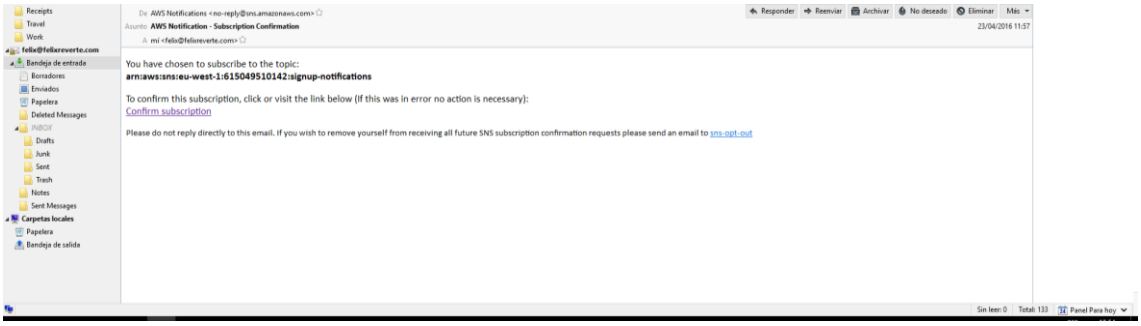
  En la página de detalles, copiar el string de **Topic ARN**, en este caso: *arn:aws:sns:eu-west-1:615049510142:signup-notifications*



4. Añadir el valor anterior al fichero *app_config.json* para el valor *"NEW_SIGNUP_TOPIC"*.
5. Seleccionar **Create Subscription**.
6. En el cuadro de dialogo de **Create Subscription**, seleccionar **Email** como **Protocol**. En el cuadro de texto **Endpoint**, introducir el email al que se recibirán las notificaciones de suscripción de un nuevo usuario. Seleccionar **Create Subscription**. En el cuadro de dialogo de confirmación seleccionar **Close**.



7. En la dirección de email introducida se habrá recibido un email en el que se pide confirmación a la suscripción.



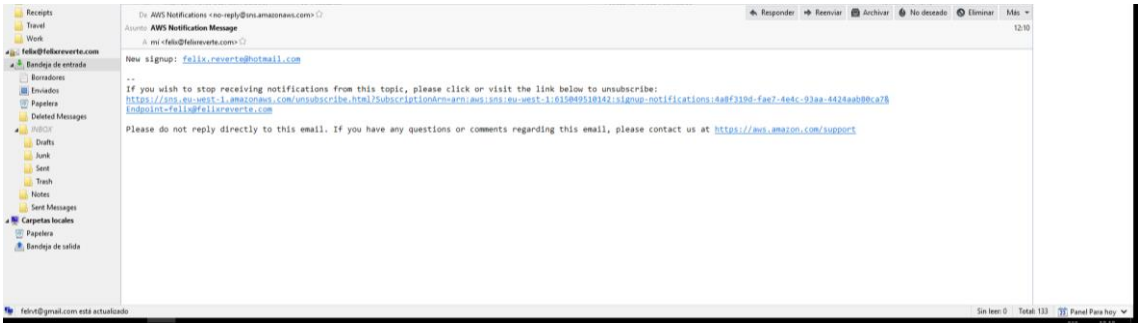
8. El fichero *app_config.json* queda finalmente como sigue:

```
{
  "AWS_REGION": "eu-west-1",
  "STARTUP_SIGNUP_TABLE": "gsg-signup-app",
  "NEW_SIGNUP_TOPIC": "arn:aws:sns:eu-west-
1:615049510142:signup-notifications"
}
```

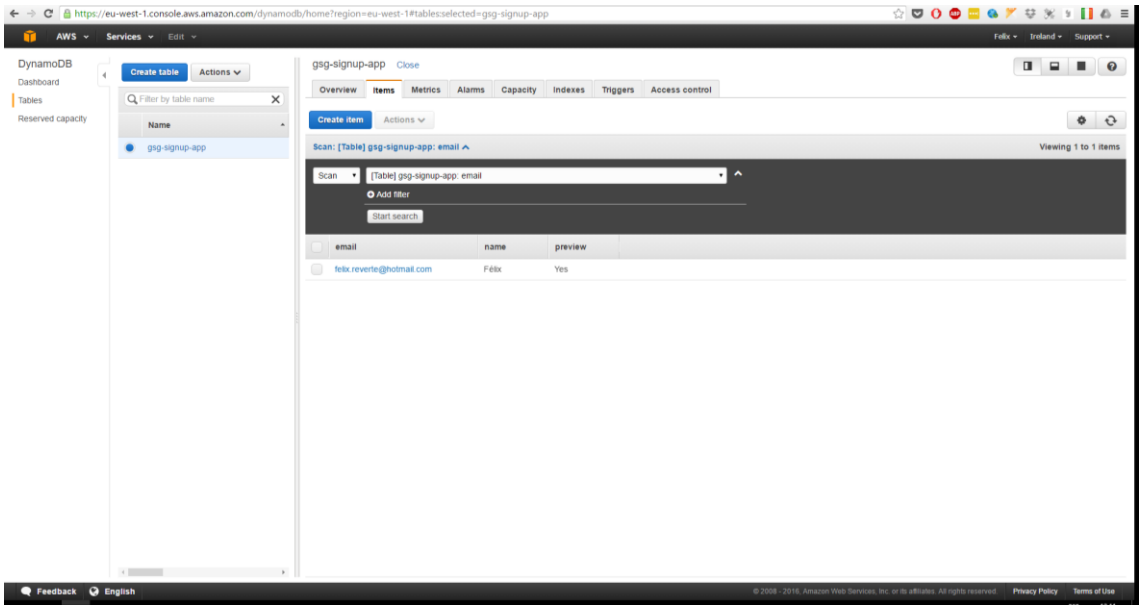
- Despliegue de la aplicación.
  1. Preparación de los ficheros a desplegar. Se deben seleccionar todos los ficheros, no la carpeta que contiene al proyecto, y comprimirlos en un fichero *.zip* o *.war*.
     En el fichero *package.json* se define el script que ejecuta la aplicación y las dependencias necesarias. En este caso además de NodeJS se necesita Jade y el sdk de AWS.
  2. Abrir la consola de Elastic Beanstalk en https://console.aws.amazon.com/elasticbeanstalk/
  3. Seleccionar **Create New Application**.
  4. En la página **Application Information**, introducir el nombre de la aplicación. Seleccionar **Next**.
  5. En la página de **New Environment**, seleccionar **Create web server**.
  6. En **Environment Type**, seleccionar en **Predefined Configuration** el servidor **Node.js** y clicar en **Next**.
  7. En **Application Version**, seleccionar **Upload your own**, seleccionar el fichero generado en el paso 1 y clicar en **Next**.
  8. En **Environment Information**, introducir un nombre de entorno y una URL de entorno única. Seleccionar **Check availability** para comprobar la disponibilidad del nombre introducido y clicar en **Next**.
  9. En **Configuration Details**, usar los valores por defecto para todas las opciones y clicar **Next**.
  10. En **Environment Tags**, clicar en **Next**.
  11. En la página de **Permissions**, aplicar en **Instance profile** el rol que se creó. No cambiar el valor de **Service role**.
  12. Seleccionar **Next**.
  13. En la página de **Review**, verificar las opciones, y seleccionar **Launch**. En el panel se puede ver en tiempo real como Elastic Beanstalk crea un entorno y despliega la aplicación. Este proceso toma unos minutos.

- Prueba de la aplicación.
  La url generada para la aplicación es: http://mystartupapp-env.eu-west-1.elasticbeanstalk.com/

Tras enviar el formulario se muestra la página principal con un mensaje informativo y se recibe un email indicando la suscripción, que se puede ver en la siguiente captura:



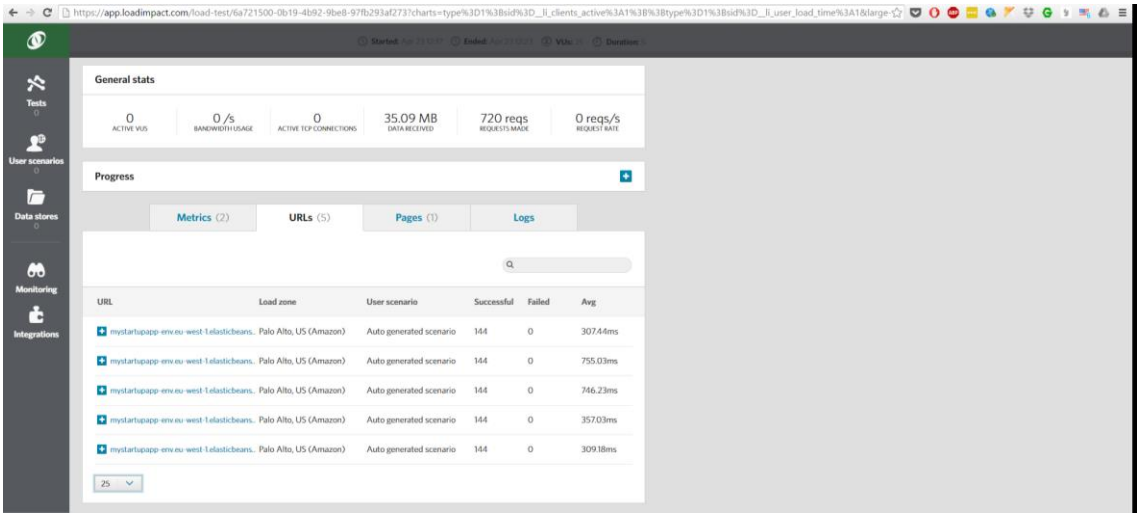
En la tabla de DynamoDB se pueden ver los datos introducidos por el usuario registrado:

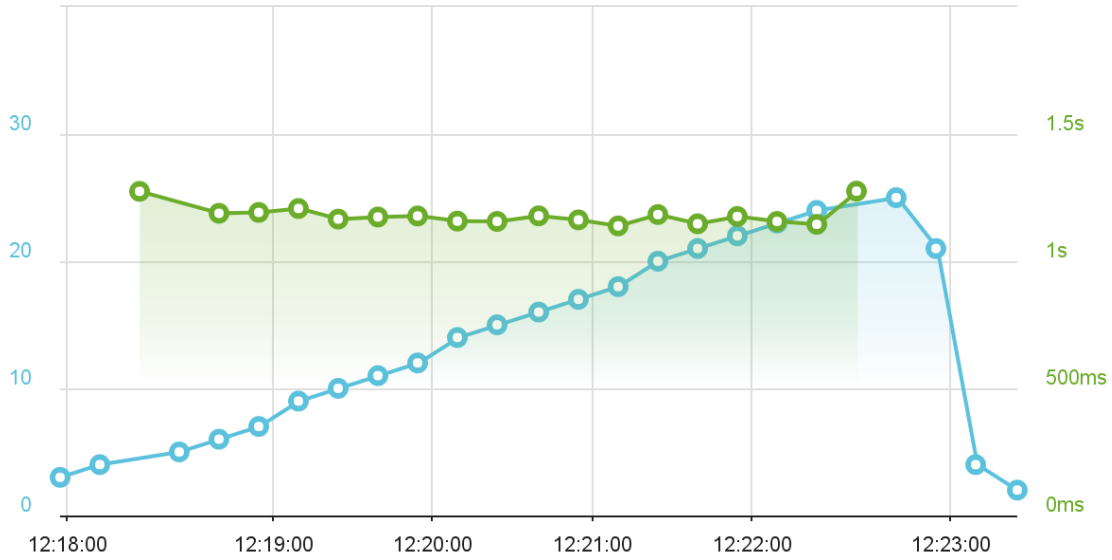


- Test de carga.

  Se realizó un test de carga en la aplicación https://loadimpact.com/ , donde se introduce la URL de nuestra aplicación y ejecuta múltiples peticiones simultaneas de usuarios virtuales (VUs, virtual users).

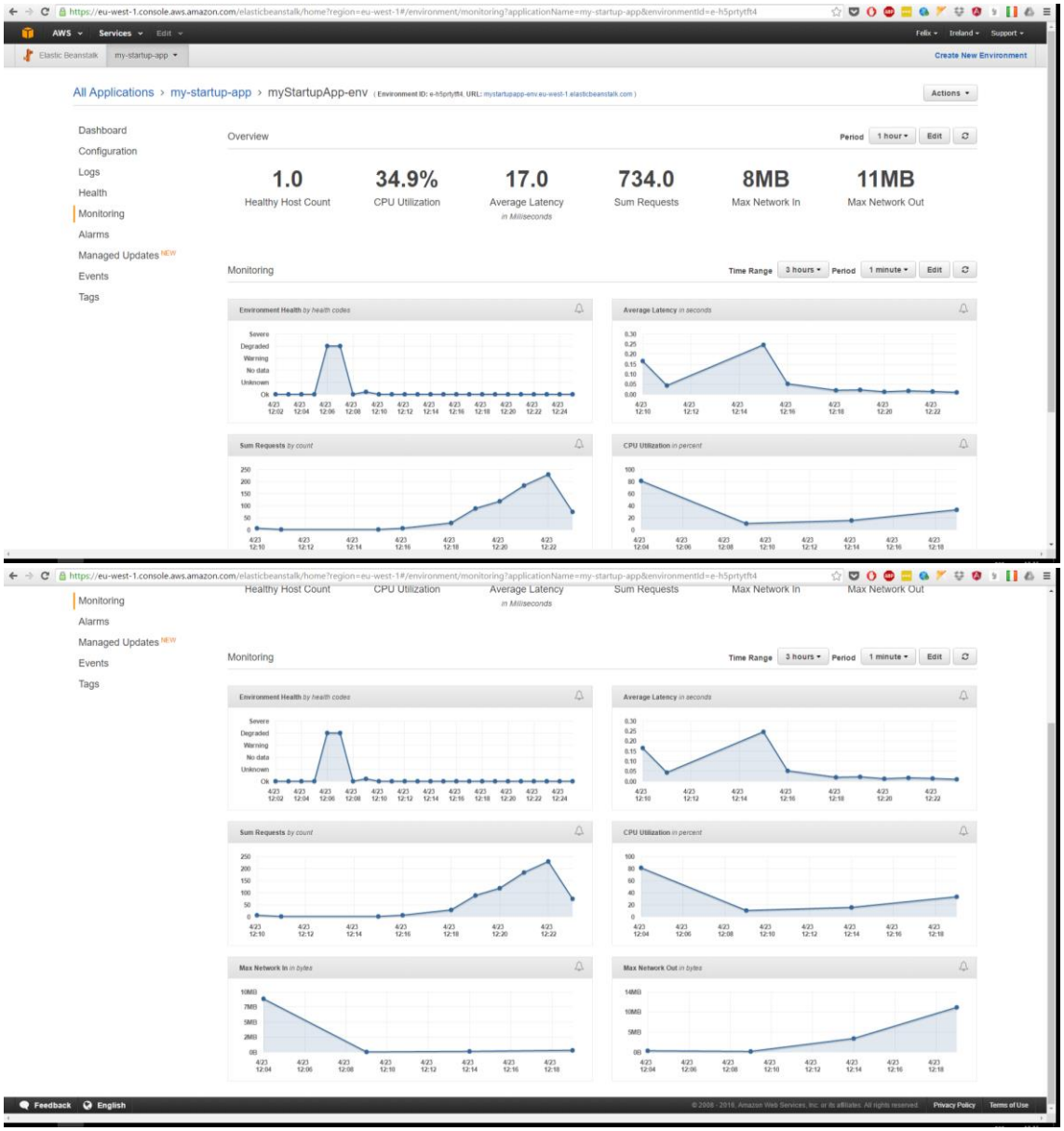
  A continuación, se muestran los resultados obtenidos en los que se aprecia como al aumentarse el número de peticiones concurrentes a la aplicación el tiempo de respuesta o latencia se sigue manteniendo prácticamente constante, habiéndose realizado 720 peticiones en un periodo de 5 minutos.





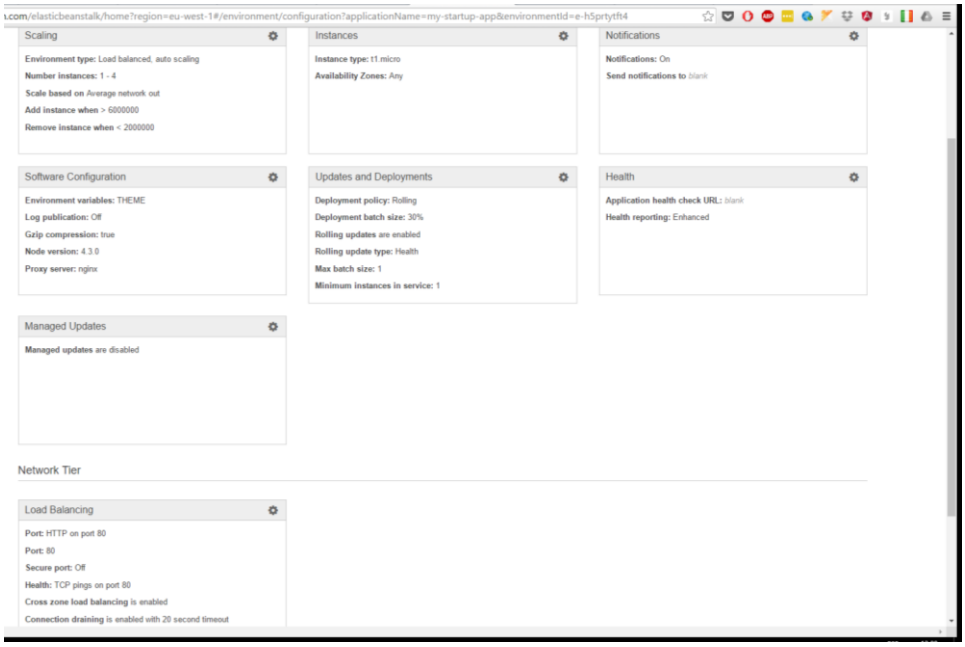
La gráfica superior representa con los puntos verdes el tiempo de carga para los usuarios, mientras que con los puntos azules representan los usuarios concurrentes. Como se aprecia, el tiempo de carga se mantiene constante a medida que el número de usuarios crece.

En el panel de monitorización de Elastic Beanstalk se puede ver el impacto de la prueba. Como se puede ver tanto en la gráfica de la prueba como en la de *Sum Requests* de AWS sobre las 12:22h se alcanza el máximo de peticiones, la CPU no alcanza el 40% y la latencia media es baja.



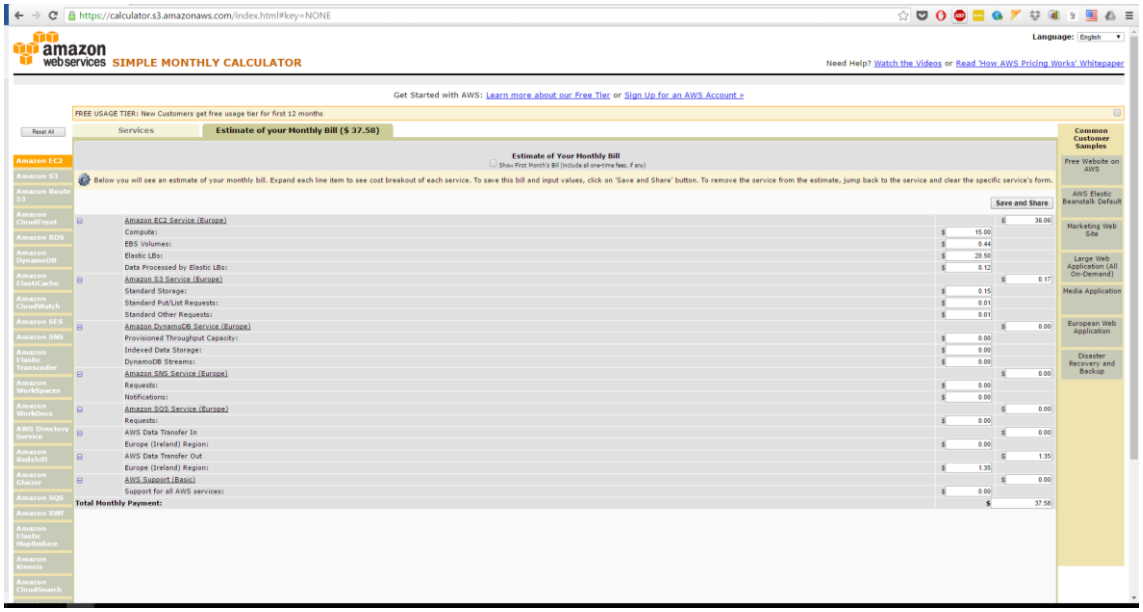
- Balanceo.

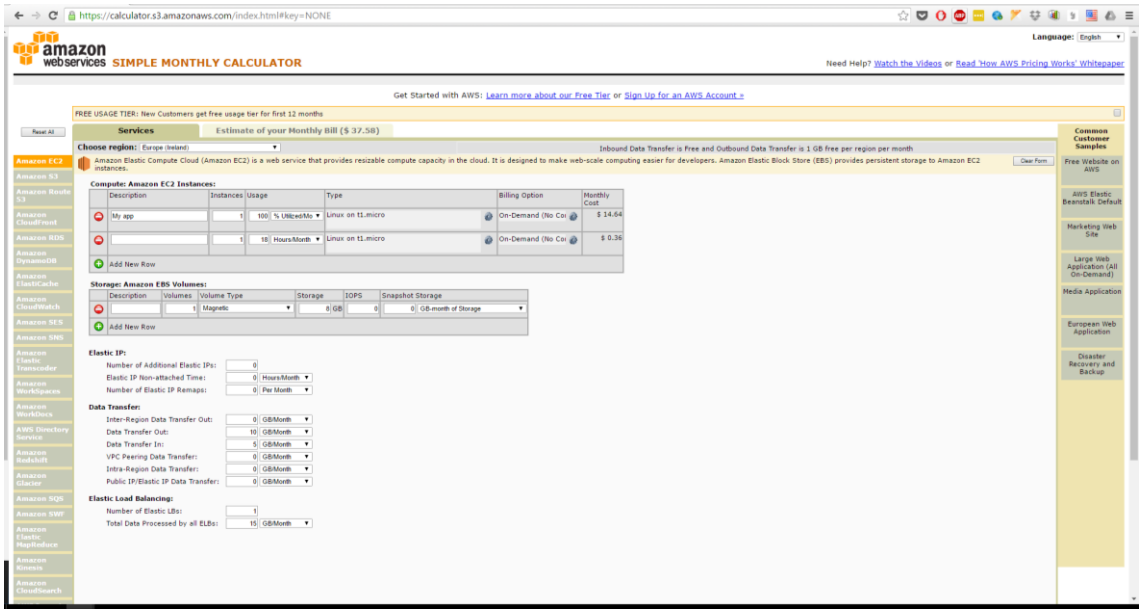
  La configuración de balanceo es la que proporciona por defecto Elastic Beanstalk (configuración mostrada en la captura siguiente), pudiendo personalizarla. Pero, en este caso, con 4 instancias en total es más que suficiente para cargas inesperadas en un supuesto éxito de nuestra aplicación.

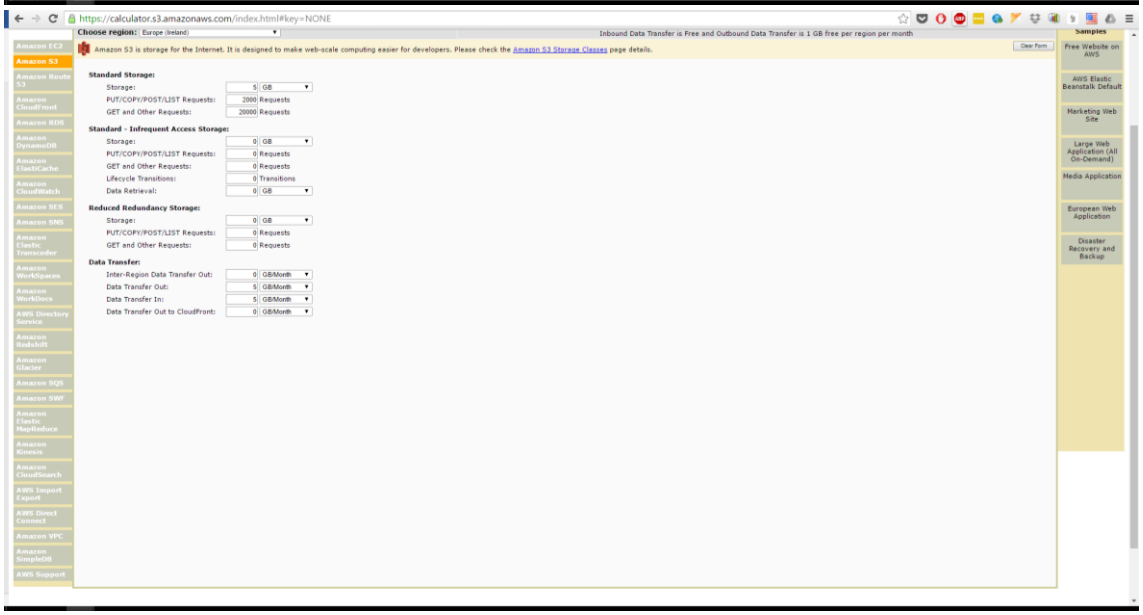


- Costes.
  El cálculo de costes se ha realizado sin tener en cuenta el descuento de capa gratuita, es decir, es coste real. El coste total es de 37.58$/mes (32.93€/mes) con servicios contratados en la región de Irlanda. En el siguiente enlace se muestra el coste calculado:
  https://calculator.s3.amazonaws.com/index.html#key=calc-FDAE0CDE-9E8C-4E1B-BA89-0D422CF8B132

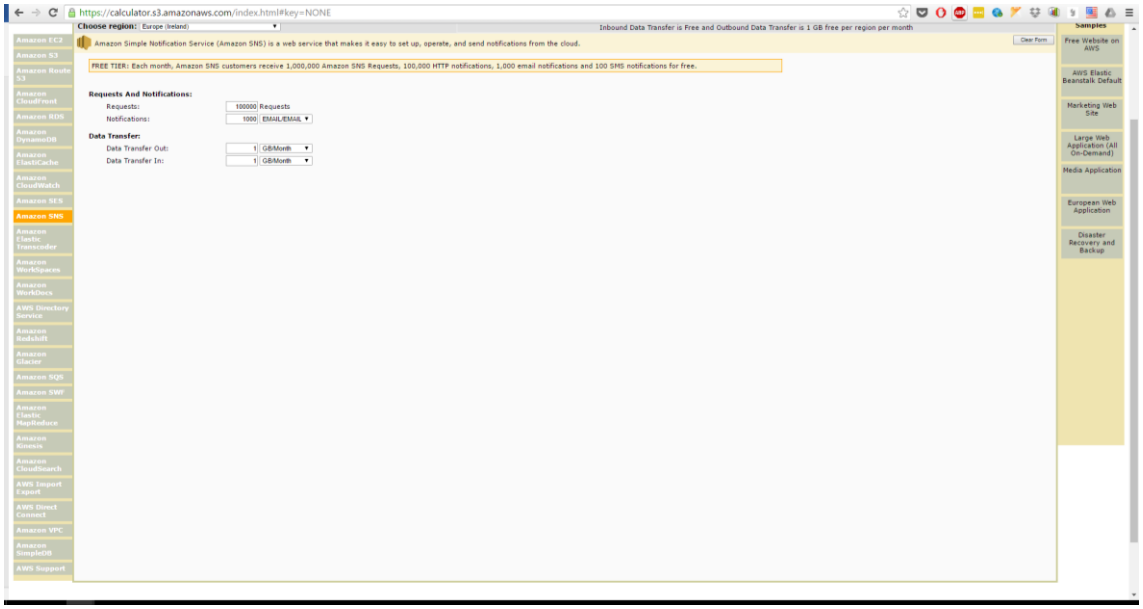
  La máquina utilizada es una instancia *t1.micro* que consta de CPU de 1 núcleo compartido y 0.6 GB de RAM donde se ejecuta Ubuntu 14.04 LTS.
  Además, en las siguientes capturas se muestra el desglose de los componentes utilizados, su capacidad y su coste.




https://calculator.s3.amazonaws.com/index.html#key=NONE
amazon web services SIMPLE MONTHLY CALCULATOR
Need Help? Watch the Videos or Read 'How AWS Pricing Works' Whitepaper
Get Started with AWS: Learn more about our Free Tier or Sign Up for an AWS Account »
FREE USAGE TIER: New Customers get free usage tier for first 12 months
Services
Estimate of your Monthly Bill ($ 37.58)
Choose region: Europe (Ireland)
Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Amazon Elastic Block Store (EBS) provides persistent storage to Amazon EC2 instances.
Compute: Amazon EC2 Instances:
Description
Instances
Usage
Type
Billing Option
Monthly Cost
My app
Linux on t1.micro
On-Demand (No Co
$ 14.64
Hours/Month
Linux on t1.micro
On-Demand (No Co
$ 0.36
Add New Row
Storage: Amazon EBS Volumes:
Description
Volumes
Volume Type
Storage
IOPS
Snapshot Storage
Magnetic
GB
GB-month of Storage
Add New Row
Elastic IP:
Number of Additional Elastic IPs:
Elastic IP Non-attached Time:
Hours/Month
Number of Elastic IP Remaps:
Per Month
Data Transfer:
Inter-Region Data Transfer Out:
Data Transfer Out:
Data Transfer In:
VPC Peering Data Transfer:
Intra-Region Data Transfer:
Public IP/Elastic IP Data Transfer:
GB/Month
Elastic Load Balancing:
Number of Elastic LBs:
Total Data Processed by all ELBs:
Common Customer Samples
Free Website on AWS
AWS Elastic Beanstalk Default
Marketing Web Site
Large Web Application (All On-Demand)
Media Application
European Web Application
Disaster Recovery and Backup

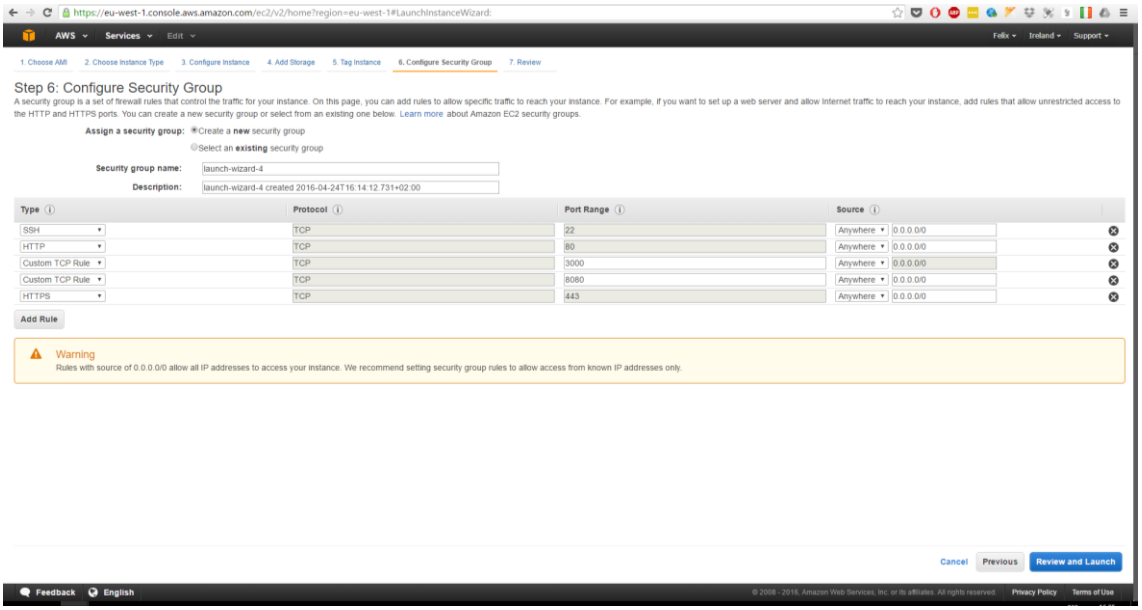

https://calculator.s3.amazonaws.com/index.html#key=NONE
Choose region: Europe (Ireland)
Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month
Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers. Please check the Amazon S3 Storage Classes page details.
Standard Storage:
Storage:
PUT/COPY/POST/LIST Requests:
Requests
GET and Other Requests:
Standard - Infrequent Access Storage:
Lifecycle Transitions:
Transitions
Data Retrieval:
Reduced Redundancy Storage:
Data Transfer:
Inter-Region Data Transfer Out:
Data Transfer Out:
Data Transfer In:
Data Transfer Out to CloudFront:
GB/Month
Samples
Free Website on AWS
AWS Elastic Beanstalk Default
Marketing Web Site
Large Web Application (All On-Demand)
Media Application
European Web Application
Disaster Recovery and Backup


https://calculator.s3.amazonaws.com/index.html#key=NONE
Choose region: Europe (Ireland)
Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month
Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.
FREE TIER: Each month, Amazon DynamoDB users pay no charges on the first 25GB of storage, the first 2.5 million DynamoDB Streams read request units, as well as 25 writes/second and 25 reads/second of ongoing throughput capacity.
Indexed Data Storage:
Dataset Size:
GB
Provisioned Throughput Capacity *:
Item Size (All attributes):
KB
Number of items read per second:
Reads/Second
Read Consistency:
Strongly Consistent
Eventually Consistent (2x cheaper)
Number of items written per second:
Writes/Second
DynamoDB Streams:
Read Request Units per month:
Units/Month
Data Transfer:
Data Transfer Out:
Data Transfer In:
GB/Month
* Every Amazon DynamoDB table has pre-provisioned the resources it needs to achieve the throughput (read/write) rate you asked for.
You are billed for these pre-provisioned resources irrespective of whether you have lower or higher throughput rate.
* You can also purchase Reserved Capacity save up to 53% with a 1-year term and up to 76% with a 3-year term. For more info visit the Amazon DynamoDB Pricing Page
Samples
Free Website on AWS
AWS Elastic Beanstalk Default
Marketing Web Site
Large Web Application (All On-Demand)
Media Application
European Web Application
Disaster Recovery and Backup

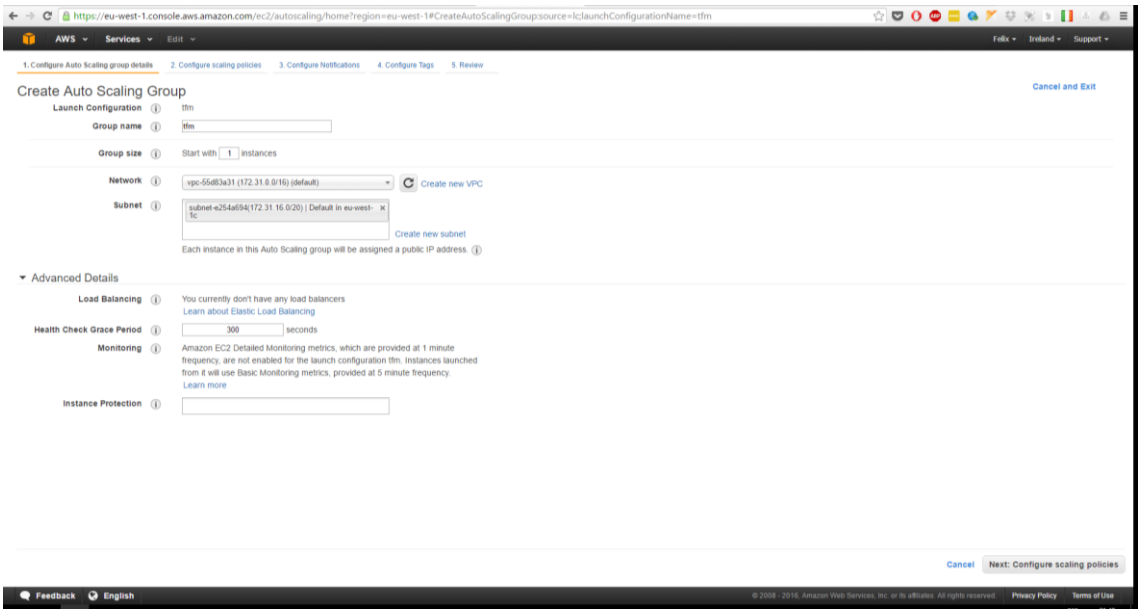


2. **Utilización de EC2 con DynamoDB y SNS.**
   En este caso ya se tiene de la configuración anterior los roles, la base de datos DynamoDB y el servicio SNS configurados, solo queda lanzar la instancia EC2 y configurarla.

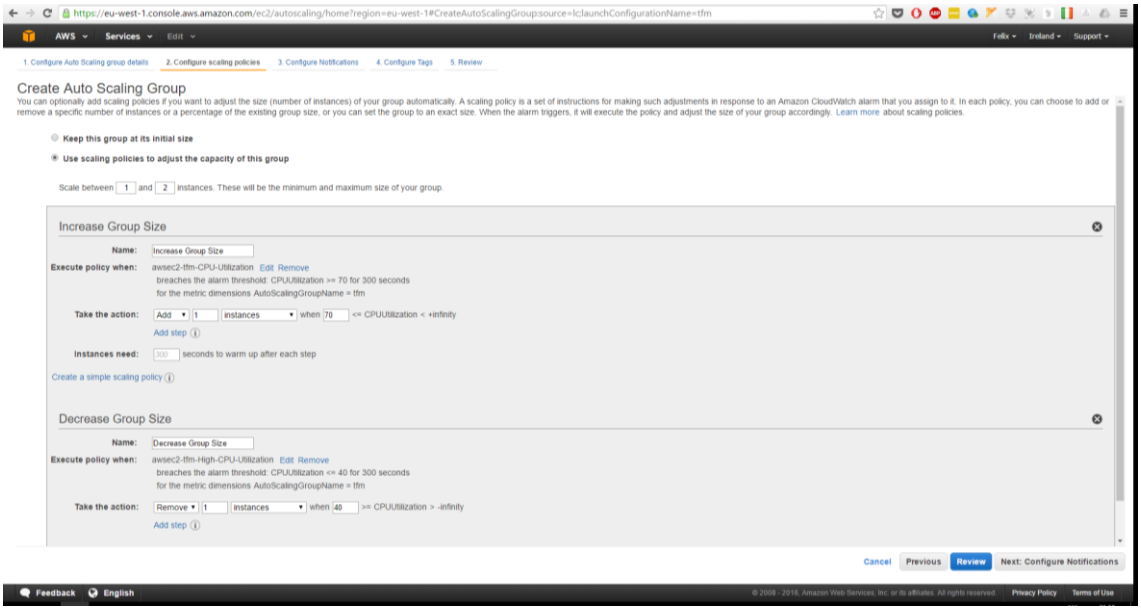
   - Creación de la instancia EC2.
     1. En la página de consola de EC2 https://eu-west-1.console.aws.amazon.com/ec2/autoscaling/home?region=eu-west-1#LaunchConfigurations , seleccionar **Create Autoscaling Group.**
     2. Seleccionar **Ubuntu**.
     3. Seleccionar la instancia marcada como **Free tier eligible**.
     4. Escribir un nombre y elegir el rol creado anteriormente.
     5. Añadir un almacenamiento dejando el que viene por defecto.
     6. Modificar los grupos de seguridad añadiendo reglas para HTTP, y puertos para el servidor ExpressJS.



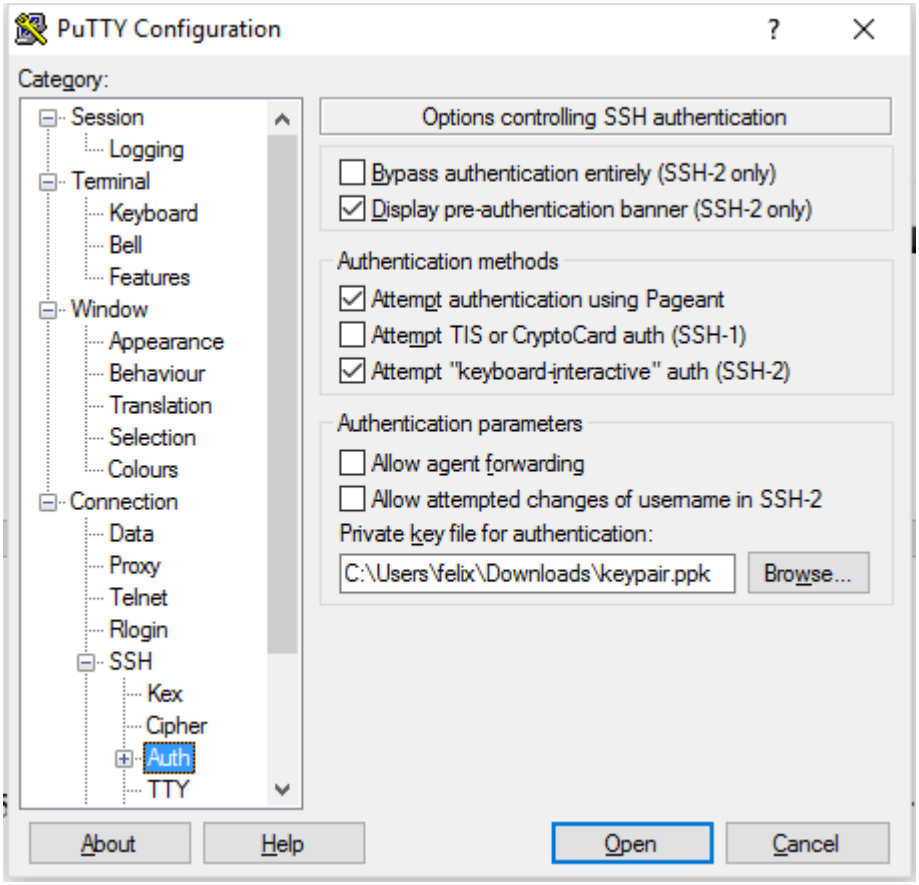
7. Seleccionamos un keypair generado anteriormente para garantizarnos acceso SSH mediante clave privada.
8. Configurar el auto-escalado con 1 instancia inicial y elegir una subred.



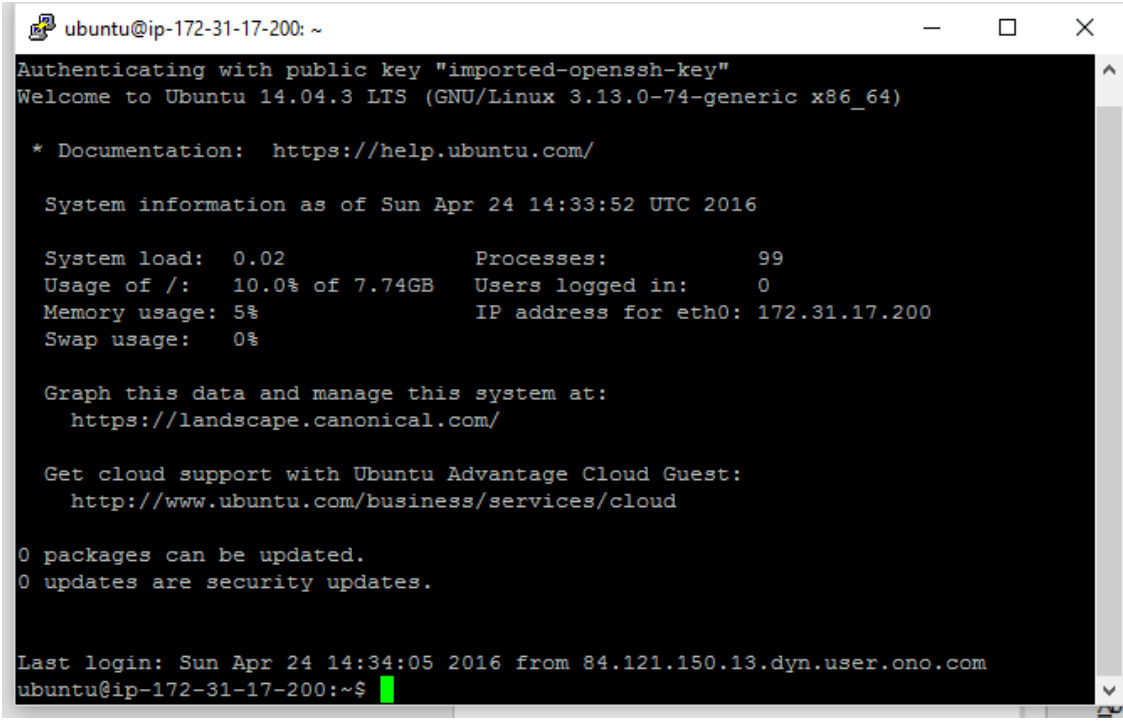
9. Configurar una alarma para encender nuevas instancias y otra para apagar instancias anteriores.



10. Añadir notificaciones.
11. Y se finaliza lanzando la instancia.
12. Instalación y ejecución.
    a. Importar la clave privada a Putty.



    b. Conectar a la instancia mediante *ubuntu@ec2-52-51-30-195.eu-west-1.compute.amazonaws.com*

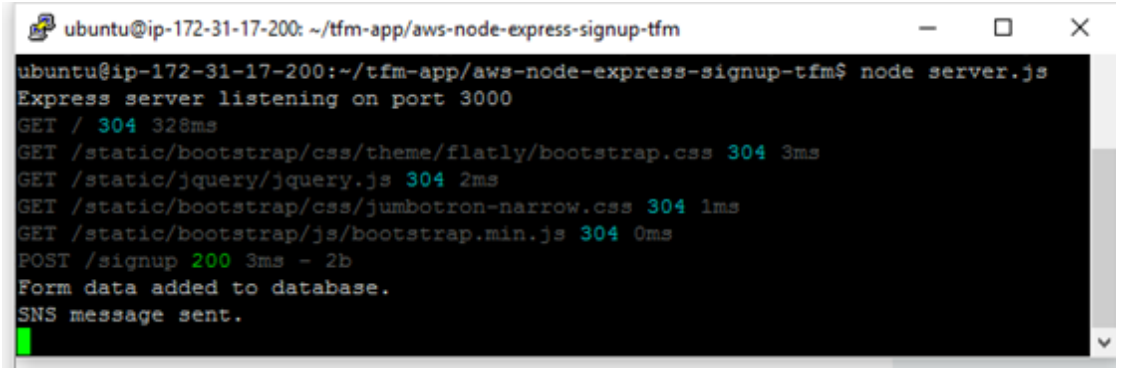


c. Comandos utilizados para actualizar, instalar y ejecutar la aplicación.

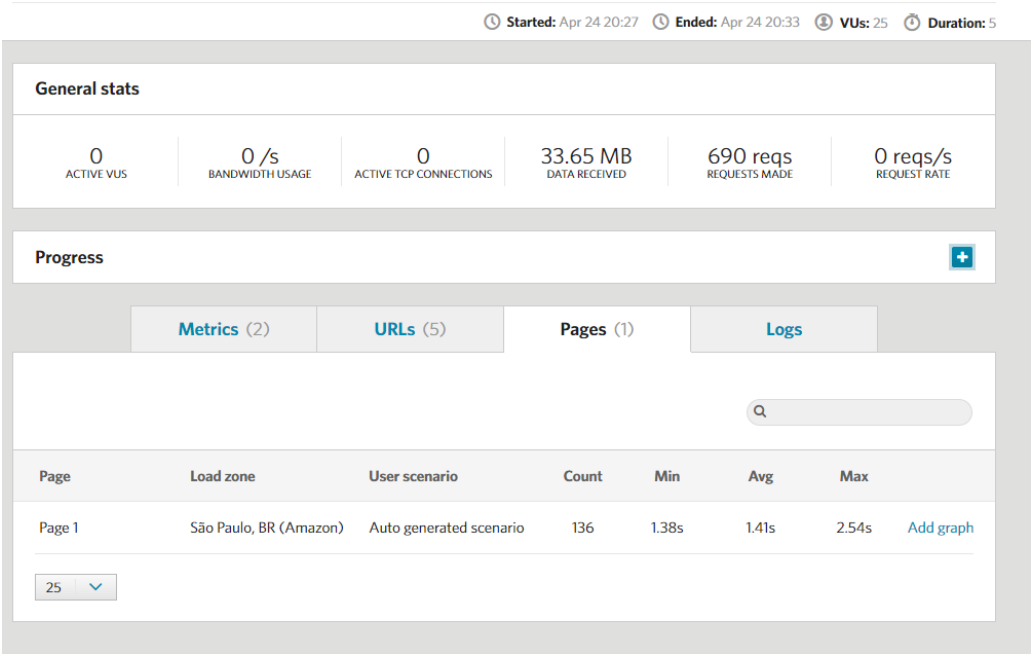
```
sudo apt-get update
sudo apt-get install nodejs –y
sudo apt-get install nodejs-legacy –y
sudo apt-get install npm –y
sudo apt-get install node-express –y
sudo apt-get install git -y
git clone https://github.com/ferrha/aws-node-express-signup-tfm.git
cd aws-node-express-signup-tfm
npm install
node server.js
```

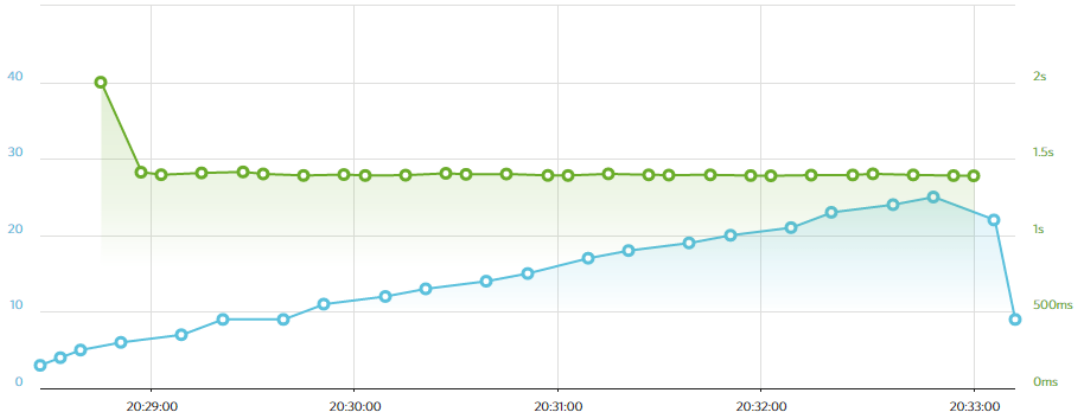
De este modo se encuentra corriendo la aplicación en la instancia.
Cabe destacar que hay que añadir a la configuración de SNS y de DynamoDB la accessKeyId y la secretAccessKey (creadas en la parte de IAM) de la cuenta de AWS utilizada para que se permita el acceso a dichos servicios.

En la siguiente captura se ve el log generado por el funcionamiento de la aplicación, en el que se aprecia cómo se realiza una petición a la raíz, se cargan los recursos estáticos como estilos CSS y scripts, un usuario realiza una suscripción y en la aplicación se añade ese usuario a la base de datos y posteriormente se envía un correo de confirmación de suscripción.



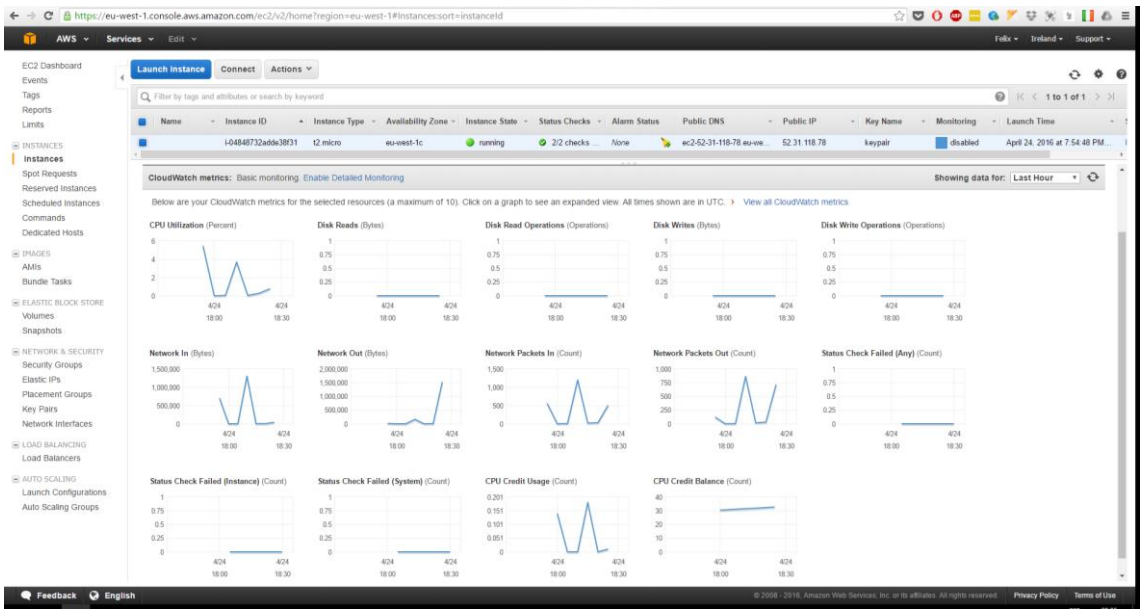
- Test de carga con https://loadimpact.com/, en el que en este caso se han realizado 690 peticiones a la aplicación notando que a medida que aumentan las peticiones el tiempo de carga sigue manteniéndose constante.





En la gráfica superior se muestran en azul los usuarios virtuales simultaneos y en verde el tiempo de carga de la aplicación. Como se puede apreciar se

mantiene constante el tiempo de carga a medida que se incrementa el número de usuarios realizando peticiones a la aplicación.



En la imagen de arriba se puede ver el comportamiento de la instancia. La utilización de CPU es inferior al caso anterior con Elastic Beanstalk.

- Coste.

  El mismo coste que para el caso con Elastic Beanstalk ya que se utilizan los mismos servicios. La única diferencia es que Elastic Beanstalk configura automáticamente el escalado de instancias mientras que con EC2 se debe configurar manualmente y en este caso la máquina utilizada es una instancia *t2.micro* que consta de CPU de 1 núcleo compartido y 1 GB de RAM donde se ejecuta Ubuntu 14.04 LTS.

  El escalado en Elastic Beanstalk por defecto viene dado por 1 instancia siempre en ejecución y 4 instancias máximas, se generan nuevas instancias cuando el tráfico de red de salida supera los 6000000 bytes y se destruyen cuando es inferior a los 2000000 bytes, aunque se puede configurar según CPU y valores máximos, mínimos o sumatorios, además de por valores medios.

4.1.2. Google Cloud Platform

En la plataforma de Google se dispone también de dos soluciones posibles, una utilizando App Engine Flexible Environment, que es el equivalente a Elastic Beanstalk, y otra, utilizando Compute Engine, que es el equivalente a las instancias EC2.

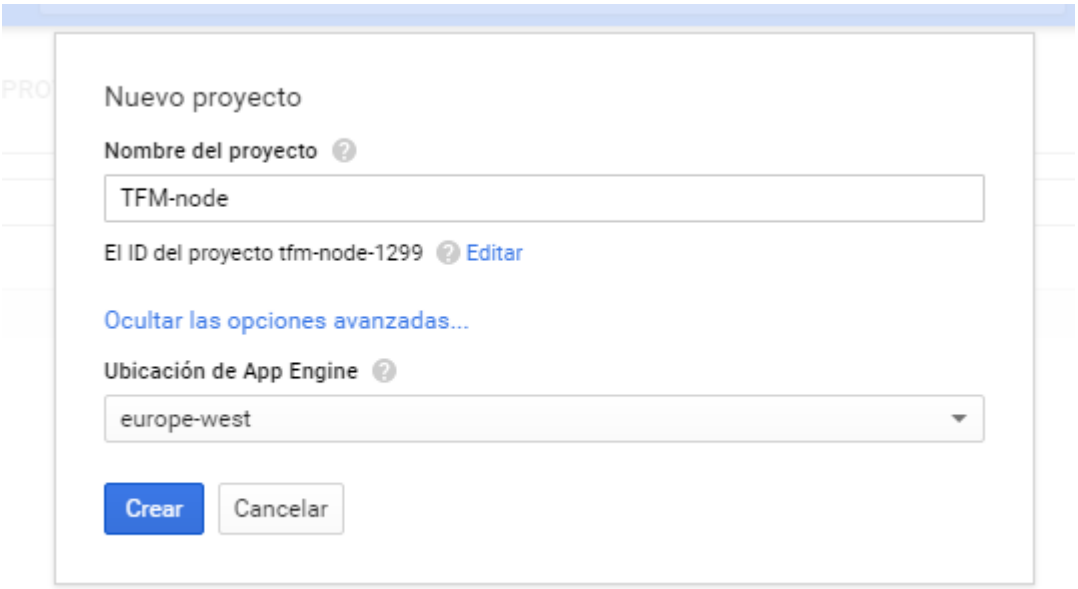
Las diferencias entre estos dos sistemas radican básicamente en el tiempo de aprovisionamiento, el escalado, el tiempo entre peticiones y la localización, estando el servicio App Engine Flexible no disponible aún en Europa y en fase Beta en Estados Unidos, por tanto, se procede a utilizar la solución utilizando Compute Engine.

En la siguiente tabla se muestran las diferencias entre el servicio App Engine (en la columna derecha) y el servicio Compute Engine (en la columna izquierda).

Feature	Standard environment	Flexible environment
Instance startup time	Milliseconds	Minutes
Maximum request timeout	60 seconds	24 hours
Background threads	Yes, with restrictions	Yes
Background processes	No	Yes
SSH debugging	No	Yes
Scaling	Manual, Basic, Automatic	Manual, Automatic
Writing to local disk	No	Yes, ephemeral (disk initialized on each VM startup)
Customizable serving stack	No	Yes (built by customizing a Dockerfile)
Automatic in-place security patches	Yes	Yes
Network access	Only via App Engine services (includes outbound sockets)	Yes
Supports installing third-party binaries	No	Yes
Location	United States or European Union	While in Beta, United States only. European Union-hosted applications should not deploy apps to the flexible environment.
Pricing	Based on Instance hours	While in Beta, based on Compute Engine Pricing for each VM. Pricing will change in the future.

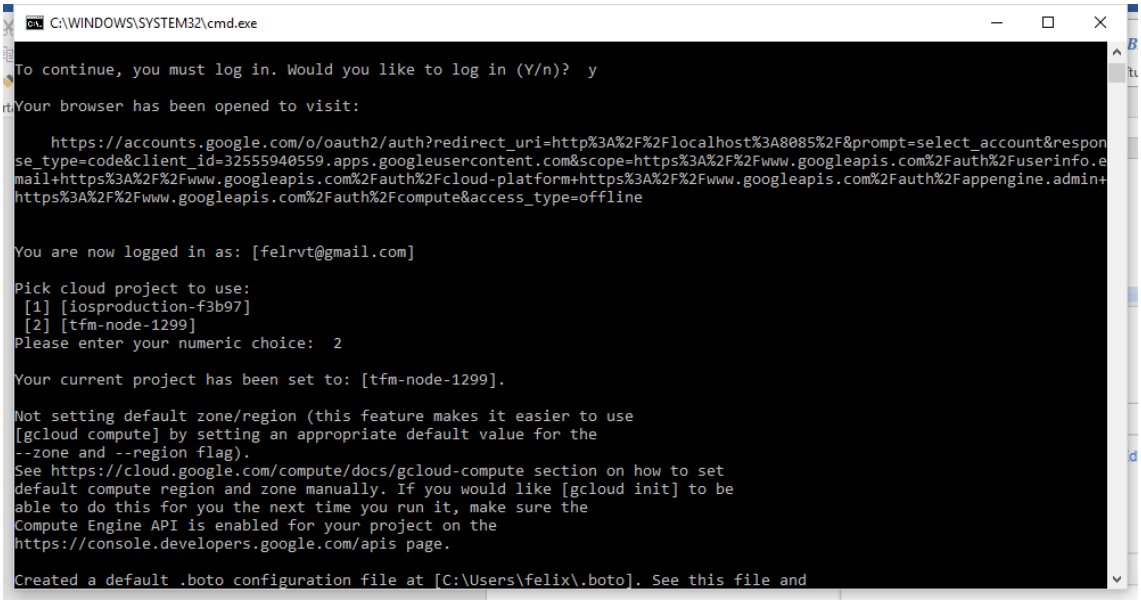
A continuación, se detallan los pasos para la ejecución de la aplicación sobre este servicio en el que, al igual que con AWS, se ha hecho uso de la capa gratuita.

- Creación de un proyecto.
  1. Abrir https://console.cloud.google.com/?_ga=1.37406944.1721234830.1458638621
  2. En el menú superior seleccionar **Crear un proyecto.**
  3. Desplegar la opción **Mostrar opciones avanzadas** y seleccionar la zona.

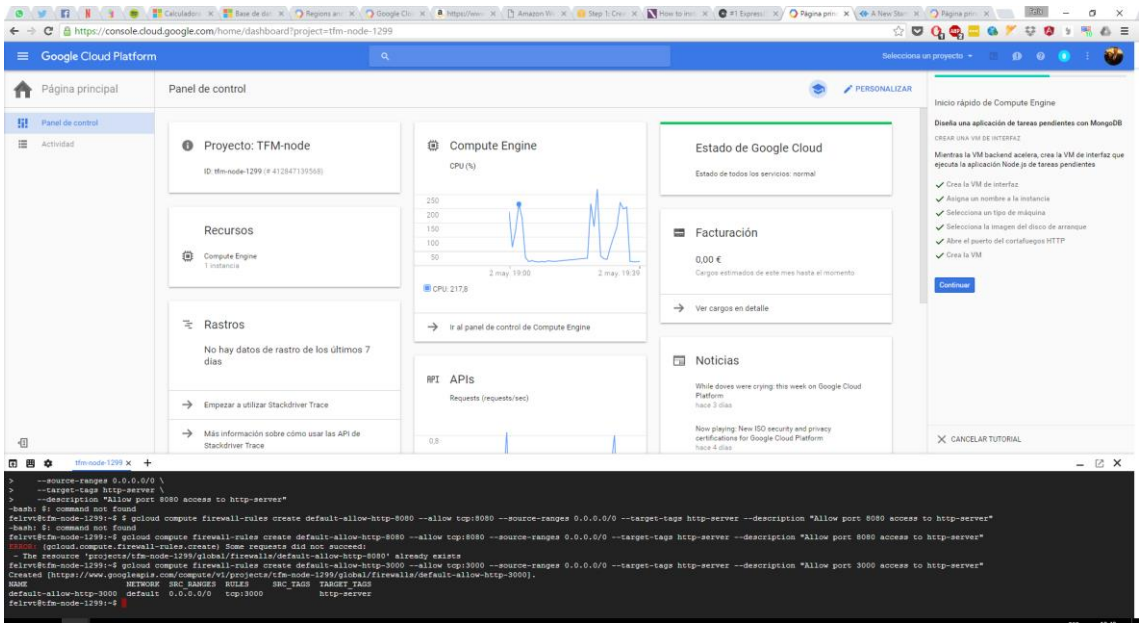


  4. Anotar el *ID del proyecto* que se utiliza con los comandos.
  5. Habilitar el cobro y elegir cuenta gratuita.

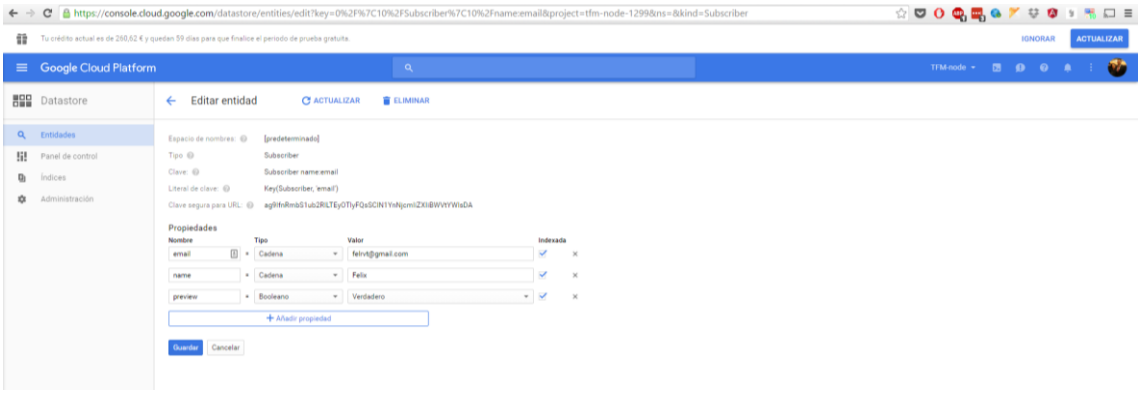
- Inicialización.
  1. Instalar *Google Cloud SDK*.
  2. Ejecutar *gcloud init*.
  3. Introducir la cuenta de Gmail en la que se encuentra el proyecto.
  4. Seleccionar el proyecto.



5. Agregar regla de firewall para permitir acceso al servidor de Express utilizando el comando:
   `gcloud compute firewall-rules create default-allow-http-3000 --allow tcp:3000 --source-ranges 0.0.0.0/0 --target-tags http-server --description "Allow port 3000 access to http-server"`



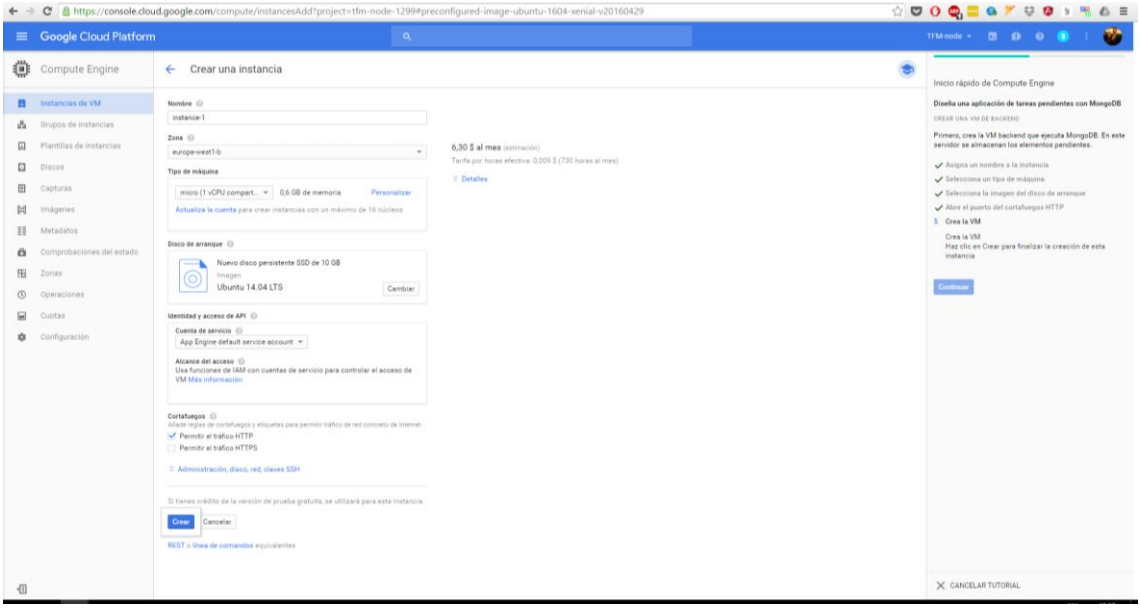
- Creación de la base de datos NoSQL con Datastore.
  1. Ir a https://console.cloud.google.com/datastore/entities/query?_ga=1.183934245.241817.1457359865
  2. Clicar en **Crear una entidad**.
  3. Usar el **Espacio de nombres** *[default]*.
  4. Escribir en el campo **Tipo** *subscriber***,** será el tipo de entidad.
  5. Añadir propiedades, en este caso las siguientes:



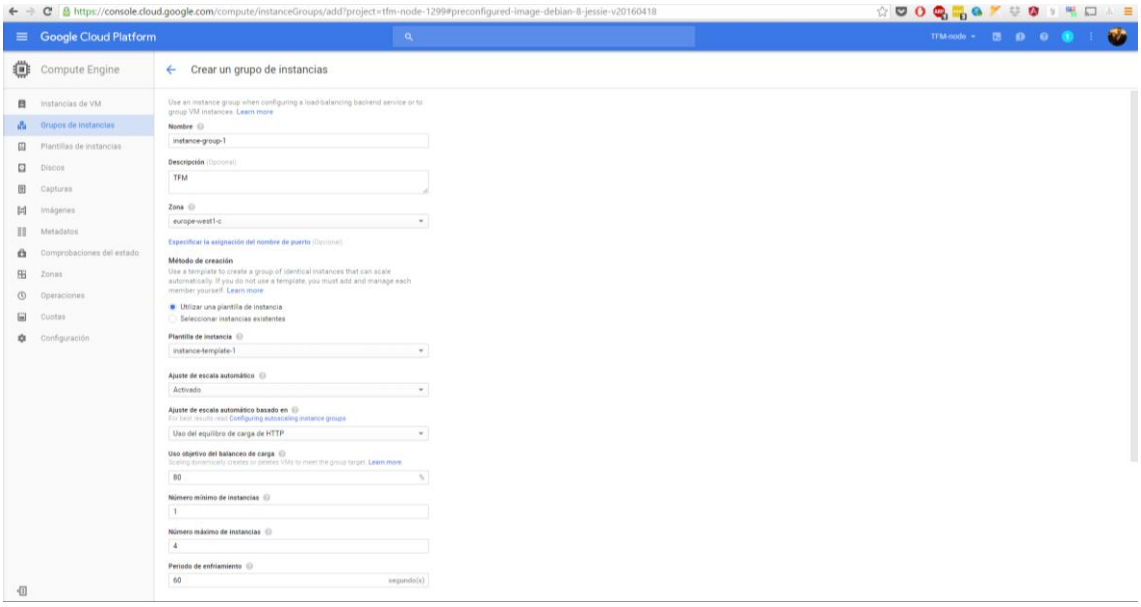
- Cambios de código.
  En este caso se han debido realizar cambios para añadir variables de entorno con el ID del proyecto para el uso de Datastore, añadiendo las dependencias a los paquetes *gcloud* y *nodemailer* al proyecto, y variables de entorno para la cuenta de Gmail desde la que se envían los correos electrónicos.

  El código se encuentra disponible en:
  https://github.com/ferrha/gc-node-express-signup-tfm.git

- Creación de la instancia inicial y el grupo de instancias.
  1. Ir a https://console.cloud.google.com/compute/instances?_ga=1.268119118.1721234830.1458638621
  2. Seleccionar el proyecto y clicar en **Continue**.
  3. Rellenar el formulario seleccionando los componentes deseados.

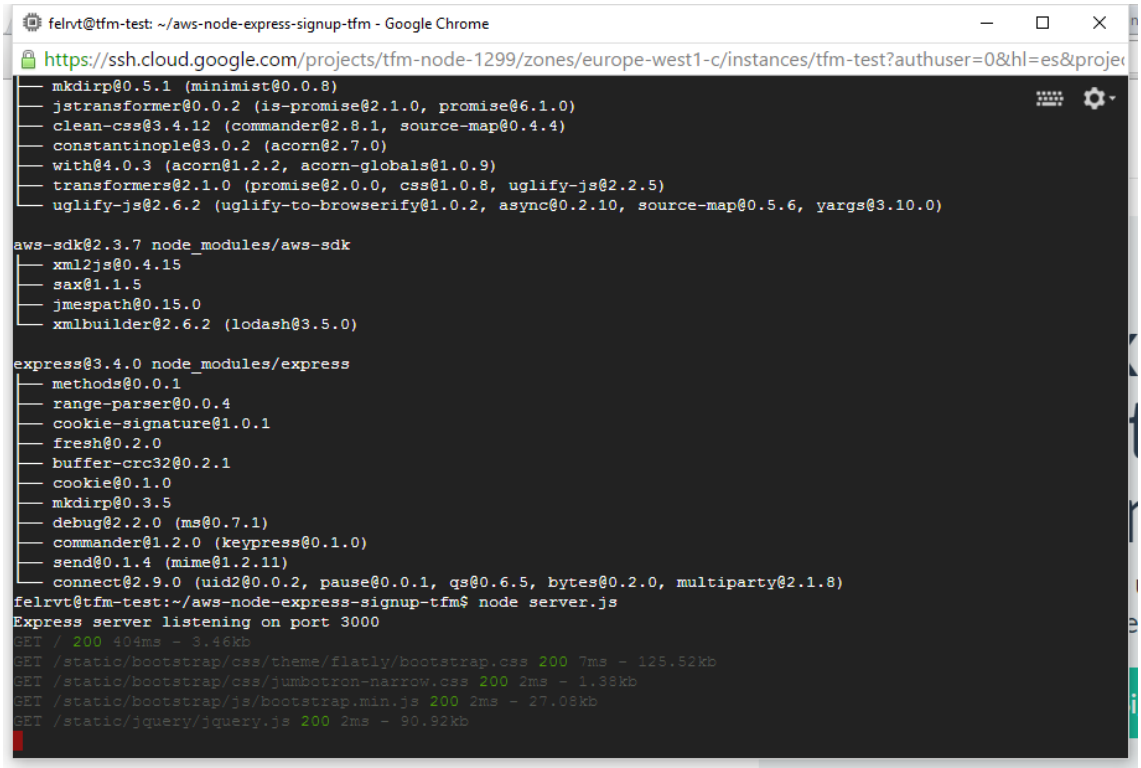


  4. Crear un grupo de instancias con la instancia anterior como plantilla e indicar los datos de escalado.



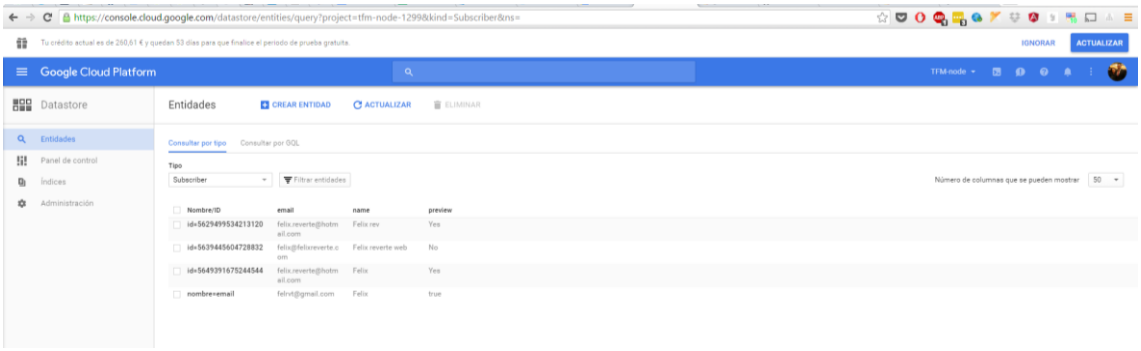
5. Ejecutar los siguientes comandos para instalar la aplicación:

```
sudo apt-get update;
sudo apt-get install -y nodejs;
sudo apt-get install -y nodejs-legacy;
sudo apt-get install -y npm;
sudo apt-get install -y node-express;
sudo apt-get install -y git;
git clone https://github.com/ferrha/gc-node-express-
signup-tfm.git
cd gc-node-express-signup-tfm
npm install
export GCLOUD_PROJECT=<project-id>
export userAccount=<gmail email>
export userPassword=<gmail password>
node server.js
```




```
Express server listening on port 3000
GET / 200 421ms - 3.46kb
GET /static/bootstrap/css/jumbotron-narrow.css 200 8ms - 1.45kb
GET /static/bootstrap/js/bootstrap.min.js 200 10ms - 27.08kb
GET /static/bootstrap/css/theme/flatly/bootstrap.css 200 16ms - 132.78kb
GET /static/jquery/jquery.js 200 11ms - 90.93kb
POST /signup 200 20ms - 2b
Service URL: https://datastore.googleapis.com/google.datastore.v1beta3.Datastore
Bearer ya29.CjLdAn8wIik7FIwDoesWFyck8UG5-1LcyrvKbOhoHVSCXJpOE4ip4rSNuxVYklx4bqJdaQ
Form data added to database.
Message sent successfully!
^C
```

En la primera captura de esta página se muestra la finalización de la instalación de todos los paquetes mediante npm y la ejecución de la aplicación. Por otro lado, en la segunda captura se ve como un usuario realiza una petición a la página principal, carga los recursos estáticos y se suscribe. La aplicación posteriormente añade los datos a la base de datos Datastore conectandose a ella y finalmente envía el mensaje de confirmación de suscripción.

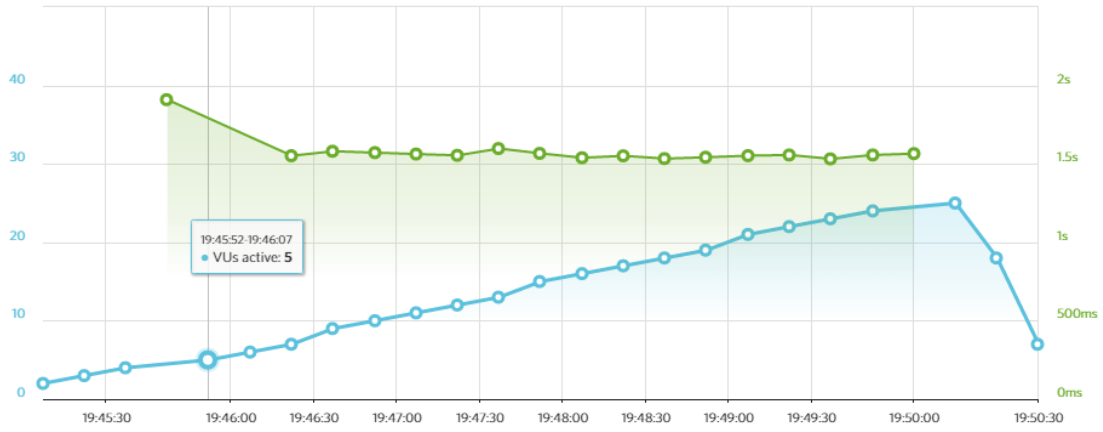


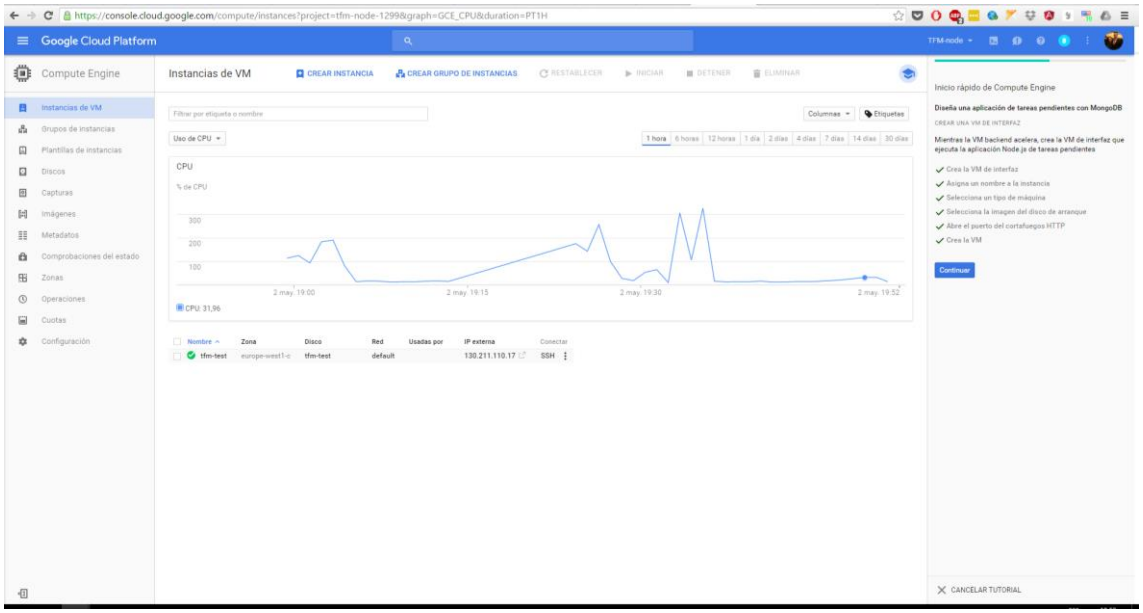
En la figura superior se ven las suscripciones realizadas que se han almacenado en la base de datos.

- Pruebas de carga.

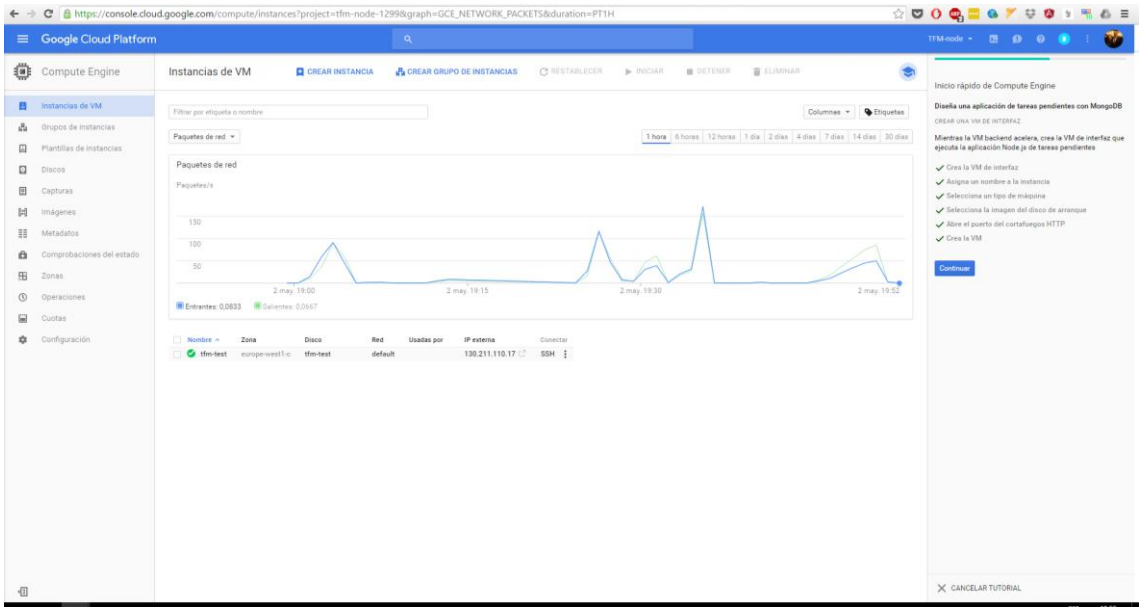
Test de carga realizado de nuevo con https://loadimpact.com/, en el que en este caso se han realizado 680 peticiones a la aplicación notando que a medida que aumentan las peticiones el tiempo de carga sigue manteniéndose constante.

En la gráfica siguiente se observan en azul el número de usuarios virtuales simultáneos generados por la aplicación de prueba de carga, mientras que en verde se muestra el tiempo de carga de nuestra aplicación, notando que a medida que el número de usuarios aumenta el tiempo de carga se mantiene prácticamente constante, aunque más elevado que para los casos de AWS.





En la imagen de arriba se puede ver el uso de CPU, de izquierda a derecha se ven los picos desde la creación de la instancia, la actualización de la misma y la instalación de la aplicación.
El uso de CPU para la hora en que se realiza la prueba no supera el 32%.



En la captura superior se aprecian los picos dados por la actividad de red en la instancia.

- Coste.

  El coste de desplegar la aplicación en Google Compute Engine viene determinada por la máquina que se elija, el escalado configurado, la base de datos NoSQL y el almacenamiento, todos estos servicios son los utilizados en la solución anteriormente desarrollada sobre AWS y se realiza el cálculo del coste con los mismos componentes.
  La máquina utilizada en este caso es una instancia *f1-micro* que consta de CPU de 1 núcleo compartido, 0.6 GB de RAM y ejecuta Ubuntu 16.04 LTS.

  Cabe destacar que la calculadora no es fiel a la configuración que luego se permite, por ejemplo, el disco local de la instancia configurada puede ser de 10GB como mínimo, mientras que en la calculadora el mínimo es de 375GB, subiendo muchísimo más el precio. En cualquier caso, el coste de utilizar este servicio es más elevado que el de utilizar AWS, rondando los 40 $/mes haciendo el cálculo con 10GB de almacenamiento SSD que se puede ver en la tabla siguiente de manera desglosada.
  Enlace con la estimación, en la que el almacenamiento de la instancia principal es de 0GB: https://cloud.google.com/products/calculator/#id=cec2afbe-6ce1-41b1-a5a8-924418a477b8

Servicio	Coste
1x Compute Engine (Balanceo)	0.16 $/mes
1x Compute Engine (Principal)	6.30 $/mes
1x Disco persistente 10GB SSD	2$/mes
1x Balanceador de carga	19$ /mes
1x Cloud Storage (5GB)	12 $/mes
1x Cloud Datastore (NoSQL)	0.19 $/mes
TOTAL	39.75 $/mes

4.1.3. Microsoft Azure

Por último, se va a probar el uso del Cloud de Microsoft utilizando su capa gratuita al igual que en los casos anteriores.

Se van a utilizar máquinas virtuales escalables y base de datos NoSQL haciendo uso del servicio DocumentDB, para el envío de emails se va a utilizar *nodemailer* en la propia aplicación como en el caso de Google Compute Engine, ya que ninguno de estos proveedores proporciona sistema de envío de mensajes, a diferencia de AWS que sí lo provee.

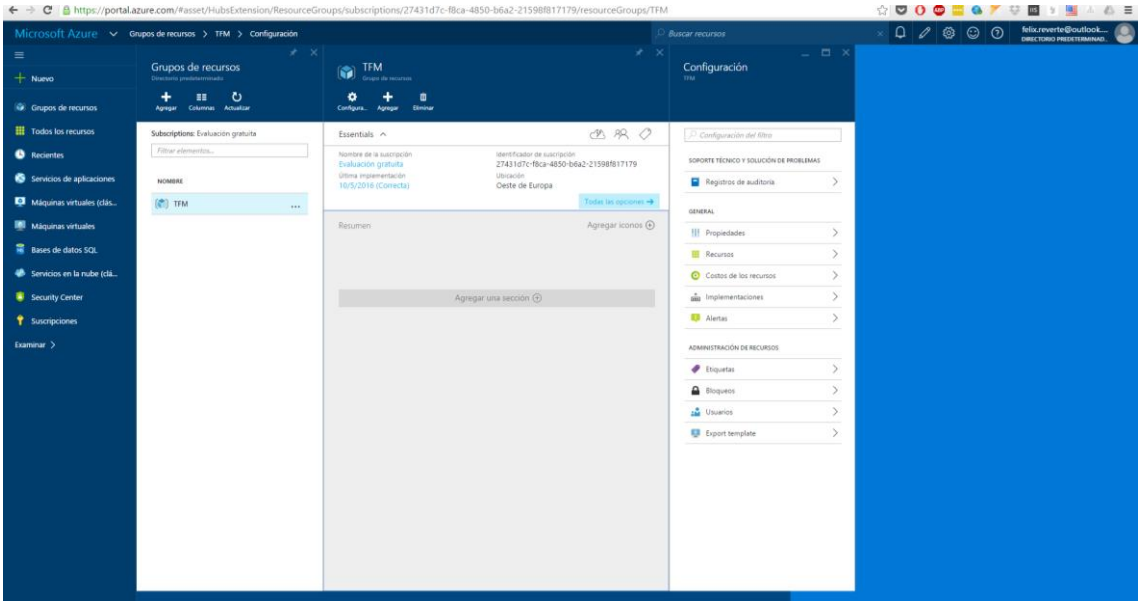
Al igual que Google Cloud, Microsoft Azure presenta unos tiempos de aprovisionamiento elevados, unos 5 minutos por instancia.

A continuación, se detallan los pasos para la ejecución de la aplicación sobre este servicio.

- Creación de un grupo de recursos.
  Con la creación de un grupo de recursos se asegura que los servicios que utilicemos se desplegarán en la ubicación seleccionada y con el tipo de suscripción elegido.

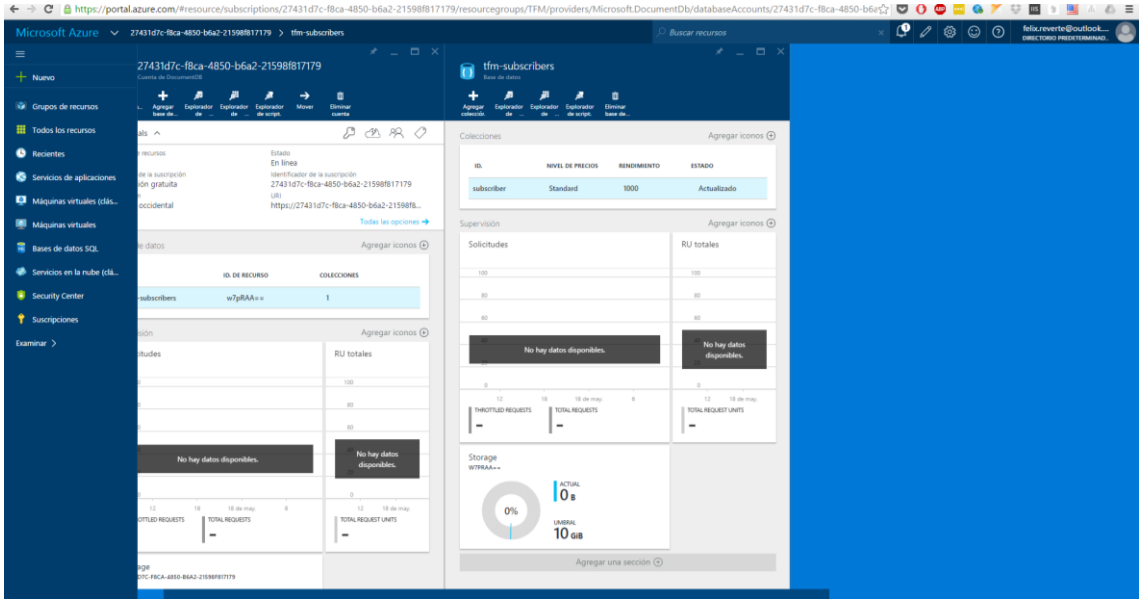
  En este caso el escalado se configura añadiendo máquinas virtuales al grupo de recursos. Se crean tantas copias de la máquina virtual original configurada como se quieran y se configura el escalado.

  1. Ir a la consola https://portal.azure.com/
  2. En la barra lateral clicar en **Grupos de recursos** y **Agregar**.
  3. Introducir un nombre, asignar evaluación gratuita a la suscripción y seleccionar la ubicación, en este caso, oeste de Europa.



- Creación la base de datos.
  1. Ir a la consola de Azure https://portal.azure.com/
  2. En la barra lateral hacer clic en **Nuevo**, en **Datos y almacenamiento** y luego en **Azure DocumentDB**.
  3. En la hoja **Nueva cuenta de DocumentDB** especificar la configuración deseada.

     En **ID** introducir un nombre para la tabla, en **Subscripción** seleccionar *Evaluación gratuita***,** y la **Ubicación** Oeste de Europa.

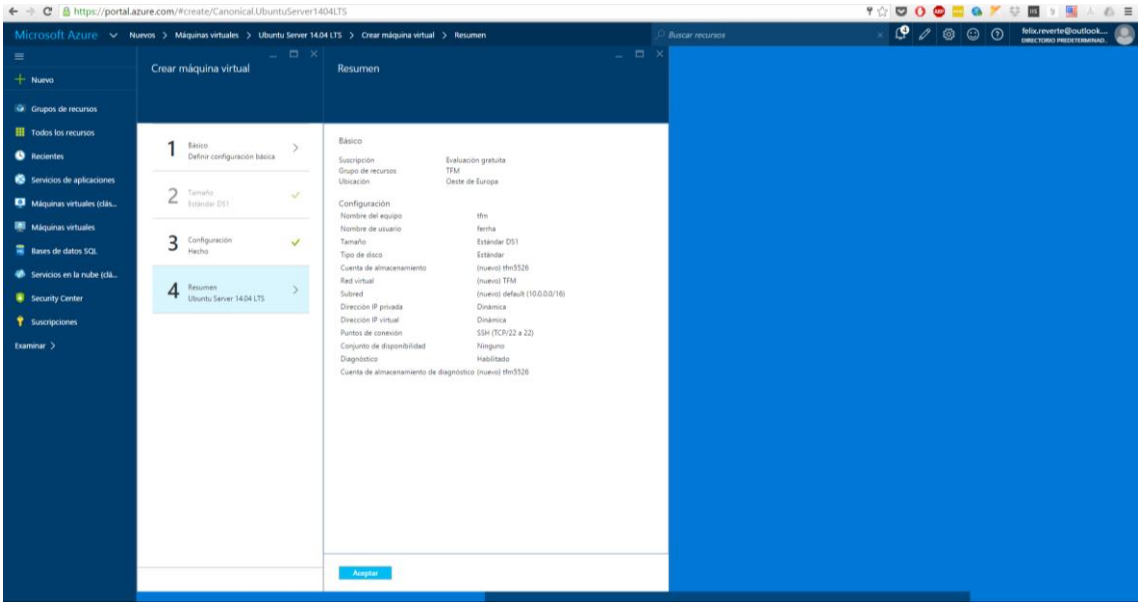


  4. Una vez creada, en **Claves** se pueden ver los datos necesarios para la conexión a la base de datos como son: la URI y las claves.
  5. Modificar el código añadiendo la siguiente configuración:

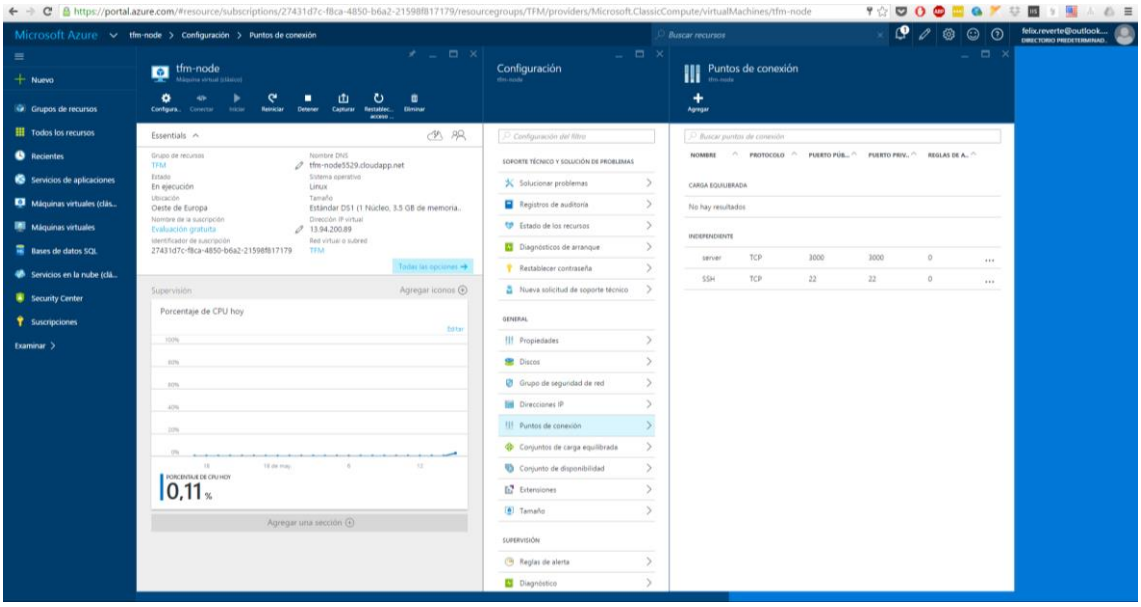
```
// config for azure
var config = {
  endpoint: process.env.dbEndpoint,
  primaryKey: process.env.primaryKey,
  database: {
    "id": "tfm-subscribers"
  },
  collection: {
    "id": "subscriber"
  }
};
```

     En la configuración anterior se añade el nombre de la base de datos, el nombre de la colección y las variables de entorno para la conexión.

- Creación de la máquina virtual.
    1. Ir a la consola https://portal.azure.com/
    2. En la barra lateral hacer clic en **Nuevo**, en **Máquinas Virtuales**, seleccionar sistema operativo, en este caso Ubuntu 14.04 LTS y **Crear máquina virtual**.
    3. Asignar un nombre a la máquina, introducir un nombre de usuario y una contraseña, asignar el tipo de suscripción a evaluación gratuita, el grupo de recursos creado anteriormente y la ubicación de oeste de Europa. Clicar en **Siguiente**.
    4. Seleccionar una máquina **DS1 Estándar** y clicar en **Seleccionar.**
    5. La configuración se selecciona **Estándar** y se dejan los valores por defecto. Aceptar.
    6. Revisar el resumen y aceptar para terminar.



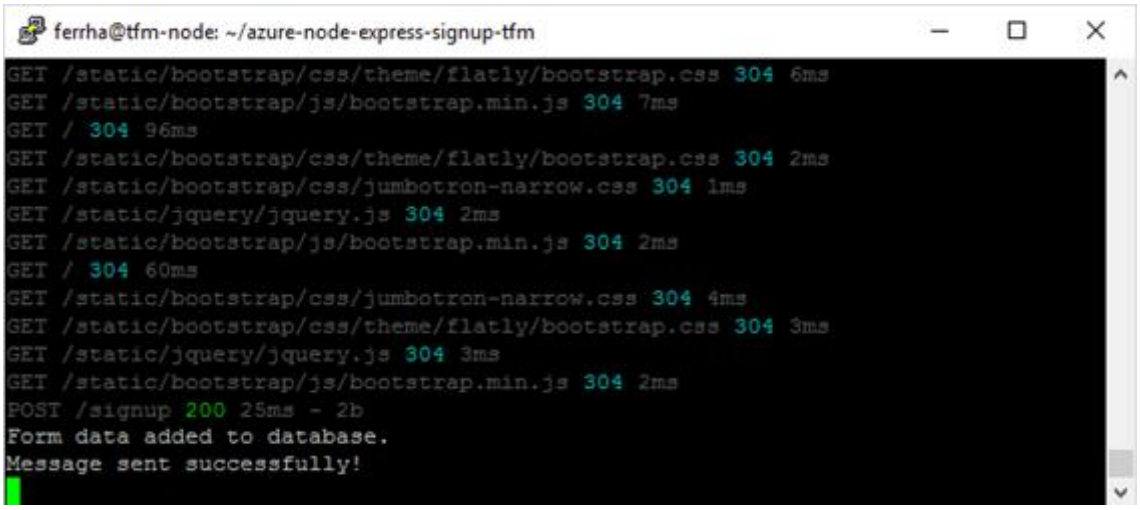
    7. Una vez lanzada la máquina virtual, ir a **Todas las opciones**, **Puntos de conexión** y agregar un nuevo punto de conexión para permitir el puerto 3000 de nuestro servidor.



8. Conectar mediante SSH a la IP de la máquina, utilizando el usuario y contraseña creados con la máquina virtual.

9. Ejecutar los siguientes comandos para actualizar e instalar los paquetes necesarios:

```
sudo apt-get update;
sudo apt-get install -y nodejs;
sudo apt-get install -y nodejs-legacy;
sudo apt-get install -y npm;
sudo apt-get install -y node-express;
sudo apt-get install -y git;
git clone https://github.com/ferrha/azure-node-express-signup-tfm.git
cd azure-node-express-signup-tfm
export userAccount=<Gmail email>
export userPassword=<Gmail email password>
export dbEndpoint=<DocumentDB URI>
export primaryKey=<DocumentDB primaryKey>
npm install
node server.js
```



En la imagen superior se puede ver la ejecución de la aplicación, en la que se cargan los recursos estáticos, se realiza el envío de los datos del formulario por parte de un usuario, se añaden a la base de datos y se envía el mensaje confirmando la suscripción.

- Cambios de código.

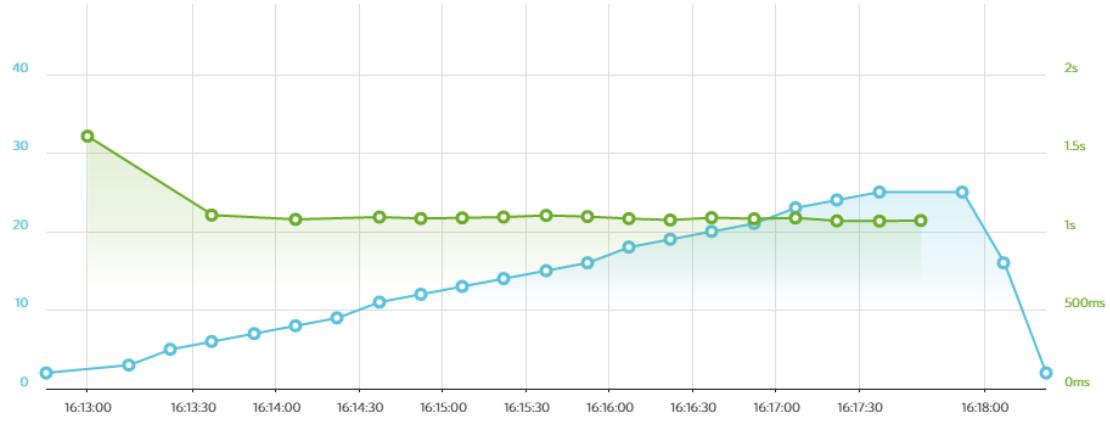
  En este caso se han debido realizar cambios para añadir variables de entorno con el endpoint y la clave de acceso para el uso de DocumentDB, añadiendo *documentdb* y *nodemailer* al proyecto, y variables de entorno para la cuenta de Gmail desde la que se envían los correos electrónicos.

  El código se encuentra disponible en:
  https://github.com/ferrha/azure-node-express-signup-tfm.git

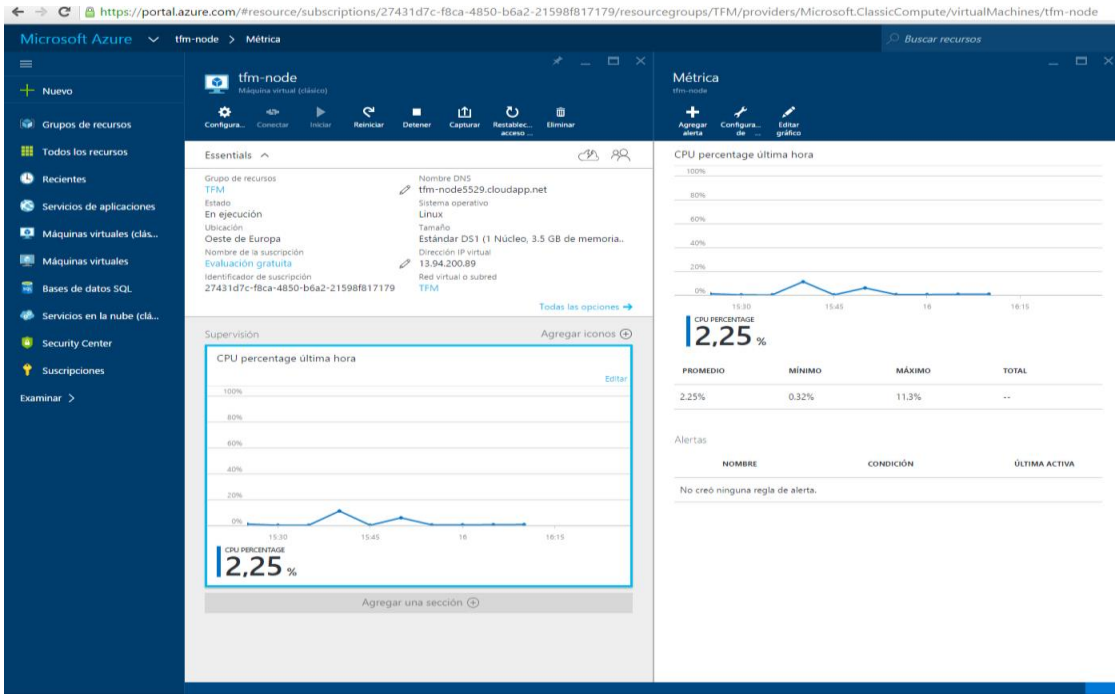
- Pruebas de carga.

  Test de carga realizado al igual que en los casos anteriores con https://loadimpact.com/, en el que en este caso se han realizado 710 peticiones a la aplicación.

  En la siguiente gráfica se pueden ver representados por un lado los puntos azules como los usuarios virtuales activos y por otro lado los puntos verdes como el tiempo de carga. Como se puede apreciar, el tiempo de carga se mantiene prácticamente constante a medida que el número de usuarios crece.



  En la siguiente captura se muestra el uso de CPU de la instancia mientras se hacía la prueba, se puede ver como el máximo uso de CPU es del 11.3%, teniendo de media un 2.25% de uso.



- Coste.

  El coste de este servicio viene dado por las máquinas virtuales escalables y la base de datos en la siguiente tabla, que genera la calculadora de Azure https://azure.microsoft.com/es-es/pricing/calculator/ , siendo el coste muy superior al de los servicios anteriores, más de 73€/mes.

  La máquina utilizada consta de una instancia *d1* con 3.5 GB de RAM y un CPU de 1 núcleo compartido que ejecuta Ubuntu 14.04 LTS

Service type	Custom name	Region	Description	Estimated Cost
Virtual Machine Scale Sets	Virtual Machine Scale Sets	West Europe	1 standard máquinas virtuales de tipo linux, d1 de tamaño	€52,70
DocumentDB	DocumentDB	West Europe	1 DocumentDB del nivel s1, 744 horas	€21,08
Support			Free level	€0,00
			Monthly Total	**€73,78**

4.2. No Cloud

En esta sección se van a explicar las soluciones ejecutadas sobre servicios No Cloud, utilizando el mismo código de aplicación para todas y eligiendo equipos con las mismas prestaciones. Por ello, solo se va a ejecutar la aplicación en un equipo, asumiendo mismos resultados en el resto, pero sí comparando precios de todas las opciones.

El código utilizado para esta sección se encuentra disponible en: https://github.com/ferrha/node-express-signup-tfm-master

Los cambios más significativos en el código son la sustitución de las bases de datos de los distintos proveedores por MongoDB, utilizando el paquete *mogoose* de npm y la inclusión de *nodemailer*, aunque ya se había utilizado en soluciones anteriores.

Para esta base de datos se ha creado el siguiente modelo:

```
var subscriberSchema = mongoose.Schema({
    email: { type: String },
    name: { type: String },
    preview: { type: Boolean }
});
```

También se debe añadir la URL de la base de datos para poder conectarla, ya sea en la propia máquina como ha sido en este caso o en un contenedor u otra máquina remota para aumentar la modularidad de la arquitectura.

```
url: "mongodb://localhost/subscriber"
```

El entorno de ejecución para la aplicación utilizada es el siguiente:

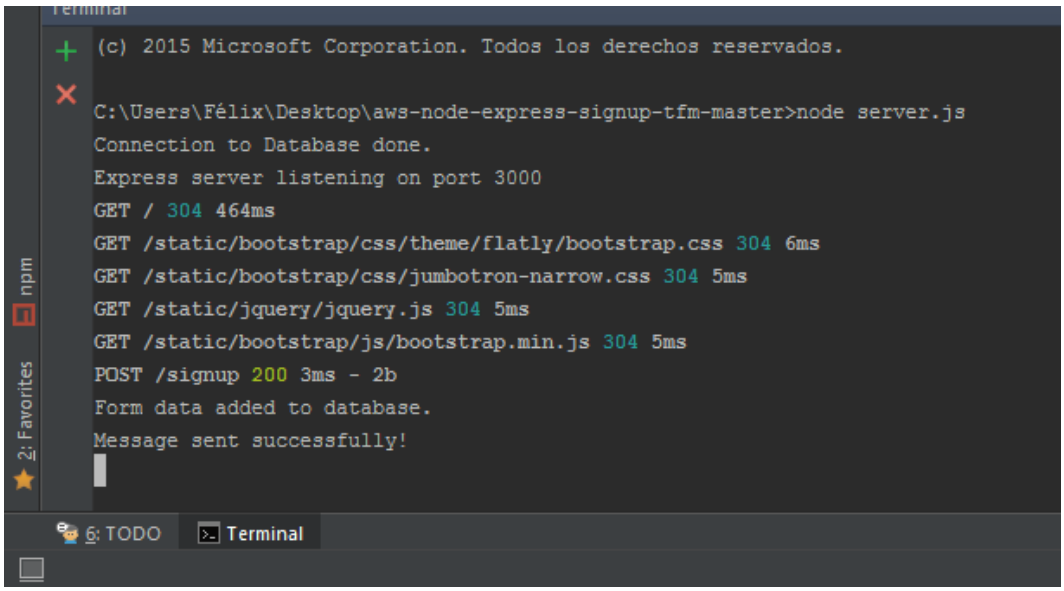
- Sistema Operativo Ubuntu 14.04 LTS o 16.04 LTS (según disponibilidad).
- NodeJS.
- Express.
- npm.
- MongoDB como base de datos NoSQL.
- Nodemailer para el envío de correos electrónicos.

Los comandos utilizados en Ubuntu para instalar todo lo necesario son prácticamente los mismos que se utilizan para la aplicación en instancias Cloud:

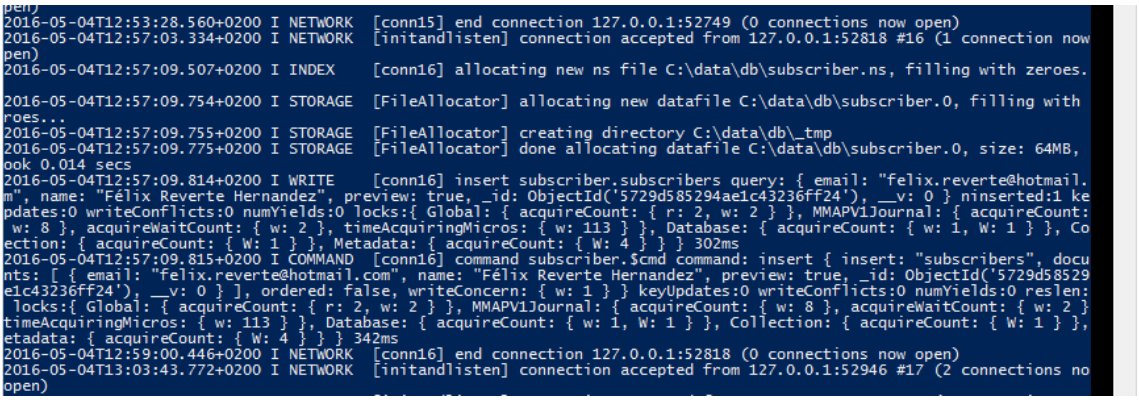
```
sudo apt-get update
sudo apt-get install -y nodejs
sudo apt-get install -y nodejs-legacy
sudo apt-get install -y npm
sudo apt-get install -y node-express
sudo apt-get install -y git
sudo apt-get install -y mongodb-org
sudo service mongod start
git clone https://github.com/ferrha/node-express-signup-tfm-master.git
cd node-express-signup-tfm-master
npm install
node server.js
```

-Capturas de la ejecución de la aplicación:

En la salida de consola proporcionada durante la ejecución de la aplicación se puede ver como se escribe correctamente en la base de datos y se envía el mensaje de confirmación de suscripción.



En la base de datos se inserta el usuario registrado.



Otros registros en la base de datos consultados.

- Pruebas de carga.

En este caso no tienen sentido las pruebas de carga al haberse probado la aplicación en una máquina local.

Se presupone que al ser máquinas más potentes que las proporcionadas por los proveedores Cloud el uso de CPU va a ser más liviano y permitirán más usuarios simultáneos, pero se verán perjudicadas por la latencia debido a las conexiones de red que tenga cada proveedor y por supuesto en el escaldo.

-Coste.

Los costes para estas implementaciones se comentan para cada proveedor y tipo de máquina a utilizar, explicándolos en las siguientes secciones.

4.2.1. VPS

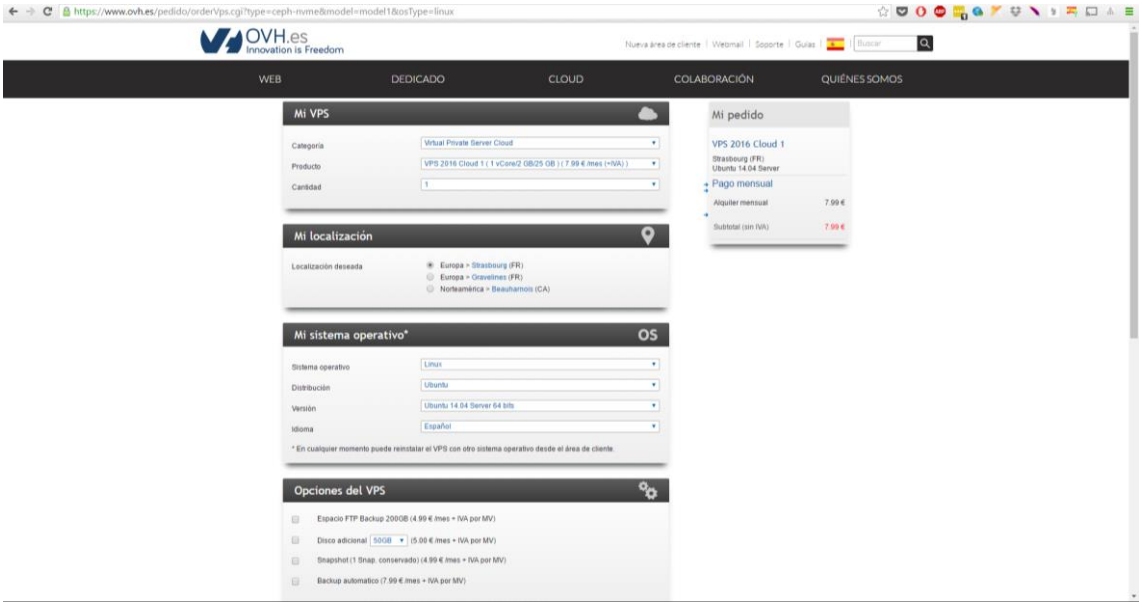
Como se comentaba en la introducción sobre los servidores virtuales, unas de las empresas más conocidas y estables respecto al uso de estos equipos son OVH, Hetzner o KIO Networks. Los precios y características de los equipos se van a tomar de ellas, pero no se van a contratar ya que son elevados y no se dispone de versiones de prueba ni de capas gratuitas.

Como se verá a continuación, el problema de estos servicios no es la potencia de las máquinas, si no el escalado. En el caso de que nuestra aplicación tenga mucho éxito y se necesite lanzar otra máquina el tiempo de lanzamiento, configuración y puesta en marcha es mucho más elevado que en el caso del Cloud, y si luego la demanda de nuestra aplicación baja, se sigue disponiendo del equipo contratado, aunque pudiendo darlo de baja el mes siguiente.
Además, puede haber fluctuaciones de carga en nuestra aplicación durante el día, por ejemplo, se puede dar el caso de que los usuarios nos visiten durante determinadas horas del día, pero el resto del día la aplicación no tenga prácticamente uso. En este caso el Cloud escala automáticamente en segundos o minutos, mientras que en estos proveedores el tiempo de aprovisionamiento es mucho más elevado.

1. **OVH.**

   Este proveedor ofrece geolocalización en Europa solamente, permite aumentar el espacio de almacenamiento del servidor virtual en caliente, ofrecen balanceo de carga por IP, bloqueo de IP, protección anti-DDoS y un SLA de hasta un 99,99%.

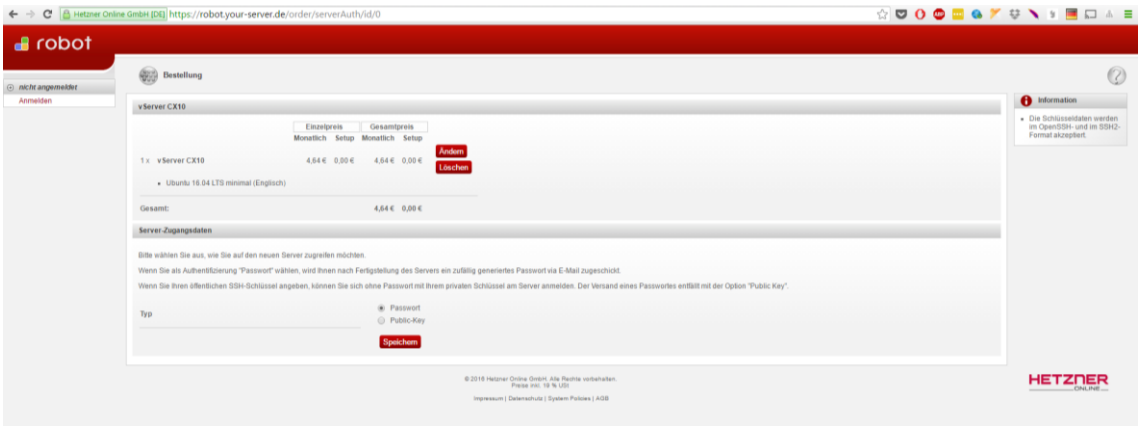
   Para seguir con las especificaciones de equipos como las de los utilizados en los servicios Cloud se elige el servidor virtual más económico, que es más potente que los proporcionados por los proveedores de Cloud.
   Ofrecen por 9,67€/mes un servidor con procesador de 1 núcleo compartido a 3,1 GHz, 2 GB de RAM y 25 GB de almacenamiento SSD.



2. Hetzner.

En este caso, se ofrece también geolocalización en Europa, se da un snapshot gratuitamente y sistema de recuperación, 2 TB de tráfico al mes y un SLA del 99%.

Las especificaciones de esta máquina son: 1 CPU compartido (no se especifica la frecuencia), 1 GB de RAM y 25 GB de SSD por 4.64 €/mes. Como se aprecia en las especificaciones, son unos equipos más modestos, además, se encontrarían los mismos problemas que en el caso anterior.



3. **KIO Networks.**

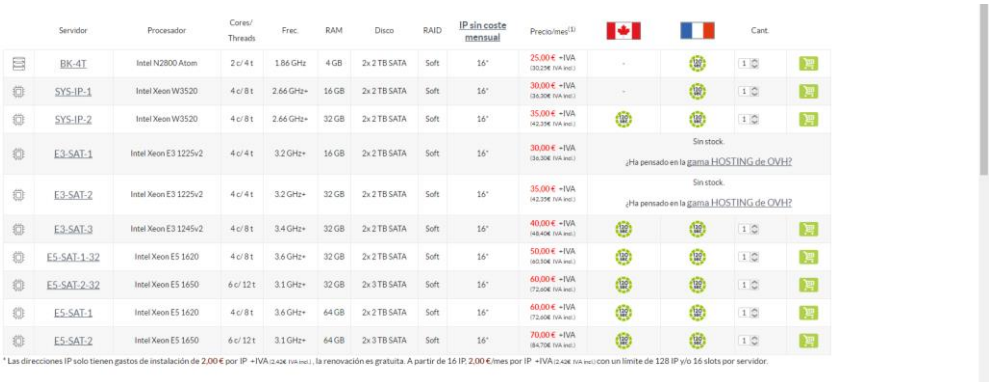
   Por último, esta empresa mexicana, con una sede actualmente en el parque científico de Murcia, ofrece servicios de pago por uso y mucha seguridad, ya que están certificados como Tier IV y por tanto de precio bastante elevado. Ofrecen precio personalizado por empresa y por tanto no se han podido obtener precios.

4.2.2. Hosting

Para este tipo de servicio también se han elegido OVH y Hetzner, siendo mucho más potentes y más caros. Estos servidores podrían servir en caso de que no se pretenda crecer, ya que tienen gran capacidad de cómputo. Ofrecen por un lado almacenamiento en directorio que solo permite albergar una web estática, proporcionando unos 100GB de almacenamiento en la opción básica, pero no permiten instalación de paquetes y solo permiten el acceso por FTP. Y por otro lado, ofrecen servidores dedicados bastante potentes y accesibles por SSH, en los que se puede instalar lo que se desee.

1. **OVH**.

   En este caso ofrecen equipos desde 30 €/mes hasta 84.70 €/mes, se elige uno del menor coste con las especificaciones siguientes:
   Procesador de 4 núcleos a 2.66 GHz, 16 GB de RAM y 2 discos de 2 TB mecánicos, contando con protección anti-DDoS, backup, firewall, IP Failover y 16 IP, localizados en Francia.

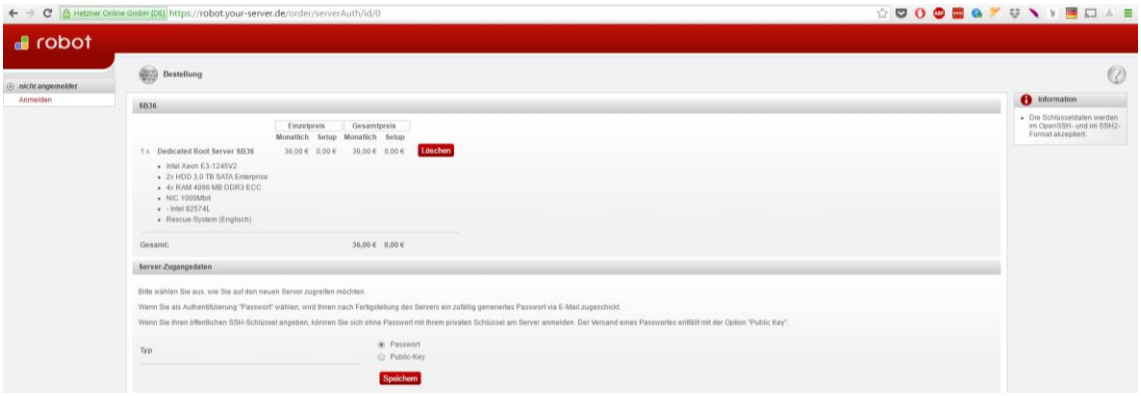


Servidor	Procesador	Cores/ Threads	Frec.	RAM	Disco	RAID	IP sin coste mensual	Precio/mes[1]			Cant.	
BK-4T	Intel N2800 Atom	2 c/ 4 t	1.86 GHz	4 GB	2x 2 TB SATA	Soft	16*	25,00 € +IVA (30.25€ IVA incl.)	-		1	
SYS-IP-1	Intel Xeon W3520	4 c/ 8 t	2.66 GHz+	16 GB	2x 2 TB SATA	Soft	16*	30,00 € +IVA (36.30€ IVA incl.)	-		1	
SYS-IP-2	Intel Xeon W3520	4 c/ 8 t	2.66 GHz+	32 GB	2x 2 TB SATA	Soft	16*	35,00 € +IVA (42.35€ IVA incl.)			1	
E3-SAT-1	Intel Xeon E3 1225v2	4 c/ 4 t	3.2 GHz+	16 GB	2x 2 TB SATA	Soft	16*	30,00 € +IVA (36.30€ IVA incl.)	Sin stock. ¿Ha pensado en la gama HOSTING de OVH?			
E3-SAT-2	Intel Xeon E3 1225v2	4 c/ 4 t	3.2 GHz+	32 GB	2x 2 TB SATA	Soft	16*	35,00 € +IVA (42.35€ IVA incl.)	Sin stock. ¿Ha pensado en la gama HOSTING de OVH?			
E3-SAT-3	Intel Xeon E3 1245v2	4 c/ 8 t	3.4 GHz+	32 GB	2x 2 TB SATA	Soft	16*	40,00 € +IVA (48.40€ IVA incl.)			1	
E5-SAT-1-32	Intel Xeon E5 1620	4 c/ 8 t	3.6 GHz+	32 GB	2x 2 TB SATA	Soft	16*	50,00 € +IVA (60.50€ IVA incl.)			1	
E5-SAT-2-32	Intel Xeon E5 1650	6 c/ 12 t	3.1 GHz+	32 GB	2x 3 TB SATA	Soft	16*	60,00 € +IVA (72.60€ IVA incl.)			1	
E5-SAT-1	Intel Xeon E5 1620	4 c/ 8 t	3.6 GHz+	64 GB	2x 2 TB SATA	Soft	16*	60,00 € +IVA (72.60€ IVA incl.)			1	
E5-SAT-2	Intel Xeon E5 1650	6 c/ 12 t	3.1 GHz+	64 GB	2x 3 TB SATA	Soft	16*	70,00 € +IVA (84.70€ IVA incl.)			1	

* Las direcciones IP solo tienen gastos de instalación de 2,00 € por IP +IVA (2.42€ IVA incl.), la renovación es gratuita. A partir de 16 IP, 2,00 €/mes por IP +IVA (2.42€ IVA incl.) con un límite de 128 IP y/o 16 slots por servidor.

2. **Hetzner.**

Este proveedor proporciona multitud de equipos, se elige el más económico por 36 €/mes con las siguientes características:
Procesador de 4 núcleos a 3.4 GHz, 16 GB de RAM y 2 discos de 3 TB mecánicos. Como en el caso anterior de servidores virtuales ofrecen menos seguridad que OVH y están ubicados en Alemania.



4.2.3. Housing

Este tipo de servicio es proporcionado por ejemplo por KIO Networks, se encuentra disponible por un coste bastante elevado debido a los servicios que proporcionan de seguridad física, virtual, protección anti-incendios, suministro eléctrico constante, redundancia de equipos y de conexión a internet, todo ello gracias a su certificación Tier IV. Estas ofertas las proporcionan bajo de manda y los precios los dan según el cliente, cantidad de equipos, etc.

4.3. Resumen

En esta sección se van a resumir primeramente en una tabla la comparación de latencias y uso de CPU en cada caso y finalmente en otra tabla los servicios contratados para cada proveedor, la escalabilidad, las características.

En primer lugar, los servicios Cloud proporcionan todos una escalabilidad alta y automática, frente a los proveedores No Cloud que no son escalables o solo permiten aumentar el almacenamiento del servidor.

En cuanto a características, los proveedores Cloud proporcionan máquinas menos potentes que las de los proveedores No Cloud, aunque sean maquinas modestas, al poder escalar rápidamente proporcionan al usuario recursos extra en pocos segundos o minutos según la necesidad.

Y, por último, en cuanto al coste, los proveedores Cloud utilizados en este trabajo presentan costes muy parecidos a excepción de Azure que presenta un coste muy elevado, dichos costes son comedidos y dado que se realiza un cobro por uso son opciones muy económicas. Por otro lado, los proveedores No Cloud en las versiones VPS ofrecen costes reducidos a costa de la escalabilidad, mientras que los servidores dedicados tienen precios algo más elevados.

En la siguiente tabla se comparan las latencias y uso de CPU obtenidos de las pruebas de carga realizadas a la aplicación sobre cada proveedor. Se aprecia que Azure es el proveedor que menor latencia tiene y menos uso de CPU, frente a Google que es el que más CPU consume y mayor latencia tiene. Las mediciones tan dispares en el uso de CPU deben darse por la diferencia de frecuencia de los procesadores en cada caso.

Proveedor	Latencia media (ms)	Uso medio de CPU (%)
AWS (Elastic Beanstalk)	1.25	35
AWS (EC2)	1.4	3.6
Google	1.51	32
Azure	1.2	2.25

Cabe destacar que, si se tiene un servicio o aplicación desplegados que tienen un uso por parte de los usuarios muy elevado en determinadas horas del día, por ejemplo, de 9 a 11H, y se necesitan 4 servidores para cubrir esa demanda sólo durante ese periodo de tiempo y el resto del día basta con un único servidor, el Cloud sería mucho más económico debido al pago por uso al tener picos y valles de uso. Mientras que en el mismo supuesto para una arquitectura No Cloud se debería tener siempre los 4 servidores contratados y ejecutándose para suplir dicha demanda.

Los costes dados en la tabla siguiente para los servicios Cloud están configurados para que se tengan instancias en espera y con uso reducido al mes, si siguiendo con el ejemplo

anterior se añaden 4 servidores para los proveedores No Cloud su coste se multiplica, mientras que el coste para el Cloud se mantiene al contemplar el uso temporal de recursos.

Proveedor	Servicio	Escalabilidad	Características	Coste (€/mes)
AWS	Elastic Beanstalk SNS DynamoDB	Muy alta y automática	CPU 1 núcleo compartido 0.6 GB RAM 10GB SSD	**32.87**
	EC2 SNS DynamoDB	Alta y automática	CPU de 1 núcleo compartido 1 GB RAM 10 GB SSD	**32.87**
Google	Compute Engine Datastore	Alta y automática	CPU de 1 núcleo compartido 0.6 GB RAM 10 GB SSD	**34.98**
Azure	Virtual Machine DocumentDB	Alta y automática	CPU de 1 núcleo compartido 3.5 GB RAM 7 GB SSD	**73.78**
OVH	VPS	Solamente en almacenamiento	CPU de 1 núcleo compartido a 3,1 GHz 2GB de RAM 25 GB SSD	**9.67**
	Servidor dedicado	Ninguna	4xCPU 2.66 GHz 16 GB de RAM 2 discos de 2 TB	**30**
Hetzner	VPS	Solamente en almacenamiento	CPU de 1 Core compartido 1 GB de RAM 25 GB SSD	**4.64**
	Servidor dedicado	Ninguna	4xCPU 3.4 GHz 16 GB de RAM 2 discos de 3 TB	**36**

Capítulo 5

5. Conclusiones

Como se ha visto en el desarrollo de este trabajo, las principales ventajas de los proveedores Cloud son el pago por uso, la escalabilidad y la alta disponibilidad, mientras que los servicios No Cloud son ventajosos a la hora de utilizarlos en aplicaciones de uso constante y predecible. Por otro lado, las desventajas de los proveedores Cloud son la seguridad y la dificultad o trabajo extra a la hora de migrar una misma aplicación entre proveedores Cloud o de proveedores No Cloud a la nube, y por su parte, las desventajas de los proveedores No Cloud son el desaprovechamiento de recursos y la no escalabilidad.

Dependiendo de la aplicación que se vaya a desarrollar puede ser interesante utilizar unos servicios u otros, si se desconoce el éxito que va a tener la aplicación es recomendable utilizar servicios Cloud, debido al pago por uso y a la facilidad de instalación. Mientras que si se tiene una aplicación acotada, que se sabe el número de usuarios y no se va a crecer es mejor utilizar un servidor VPS debido a que es más económico.

Durante la realización de este trabajo me ha sorprendido la facilidad con la que, por ejemplo, AWS permite encender una instancia EC2 con Ubuntu, instalar el software necesario y configurar el escalado que, además, arranca muy rápido nuevas instancias.

Por parte de Google Cloud Platform me ha parecido que es más lento que AWS a la hora de lanzar nuevas instancias, poco más caro y un poco más difícil de configurar. Por parte de Microsoft Azure me ha parecido muy caro, aunque las máquinas son más potentes, lento de configurar y lo peor su velocidad a la hora de crear instancias ya que se va a más de 5 minutos, incluso al lanzar una instancia obtuve un error y tuve que rehacerla. También me ha sorprendido que solo Google proporcione instancias con Ubuntu 16.04 LTS y el resto sigan utilizando 14.04 LTS.

Por último, este proyecto sirve como guía para el desarrollo y despliegue de aplicaciones en el Cloud, de una manera guiada y proporcionando ejemplos funcionales adaptados a cada uno de los proveedores utilizados.

5.1. Líneas de desarrollo futuro

Se debe permanecer muy atento a la evolución de las tecnologías y servicios Cloud ya que son relativamente nuevos y deben aparecer nuevos servicios ofrecidos por los proveedores estudiados en este trabajo. Hay diferentes servicios en fase Beta y seguro que muchos por venir que ayuden a los desarrolladores y administradores de sistemas a hacer su trabajo más eficiente y cómodo.

En cuanta la aplicación utilizada se pueden añadir test unitarios a la función *signup* existente y a nuevos "end points" que se añadan posteriormente, con este testeo se aprende a testear NodeJS y a mejorar la calidad del código asegurando su correcto funcionamiento. Todo ello siguiendo la práctica del TDD (Test Driven Development o Desarrollo Guiado por Pruebas), para llevar este testeo a cabo se puede utilizar Mocha, Supertest y Chai. Además de poderse añadir test funcionales que simulen la utilización de la plataforma por parte de un usuario, utilizando por ejemplo Selenium.

Por otro lado, se deben tener presentes las dependencias del proyecto para evitar sorpresas a la hora de realizar actualizaciones que hagan que deje de funcionar nuestra aplicación o de versiones obsoletas.

Bibliografía

[1] Van Vliet, J., Paganelli, F. (2011). Programming Amazon EC2. O'Reilly.

[2] Sosinsky, B. (2011). Cloud Computing Bible. Wiley Publishing, Inc.

[3] Marinescu, Dan C. (2012). Cloud Computing: Theory and Practice. Computer Science Division Department of Electrical Engineering & Computer Science. University of Central Florida, Orlando, FL 32816, USA

[4] Amazon EC2 Container Service (ECS) Developer Guide (English Edition). (2015)

[5] AWS Elastic Beanstalk Developer Guide (English Edition). (2015)

[6] Collier, M. Microsoft Azure Esentials – Fundamentals of Azure. (2014)

[7] Guthrie, S., Simms, M., Dykstra, T., Anderson, R., Wasson, M. (2014). Building Cloud Apps with Microsoft Azure: Best Practices for DevOps, Data Storage, High Availability, and More (Developer Reference). Microsoft Press.

[8] AWS Command Line Interface User Guide (English Edition). (2015)

[9] Amazon Simple Storage Service (S3) Getting Started Guide (English Edition). (2015)

[10] Amazon Elastic Compute Cloud (EC2) User Guide for Linux Instances (English Edition). (2015)

[11] Getting Started with AWS: Deploying a Web Application (English Edition). (2015)

[12] WEB https://aws.amazon.com/es/documentation/

[13] WEB https://cloud.google.com/docs/

[14] WEB https://azure.microsoft.com/en-us/develop/nodejs/

[15] WEB https://github.com/nodemailer/nodemailer

[16] WEB http://mongoosejs.com/docs/