

Título



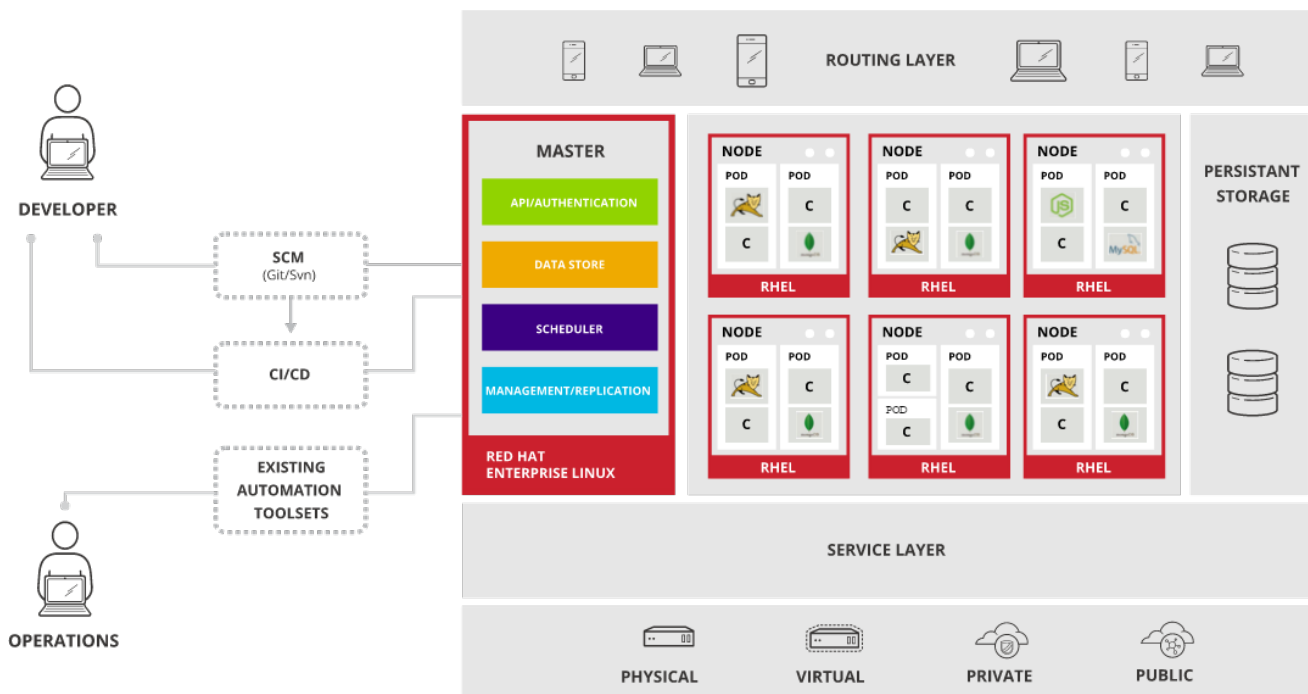
Rafael Martín Guerrero

Introducción

OpenShift 3 es un PaaS (*Platform as a Service o Plataforma como Servicio*) que utiliza contenedores para la construcción, ejecución y despliegue de aplicaciones. Ha sido desarrollado por Red Hat y actualmente existen tres soluciones disponibles según el ámbito en el que nos movamos:

- **OpenShift Online:** permite crear y ejecutar aplicaciones en el cloud público, cualquiera puede crearse una cuenta gratuita y desplegar su aplicación. Actualmente **no tiene disponible la versión 3**, que es la que trabaja con contenedores.
- **OpenShift Dedicated:** te permite disponer de un cluster de OpenShift gestionado por Red Hat para que despliegues tus aplicaciones.
- **OpenShift Enterprise:** admite construir el PaaS OpenShift sobre la infraestructura privada de tu compañía.

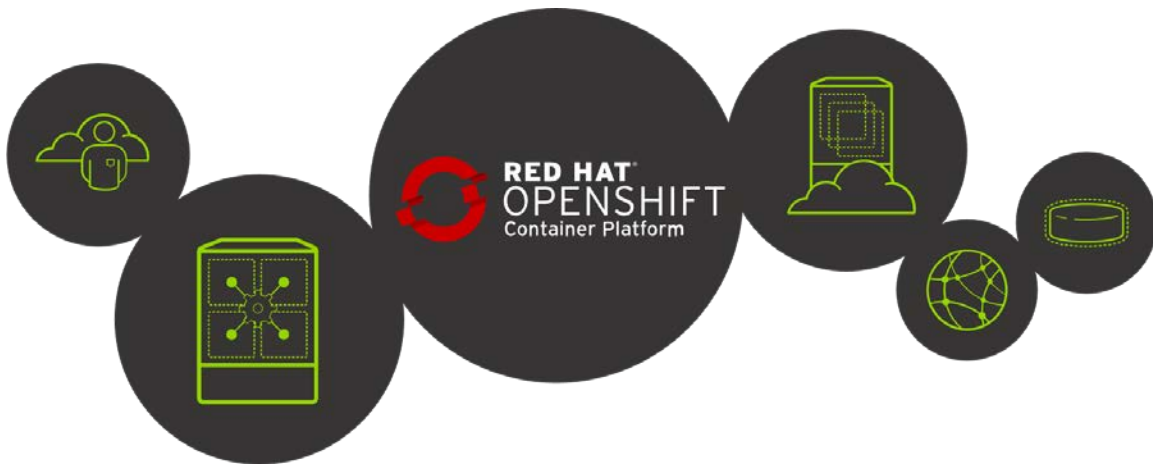
OpenShift Enterprise 3 está construido con tecnologías OpenSource en base a la filosofía de contenedores, utilizando Docker, con Kubernetes como solución de orquestación y gestión y sobre la base de Red Hat Enterprise Linux.



Contenido

Empezando con Openshift	3
Configurando nuestro proyecto	4
Crear proyecto con GIT.....	6

Empezando con Openshift



El primer paso es crearse una cuenta en su plataforma desde [aquí](#), en nuestro caso iniciaremos sesión con GitHub. Una vez que se haya confirmado la creación de la cuenta, podremos empezar a crear nuestro **proyecto**.

New Project

*** Name**

A unique name for the project.

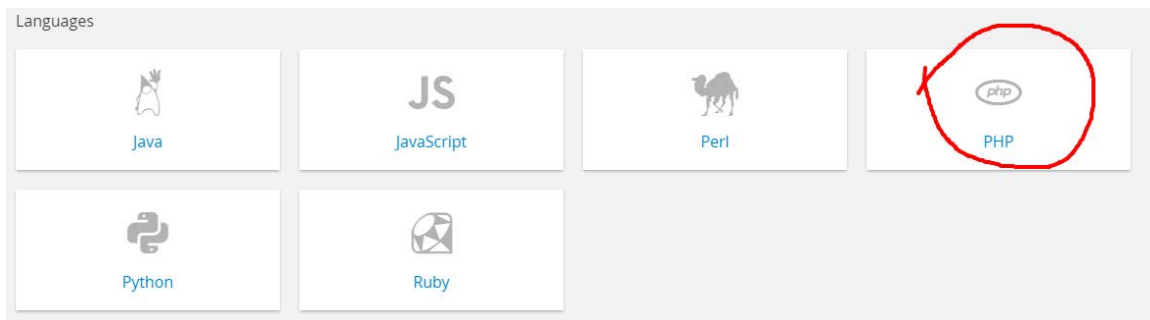
Display Name

Description

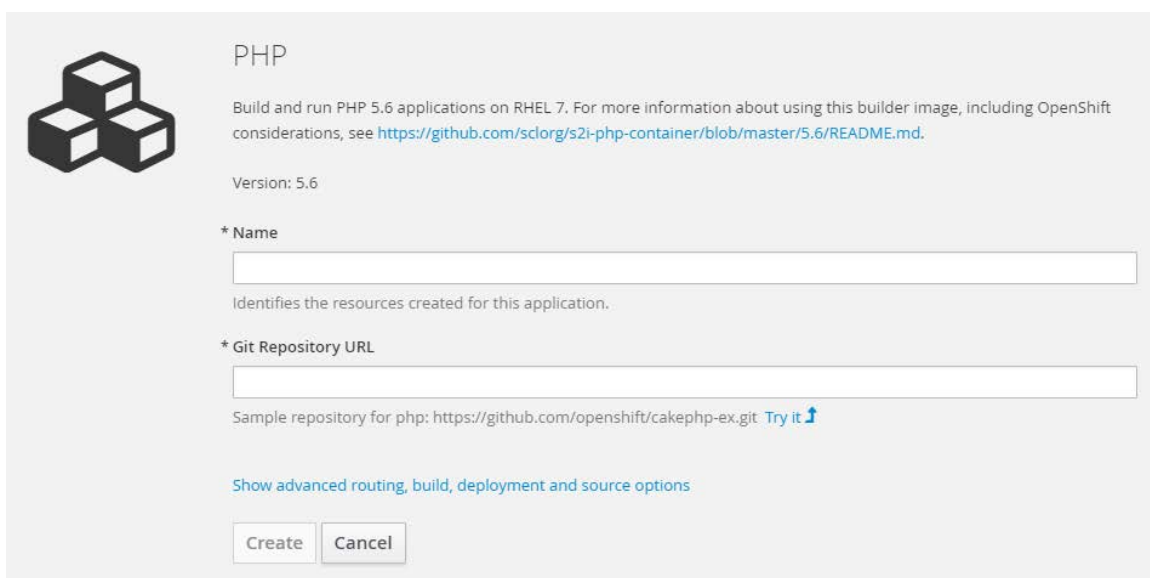
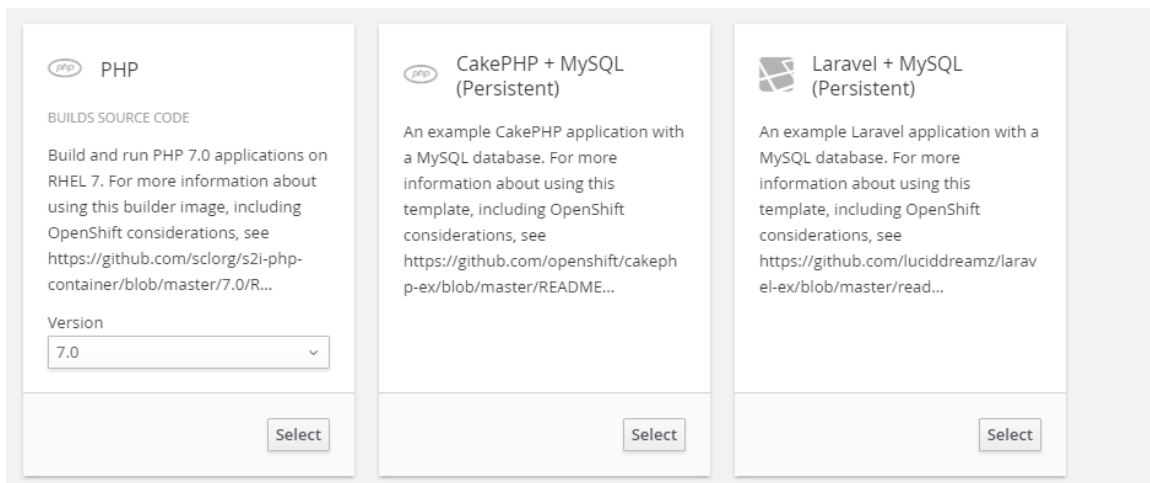
CreateCancel

Configurando nuestro proyecto

Una vez creado nuestro proyecto, tendremos que seleccionar con qué lo vamos a realizar, es decir con qué recursos vamos a trabajar. Para ello podemos elegir entre lenguajes como php, ruby, java... entre otros.



En nuestro caso usaremos php sin frameworks.



Le daremos el nombre que queramos y añadiremos la url de nuestro proyecto que tengamos creados en git, en mi caso añadiré: <https://github.com/rafots/proyectoDespliegue>.

Una vez añadido la opción deseada a nuestro proyecto, podremos empezar a montarlo, para ello nos descargamos OC (Openshift Console) , extraemos la carpeta y el archivo .exe resultante lo metemos en las variables de entorno de nuestro sistema para poder usarlo en la consola, en mi caso he creado una carpeta en C:\oc y he añadido ahí el archivo.

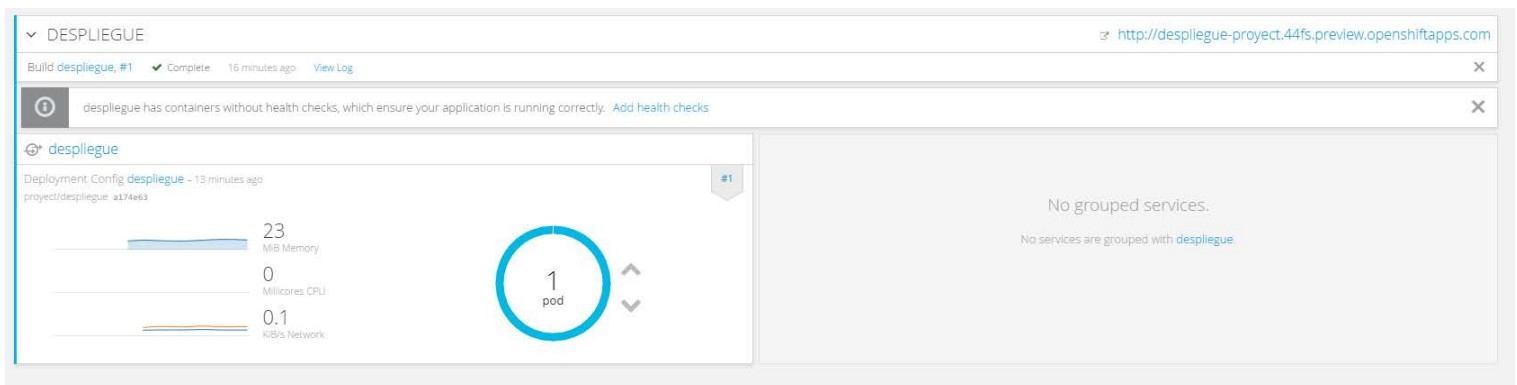
Una vez instalado ejecutaríamos la siguiente línea:

```
oc login https://api.preview.openshift.com --  
token=YdxlejYXkWmSCqd4hJ_9g0H2mmAjqz7wIGA8pifk
```

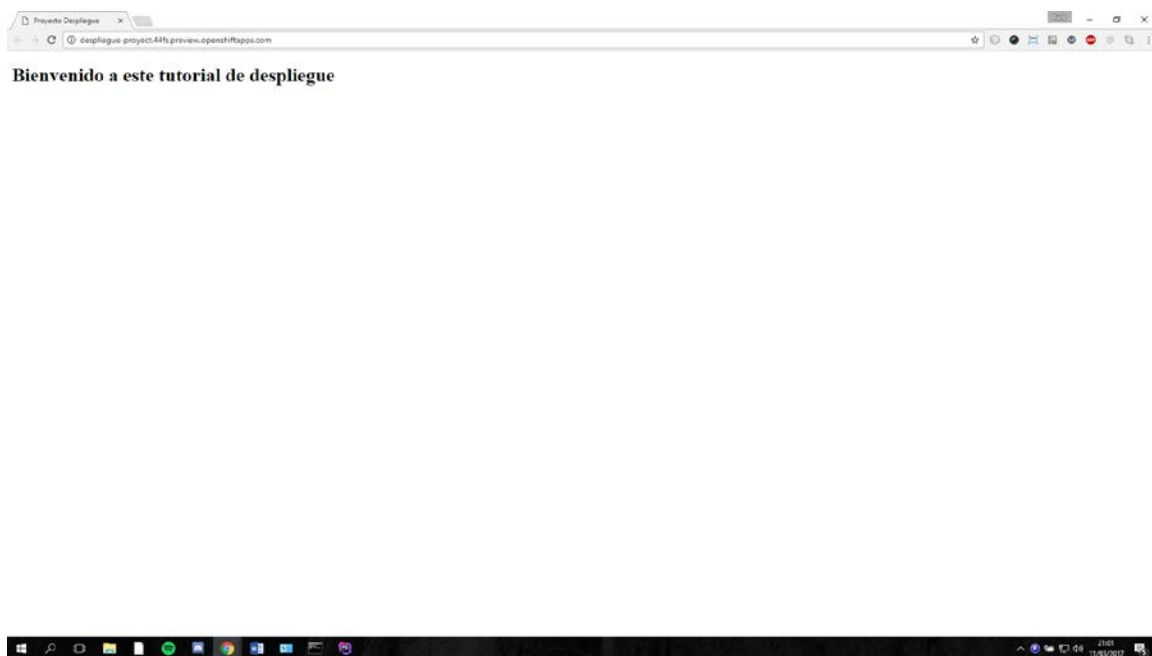
El token es una forma de contraseña también

Mediante la consola, se nos seleccionara el Proyecto que hemos creado y con 'oc status' podremos ver el estado del proyecto , por si hubiese algun error o alguna anomalía.

Una vez creado, podremos irnos a la pagina principal del Proyecto en Openshift, y tendremos que esperar que se realizan las configuraciones y se suba el repositorio , una vez realizado todo correctamente nos debería de salir algo como esto:



Y cuando entrásemos en <http://despliegue-proyect.44fs.preview.openshiftapps.com> tendríamos nuestro proyecto web funcionando perfectamente:



Crear proyecto con GIT

En este punto explicaré como crear un proyecto en git:


En la pagina oficial de GitHub , una vez nos hemos registrado, le damos a “Start a Project”, y nos saldrá algo parecido a esto:

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 rafots ▼

 /

proyectoPrueba ✓

Great repository names are short and memorable. Need inspiration? How about **friendly-funicular**.

Description (optional)

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▼

Add a license: None ▼ ⓘ

Create repository

Después de tener creado nuestro proyecto en GitHub, lo primero de todo es realizar las configuraciones para git una vez instalado. Abrimos la consola de git y escribimos:

- git config --global user.name “Rafa”
- git config --global user.email rafapacense@gmail.com

Una vez hecho esto, supongamos que el proyecto que queremos subir lo tenemos en una carpeta de nuestro ordenador, lo que tendríamos que hacer sería, desde la consola de git(recordemos que utiliza los mismos comandos que en Linux) con cd “rutaProyecto” nos situamos dentro de él, y ejecutamos el siguiente comando: git init.

```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba
$ git init
Initialized empty Git repository in C:/Users/Rafa/Desktop/proyecto prueba/.git/
```

Como vemos se ha inicializado nuestro repositorio.

Ahora realizamos lo siguiente, con git add . , añadimos todos los archivos y carpetas que haya en ese proyecto y realizamos un git commit -m “Mi primer commit” para marcar un punto en el proyecto.


```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git add .

Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git commit -m "Mi primer Commit"
[master (root-commit) cd02686] Mi primer Commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
```

Después de haber hecho nuestro comit, conectaremos con nuestro proyecto en GitHub, para ello haremos lo siguiente: `git remote add origin https://github.com/usuario/nombreProyecto.git`

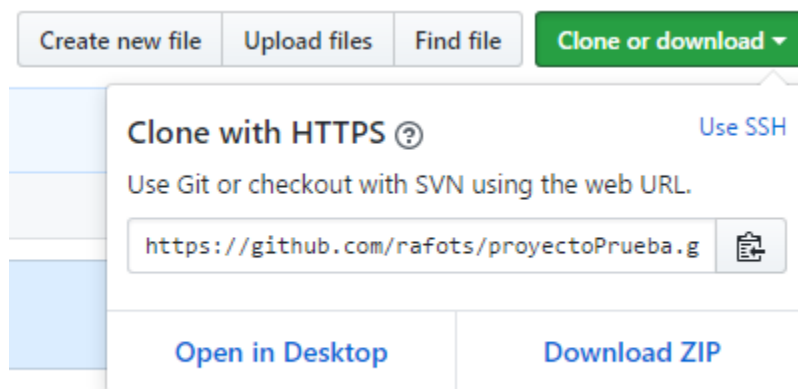
```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git remote add origin https://github.com/rafots/proyectoPrueba.git
```

Y por último subiríamos el commit(que es el que tiene añadido los archivos de nuestro repositorio creado al principio) con : `git push origin master`:

```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 216 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/rafots/proyectoPrueba.git
* [new branch]      master -> master
```

Ya tendríamos en la página oficial de github nuestro proyecto subido con los archivos que hemos creado.

Para descargar un proyecto de git a nuestro ordenador tenemos dos opciones:



Lo descargamos como zip o bien usamos :

`git clone https://github.com/rafots/proyectoPrueba.git`

Ahora bien, imaginaos que varias personas están trabajando en un mismo proyecto, alguien realiza un cambio , pero obviamente los demás no lo pueden ver aunque él haya hecho un push, entonces. ¿Qué habría que hacer para implementar en tu repositorio los nuevos cambios? :

Git fetch y git pull.

```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/rafots/proyectoPrueba
   cd02686..f7fd346  master       -> origin/master
```

Si hubiese cambios en el proyecto cuando ejecutamos el fetch nos saldría algo como esto, en el caso de que no haya nada nuevo no aparecería nada.

```
Rafa@DESKTOP-LIVBIBG MINGW64 ~/Desktop/proyecto prueba (master)
$ git pull origin master
From https://github.com/rafots/proyectoPrueba
 * branch                master       -> FETCH_HEAD
Updating cd02686..f7fd346
Fast-forward
 hola.html | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 hola.html
```

Después , con el pull, como vemos se ha añadido a nuestro repositorio un nuevo archivo, es decir ha actualizado nuestro proyecto y como el compañero usó un push , con pull hemos recogido los cambios.