Universidad de La Laguna

Práctica 6

Computación en la Nube

Pierre Simon Callist Yannick Tondreau
Servicio Cloud: http://10.6.129.171

Propuesta 1

Incorpora algún mecanismo de seguridad que permita gestionar la autorización a la incorporación de un nuevo cliente software.

Para realizar esta propuesta, se ha decidido implementar la autenticación mediante el uso de JWT (JSON Web Token). Para implementar esta funcionalidad en el servidor backend de express, se han usado las librerías adicionales ***passport*** y ***passport-jwt.***

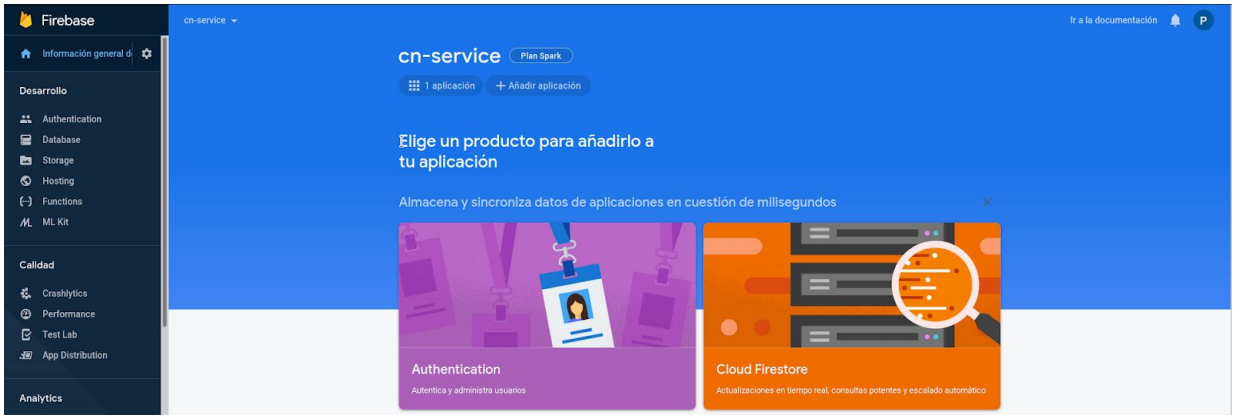
- **passport**: Passport es un middleware de autenticación para Node.js. Extremadamente flexible y modular, Passport se puede colocar discretamente en cualquier aplicación web basada en Express.
- **passport-jwt:** Estrategia de passport que permite el uso de JWT, para la autenticación.

```
passport.use('jwt', new JWTstrategy(opts, (jwt_payload, done) => {
    try {
        console.log(jwt_payload.id);
        User.findOne({
            username: jwt_payload.id,
        }).then(user => {
            if (user) {
                console.log('User found in database');
                done(null, user);
            } else {
                console.log('User not found in db');
                done(null, false);
            }
        })
    } catch (err) {
        done(err);
    }
}))
```

De esta forma, se ha añadido una medida de autorización para permitir al servicio frontend usar la API.

También se ha decidido implementar una medida de autenticación para el frontend, para ello se ha usado ***firebase***.

- **Firebase**: Es una plataforma ofrecida por Google que permite integrar de forma sencilla el uso de una base de datos, sistemas de analíticas y muchas funcionalidades más.



Para implementar lo mencionado anteriormente en el servicio frontend, es tan simple como añadir las siguientes funciones en el código fuente:

```
var app = firebase.initializeApp({
  apiKey: "AIzaSyD8thscCqXq3Kc2COQE_6LQ-uijyJVWA3Q",
  authDomain: "cn-service.firebaseapp.com",
  databaseURL: "https://cn-service.firebaseio.com",
  projectId: "cn-service",
  storageBucket: "cn-service.appspot.com",
  messagingSenderId: "615852070953",
  appId: "1:615852070953:web:9713c9fd0484858144e438"
});
```

Y finalmente, usar las funciones que se indican para realizar la autenticación y el registro de usuarios al servicio frontend.

```
export const registerUser = account => dispatch => {
    firebase.auth().createUserWithEmailAndPassword(account.email, account.password).then(val => {
        firebase.auth().signInWithEmailAndPassword(account.email, account.password).then(user => {
            dispatch(authenticate());
        }, err => {
            console.log(err.code);
            console.log(err.message);
        })
    }, err => {
        console.log(err.code);
        console.log(err.message);
    });
};

export const authenticateUser = account => dispatch => {
    firebase.auth().signInWithEmailAndPassword(account.email, account.password).then(val => {
        dispatch(authenticate());
    }, err => {
        console.log(err.code);
        console.log(err.message);
    })
};
```

Propuesta 2

Realiza el análisis de costes de incorporar la infraestructura de alguno de los siguientes proveedores, Amazon, Google, Microsoft, Alibaba. Tienen que ponerse previamente de acuerdo para elegir uno diferente. El análisis de costes tiene que tener en cuenta los aspectos descritos en el documento "Your cloud bill climbs due to these five overlooked costs" que puedes encontrar en el campus.

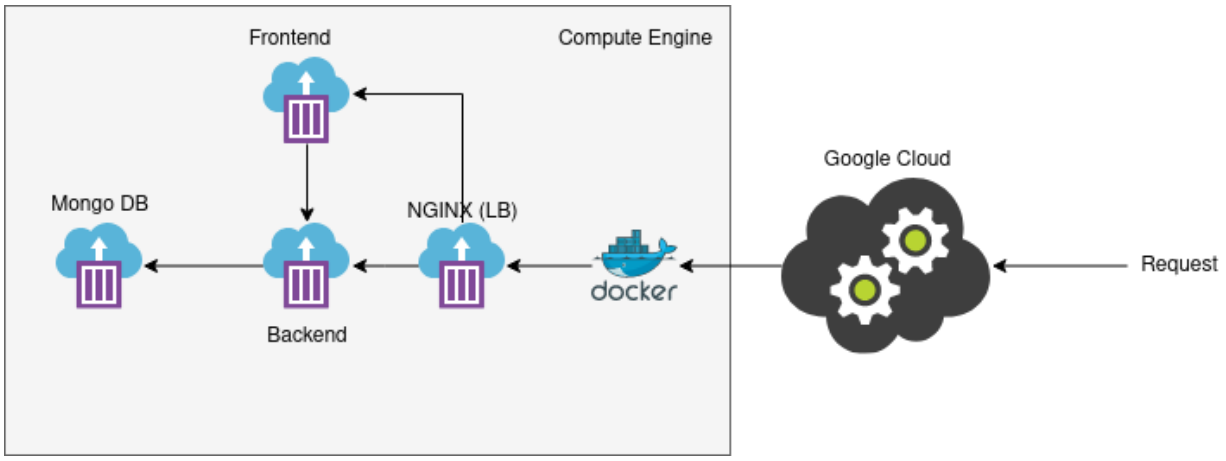
La plataforma que se ha escogido para el análisis de coste de despliegue es **Google Cloud Platform**. Dicha plataforma ofrece una gran cantidad de servicios distintos que permiten desplegar aplicaciones de diferentes tipos.

Para el caso de la aplicación de esta práctica, se requieren los siguientes servidores:

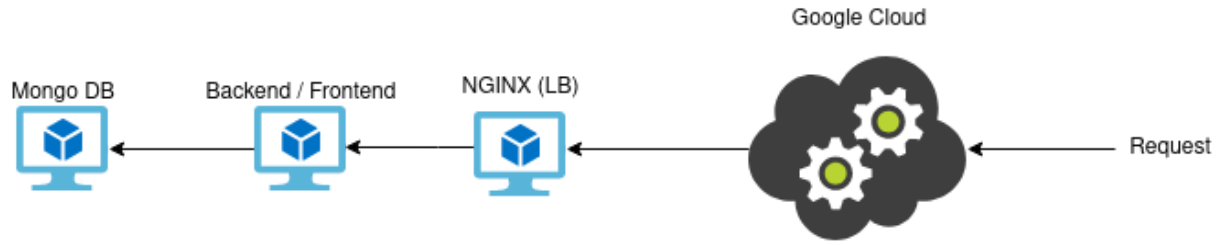
- Base de datos
- Frontend
- Backend
- Balanceador de Carga

El despliegue de los servidores mencionados anteriormente se puede realizar de diversas formas.

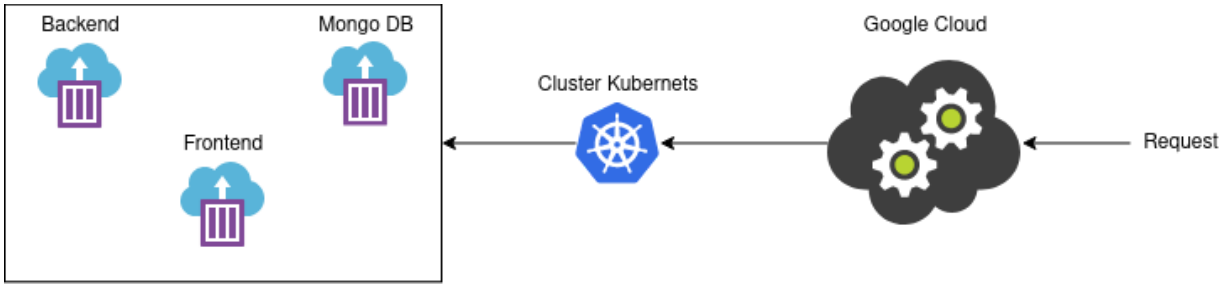
- La primera de ellas es desplegar los servicios en un único Compute Engine mediante el uso de una tecnología de contenedores como Docker, el resultado es el siguiente:



- La segunda solución es desplegar tres instancias de Compute Engines, en las cuales una de ellas ejecutará el backend y frontend, en otra de ellas ejecuta el balanceador de carga y en la última se ejecuta la base de datos.



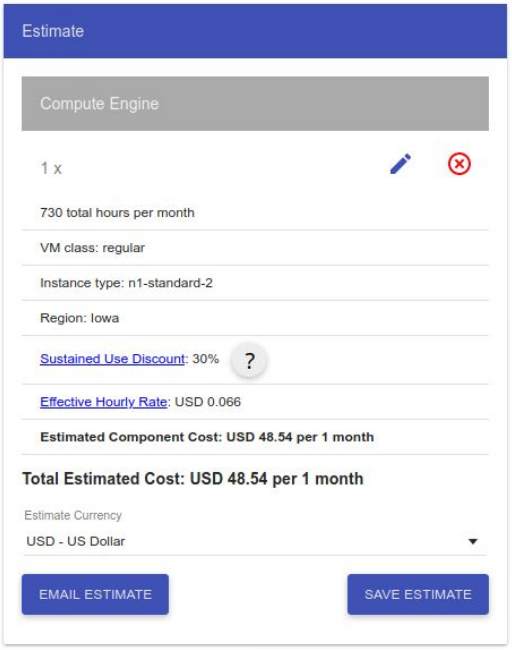
- La tercera solución es desplegar cada uno de los servicios en un cluster de Kubernetes. Los servicios de backend, frontend y base de datos, se ejecutan en el cluster de Kubernetes, permitiendo una escalabilidad horizontal más efectiva dado que se encarga del balanceo interno.



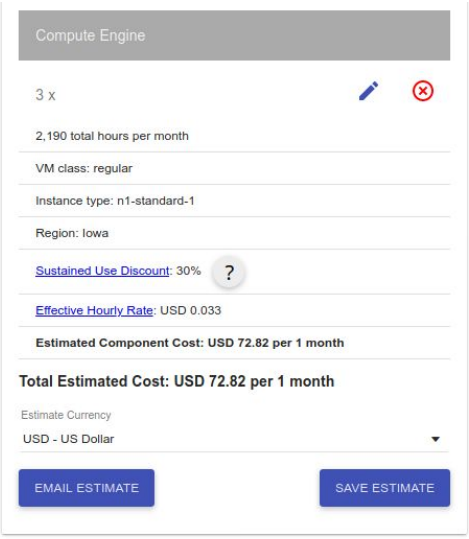
Análisis de Costes

A continuación veremos el coste asociado a cada una de las soluciones mencionadas anteriormente (suponiendo que las máquinas se ejecutan 24/7).

- Para desplegar el servicio siguiendo la estructura de la primera solución, solo es necesario una instancia de un Compute Engine. El tipo de instancia que se usaría es del tipo *n1-standard-2*, este tipo de máquina ofrece 2vCPUs y 7.5GB de RAM por un precio de 0.0950$/hora. Mensualmente esto es equivalente a 48.5500$/mes. Si calculamos usando el calculador de precios de Google, obtenemos el siguiente resultado:



- Para desplegar el servicio siguiendo la estructura de la segunda solución, es necesario de tres instancias Compute Engine. El tipo de instancia usados serían del tipo *n1-standard-1*, este tipo de máquina ofrece 1vCPU y 3.75GB de RAM por un precio de 0.0475$/hora. Mensualmente esto es equivalente a 24.2725$/mes. Dado que necesitamos tres instancias el precio final calculado usando el calculador de precios de Google, es el siguiente.



- Finalmente, para desplegar el servicio siguiente la estructura de la tercera solución, es necesario desplegar un Kubernetes Engine con tres nodos de tipo *n1-standard-1*, los nodos del cluster tienen las siguientes especificaciones: 1vCPU y 3.75GB de RAM. El precio del despliegue es de 0.0475$/hora por nodo o 24.2725$/mes por nodo. Adicionalmente se paga el mantenimiento del cluster de Kubernetes por lo tanto, se añade al precio la cantidad de 0.10$/hora por cluster. El precio final usando el calculador de precios de Google es el siguiente: