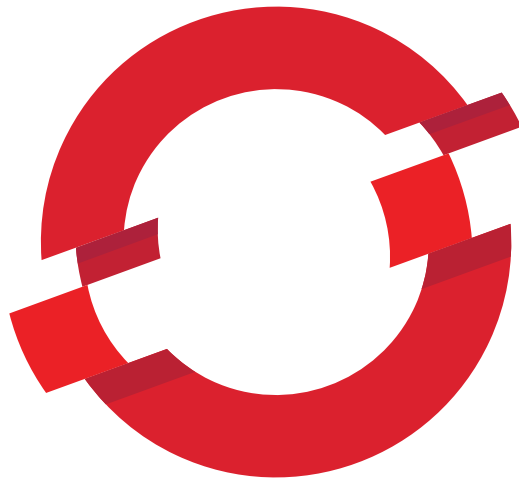


OpenShift v3

Minishift

**Develop Applications Locally in a
Containerized OpenShift Cluster**



Índice

Introducción	3
Cómo surge el proyecto Minishift	4
Hipervisores	5
Pre-requisitos	5
Instalación y configuración de Minishift	6
Instalación del driver docker-machine para KVM.....	6
Red 'default' de KVM.....	7
Minishift - Ciclo de vida	8
Minishift start.....	8
Minishift stop.....	8
Minishift delete.....	8
Administración de Minishift	9
Creación del cluster OpenShift	11
Interactuando con OpenShift	12
Funcionamiento interno de Minishift	15
Configuración en de la máquina en KVM.....	15
Configuración en de las redes en KVM.....	16
Directorios generados en el host anfitrión.....	19
Minishift Docker daemon	21
Command Line Tools	22
Conclusión	35
Credits	36

Introducción

Minishift es una herramienta que proporciona y gestiona un cluster OpenShift de un solo nodo dentro de una máquina virtual. Está optimizada para flujos de trabajo en entornos de desarrollo y requiere un hipervisor para iniciar la máquina virtual en la que se proporciona el cluster. Minishift utiliza [libmachine](#) para la administración de la máquina, [cluster up](#) para el aprovisionamiento del cluster y [OpenShift Origin](#) para ejecutar el cluster.

Minishift es un proyecto open source dedicado a desarrollar y dar soporte a Minishift. El código fuente está basado en un fork del proyecto original [Minikube](#).

Este proyecto se ha llevado a cabo sobre el sistema operativo Centos 7.3 GNU/Linux. El proyecto Minishift es reciente y está en continuo desarrollo por parte de la comunidad que lo mantiene. Como consecuencia, las versiones de Minishift que han sido usadas durante el desarrollo de mi proyecto han ido cambiando junto con las del proyecto Minishift. Finalmente, este proyecto ha terminado por desarrollarse en la que actualmente (16/06/2016) es la última versión disponible, Minishift 1.1.0

Las diferentes formas que existen actualmente para desplegar Openshift localmente serán comentadas. Los conceptos y el funcionamiento interno de Minishift serán explicados, así como se realizarán despliegues de aplicaciones usando tanto la consola web de OpenShift como el cliente de línea de comandos.

Cómo surge el proyecto Minishift

La comunidad de OpenShift comenzó proporcionando una máquina virtual de OpenShift Origin que usaba como imagen base un box de Vagrant y soportaba VirtualBox como único hipervisor. Permitía ejecutar un cluster de OpenShift localmente, simplemente ejecutando un par de comandos de Vagrant, abstrayendo al desarrollador o usuario de realizar cualquier tipo de configuración. Como requisitos del entorno, tan sólo debían estar instalados Vagrant y VirtualBox.

Una vez que la máquina había iniciado, se aprovisionaba a través de una serie de scripts con toda la configuración necesaria que realizaba automáticamente el despliegue y configuración de OpenShift. Si el Vagrantfile usado para crear la máquina no era personalizado, por defecto la máquina virtual se creaba con recursos de 2 vCPUs y 5GiB de memoria RAM.

Esta forma de desplegar un entorno de desarrollo local con OpenShift, dejó de tener soporte como "All-In-One Virtual Machine" en la versión 1.3.0 de OpenShift Origin. Surgieron problemas consistentes para que Vagrant funcionara correctamente en las tres plataformas (Linux, Mac y Windows) y finalmente no era factible soportar esta configuración.

Mientras tanto, surgió un movimiento dentro de la comunidad de Kubernetes para crear MiniKube. Desde RedHat vieron el trabajo y se decidió iniciar el proyecto Minishift, proyecto que actualmente está compuesto por un equipo de trabajo con más de 30 colaboradores. Por ello, se decidió dirigir a los usuarios a Minishift en lugar de seguir usando "All-In-One". Inicialmente el proyecto salió en versión beta, aunque funcionando mejor que "All-In-One" y además actualizado a la versión de OpenShift Origin 1.4.1 (OpenShift 3.4). El proyecto cumplió con todos los casos de uso originales que se tuvieron con "All-In-One", con la ventaja añadida de tener ahora un equipo que lo mantiene.

Actualmente, es posible todavía descargar el box desde los repositorios de Vagrant y ejecutar la máquina virtual 'All-in-one'. No es recomendable.

Hipervisores

A diferencia del proyecto original, Minishift está soportado en varios hipervisores dependiendo del sistema operativo base que se utilice. Minishift requiere un hipervisor para ejecutar la máquina virtual que va a contener OpenShift. En este caso de uso, nos vamos a centrar en los hipervisores soportados para Minishift en sistemas operativos GNU/Linux.

- KVM

Es el hipervisor que se utilizar por defecto. Las versiones más recientes de **qemu-kvm** y **libvirt** se encuentran instaladas por defecto en CentOS. Si los paquetes no estuvieran instalados, habría que realizar su instalación:

```
~]# yum install qemu-kvm libvirt virt-manager
```

El uso de KVM requiere además de pasos específicos de instalación y configuración que se describen en la sección de instalación del driver docker-machine para KVM.

- VirtualBox

No requiere ningún tipo de configuración extra.

Pre-requisitos

El usuario que vaya a realizar la instalación de Minishift y todos los componentes necesarios, deberá pertenecer al grupo wheel para ejecutar comandos con privilegios de superusuario.

```
~]# usermod -a -G wheel fjhidalgo
~]$ newgrp wheel
~]$ groups
fjhidalgo wheel
```

Activación del repositorio EPEL:

```
~]$ sudo yum --enablerepo=extras install epel-release
```

Desactivar SELinux:

```
~]$ sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config
```

Instalación y configuración de Minishift

Descarga del fichero correspondiente a mi distribución:

```
~]$ curl -L
https://github.com/minishift/minishift/releases/download/v1.1.0/minishift-1.1.0-linux-amd64.tgz -o /tmp/minishift-1.1.0.tgz
~]$ mkdir /tmp/minishift/

~]$ tar -zxf /tmp/minishift-1.1.0.tgz -C /tmp/minishift && rm -rf
/tmp/minishift-1.1.0.tgz
~]$ ls /tmp/minishift/
LICENSE  minishift  README.adoc
```

Tras desempaquetar y descomprimir Minishift, el contenido es alojado en un nuevo directorio, dentro de /opt:

```
~]$ sudo mkdir /opt/minishift/
~]$ sudo chown -R fjhidalgo:fjhidalgo /opt/minishift/
~]$ mv /tmp/minishift/* /opt/minishift/.
```

El resultado es un fichero binario ejecutable. Para manejarlo adecuadamente se creará un enlace simbólico al directorio /usr/local/bin:

```
~]$ sudo ln -s /opt/minishift/minishift /usr/local/bin/
```

Instalación del driver docker-machine para KVM

Minishift utiliza Docker Machine y su arquitectura para proporcionar una forma consistente de administrar la máquina virtual de OpenShift.

El siguiente error se produce tras intentar realizar la creación de la máquina virtual sin realizar previamente la instalación del driver de **docker-machine**:

```
E0511 19:46:18.244004      8749 start.go:171] Error starting the VM:
Driver "kvm" not found. Do you have the plugin binary "docker-
machine-driver-kvm" accessible in your PATH?. Retrying.

Error starting the VM:  Driver "kvm" not found. Do you have the
plugin binary "docker-machine-driver-kvm" accessible in your PATH?

Driver "kvm" not found. Do you have the plugin binary "docker-
machine-driver-kvm" accessible in your PATH?
```

La forma correcta de instalar y ejecutar el binario de KVM para CentOS sería la siguiente:

```
~]$ sudo curl -L https://github.com/dhiltgen/docker-machine-kvm/releases/download/v0.7.0/docker-machine-driver-kvm -o /usr/local/bin/docker-machine-driver-kvm

~]$ sudo chmod +x /usr/local/bin/docker-machine-driver-kvm
```

Aunque existen versiones más recientes, se utiliza la versión 0.7.0 porque es la que ha sido probada con Minishift y que funciona correctamente.

Es conveniente que el usuario que cree la máquina virtual se añada al grupo libvirt, para así no necesitar sudo. De esta manera podría usar en KVM la sesión de SYSTEM (root).

```
~]$ sudo usermod -a -G libvirt fjhidalgo
```

Para actualizar la sesión actual y aplicar el cambio de grupo sin tener que cerrar sesión:

```
~]$ newgrp libvirt
~]$ groups
fjhidalgo wheel libvirt
```

Red 'default' de KVM

Al intentar crear la máquina de Minishift, se produce el siguiente error debido a que la red 'default' de KVM no está activa. Esta red es necesaria para dotar al cluster de la dirección IP que será usada para manejarlo, bien desde el navegador web o vía ssh.

```
Starting local OpenShift cluster using 'kvm' hypervisor...
E0507 14:12:36.874356      4762 start.go:171] Error starting the VM:
Error creating the VM. Error creating machine: Error in driver
during machine creation: [Code-55] [Domain-19] Requested operation
is not valid: network 'default' is not active. Retrying.
E0507 14:12:36.916564      4762 start.go:171] Error starting the VM:
Error starting stopped host: [Code-55] [Domain-19] Requested
operation is not valid: network 'default' is not active. Retrying.
Error starting the VM: Error creating the VM. Error creating
machine: Error in driver during machine creation: [Code-55]
[Domain-19] Requested operation is not valid: network 'default' is
not active
Error starting stopped host: [Code-55] [Domain-19] Requested
operation is not valid: network 'default' is not active
```

Para configurar que la red 'default' arranque automáticamente en cada inicio:

```
~]$ virsh -c qemu:///system net-autostart default
Network default marked as autostarted
```

Para ver las redes disponibles y arrancar la red 'default':

```
~]$ virsh -c qemu:///system
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # net-list --all
Name                               State      Autostart  Persistent
-----
default                             inactive  yes        yes

virsh # net-start --network default
Network default started
```

Minishift - Ciclo de vida

Durante el uso de Minishift, se interactúan con 2 componentes:

- la máquina virtual creada por Minishift
- el cluster de OpenShift provisionado dentro de la máquina virtual

Minishift start

El comando 'minishift start' crea y configura la máquina virtual y provisiona OpenShift dentro de la máquina virtual. Descarga además algunos componentes que son necesarios para ejecutar la máquina virtual.

Minishift stop

El comando minishift stop para el cluster OpenShift y apaga la máquina virtual, preservando el estado del cluster. Cuando se inicie de nuevo Minishift se restaurará el cluster OpenShift, lo que permitirá continuar trabajando desde la última sesión. Sin embargo, se deben introducir también los mismos parámetros que se utilizaron con el comando de inicio original.

Minishift delete

El comando minishift delete elimina el cluster OpenShift y además apaga y elimina la máquina virtual. No se conservan datos ni estados. Sólo se mantienen los directorios:

```
$MINISHIFT_HOME/.minishift
$MINISHIFT_HOME/.kube
```

Para completar la eliminación, ambos directorios deben ser eliminados de forma manual.

Administración de Minishift

El comportamiento de Minishift en el momento de ejecución puede controlarse mediante flags, variables de entorno y opciones de configuración persistentes.

Flags

Es posible utilizar 'flags' para especificar opciones de configuración y cambiar el comportamiento durante la ejecución de Minishift. Dicha configuración será la que prevalezca cuando se cree la máquina. Casi todos los comandos tienen flags, algunos de los más utilizados son con el comando `minishift start`: `--cpus`, `--memory` o `--vm-driver`.

Ejemplo:

```
~] $ minishift start --cpus 4 --memory 8096
```

Variables de entorno

Minishift permite especificar valores en los 'flags' a través de variables de entorno.

Ejemplo:

```
~] $ export MINISHIFT_VM_DRIVER=kvm
~] $ export MINISHIFT_CPUS=4
~] $ export MINISHIFT_MEMORY=8096
```

Valores de configuración persistentes

La forma más sencilla de cambiar una opción de configuración persistente es con el subcomando 'config set'.

Ejemplo para establecer la memoria de la máquina virtual a 4096 MiB por defecto:

```
~] $ minishift config set memory 4096
```

Para ver todos los valores de configuración persistentes, se puede usar el subcomando 'view'

```
~] $ minishift config view
- memory: 4096
```

Alternativamente se puede mostrar un valor único con el subcomando 'get':

```
~] $ minishift config get memory
4096
```

Para deshacer los valores de configuración persistentes, es posible usar el subcomando `unset`. Ejemplo:

```
~] $ minishift config unset memory
```

Volúmenes persistentes

Como parte del aprovisionamiento, se crean en la máquina 100 volúmenes persistentes para usarlos en el cluster OpenShift. Esto permite a las aplicaciones poder reclamar volúmenes persistentes y usarlos en los despliegues.

La ubicación de los datos persistentes se determina a través del flag '`--host-pv-dir`' con el comando `minishift start`. Por defecto, los volúmenes se crean dentro de la máquina virtual en el directorio:

```
docker@minishift:~$ ls -l /var/lib/minishift/openshift.local.pv/  
pv0001/  
pv0002/  
pv0003/  
.  
.  
.  
pv0099/  
pv0100/  
registry/
```

Redes

La máquina virtual Minishift está expuesta al sistema con una dirección IP host-only que pertenece a la red 'default' de KVM y que se puede obtener con el comando `minishift ip`.

```
~] $ minishift ip  
192.168.42.82
```

Logs

Para acceder a los logs de OpenShift, se puede ejecutar el siguiente comando justo después de que Minishift haya iniciado:

```
~] $ minishift logs
```

Creación del cluster OpenShift

```
~]$ export MINISHIFT_HOME=/opt/minishift/
~]$ minishift start
Starting local OpenShift cluster using 'kvm' hypervisor...
Downloading ISO 'https://github.com/minishift/minishift-b2d-iso/releases/download/v1.0.2/minishift-b2d.iso'
 40.00 MiB / 40.00 MiB
[=====
=====] 100.00% 0s
Downloading OpenShift binary 'oc' version 'v1.5.1'
 19.96 MiB / 19.96 MiB
[=====
=====] 100.00% 0s
-- Checking OpenShift client ... OK
-- Checking Docker client ... OK
-- Checking Docker version ... OK
-- Checking for existing OpenShift container ... OK
-- Checking for openshift/origin:v1.5.1 image ...
  Pulling image openshift/origin:v1.5.1
  Pulled 0/3 layers, 3% complete
  Pulled 0/3 layers, 63% complete
  Pulled 1/3 layers, 81% complete
  Pulled 2/3 layers, 96% complete
  Pulled 3/3 layers, 100% complete
  Extracting
  Image pull complete
-- Checking Docker daemon configuration ... OK
-- Checking for available ports ... OK
-- Checking type of volume mount ...
  Using Docker shared volumes for OpenShift volumes
-- Creating host directories ... OK
-- Finding server IP ...
  Using 192.168.42.82 as the server IP
-- Starting OpenShift container ...
  Creating initial OpenShift configuration
  Starting OpenShift using container 'origin'
  Waiting for API server to start listening
  OpenShift server started
-- Adding default OAuthClient redirect URIs ... OK
-- Installing registry ... OK
-- Installing router ... OK
-- Importing image streams ... OK
-- Importing templates ... OK
-- Login to server ... OK
-- Creating initial project "myproject" ... OK
-- Removing temporary directory ... OK
-- Checking container networking ... OK
-- Server Information ...
```

```
OpenShift server started.
The server is accessible via web console at:
  https://192.168.42.82:8443

You are logged in as:
  User:      developer
  Password: developer

To login as administrator:
  oc login -u system:admin
```

OpenShift se ha iniciado correctamente.

Interactuando con OpenShift

Existen varias posibilidades de acceder / interactuar con la máquina.

Console

Ejecutando el siguiente comando en una shell se obtiene la dirección URL de acceso a la consola Web de OpenShift:

```
~] minishift console --url
https://192.168.42.82:8443
```

Alternativamente, ejecutando el siguiente comando se abre directamente la consola Web de OpenShift en el navegador:

```
~] minishift console
```

Login

El cluster utiliza por defecto [AllowAllPasswordIdentityProvider](#) para la autenticación contra el cluster local. Para iniciar sesión como administrador, se utiliza la cuenta system:

```
~] oc login -u system:admin
```

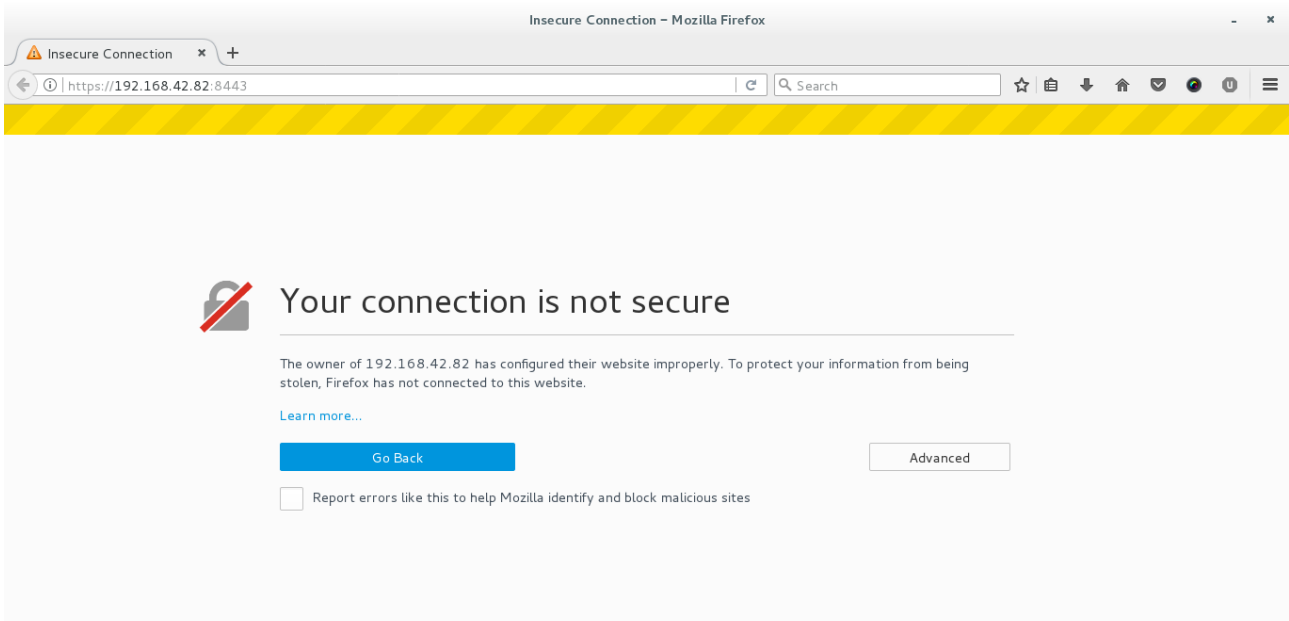
SSH

Es posible iniciar sesión o ejecutar un comando remoto en la máquina virtual Minishift con ssh:

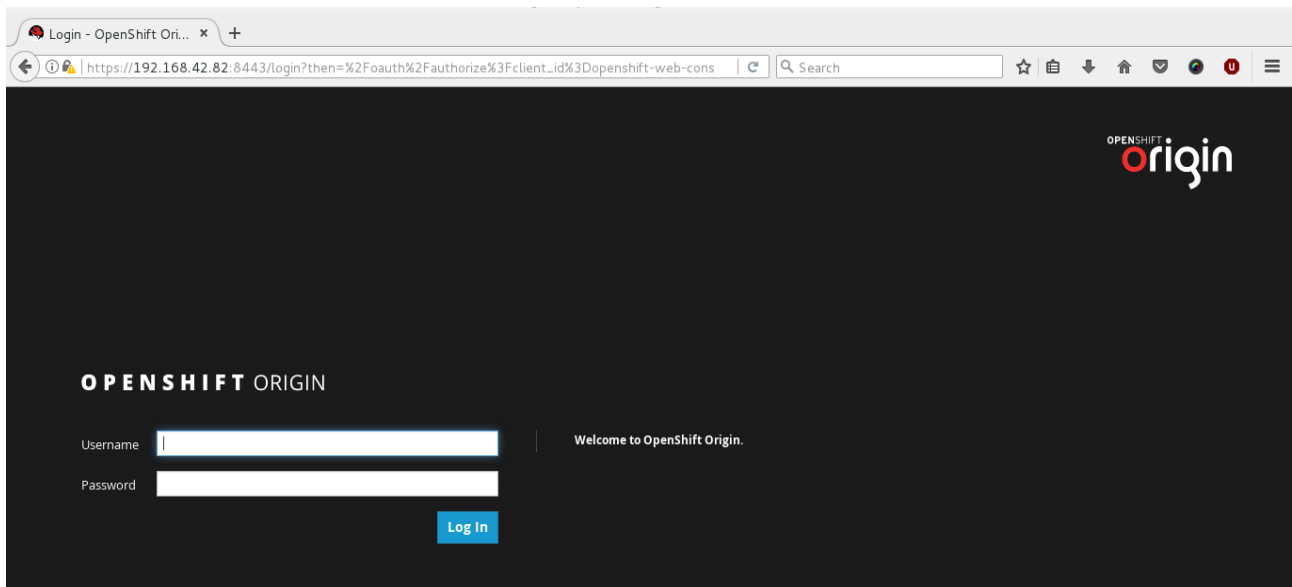
```
~] minishift ssh "echo Hello World"
Hello World
~] minishift ssh
docker@minishift:~$
```

Consola web

Cuando se abre por primera vez la consola web desde el navegador, es necesario confirmar la excepción de seguridad para aceptar el certificado de Minishift:



Una vez se muestra la consola web, es necesario autenticarse para acceder. Se muestra la aplicación que está desplegada dentro del contenedor de OpenShift Origin (v1.5.1)

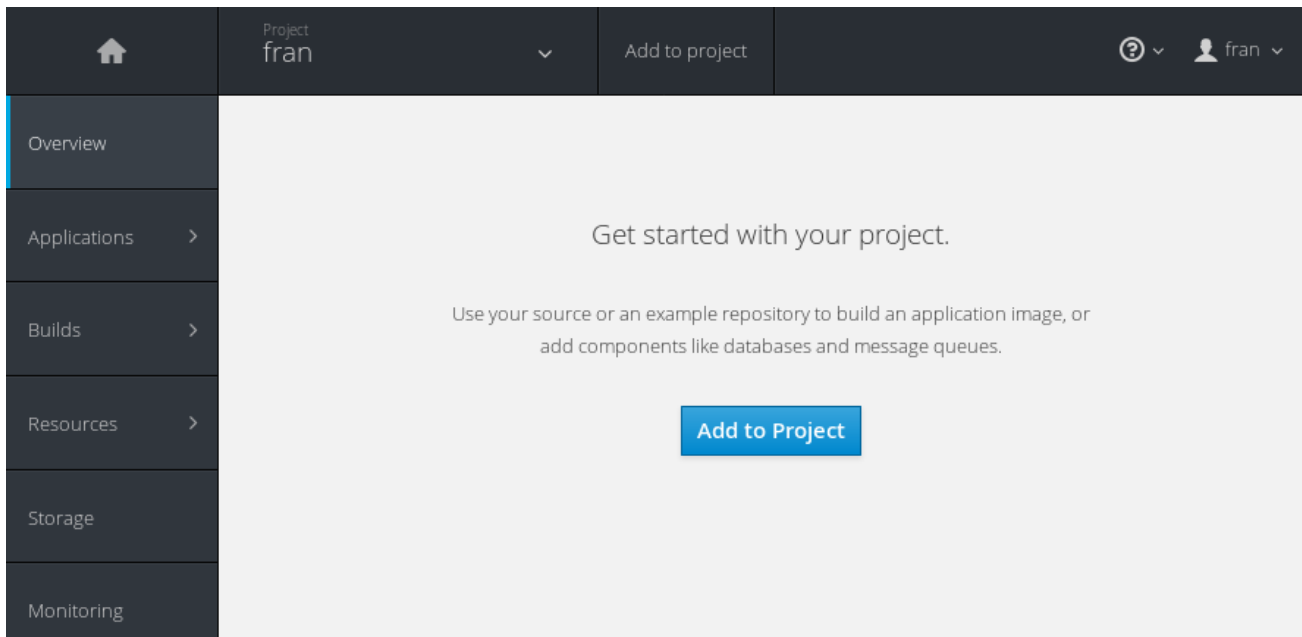


Las credenciales de acceso por defecto son:

```
User:      developer
Password:  developer
```

```
User:      system
Password:  admin
```

Una vez dentro, nos encontramos el siguiente panel web:



Overview - La pestaña Overview (actualmente seleccionada) visualiza el contenido de su proyecto con una vista general sobre cada componente.

Applications - Aquí es posible examinar y realizar acciones en sus pods, servicios rutas e implementaciones.

Builds - El proceso de construcción (build) en OpenShift Origin transforma los parámetros de entrada en un objeto resultante. Se utiliza para transformar el código fuente de una aplicación en una imagen de contenedor que se pueda ejecutar.

Los mecanismos de construcción son:

Source-to-Image (S2I): Es una herramienta para crear imágenes reproducibles de contenedores con formato Docker. Produce imágenes 'ready-to-run' inyectando el código fuente de una aplicación en una imagen de contenedor y ensamblando una imagen nueva.

Pipeline : se usa desde Jenkins. La compilación puede ser iniciada, monitorizada y administrada por OpenShift de la misma forma que cualquier otro tipo de compilación.

Docker : invoca al comando "docker build" y espera a que se genere un repositorio con un Dockerfile con todos los elementos necesarios para producir una imagen en el Docker Registry para poder utilizarla.

Custom: Este mecanismo permite definir una imagen que será responsable de todo el proceso de generación. Usar tu imagen propia permite personalizar todo el proceso.

Resources - Permite ver el consumo de cuota actual y otros recursos.

Storage - Permite ver las solicitudes de volúmenes persistentes y almacenamiento hechas por las aplicaciones.

Monitoring - Permite ver los logs que se han generado con las compilaciones (cada versión de la aplicación), pods e implementaciones, así como todas las notificaciones para todos los objetos del proyecto.

Funcionamiento interno de Minishift

Es una máquina que se crea por defecto con 2 GiB de RAM, 2 VCPUs y 20 GiB de disco.

Configuración en de la máquina en KVM

El fichero XML que define a la máquina virtual Minishift presenta la siguiente configuración:

```
~] $ sudo cat /etc/libvirt/qemu/minishift.xml
<domain type='kvm'>
  <name>minishift</name>
  <uuid>7f81860e-349d-4b33-8a13-1726ce79530e</uuid>
  <memory unit='KiB'>2097152</memory>
  <currentMemory unit='KiB'>2097152</currentMemory>
  <vcpu placement='static'>2</vcpu>
```

El almacenamiento de la máquina virtual se encuentra por defecto en:

```
<devices>
<disk type='file' device='disk'>
<source
file='/opt/minishift/.minishift/machines/minishift/minishift.img' />
</disk>
```

No usa el formato por defecto .qcow2

La imagen que descarga el comando 'minishift start' es usada por la máquina virtual como una Live ISO directamente en modo lectura dentro de una unidad virtual CD-Rom. La imagen contiene el sistema operativo que usará la máquina virtual 'minishift' en KVM.

```
<disk type='file' device='cdrom'>
<source
file='/opt/minishift/.minishift/machines/minishift/boot2docker.iso' />
<readonly/>
</disk>
```

Configuración en de las redes en KVM

La máquina dispone de **2 adaptadores de red**, uno en la red 'default' y otro en la red 'docker-machines'

- La primera red está expuesta al sistema en modo host-only.
- La segunda red se usa para la comunicación interna entre los contenedores.

```
<interface type='network'>
  <mac address='52:54:00:d5:cb:4c' />
  <source network='default' />
  <model type='rtl8139' />
  <address type='pci' domain='' bus='' slot='' function='' />
</interface>

<interface type='network'>
  <mac address='52:54:00:2f:f0:83' />
  <source network='docker-machines' />
  <model type='rtl8139' />
  <address type='pci' domain='' bus='' slot='' function='' />
</interface>
```

Redes Virtuales usadas por la máquina virtual Minishift:

```
~]# sudo ls -lF /etc/libvirt/qemu/networks
autostart/
default.xml
docker-machines.xml
```

El fichero XML que define la red default, presenta la siguiente configuración:

```
~]# sudo cat /etc/libvirt/qemu/networks/default.xml
<network>
  <name>default</name>
  <uuid>107aleba-0d07-4ccc-a652-e7b1ebe75940</uuid>
  <forward mode='nat' />
  <bridge name='virbr0' stp='on' delay='0' />
  <mac address='52:54:00:11:bf:6a' />
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
</network>
```


El dispositivo de red virtual usado por la red default en el equipo anfitrión es virbr0:

```
[fjhidalgo@i7core ~]$ cat /var/lib/libvirt/dnsmasq/virbr0.status
[
  {
    "ip-address": "192.168.122.117",
    "mac-address": "52:54:00:d5:cb:4c",
    "hostname": "minishift",
    "client-id": "01:52:54:00:d5:cb:4c",
    "expiry-time": 1497594063
  }
]
```

El fichero XML que define la red docker-machines, presenta la siguiente configuración (es un fichero autogenerado durante la creación de la máquina virtual minishift):

```
~]$ sudo cat /etc/libvirt/qemu/networks/docker-machines.xml
<network>
  <name>docker-machines</name>
  <uuid>7c360b24-a5aa-4084-b5cb-02b750878a1a</uuid>
  <bridge name='virbr1' stp='on' delay='0' />
  <mac address='52:54:00:b9:55:43' />
  <ip address='192.168.42.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.42.2' end='192.168.42.254' />
    </dhcp>
  </ip>
</network>
```

El dispositivo de red virtual usado por la red docker-machines en el equipo anfitrión es virbr1:

```
~]$ cat /var/lib/libvirt/dnsmasq/virbr1.status
[
  {
    "ip-address": "192.168.42.82",
    "mac-address": "52:54:00:17:c5:a1",
    "hostname": "minishift",
    "client-id": "01:52:54:00:2f:f0:83",
    "expiry-time": 1497594962
  }
]
```



```
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1
192.168.42.0/24 dev eth1 proto kernel scope link src 192.168.42.82
192.168.122.0/24 dev eth0 proto kernel scope link src
192.168.122.132
```

Directorios generados en el host anfitrión

La ejecución del binario '**minishift**' genera los directorios '.minishift' y '.kube' dentro de \$MINISHIFT_HOME.

Directorio .minishift:

```
~]# tree $MINISHIFT_HOME/.minishift
.
|-- addons
|-- cache
|   |-- iso
|   |   |-- minishift-b2d.iso
|   |-- oc
|       |-- v1.5.1
|           |-- oc
|-- ca.pem
|-- cert.pem
|-- certs
|   |-- ca-key.pem
|   |-- ca.pem
|   |-- cert.pem
|   |-- key.pem
|-- config
|   |-- allinstances.json
|   |-- config.json
|-- key.pem
|-- logs
|-- machines
|   |-- minishift
|   |   |-- boot2docker.iso
|   |   |-- config.json
|   |   |-- id_rsa
|   |   |-- id_rsa.pub
|   |   |-- minishift.img
|   |-- minishift.json
|   |-- minishift_kubeconfig
|   |-- server-key.pem
|   |-- server.pem
|-- tmp
```

Directorio '.kube':

```
~] $ tree $MINISHIFT_HOME/.kube
.
|-- 192.168.42.82_8443
|   |-- apps
|   |   |-- v1beta1
|   |   |   |-- serverresources.json
|   |-- authentication.k8s.io
|   |   |-- v1beta1
|   |   |   |-- serverresources.json
|   |-- autoscaling
|   |   |-- v1
|   |   |   |-- serverresources.json
|   |-- batch
|   |   |-- v1
|   |   |   |-- serverresources.json
|   |   |-- v2alpha1
|   |   |   |-- serverresources.json
|   |-- certificates.k8s.io
|   |   |-- v1alpha1
|   |   |   |-- serverresources.json
|   |-- extensions
|   |   |-- v1beta1
|   |   |   |-- serverresources.json
|   |-- policy
|   |   |-- v1beta1
|   |   |   |-- serverresources.json
|   |-- servergroups.json
|   |-- storage.k8s.io
|   |   |-- v1beta1
|   |   |   |-- serverresources.json
|   |-- v1
|   |   |-- serverresources.json
|-- config
```

Minishift Docker daemon

Al estar ejecutando OpenShift en una única máquina virtual, es posible rehusar el demonio de Docker que es usado por Minishift y usarlo para otros casos. Por ejemplo, podemos acelerar el desarrollo local usando el mismo daemon de Docker que usa Minishift.

Es posible configurar dicho daemon y rehusarlo desde la máquina anfitriona contra la máquina virtual Minishift. Así, es posible ejecutar comandos remotamente desde el host contra la máquina virtual Minishift. Para configurar nuestra terminal es necesario previamente cumplir con los siguientes requisitos:

- Tener instalado el cliente de Docker en el host anfitrión.
- Que la máquina de Minishift esté iniciada.
- Usar el siguiente comando para configurar en nuestra terminal el cliente de Docker:

```
[fjhidalgo@administrador ~]$ minishift docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.42.82:2376"
export DOCKER_CERT_PATH="/opt/minishift/.minishift/certs"
export DOCKER_API_VERSION="1.24"
# Run this command to configure your shell:
# eval $(minishift docker-env)

[fjhidalgo@administrador ~]$ eval $(minishift docker-env)
```

Si se desea tener una configuración persistente para poder ejecutar comandos usando el cliente de Docker desde el host anfitrión, tan sólo habría que añadir las variables de entorno al fichero `.profile` del usuario que ejecuta la sesión.

Para probar la conexión, basta con ejecutar cualquier comando de Docker:

```
~]$ docker ps --format 'table {{.Image}}\t{{.Names}}'
IMAGE                                NAMES
k8s_router.8b083074_router              openshift/origin-haproxy-router:v1.5.1
k8s_registry.1252d4e9_docker-registry   openshift/origin-docker-registry:v1.5.1
k8s_POD.a71dfe85_docker-registry        openshift/origin-pod:v1.5.1
k8s_POD.50d4dca2_router                 openshift/origin-pod:v1.5.1
origin                                   openshift/origin:v1.5.1
```

En la terminal deben aparecer la lista de contenedores que se están ejecutando en la máquina virtual de Minishift.

Command Line Tools

Por defecto, cuando la máquina inicia por primera vez, te encuentras logueado en la terminal como:

```
User: developer
Password: developer
```

Para loguearte como administrador:
oc login -u system:admin

Es posible usar con Minishift el cliente de línea de comandos que permite realizar la misma configuración que desde la interfaz web gráfica.

Está disponible en el mismo directorio donde se encuentra la imagen boot2docker.iso

```
~]$ ls .minishift/cache/
iso oc
```

Para usar el cliente de línea de comandos sin indicar su ruta completa, podemos establecer la siguiente configuración con los comandos que establecen la ruta de acceso al binario 'oc':

```
~]$ minishift oc-env
export PATH="/opt/minishift/.minishift/cache/oc/v1.5.1:$PATH"
# Run this command to configure your shell:
# eval $(minishift oc-env)

~]$ eval $(minishift oc-env)
~]$ cat <<EOF >> .profile
export PATH="/opt/minishift/.minishift/cache/oc/v1.5.1:$PATH"
EOF
```

Ejecución del comando:

```
~]$ oc
OpenShift Client

This client helps you develop, build, deploy, and run your applications
on any OpenShift or Kubernetes compatible platform. It also includes the
administrative commands for managing a cluster under the 'adm'
subcommand.

To create a new application, login to your server and then run new-app:

oc login https://mycluster.mycompany.com
oc new-app centos/ruby-22-centos7-https://github.com/openshift/ruby-ex.git
oc logs -f bc/ruby-ex

This will create an application based on the Docker image 'centos/ruby-
22-centos7' that builds the source code from GitHub. A build will start
automatically, push the resulting image to the registry, and a
```

deployment will roll that change out in your project.

Once your application is deployed, use the status, describe, and get commands to see more about the created components:

```
oc status
oc describe deploymentconfig ruby-ex
oc get pods
```

To make this application visible outside of the cluster, use the expose command on the service we just created to create a 'route' (which will connect your application over the HTTP port to a public domain name).

```
oc expose svc/ruby-ex
oc status
```

You should now see the URL the application can be reached at.

To see the full list of commands supported, run 'oc --help'.

Si cierras sesión desde la línea de comandos debes volver a loguearte antes de usar de nuevo el cliente:

```
~]$ oc status
```

```
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get persistentvolumeclaims))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get secrets))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get routes))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get serviceaccounts))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get buildConfigs))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get builds))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get imageStreams))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get pods))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get services))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get replicationcontrollers))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get statefulsets.apps))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get deploymentConfigs))
error: You must be logged in to the server (the server has asked for the client
to provide credentials (get horizontalpodautoscalers.autoscaling))
```

Para hacer login con el usuario system:

```
[fjhidalgo@i7core ~]$ oc login -u system:admin
Logged into "https://192.168.42.82:8443" as "system:admin" using existing
credentials.
```

You have access to the following projects and can switch between them with 'oc project <projectname>':

```
* default
  kube-system
  myproject
  openshift
  openshift-infra
```

Using project "default".

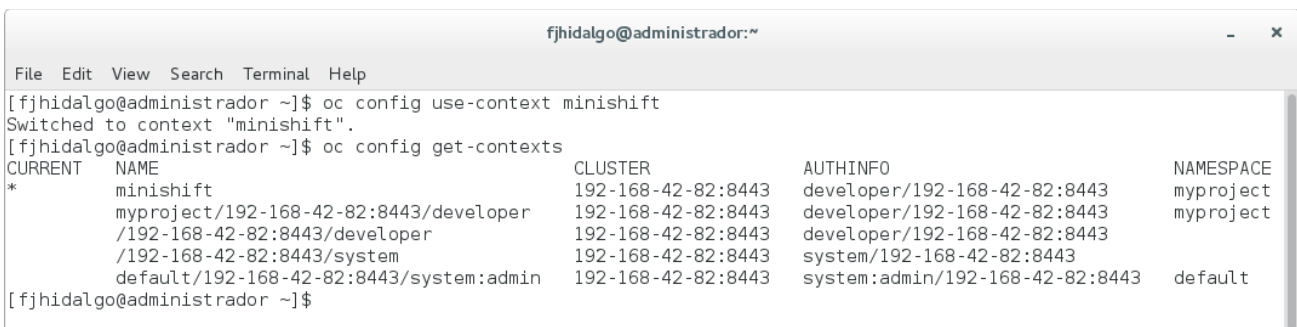
Como parte de la ejecución del comando 'minishift start' se crea un contexto. Este contexto contiene la configuración que permite la comunicación con el cluster OpenShift.

Minishift activa este contexto automáticamente. Si se necesita volver a él después de, por ejemplo, iniciar sesión en otra instancia de OpenShift, se debe ejecutar:

```
~]$ oc config use-context minishift
Switched to context "minishift".
```

El siguiente comando muestra uno o varios contextos creados en el fichero kubeconfig.

```
~]$ oc config get-contexts
```



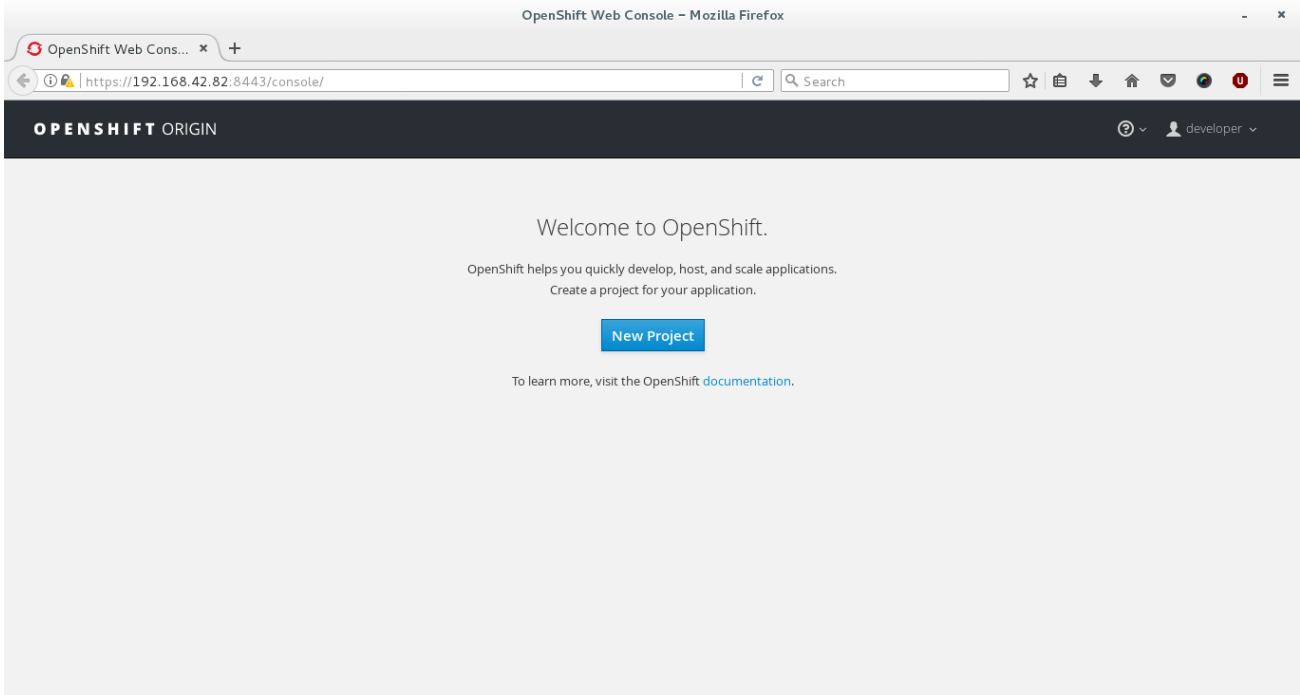
```
fjhidalgo@administrador:~
File Edit View Search Terminal Help
[fjhidalgo@administrador ~]$ oc config use-context minishift
Switched to context "minishift".
[fjhidalgo@administrador ~]$ oc config get-contexts
CURRENT  NAME                                     CLUSTER                AUTHINFO                NAMESPACE
*        minishift                               192-168-42-82:8443     developer/192-168-42-82:8443  myproject
         myproject/192-168-42-82:8443/developer  192-168-42-82:8443     developer/192-168-42-82:8443  myproject
         /192-168-42-82:8443/developer          192-168-42-82:8443     developer/192-168-42-82:8443
         /192-168-42-82:8443/system            192-168-42-82:8443     system/192-168-42-82:8443
         default/192-168-42-82:8443/system:admin 192-168-42-82:8443     system:admin/192-168-42-82:8443  default
[fjhidalgo@administrador ~]$
```

El fichero kubeconfig se encuentra en la siguiente ruta:

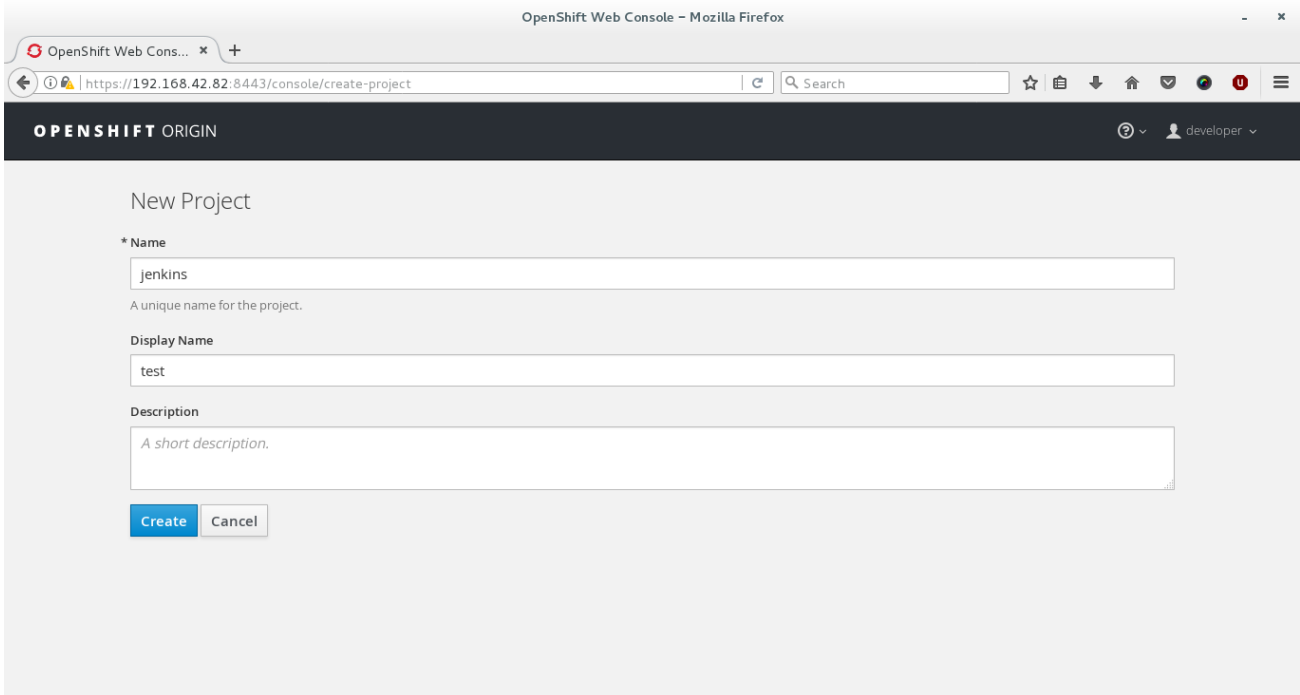
```
$MINISHIFT_HOME/.minishift/machines/minishift_kubeconfig
```


Despliegue de la aplicación Jenkins usando el panel web gráfico

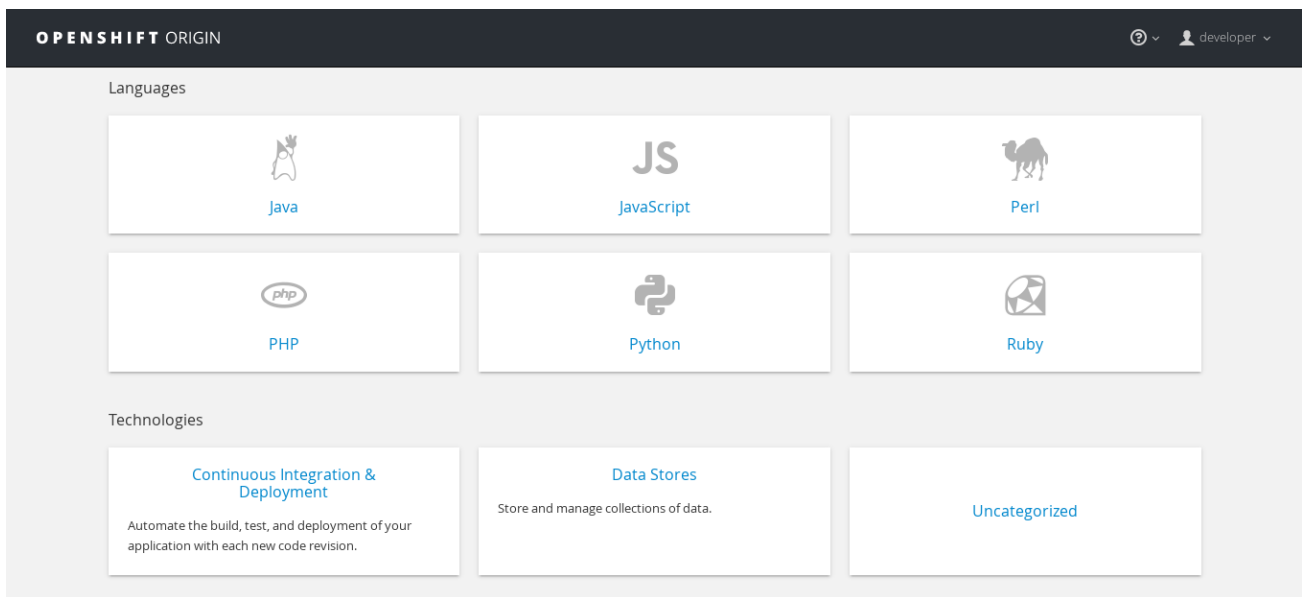
Lo primera acción que hay que hacer tras el login, es crear un nuevo proyecto.



Nuevo proyecto:

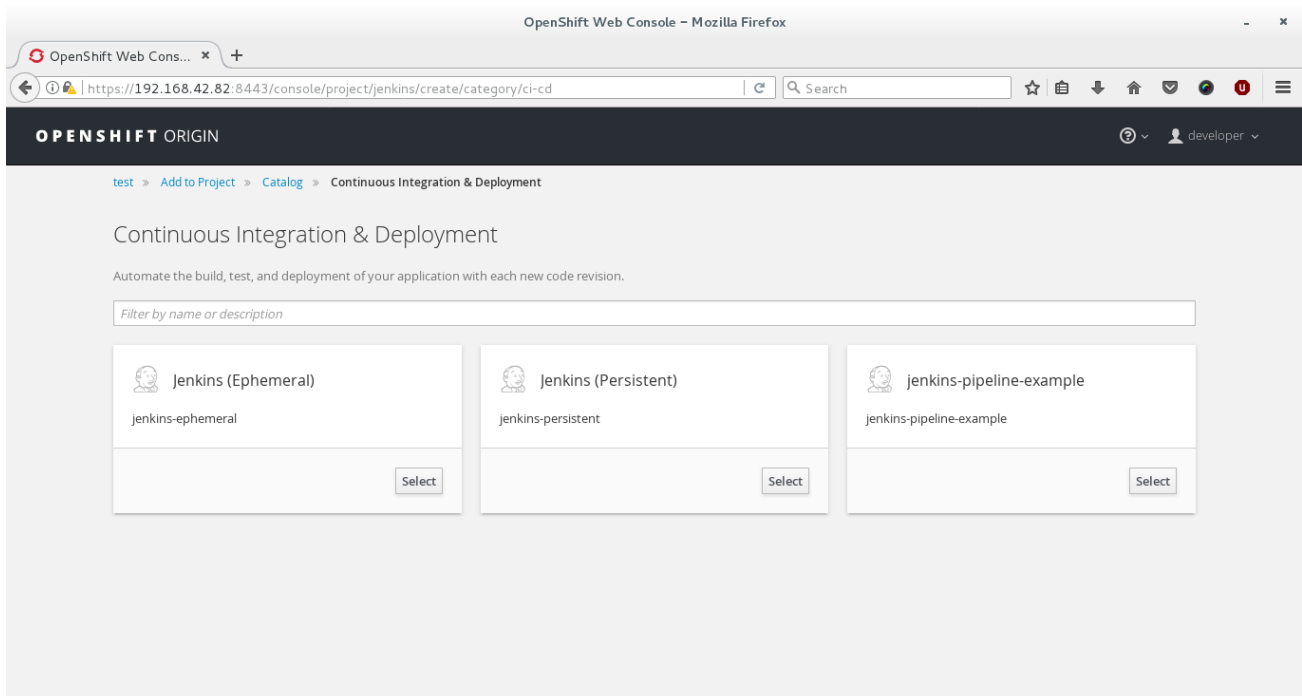


A continuación podemos elegir los lenguajes de programación o tecnologías que queremos usar para el despliegue de las aplicaciones:



Permite utilizar diferentes lenguajes de programación, tecnologías como la integración continua y entrega continua con Jenkins, bases de datos ... etc

En este ejemplo se realizará el despliegue de Jenkins (Ephemeral):



Independientemente de la versión elegida de las 3 que se muestran en la imagen, lo que va a hacer OpenShift es coger un fichero de formato **.json** que está alojado en Github y usarlo como base para crear la aplicación, que se encontrará definida dentro.

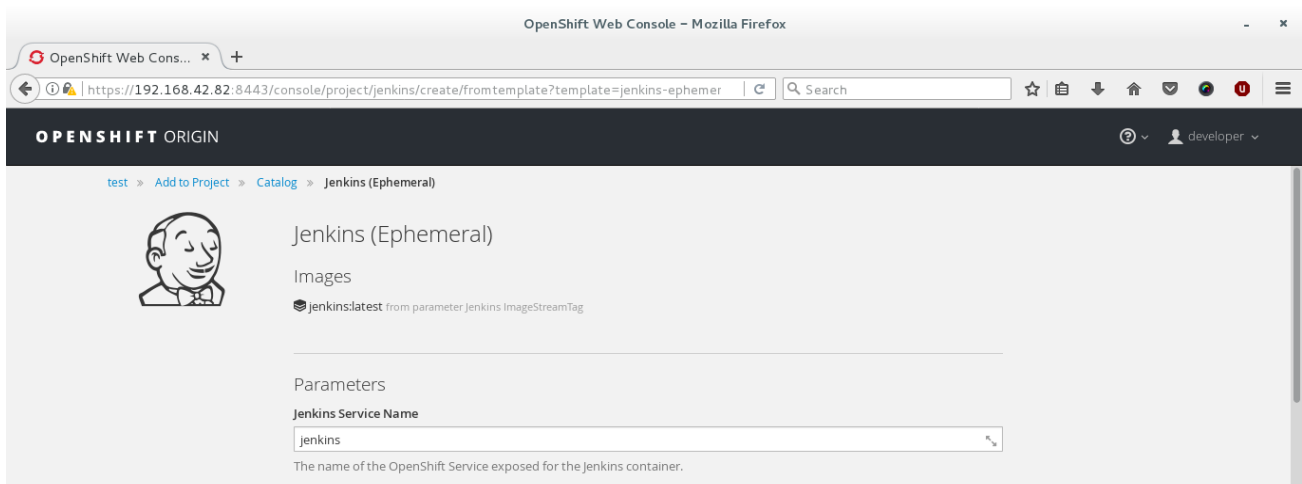
Parámetros con los que va a ser creada la aplicación.

Por defecto se toma como valor de memoria límite 512 MiB de RAM por cada contenedor que es desplegado en un pod. La autenticación OAUTH está activada por defecto; y requiere autenticación por contraseña y nombre de usuario del proyecto donde se cree la aplicación.

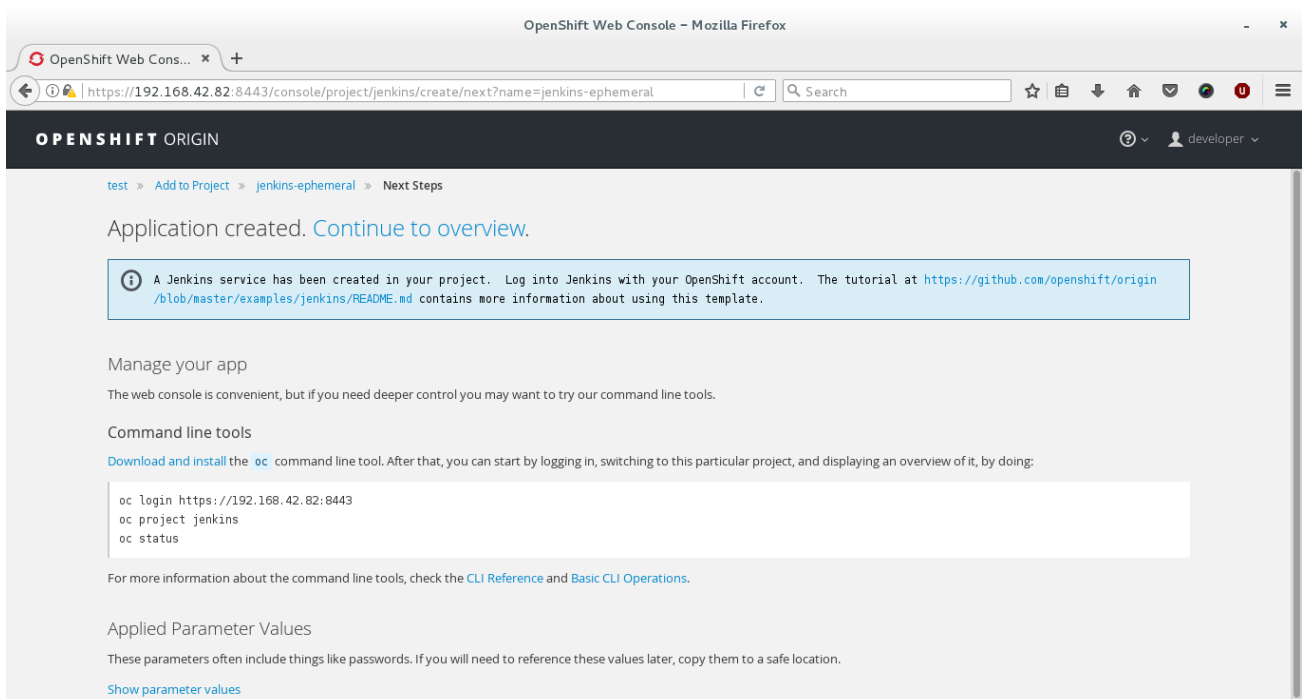
La aplicación tomará como base:

- La imagen **jenkins:latest** que será descargada desde ImageStream. Es el espacio de nombre de OpenShift donde residen sus imágenes.
- Una plantilla de nombre **jenkins-ephemeral-template** que será descargada desde GitHub. Contiene toda la configuración necesaria para desplegar Jenkins (master) junto con otro nodo Jenkins (slave).

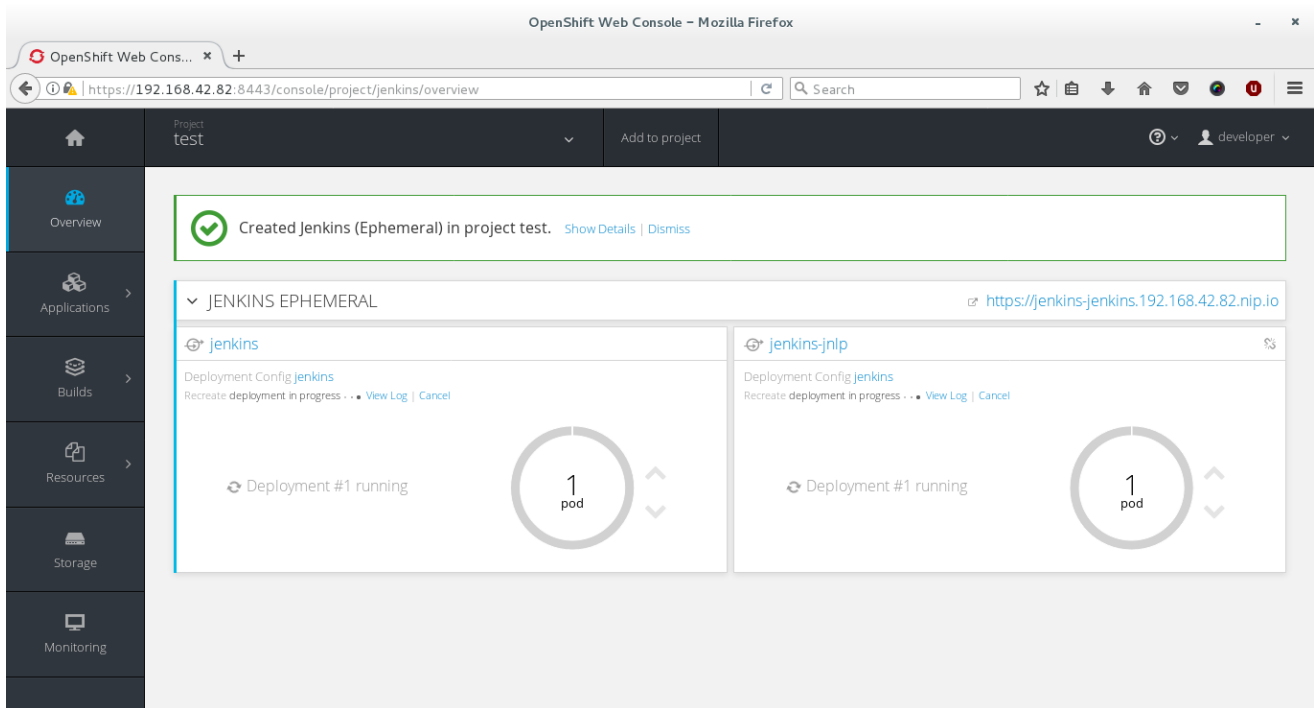
Con estos 2 componentes se creará la aplicación de Jenkins en OpenShift.



La aplicación ha sido creada y comienza a desplegarse:

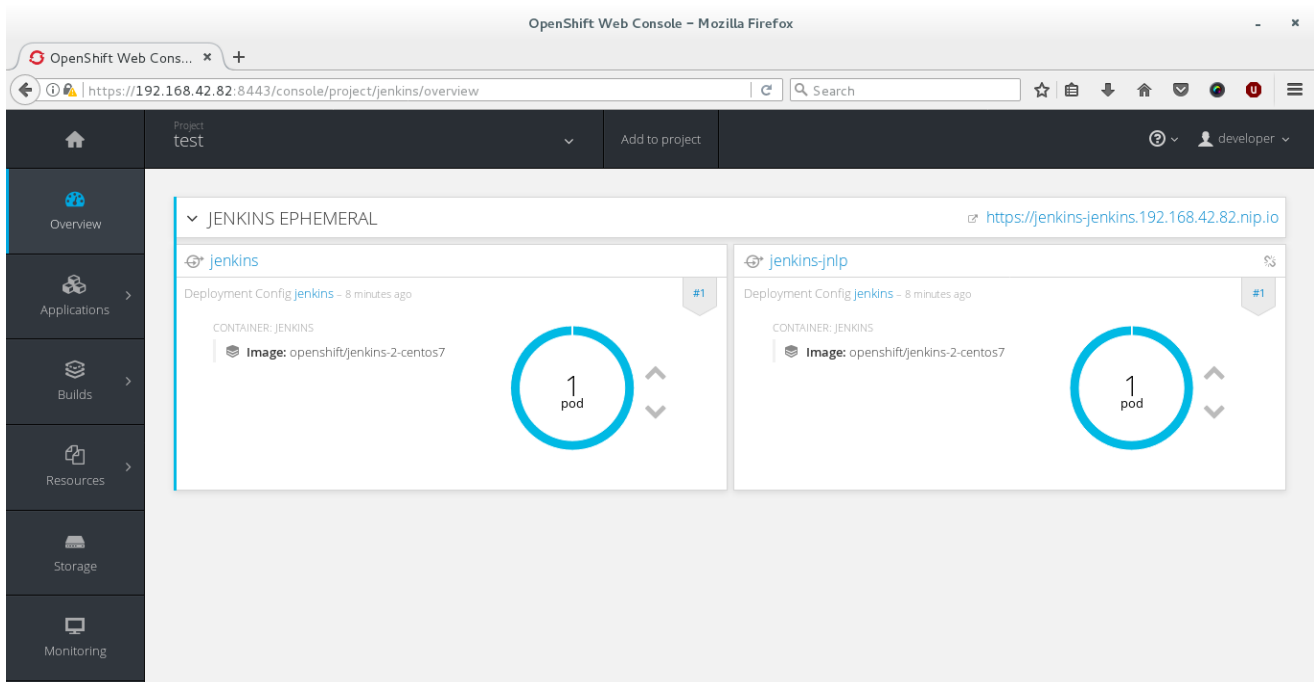


Proceso del despliegue de Jenkins (master) y Jenkins slave (jnlp) dentro de un pod:



OpenShift Origin aprovecha el concepto de un pod de Kubernetes, que es uno o más contenedores desplegados juntos en un mismo host. Un pod es la unidad más pequeña que se puede definir, desplegar y administrar dentro de Kubernetes.

Despliegue finalizado:

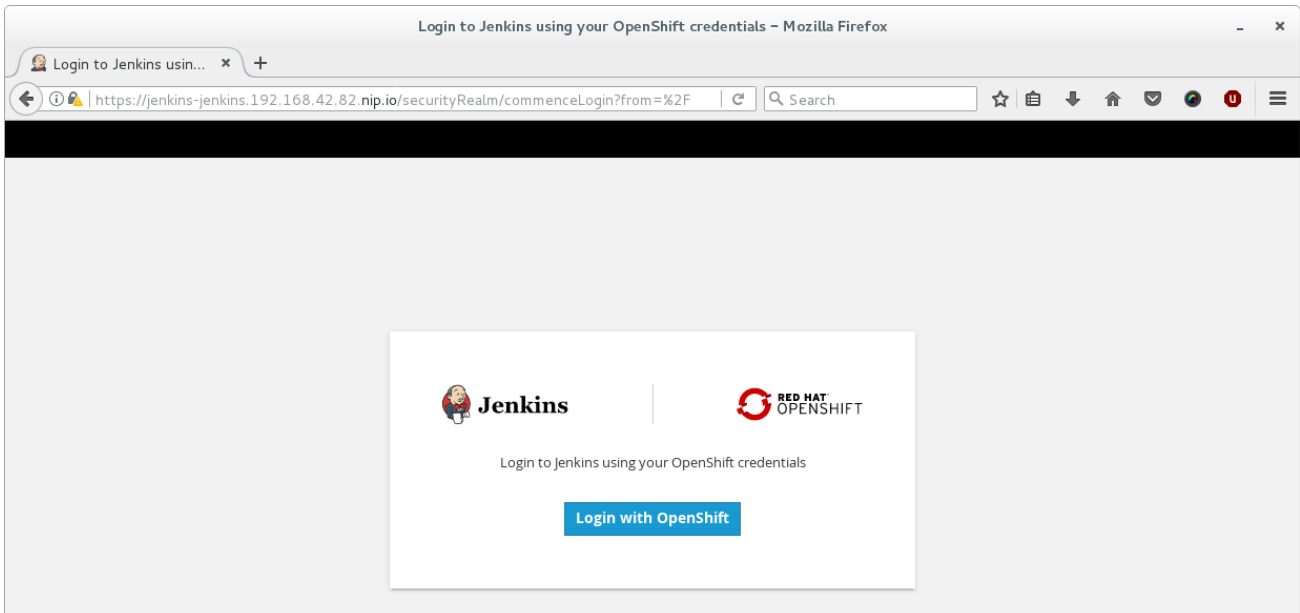


Openshift automatiza todo el proceso de creación de la aplicación, configuración del despliegue, la ruta en la que se expone la aplicación, el escalado, el estado, ... etc

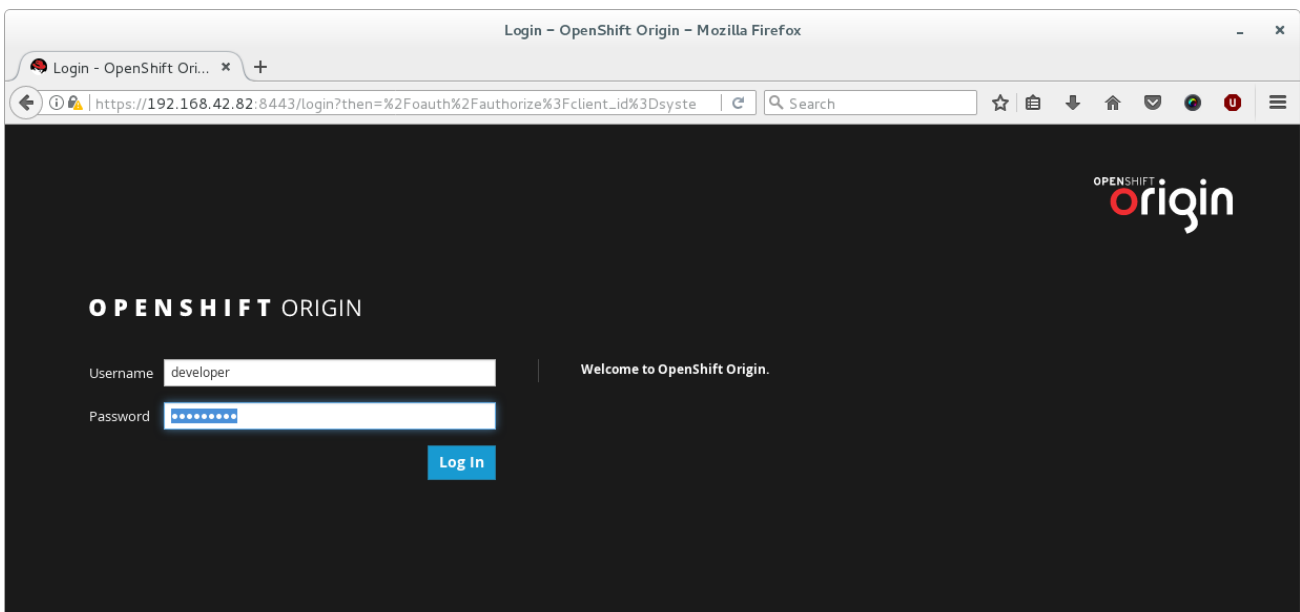
Para finalizar la configuración de Jenkins debemos acceder a la aplicación desde el navegador usando la ruta que se crea con el dominio público **nip.io**:

Es necesario añadir a la URL los siguientes parámetros para finalizar la instalación:

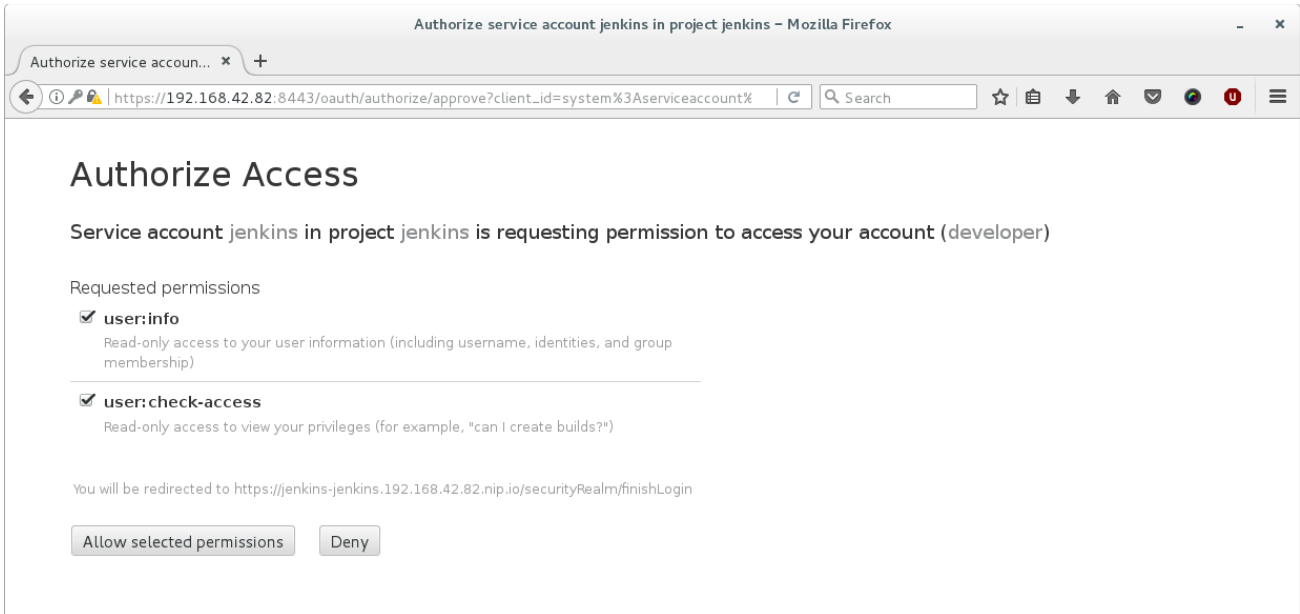
```
/securityRealm/finishLogin
```



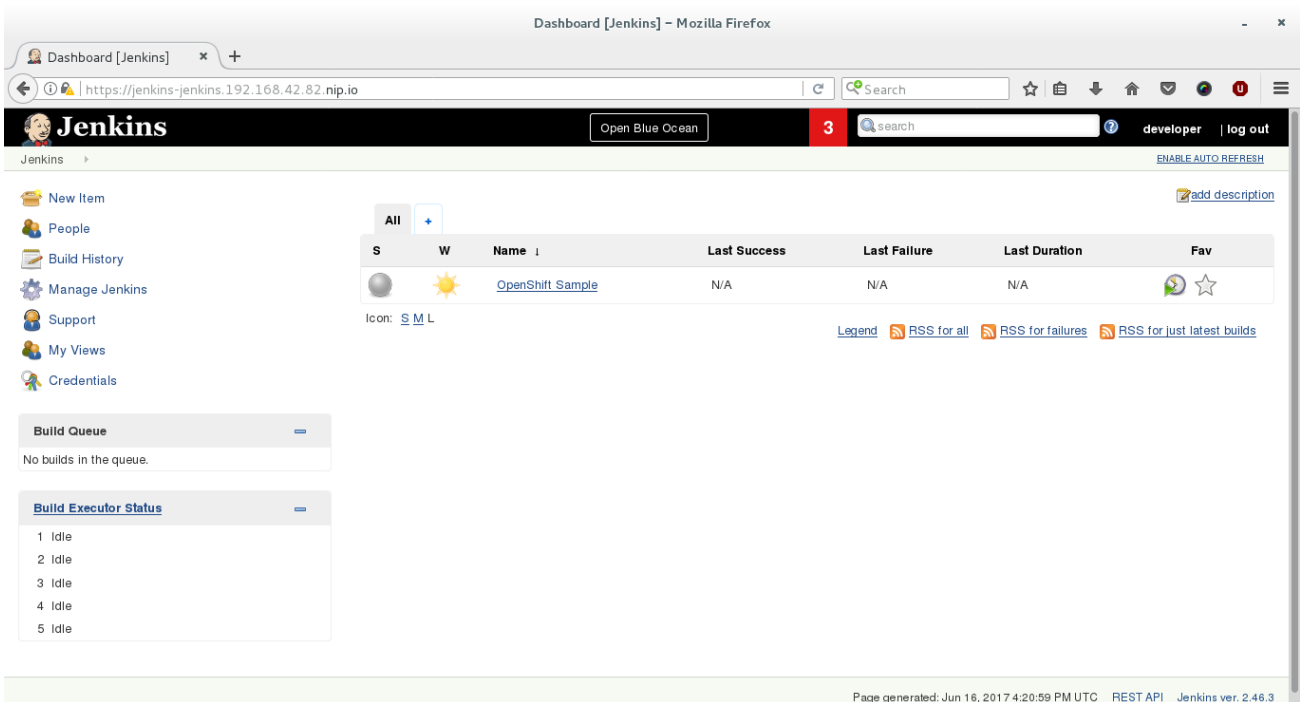
Login con el usuario 'developer'. Usuario que ha sido usado durante la creación y despliegue de la aplicación.



Concesión de los permisos necesarios para poder acceder a la aplicación:



Jenkins desplegado completamente:



Creación de la aplicación Jenkins, ahora desde la línea de comandos

```
~]$ oc login https://192.168.42.154:8443
Authentication required for https://192.168.42.154:8443 (openshift)
Username: fran
Password:
Login successful.

You don't have any projects. You can try to create a new project, by
running

    oc new-project <projectname>

~]$ oc new-project test
Now using project "test" on server "https://192.168.42.154:8443".

~]$ oc status
In project test on server https://192.168.42.154:8443

~]$ oc new-app jenkins-ephemeral

~]$ oc new-app -f
https://raw.githubusercontent.com/openshift/origin/master/examples/j
enkins/application-template.json
--> Deploying template "openshift/jenkins-ephemeral" to project test

    Jenkins (Ephemeral)
    -----
    Jenkins service, without persistent storage.

    WARNING: Any data stored will be lost upon pod destruction.
    Only use this template for testing.

    * With parameters:
      * Jenkins Service Name=jenkins
      * Jenkins JNLP Service Name=jenkins-jnlp
      * Enable OAuth in Jenkins=true
      * Memory Limit=512Mi
      * Jenkins ImageStream Namespace=openshift
      * Jenkins ImageStreamTag=jenkins:latest

--> Creating resources ...
    route "jenkins" created
    deploymentconfig "jenkins" created
    serviceaccount "jenkins" created
    rolebinding "jenkins_edit" created
    service "jenkins-jnlp" created
    service "jenkins" created
--> Success
```

Todo el proceso de creación de los recursos ha sido realizado de forma automatizada

Podemos ver los logs de la aplicación para comprobar cuando ha terminado de desplegarse:

```
~] $ oc logs -f dc/jenkins
```

El pod donde se aloja la aplicación:

```
~] $ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
jenkins-1-ftn7s	1/1	Running	0	5m

La ruta con la que ha sido expuesta la aplicación en el dominio público nip.io:

```
~] $ oc get route
```

NAME	HOST/PORT	PATH	SERVICES
jenkins	jenkins-test.192.168.42.154.nip.io		jenkins

PORT	TERMINATION	WILDCARD
<all>	edge/Redirect	None

La dirección ip y puerto del contenedor que se aloja dentro del pod:

```
~] $ oc get svc
```

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
jenkins	172.30.246.27	<none>	80/TCP	12m
jenkins-jnlp	172.30.83.60	<none>	50000/TCP	12m

Una vez que la aplicación es desplegada, para finalizar la configuración es necesario ejecutar el siguiente comando o alternativamente abrir la URL de la aplicación desde el navegador:

```
~] $ oc annotate sa/jenkins serviceaccounts.openshift.io/oauth-redirecturi.l=http://jenkins-test.192.168.42.154.nip.io/securityRealm/finishLogin --overwrite
```

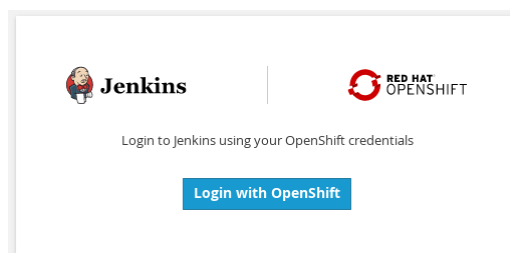
URL:

```
http://jenkins-test.192.168.42.154.nip.io/securityRealm/finishLogin
```

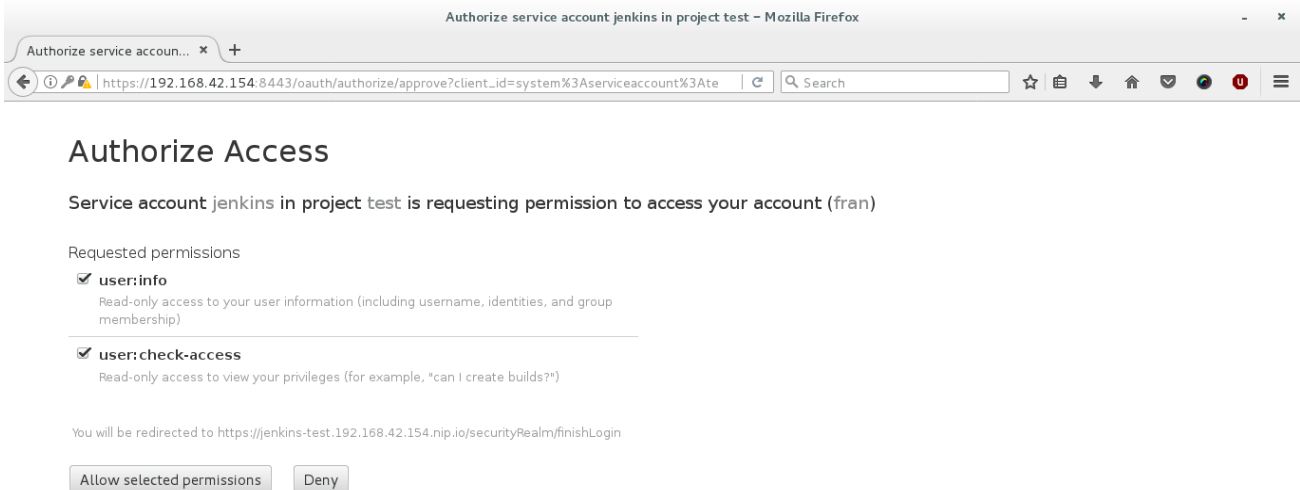
Si hemos finalizado la instalación de Jenkins usando la primera opción, el navegador no se abrirá. Es posible abrirlo automáticamente para ver la aplicación con el siguiente comando. Tan sólo hay que indicar el nombre del servicio y el proyecto al que pertenece:

```
~] $ minishift openshift service jenkins -n test
```

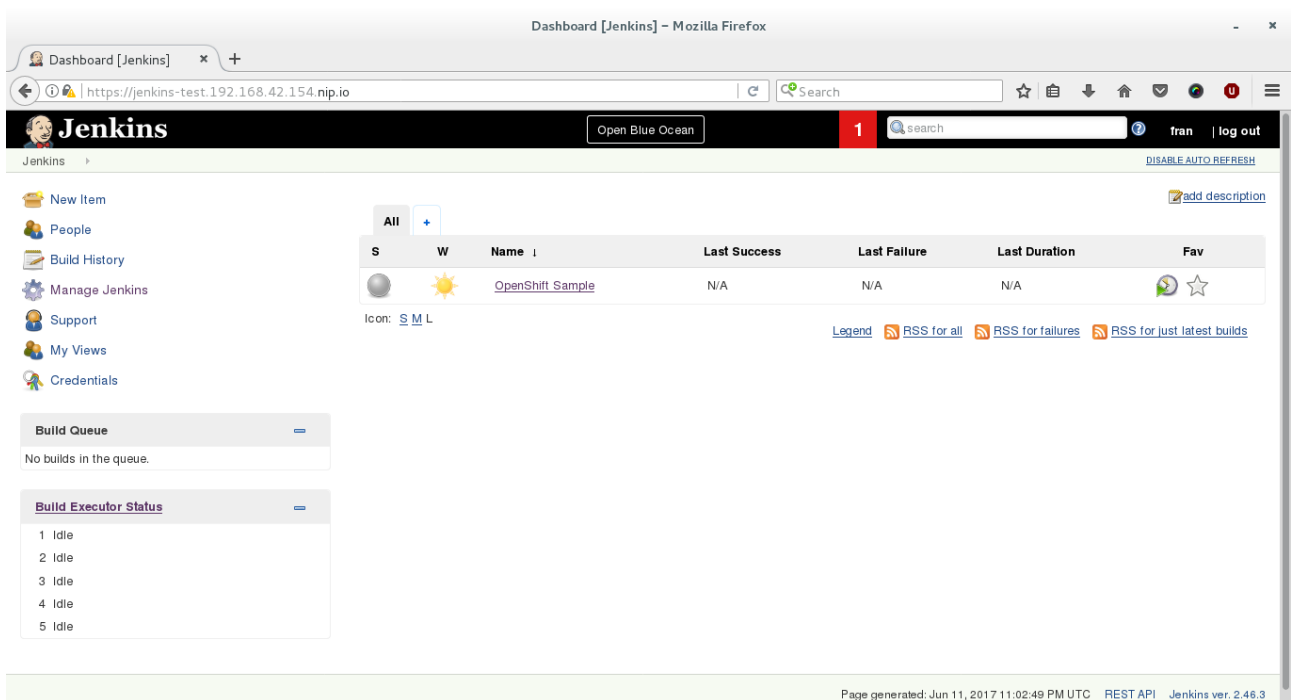
```
Opening the service test/jenkins in the default browser...
```



Concesión de los permisos que solicita:



Jenkins ha sido nuevamente desplegado:



¿Qué ha pasado internamente dentro de la máquina virtual?

Se han creado 2 nuevas imagenes

```
[fjhidalgo@administrador ~]$ minishift ssh
```

```
docker@minishift:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
openshift/jenkins-slave-maven-centos7	latest	bbbc602acc50	About an hour ago	899.3 MB
openshift/jenkins-2-centos7	<none>	936fa62bc19a	6 days ago	1.281 GB

La imagen 'jenkins-slave-maven-centos7' es la imagen usada para crear contenedores que

funcionen como nodos esclavos del nodo maestro Jenkins. Los nodos esclavos son necesarios para poder realizar la ejecución de jobs.

Para ver el último contenedor creado (en este caso, el contenedor de Jenkins):

```
docker@minishift:~$ docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED
193cfb6d14f2	openshift/jenkins-2-centos7	"/usr/libexec/s2i/run"	14 minutes ago

STATUS	NAMES
Up 14 minutes	k8s_jenkins.14ba0cb4_jenkins-1-ftn7s_test

El contenedor ha sido creado con la herramienta **S2I** (Source-to-Image)

Contenedores creados y activos dentro de la máquina virtual Minishift

Es posible verlos de una manera más clara, filtrando por el formato table, sacando los campos que sólo nos interesen mostrar.

```
docker@minishift:~$ docker ps --format 'table {{.Image}}\t{{.Command}}'
```

IMAGE	COMMAND
openshift/jenkins-2-centos7@sha256:25f1e	"/usr/libexec/s2i/run"
openshift/origin-docker-registry:v1.5.1	"/bin/sh -c 'DOCKER_R"
openshift/origin-haproxy-router:v1.5.1	"/usr/bin/openshift-r"
openshift/origin-pod:v1.5.1	"/pod"
openshift/origin-pod:v1.5.1	"/pod"
openshift/origin-pod:v1.5.1	"/pod"
openshift/origin:v1.5.1	"/usr/bin/openshift s"

```
docker@minishift:~$ docker ps -n 2 --format \
```

```
"\n- CONTAINER ID : {{.ID}}\n  IMAGE : {{.Image}}\n  COMMAND : {{.Command}}\n  CREATED : {{.CreatedAt}}\n  STATUS : {{.Status}}\n  NAME : {{.Names}}\n"
```

```
- CONTAINER ID : c407ff0dc074
```

```
  IMAGE : openshift/origin-docker-registry:v1.5.1
```

```
  COMMAND : "/bin/sh -c 'DOCKER_R"
```

```
  CREATED : 2017-06-12 17:52:32 +0000 UTC
```

```
  STATUS : Up 17 minutes
```

```
  NAME : k8s_registry.e16ed372_docker-registry-1-kxglf_default_8
```

```
- CONTAINER ID : c289ala6061d
```

```
  IMAGE : openshift/jenkins-2-centos7@sha256:25f1e38572b0aed33af
```

```
  COMMAND : "/usr/libexec/s2i/run"
```

```
  CREATED : 2017-06-12 17:52:32 +0000 UTC
```

```
  STATUS : Up 17 minutes
```

```
  NAME : k8s_jenkins.14ba0cb4_jenkins-1-ftn7s_test_66e2bdc8-4ef5
```

Conclusión

Minishift es una herramienta práctica, fácil de usar y que permite realizar múltiples despliegues y configuraciones. Usar como backend una plataforma que ejecuta un entorno de microservicios, es ideal para que desarrolladores o personas que se inicien en el mundo de los contenedores, puedan usar esta herramienta para abstraerse de configuraciones y detalles más técnicos, centrándose simplemente en conocer y aprender el despliegue de servicios en contenedores.

El hecho de poder usar esta herramienta en tu propia máquina local, realizando despliegues de aplicaciones que te permiten incluso compartirlas en Internet con otras personas o desarrolladores, hacen de Minishift una gran herramienta para usar, tirar y volver a desplegar aplicaciones.

Credits

<https://www.openshift.org/minishift/>

<https://docs.openshift.org/latest/minishift/getting-started/installing.html>

<https://docs.openshift.org/latest/minishift/getting-started/docker-machine-drivers.html>

<https://docs.openshift.org/latest/minishift/index.html>

<https://docs.docker.com/engine/installation/linux/centos/#install-from-a-package>

https://github.com/openshift/origin/blob/master/docs/cluster_up_down.md

<https://docs.docker.com/engine/admin/systemd/>

<https://docs.docker.com/engine/installation/linux/linux-postinstall/>

<https://docs.openshift.org/latest/minishift/using/managing-minishift.html>

<https://docs.openshift.org/latest/minishift/command-ref/minishift.html>

<https://docs.openshift.org/latest/minishift/using/docker-daemon.html#reusing-docker-daemon>

<https://docs.openshift.org/latest/minishift/using/interacting-with-openshift.html>