Prácticas Openshift

# Crear un nuevo proyecto

**oc new-project proyecto1**

# Creamos una nueva plantilla desde un fichero.

* En primer lugar vamos a crear una image-stream desde fichero, basada en jboss. Descargamos el fichero imagen-jboss.yaml
* Lo cargamos

oc create -f imagen-jboss.yaml

* Comprobamos que está

**oc get is**

NAME DOCKER REPO TAGS UPDATED

jboss-eap64-openshift 172.30.1.1:5000/pipelines/jboss-eap64-openshift 1.8,1.9,latest + 7 more...

mlbparks 172.30.1.1:5000/pipelines/mlbparks

newbuildtest 172.30.1.1:5000/pipelines/newbuildtest

nodejs-010-centos7 172.30.1.1:5000/pipelines/nodejs-010-centos7 latest 6 minutes ago

nodejsex 172.30.1.1:5000/pipelines/nodejsex

* Para la plantilla, descargamos un fichero denominado plantillacurso.yml que será entregado por el profesor
* La cargamos con el comando

**oc create -f plantilla-curso.yaml**

template.template.openshift.io/plantilla-curso created

* Comprobamos que la ha creado

**oc get templates**

NAME DESCRIPTION PARAMETERS OBJECTS

mlbparks-eap Application template for MLB Parks application on EAP 6 & MongoDB built using... 14 (5 blank) 8

* Vemos los parámetros que necesita

**oc process mlbparks-eap --parameters**

NAME DESCRIPTION GENERATOR VALUE

APPLICATION\_NAME The name for the application. mlbparks

APPLICATION\_HOSTNAME Custom hostname for service routes. Leave blank for default hostname, e.g.: <application-name>.<project>.<default-domain-suffix>

GIT\_URI Git source URI for application https://github.com/jorgemoralespou/openshift3mlbparks.git

GIT\_REF Git branch/tag reference master

MAVEN\_MIRROR\_URL Maven mirror url. If nexus is deployed locally, use nexus url (e.g. http://nexus.ci.apps.10.2.2.2.xip.io/content/groups/public/)

MONGODB\_DATABASE Database name root

MONGODB\_NOPREALLOC Disable data file preallocation.

MONGODB\_SMALLFILES Set MongoDB to use a smaller default data file size.

MONGODB\_QUIET Runs MongoDB in a quiet mode that attempts to limit the amount of output.

MONGODB\_USER Database user name expression user[a-zA-Z0-9]{3}

MONGODB\_PASSWORD Database user password expression [a-zA-Z0-9]{8}

MONGODB\_ADMIN\_PASSWORD Database admin password expression [a-zA-Z0-9]{8}

GITHUB\_TRIGGER\_SECRET Github trigger secret expression [a-zA-Z0-9]{8}

GENERIC\_TRIGGER\_SECRET Generic build trigger secret expression [a-zA-Z0-9]{8}

* Creamos una aplicación con la plantilla

**oc process mlbparks-eap | oc create -f -**

buildconfig.build.openshift.io/mlbparks created

imagestream.image.openshift.io/mlbparks created

deploymentconfig.apps.openshift.io/mlbparks-mongodb created

deploymentconfig.apps.openshift.io/mlbparks created

route.route.openshift.io/mlbparks created

service/mongodb created

service/mlbparks created

service/mlbparks-ping created

* Comprobamos que se han creado los objetos y probamos la aplicación

# Instalar una aplicación con new-app

**oc new-app --name=cotd --labels name=cotd php~https://github.com/devops-with-openshift/cotd.git -e SELECTOR=cats**

# Crear el route

oc expose svc/cotd

* Comprobar desde la consola de Openshift que la aplicación está funcionando
* Acceder al route y comprobar mediante la URL propuesta que la aplicación está funcionando

# Crear un almacenamiento permanente

* Primero creamos el el volumeclaim desde un fichero de configuración

apiVersion: "v1"

kind: "PersistentVolumeClaim"

metadata:

name: "disco1"

spec:

accessModes:

- "ReadWriteOnce"

resources:

requests:

storage: "1Mi"

volumeName: "pv100"

* Lo generamos desde fichero y lo acoplamos a la aplicación

oc create -f nombre-fichero.yaml

* También podíamos haber usado esta línea de comandos para crearlo

oc set volume dc/cotd --add --name=images --type=persistentVolumeClaim --mount-path=/opt/app-root/src/data/images --claim-name=disco1 --claim-size=100m

* Comprobamos que está creado

oc get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

disco1 Bound pv0096 100Gi RWO,ROX,RWX 10m

* Creamos una Nueva aplicación

oc new-app devopswithopenshift/welcome:latest --name=myapp

* La asociamos sondas

oc set probe dc myapp --readiness --open-tcp=8080 --initial-delay-seconds=5 --timeout-seconds=5

oc set probe dc myapp --liveness -- echo ok

* Creamos un route

oc expose svc/myapp

* Describimos la aplicación

oc describe dc myapp

* Podemos recrear manualmente el despliegue

oc rollout latest myapp

* Podemos ver el histórico de cambios

oc rollout history dc/myapp

deploymentconfigs "myapp"

REVISION STATUS CAUSE

1 Complete config change

2 Complete config change

3 Complete config change

4 Complete manual change

* Comprobamos la URL de acceso y probamos la aplicación desde un navegador

oc get route/myapp

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

myapp myapp-proyecto1.192.168.99.101.nip.io myapp 8080-tcp None

* Comprobamos los triggers existentes

oc set triggers dc/myapp

NAME TYPE VALUE AUTO

deploymentconfigs/myapp config true

deploymentconfigs/myapp image myapp:latest (myapp) true

* Podemos por ejemplo desactivar el trigger para config si no queremos que se actualize con un cambio de configuración

oc set triggers dc/myapp --from-config –remove

* Comprobamos que se ha quitado y lo volvemos a activar

oc set triggers dc myapp --from-config

* Vamos a cambiar ahora a la estrategia de Recreate

# PIPELINES

* Creamos un nuevo proyecto para probarlo

**oc new-project pipelines**

Now using project "pipelines" on server "https://192.168.99.101:8443".

You can add applications to this project with the 'new-app' command. For example, try:

oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git

to build a new example application in Ruby.

* Indicamos lo siguiente

**oc get templates -n openshift**

* Debe aparecer al menos dos plantillas de Jenkins una del propio jenkins y otra de prueba de pipelines

jenkins-ephemeral Jenkins service, without persistent storage.... 7 (all set) 6

jenkins-pipeline-example This example showcases the new Jenkins Pipeline integration in OpenShift,... 16 (4 blank) 8

* **NOTA : si no tenemos la plantilla podemos cargarla desde**

oc create -f \

https://raw.githubusercontent.com/openshift/origin/

master/examples/jenkins/pipeline/samplepipeline.yaml

* Creamos un nuevo Jenkins a partir de la plantilla

**oc new-app jenkins-ephemeral**

--> Deploying template "openshift/jenkins-ephemeral" to project proyecto1

Jenkins (Ephemeral)

---------

Jenkins service, without persistent storage.

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

A Jenkins service has been created in your project. Log into Jenkins with your OpenShift account. The tutorial at https://github.com/openshift/origin/blob/master/examples/jenkins/README.md contains more information about using this template.

\* With parameters:

\* Jenkins Service Name=jenkins

\* Jenkins JNLP Service Name=jenkins-jnlp

\* Enable OAuth in Jenkins=true

\* Memory Limit=512Mi

\* Jenkins ImageStream Namespace=openshift

\* Disable memory intensive administrative monitors=false

\* Jenkins ImageStreamTag=jenkins:2

--> Creating resources ...

route.route.openshift.io "jenkins" created

deploymentconfig.apps.openshift.io "jenkins" created

serviceaccount "jenkins" created

rolebinding.authorization.openshift.io "jenkins\_edit" created

service "jenkins-jnlp" created

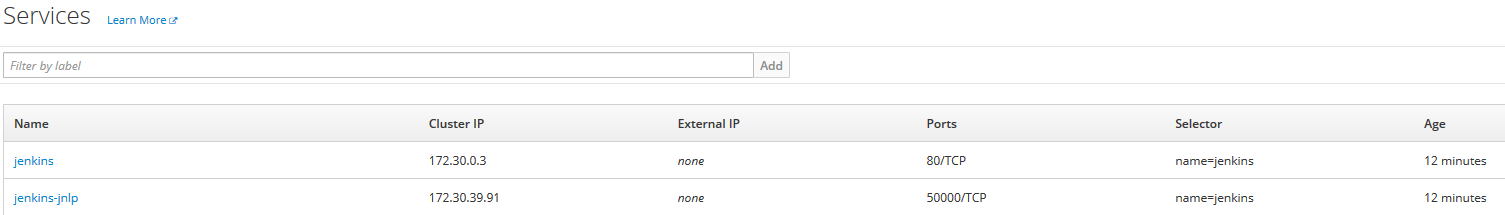
service "jenkins" created

--> Success

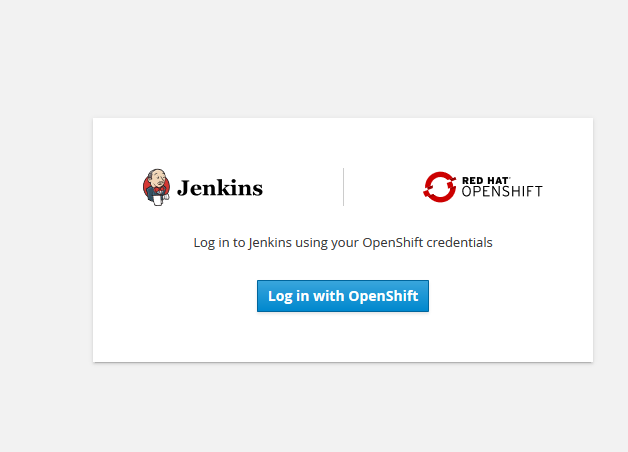
Access your application via route 'jenkins-proyecto1.192.168.99.101.nip.io'

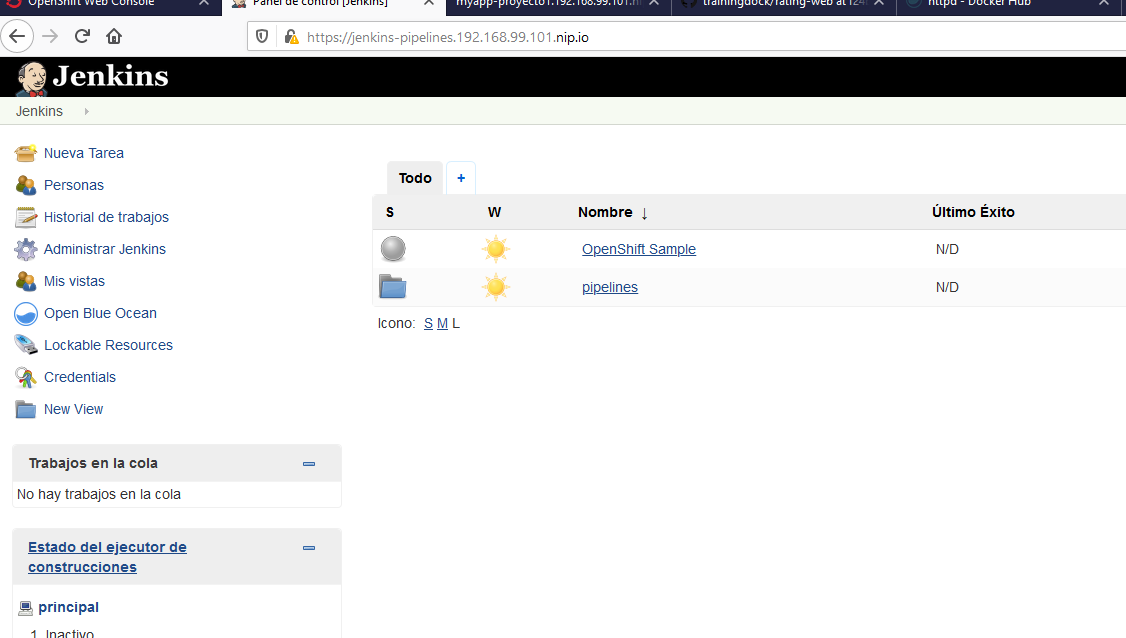
Run 'oc status' to view your app.

* Deberiamos tener dos servicios: uno para Jenkins web-ui y prar jenkins-jnlp



* Si probamos el route del acceso WEB





* Descargamos el fichero de una plantilla de pipeline desde

<https://github.com/sclorg/nodejs-ex/blob/master/openshift/templates/nodejs-mongodb.json>

* Lo llamamos por ejemplo “plantilla-pipeline.json” y la cargamos

**oc create -f plantilla-pipeline.json**

template.template.openshift.io/nodejs-mongodb-example created

* Comprobamos que está

**oc get templates**

NAME DESCRIPTION PARAMETERS OBJECTS

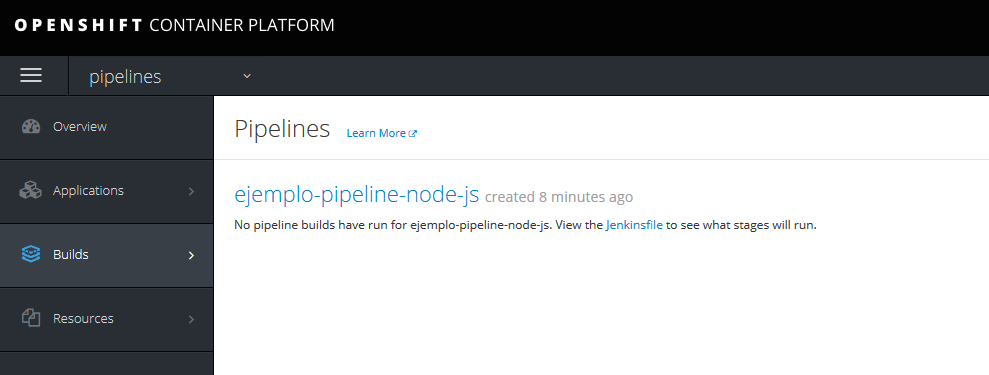
nodejs-mongodb-example An example Node.js application with a MongoDB database. For more information... 18 (4 blank) 8

* Descargamos el fichero pipeline.yaml desde nos indique le profesor
* Lo instalamos

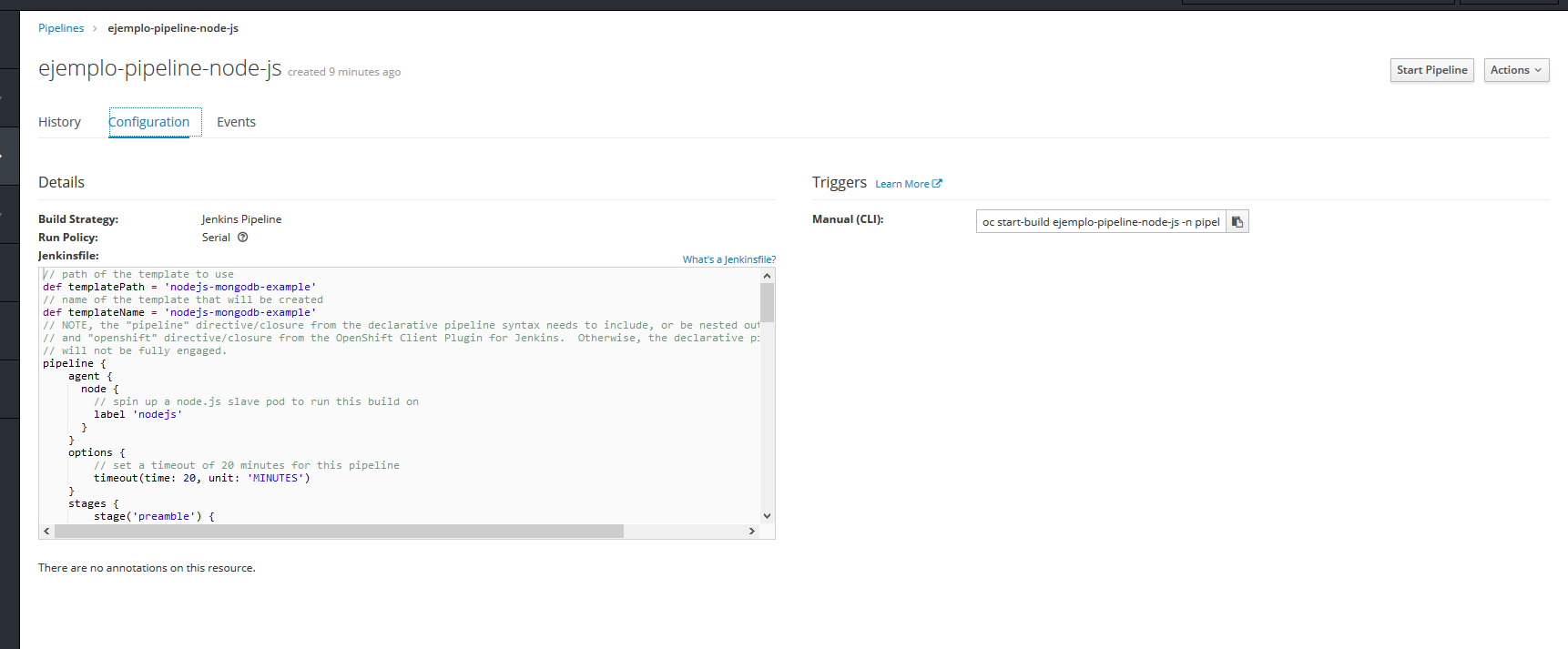
**oc create -f pipeline.yaml**

buildconfig.build.openshift.io/ejemplo-pipeline-node-js created

* Podemos comprobar desde la consola WEB, en build🡪pipelines que lo tenemos cargado
* Todavía no tenemos ningún build de la pipeline



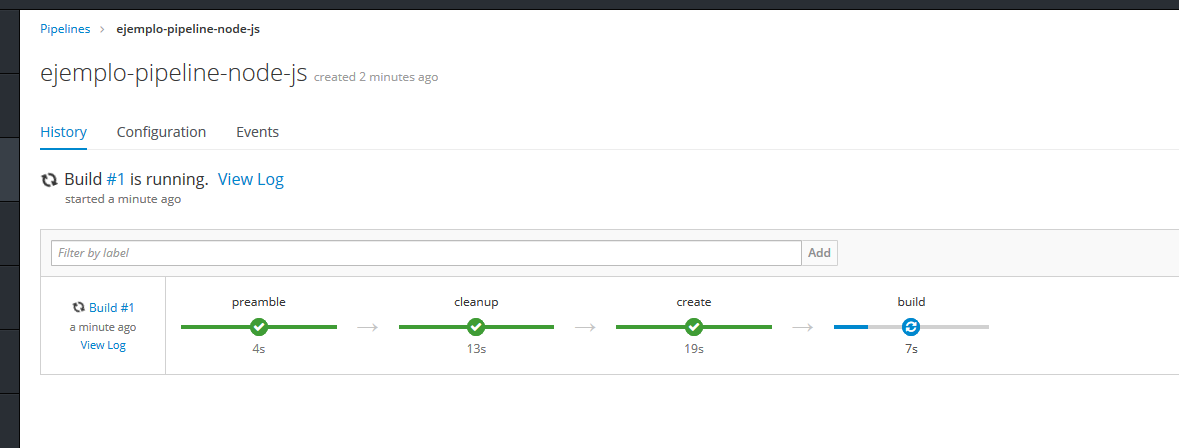
* Podemos lanzar un pipeline directamente desde el botón de la pantalla



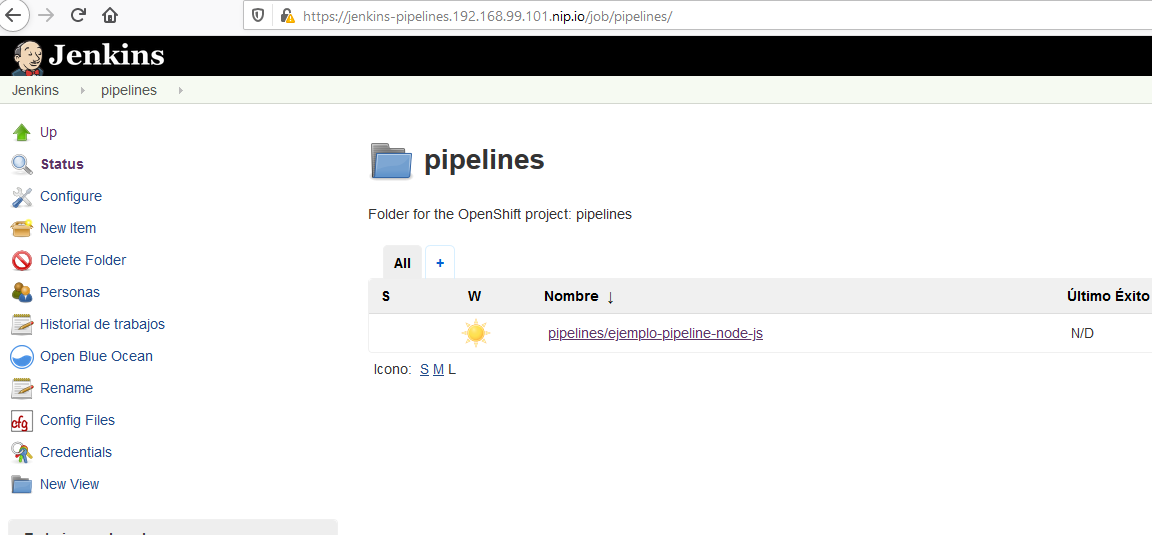
* O Tambien lo Podemos hacer desde línea de comandos

oc start-build nodejs-sample-pipeline

* Podemos ver el proceso en la consola



* También podemos verlo desde la consola de Jenkins



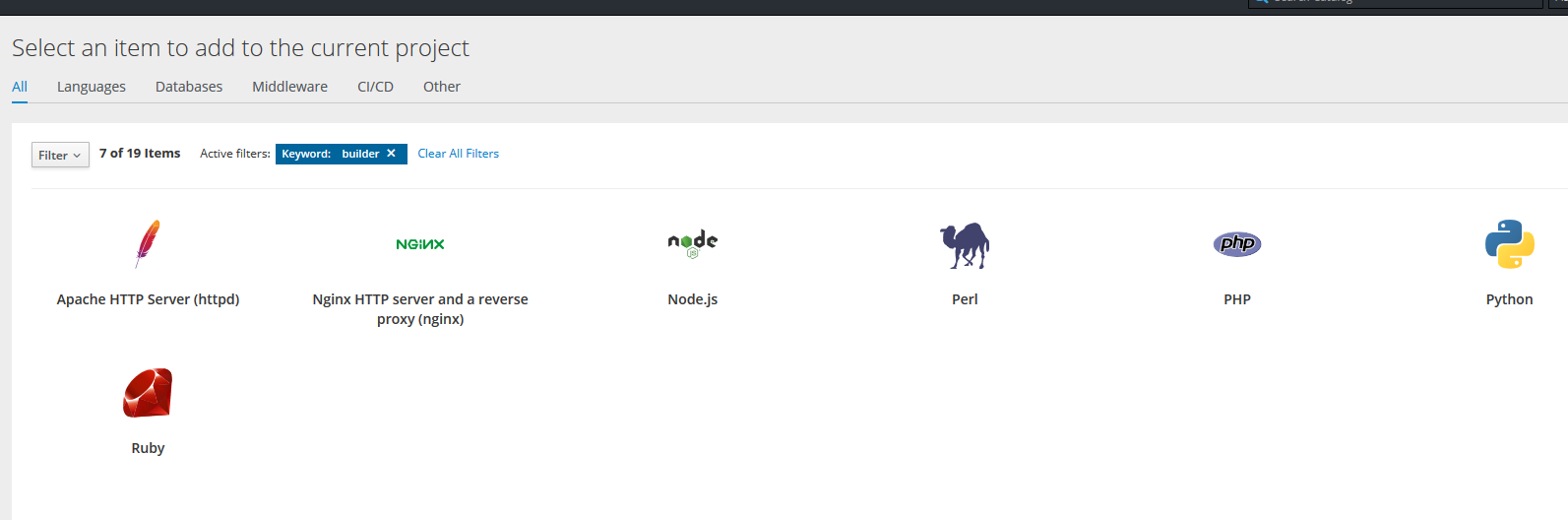
* En resumen:
  + Se lanza una instancia del trabajo en el servidor Jenkins.
  + También se puede lanzar un pod esclavo si es necesario.
  + La pipeline se ejecuta en el master o en el esclavo
  + Se borra cualquier recurso anterior con la label template=nodejs-mongodb-example
  + Se crea una nueva aplicación con todos los recursos de la plantilla nodejs-mongodb-example
  + Se comienza un build usando el Build config nodejs-mongodb-example.
    - La pipeline espera a que termine el build para ir a la siguiente etapa
  + Se comienza un deploy con la configuración de nodejs-mongodb-example.
  + Si todo es correcto se crea una imagen  nodejs-mongodb-example:latest cokmo  nodejs-mongodb-example:stage.
  + Se borra el esclavo si no es necesario

# BUILDS

* Como ejemplo, creamos un build desde un fuente

oc new-build openshift/nodejs-010-centos7~https://github.com/trainingdock/nodejs-ex --name=prueba

* Comprobamos ahora los builder que tenemos en nuestra página



* Ahora subimos el builder como imagen a nuestro repositorio

oc new-build <https://github.com/trainingdock/imagen_java>

* Comprobamos

oc get is

NAME DOCKER REPO TAGS UPDATED

centos 172.30.1.1:5000/pipelines/centos latest About a minute ago

imagenjava 172.30.1.1:5000/pipelines/imagenjava

jboss-eap64-openshift 172.30.1.1:5000/pipelines/jboss-eap64-openshift 1.3,1.4,latest + 7 more...

mlbparks 172.30.1.1:5000/pipelines/mlbparks

newbuildtest 172.30.1.1:5000/pipelines/newbuildtest

nodejs-010-centos7 172.30.1.1:5000/pipelines/nodejs-010-centos7 latest 29 minutes ago

prueba 172.30.1.1:5000/pipelines/prueba

prueba1 172.30.1.1:5000/pipelines/prueba1 latest 24 minutes ago

* Construimos

oc start-build imagenjava

build.build.openshift.io/imagenjava-2 started

* Creamos una aplicación con una base como ejemplo

oc new-app imagenjava~https://github.com/trainingdock/ItemsWS --context-dir=app --allow-missing-imagestream-tags --strategy=source