Openshift

Autor: Antonio Guzmán Martín

¿Qué es openshift?

Es un producto de computación en la nube de plataforma como servicio de Red Hat.

Los desarrolladores pueden usar Git para desplegar sus aplicaciones Web en los diferentes lenguajes de la plataforma.

OpenShift también soporta programas binarios que sean aplicaciones Web, con tal de que se puedan ejecutar en RHEL Linux. Esto permite el uso de distintos lenguajes y frameworks.

OpenShift se encarga de mantener los servicios subyacentes a la aplicación y la escalabilidad de la aplicación como se necesite. Una plataforma como servicio.

¿Qué es PAAS?

Es la abstracción de un ambiente de desarrollo y el empaquetamiento de una serie de módulos o complementos que proporcionan, normalmente, una funcionalidad horizontal (persistencia de datos, autenticación, mensajería, etc.). De esta forma, un arquetipo de plataforma como servicio podría consistir en un entorno conteniendo una pila básica de sistemas, componentes o APIs preconfiguradas y listas para integrarse sobre una tecnología concreta de desarrollo (por ejemplo, un sistema Linux, un servidor web, y un ambiente de programación como Perl o Ruby). Las ofertas de PaaS pueden dar servicio a todas las fases del ciclo de desarrollo y pruebas del software, o pueden estar especializadas en cualquier área en particular, tal como la administración del contenido. Algunos ejemplos destacables son Google App Engine, Microsoft Azure, Heroku o la propia Openshift.

Openshift vs otros servicios PAAS.

Amazon web Services

Es la plataforma más extendida y la primera en entrar al mercado.Está muy por encima de la competencia en cuanto a funcionalidades, número de datacenters y número de clientes.Tienen una pega se basan en tecnologías propias

Google Cloud Platform

En poco tiempo ha conseguido tener una oferta muy compacta.Su oferta es más reducida que la de AWS pero por contra es más simple de utilizar y más económica.

Podemos destacar también que está basada principalmente en estándares abiertos y que usa los mismos datacenters de Google, conectados por fibra propia y en constante innovación.

Windows Azure

Tiene una oferta muy completa no solo de tecnologías Microsoft sino también de tecnologías open source, por las que sorprendentemente los de Redmond están haciendo una gran apuesta en los últimos años.

OpenshiftOrigin

Origin es la versión de código abierto de OpensShift. Es ahora mismo una de las opciones más interesantes para Cloud privadas, ya que está basada en estándares abiertos, integrada con las tecnologías open source de referencia y tiene un market en constante crecimiento.
Los productos OpenShift Online, Openshift Dedicated y OpenShift Enterprise son implementaciones de Origin.
Origin utiliza Docker para la gestión de contenedores y Kubernetes para la gestión de grupos de contenedores.

Arquitectura de Openshift

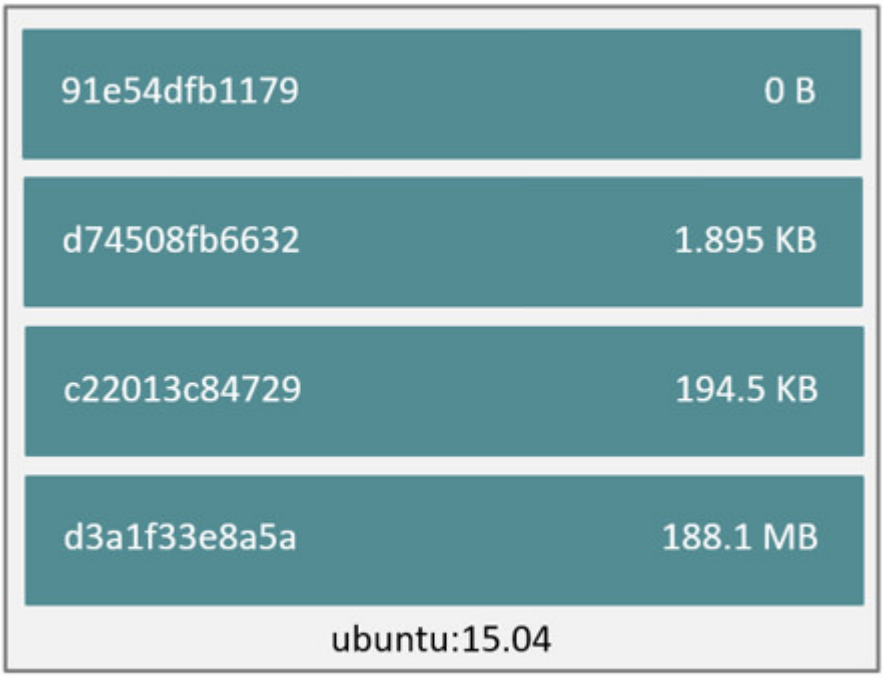
Componentes:

Kubernetes plataforma comenzada por Google en 2014, automatiza el desarrollo, escalado y gestión de contenedores. Está construido para trabajar junto a:

Docker (basado en crear contenedores portables para ser ejecutados en cualquier máquina con Docker instalado independientemente de el SO que tenga por debajo). Un contenedor contiene todo aquello que necesite mi aplicación para ser ejecutada como (java,Maven,tomcat…) y la propia aplicación. La idea es que el programador pueda centrarse en desarrollar su proyecto sin preocuparse si el código funcionará en la maquina que ejecute. Un contenedor es mucho más ligero que una máquina virtual, funciona con el SO de la máquina en la que se ejecuta el ordenador.

Con Kubernetes podemos ver el conjunto de contenedores como un inmenso clúster de aplicaciones donde desacoplar nuestras aplicaciones.
El escalado ya sea “hacia arriba” o “hacia abajo”; con Kubernetes es fácil, solo se tiene que especificar la cantidad de nuevas aplicaciones y si quieres hacia arriba o hacia abajo a través de un comando

OpenShift Online puede usar cualquier servidor implementando la API registry de Docker como proveedor de imágenes incluyendo el Hub Docker (registro docker). Docker registry es un servicio para manejar información sobre las imágenes docker y habilitar su distribución.
Para almacenar drivers eficazmente, se debe entender como Docker almacena y compila imágenes. Una imagen Docker contiene una lista de capas que representan diferencias de sistemas de archivos . Las capas son puestas encima una de otra para contener el sistema de archivos root. El storage de Docker es responsable de apilar las capas proporcionando una capa unificada. Imagen de Ubuntu 15.04 apiliadas en 4 capas de imagen apiladas y comprimidas.



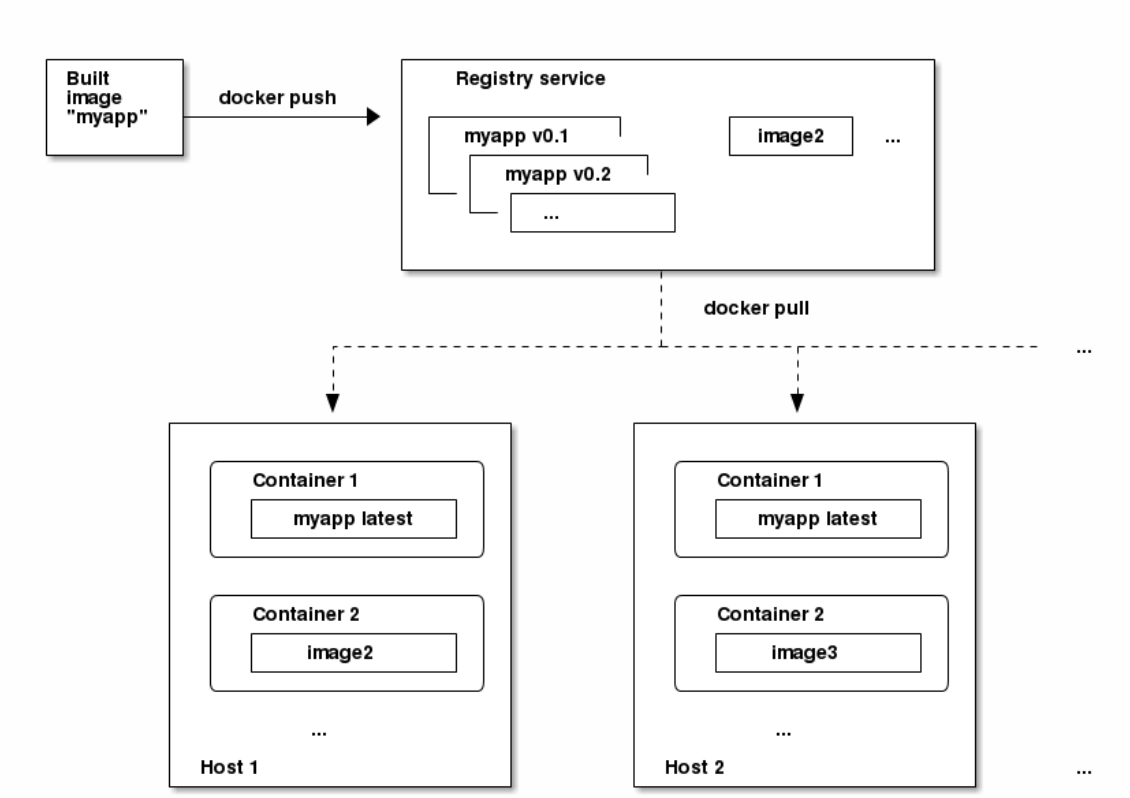
Image

Siempre que una nueva imagen es empujada para ser integrada en un registro , el registro notifica a Openshift Online sobre la nueva imagen pasándole toda la información sobre ella, como el espacio de nombres, el nombre y los metadatos de la imagen.

Desplegando la misma imagen en múltiples contenedores sobre múltiples hosts y cargando y balanceándolos entre ellos Openshift Online puede proveer de redundancia y escalado horizontal para un servicio empacado en una imagen

Por último Openshift Online nos ofrece una consola web donde podremos visualizar, buscar y manejar los contenidos de nuestros proyectos. Openshift proporciona su propio

registro para manejar sus propios contenedores de imágenes (representación).

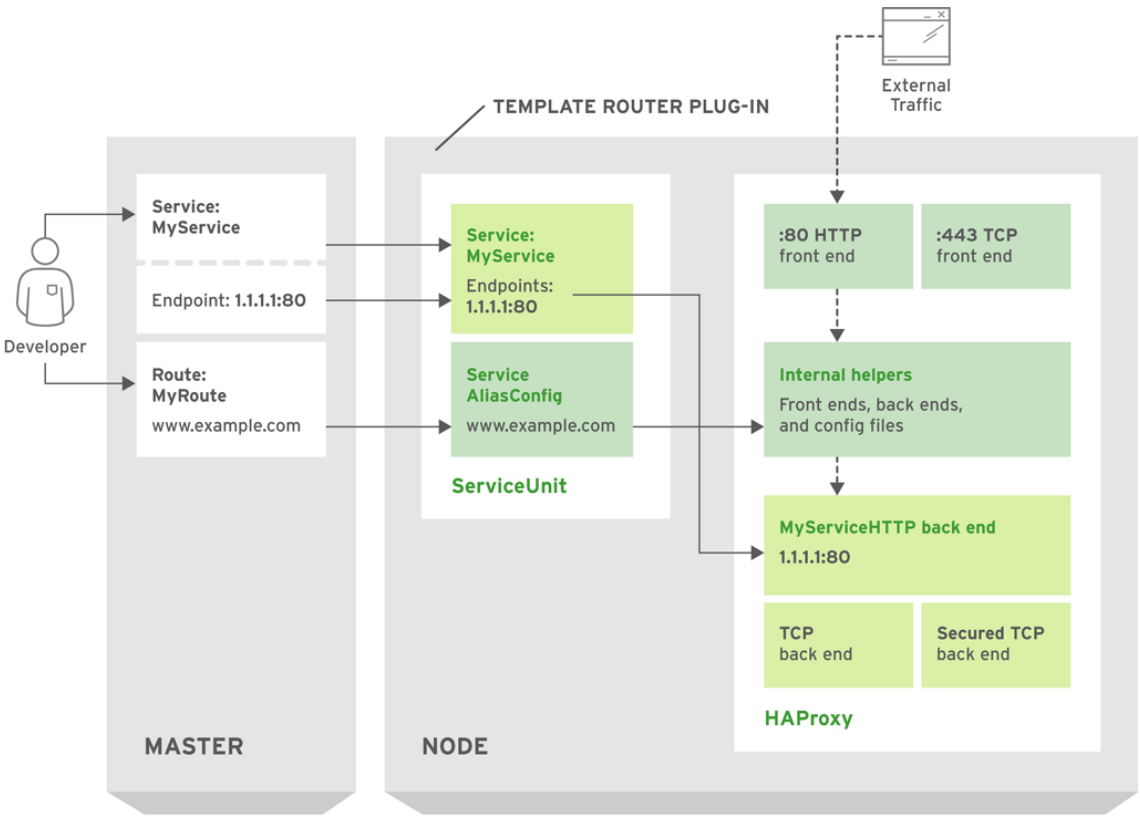


Pod es un grupo de grupo de uno o más contenedores, el almacenamiento compartido para almacenar dichos contenedores y como ejecutarlos. Siempre se ejecutan en un contexto compartido.En un mundo pre-contenedor, seria ejecutado físicamente como una máquina virtual. El contexto de un Pod es un conjunto de espacios de nombre Linux, cgroups y otras facetas de aislamiento.

Proyectos: Un proyecto permite a un grupo de usuarios organizar y administrar su contenido aisladamente de otros. Los administradores permiten a los desarrolladores crear proyectos.
Builds: Para las compilaciones Docker y S2Ibuilds el resultado son imágenes ejecutables

Rutas: una ruta expone un servicio en un hostname como : example.com, así los clientes pueden llegar a ellos através de su nombre.

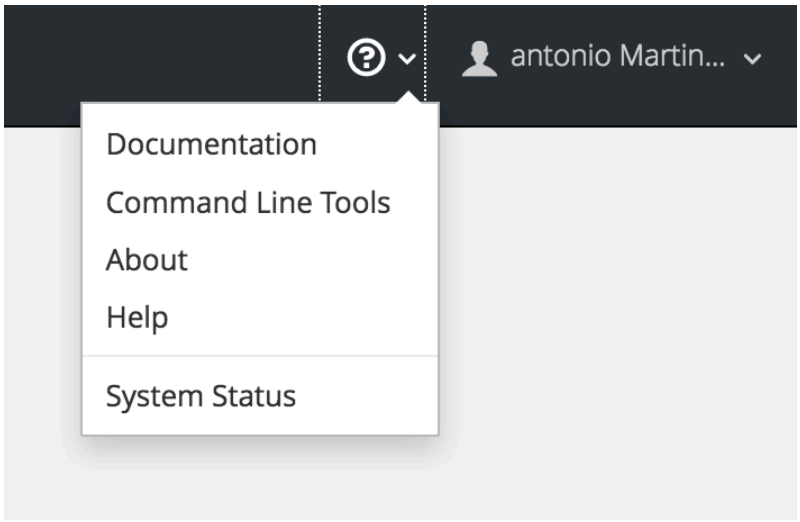
Un router usa el selector de servicio para encontrar el servicio y los endpoints detrás del servicio. Openshift usa el balanceo de carga del router. Un router detecta los cambios en las direcciones IP de sus servicios y adapta la configuración de forma acorde. Esto es útil para comunicar modificaciones de los objetos de la API a una solución externa de enrutamiento
El Haproxy template router es la implementación de referencia para una plantilla router pug-in. La siguiente imagen ilustra como los datos fluyen desde el master a través del plugin y finalmente en la configuración Haproxy



Por último un template describe un conjunto de objetos que pueden incluir cualquier cosa que los usuarios tengan permiso de crear dentro de un proyecto (servicos, configuraciones de ejecución, configuraciones de desarrollo)

Tutorial para crear aplicación en Openshift para Linux/MacOS X

1. Debemos acceder a About y seleccionar Command Line tools



2. Descargar oc.tar.gz, desempaquetarlo y añadirlo al $PATH

```
$ echo $PATH
```

3. Despues iniciar sesión y suministrar nombre de usuario y contraseña

```
$ oc login
```

4. Crear nuevo proyecto nodejs.

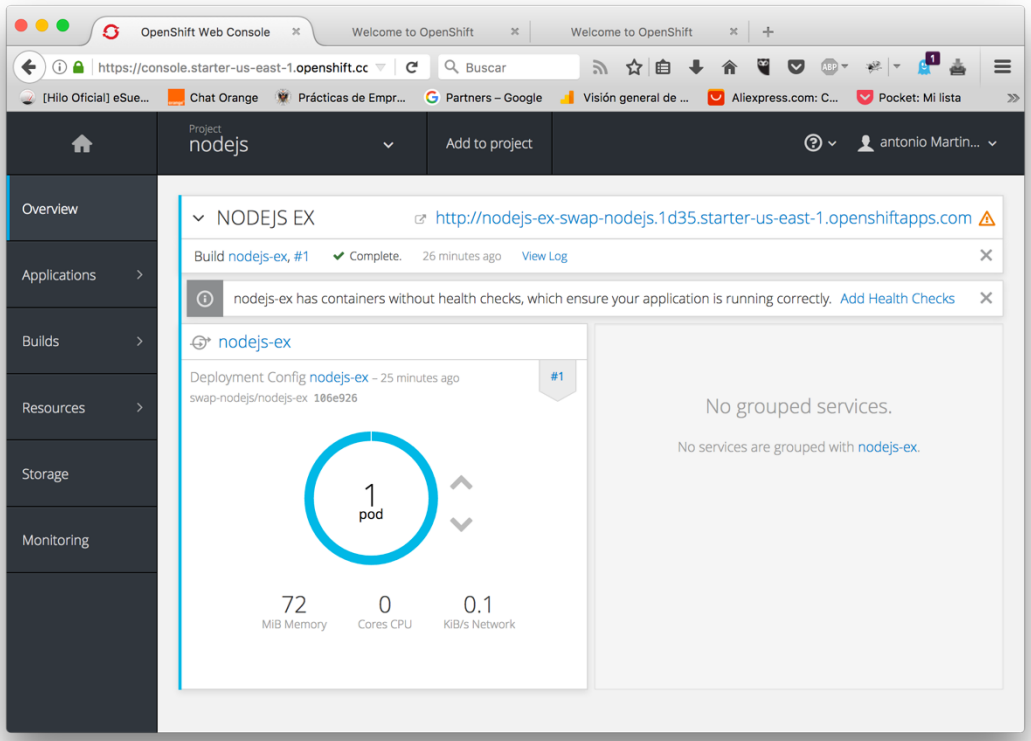
```
 oc new-project <Project-name> \
--display-name="nodejs" --description="Sample Node.js app"
```

5. Nos situamos en el proyecto que acabamos de crear

```
oc project <Project-name>
```

6. Creamos una nueva app, escogemos un ejemplo del repositorio de openshift

```
oc new-app https://github.com/openshift/nodejs-ex -l name=myapp
```



la herramienta se encarga de inspeccionar el código , localizar una imagen apropiada den DockerHub, crear una ImagenStream (contiene una o más imágenes docker)para esa imagen y crear una configuración de ejecución, de desarrollo y una definición de servicio.

7. Estructura del proyecto nodejs, también se crean archivos en el directorio openshift > template.

```
nodejs-ex
├── openshift
│   └── templates
│       ├── nodejs.json
│       ├── nodejs-mongodb.json
│       └── nodejs-mongodb-persistent.json
├── package.json
├── README.md
├── server.js
├── tests
│   └── app_test.js
└── views
    └── index.html
```

Package.json: fichero en formato json con los metadatos que identifican nuestro proyecto. El nombre y versión son obligatorios.

Server.js: Si hay un archivo server.js npm por defecto lo iniciará. En este archivo se usa el framework Express que nos ayuda a organizar la aplicación web según la arquitectura MVC en el lado del servidor. Básicamente nos ayuda a organizar todo,rutas, manejo de peticiones y vistas

Directorio de test: contendrá ficheros para realizar test de la aplicación
Directorio Views: El directorio donde se encuentran los archivos de plantilla.

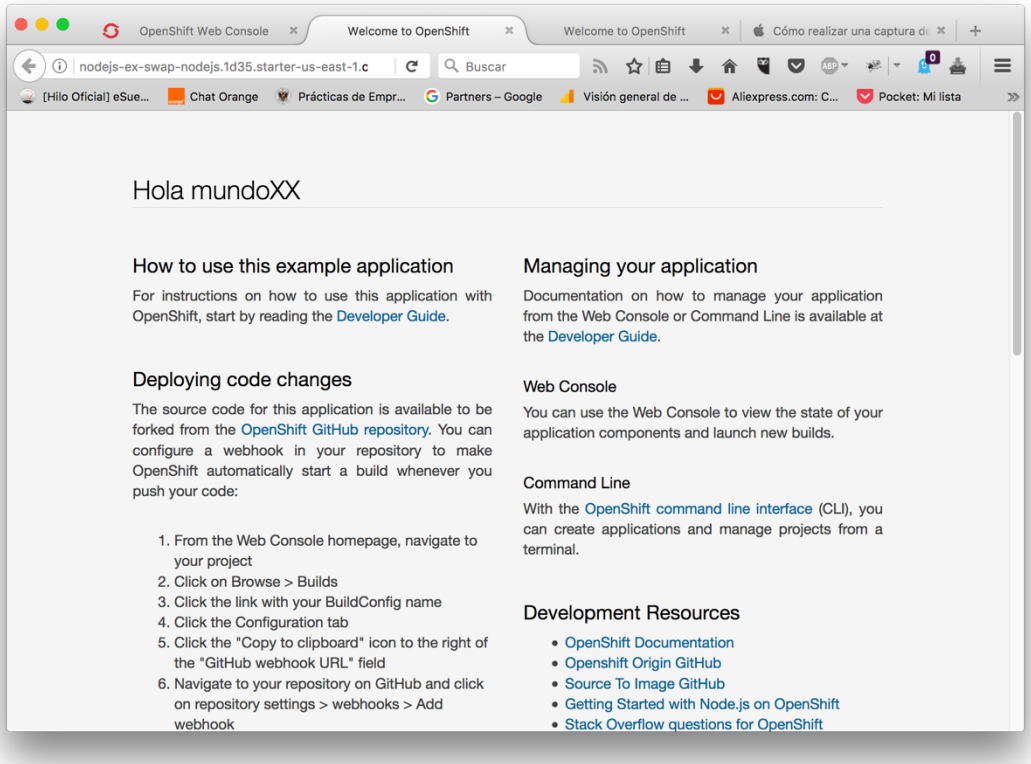
Directorio Openshift: Es una carpeta con templates que contiene nuestro proyecto que podemos para una futura creación de otro proyecto a partir del actual

8. Comprobamos todo lo que ha creado el openshift con oc get status:

Exponemos nuestro servicio mongo con la orden (creamos un route inseguro).

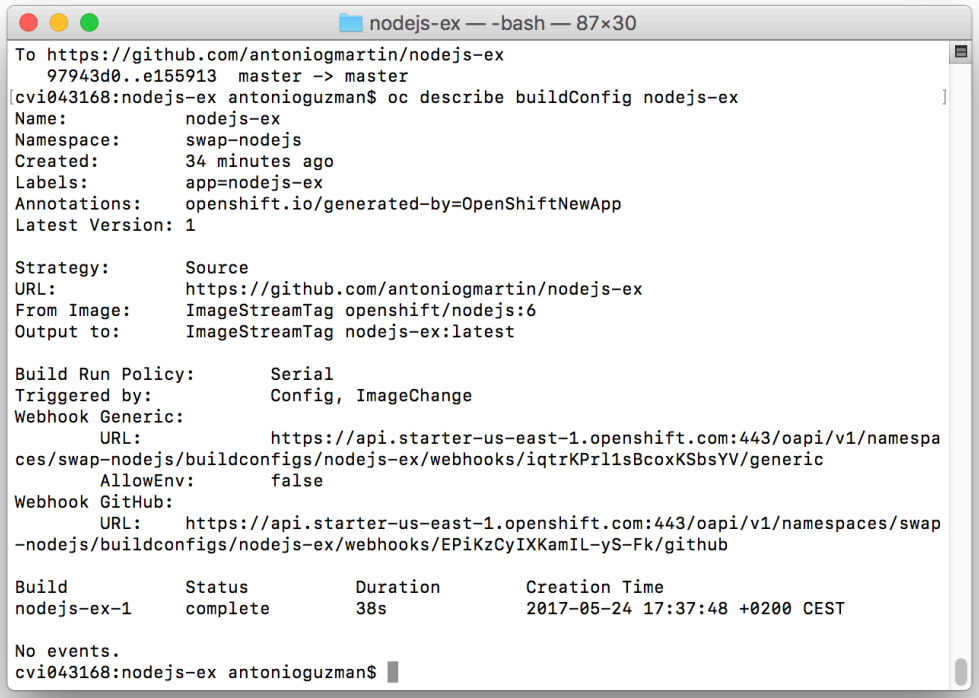
```
oc expose svc/<nombre_servicio>
```

Visión del Proyecto antes del cambio “Hola mundoXX”

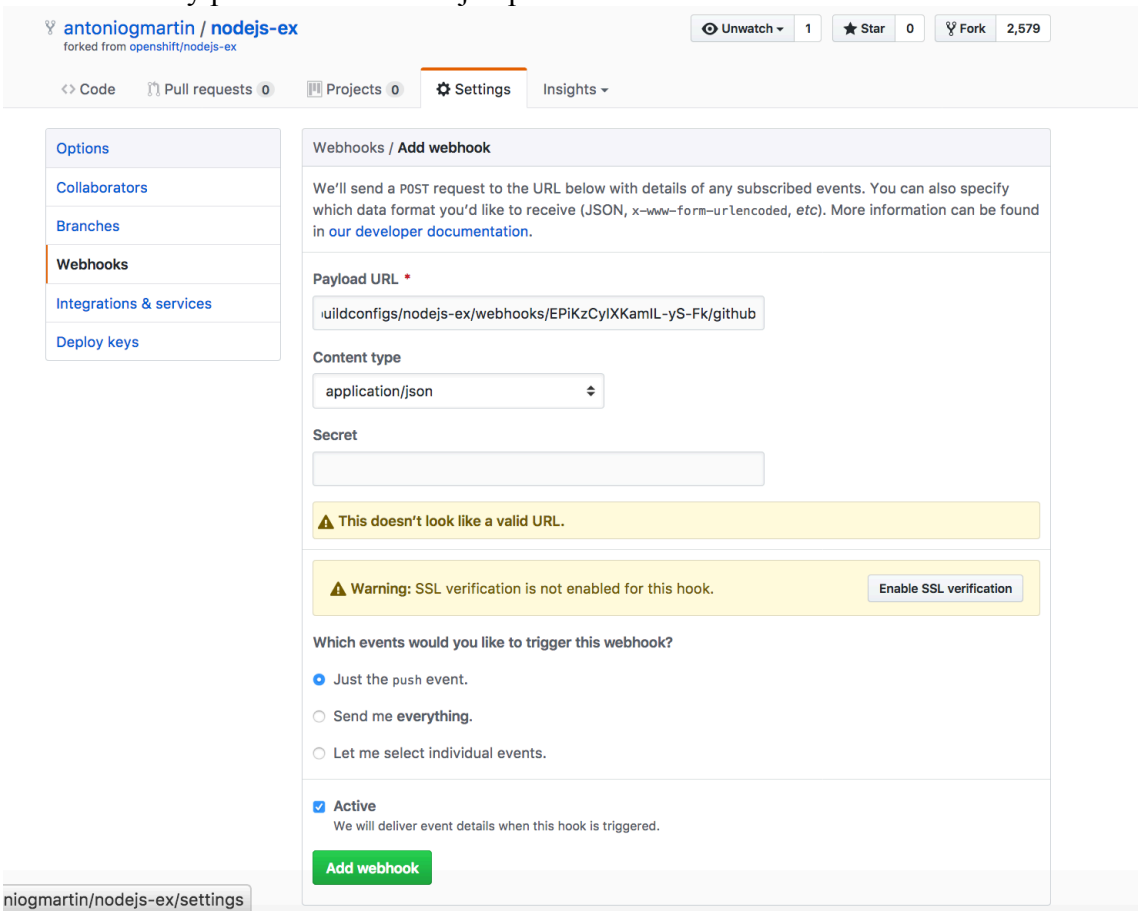


Ahora podremos acceder al proyecto a través de una url autogenerada.

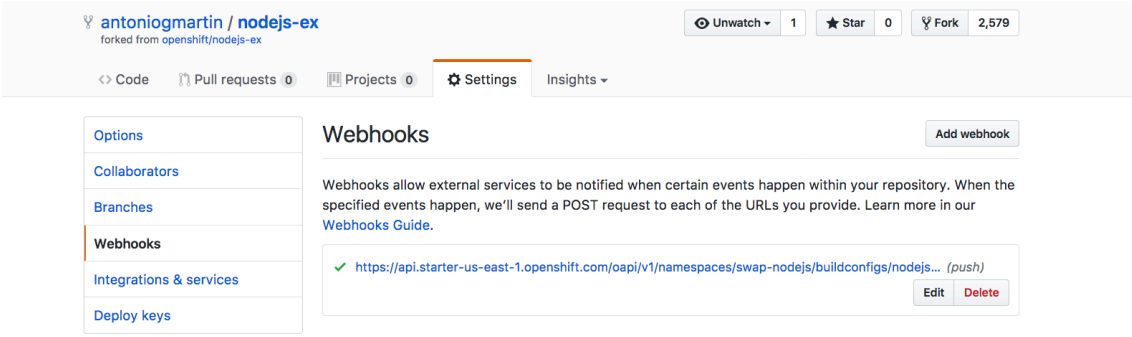
```
oc describe buildConfig <nombre_proyecto>
```



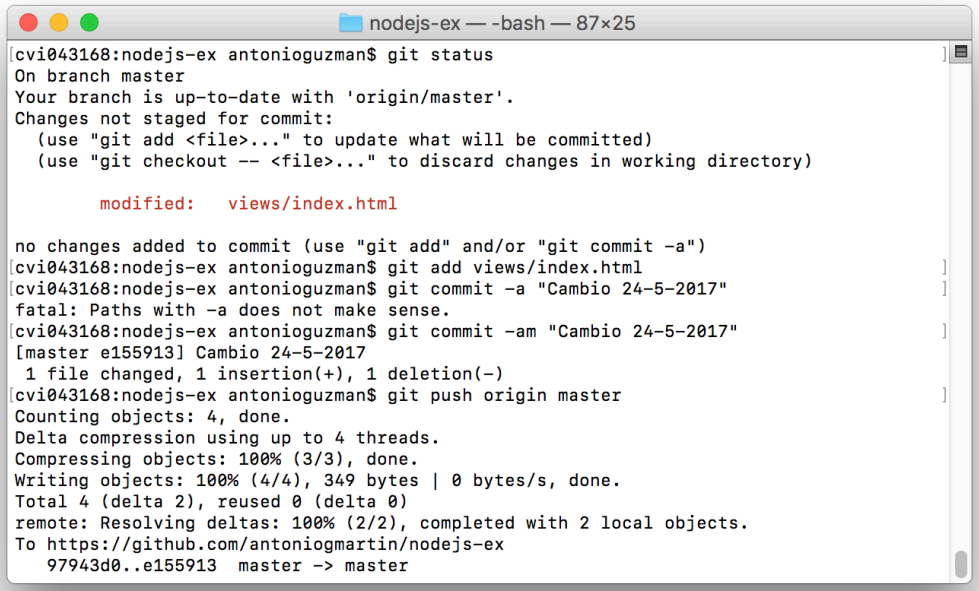
Copiar webhook a github ,luego realizar cambios en nuestro proyecto y por último hacer commit y push de ellos. Por ejemplo modificando index.html



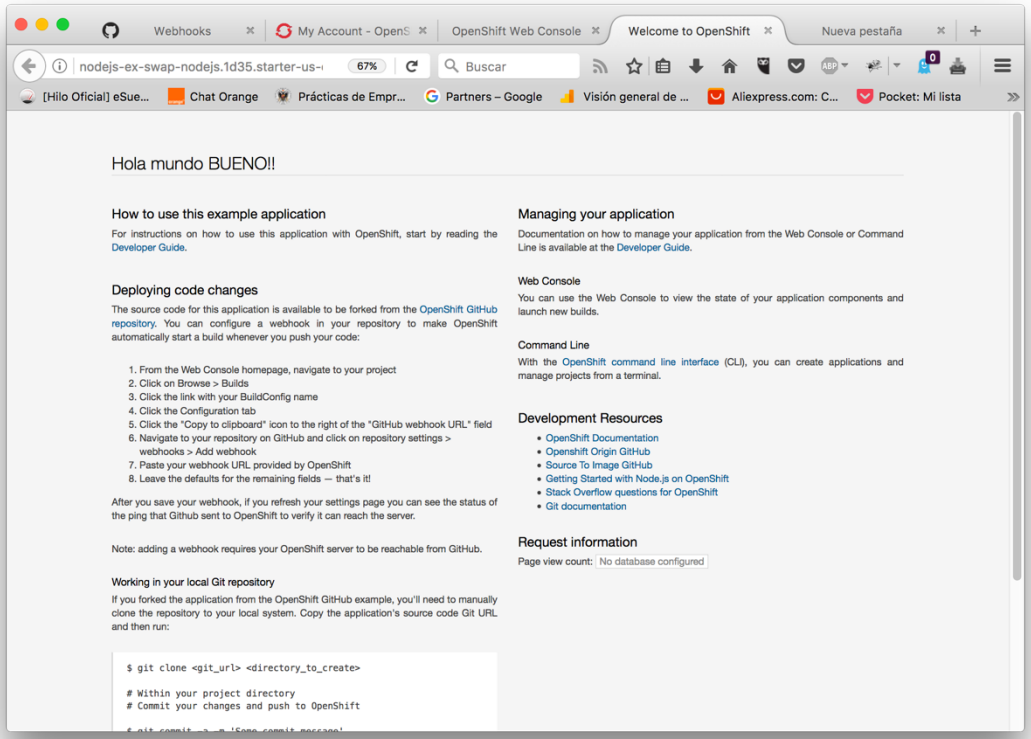
EL webhook funciona correctamente (símbolo verde)




```
git add views/index.html
git push origin master
git commit -am "comentario"
git push origin master
```



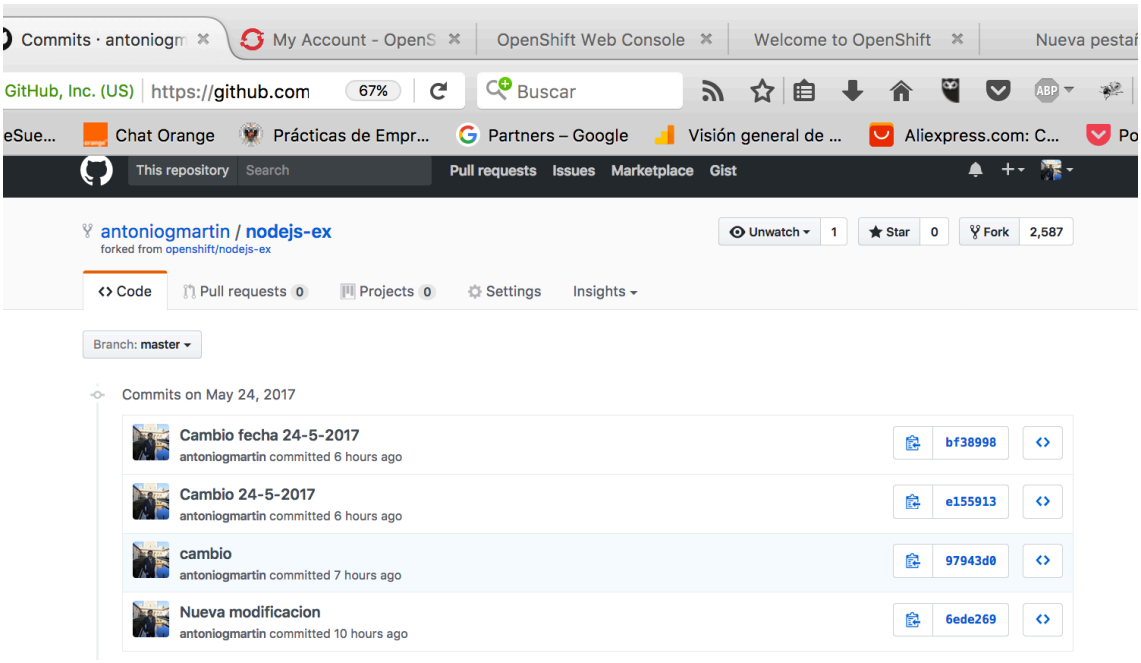
Debemos hacer redeliver hasta que se produzca el cambio. Después accedemos a la url de nuestro proyecto



Puede comprobar el estado actual del servicio en este enlace.

http://nodejs-ex-swap-nodejs.1d35.starter-us-east-1.openshiftapps.com/

También puede observar los commits que he realizado el 24/5/17 en:
https://github.com/antoniogmartin/nodejs-ex/commits/master



9. El desarrollo de la app pasa automáticamente una vez la imagen de la aplicación esté disponible.Para monitorizar su estado vemos los pods que están levantados y los servicios

```
oc get pods
```

10. Creación base de datos MongoDB ejecutando un contenedor con un volumen efímero

```
oc new-app \
    -e MONGODB_USER=admin \
    -e MONGODB_PASSWORD=save \
    -e MONGODB_DATABASE=save \
    -e MONGODB_ADMIN_PASSWORD=supersave \
    mongodb:2.6
```

```
MacBook-Pro-de-Antonio:~ antonioguzman$ oc get po
NAME                READY     STATUS      RESTARTS   AGE
mongodb-1-tvxrp     1/1       Running     0          1m
nodejs-ex-1-build   0/1       Completed   0          6h
nodejs-ex-2-build   0/1       Completed   0          5h
nodejs-ex-2-nkrz3   1/1       Running     0          5h
MacBook-Pro-de-Antonio:~ antonioguzman$
```

11. Podremos conectarnos a nuestra base de datos Mongo con el siguiente comando:

```
oc rsh <nombre_po_bd>
```

```
mongo -u admin -p save save \
 MongoDB shell version: 2.4.9
```

```
sh-4.2$ mongo -u admin -p save save
MongoDB shell version: 2.6.9
connecting to: save
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
> exit
bye
sh-4.2$ mongo -u admin -p save save
MongoDB shell version: 2.6.9
connecting to: save
> db
save
>
```

Comandos de administrador oc adm.

```
Para borrarlo todo : oc delete all --all
```