

# Grupo 13

## Ejercicio práctico de Jenkins

Ejecutar proyecto de Jenkins con Docker y levantar flujo de trabajo de CI/CD desde un repositorio de GitHub.

## Requisitos

- GitHub
  - Cuenta de GitHub
  - Fork
- Docker
- Docker-compose

## Instrucciones

### GitHub - Cuenta

Se debe crear una cuenta en [github.com](https://github.com) de la manera habitual que se crea una cuenta en cualquier sitio web.

### GitHub Fork repositorio

Fork el repositorio de GitHub en su cuenta personal. (Fork -> Bifurcación) desde el siguiente [link](#)

### Docker y Docker-compose

Para instalar docker en Ubuntu

- Actualizar el sistema operativo

```
sudo apt-get update && upgrade
```

- Instalar servidor docker-ce y docker-compose

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

- Comprobamos el estado de docker

```
sudo systemctl status docker
```

- Para validar que está correctamente instalado se ejecuta:

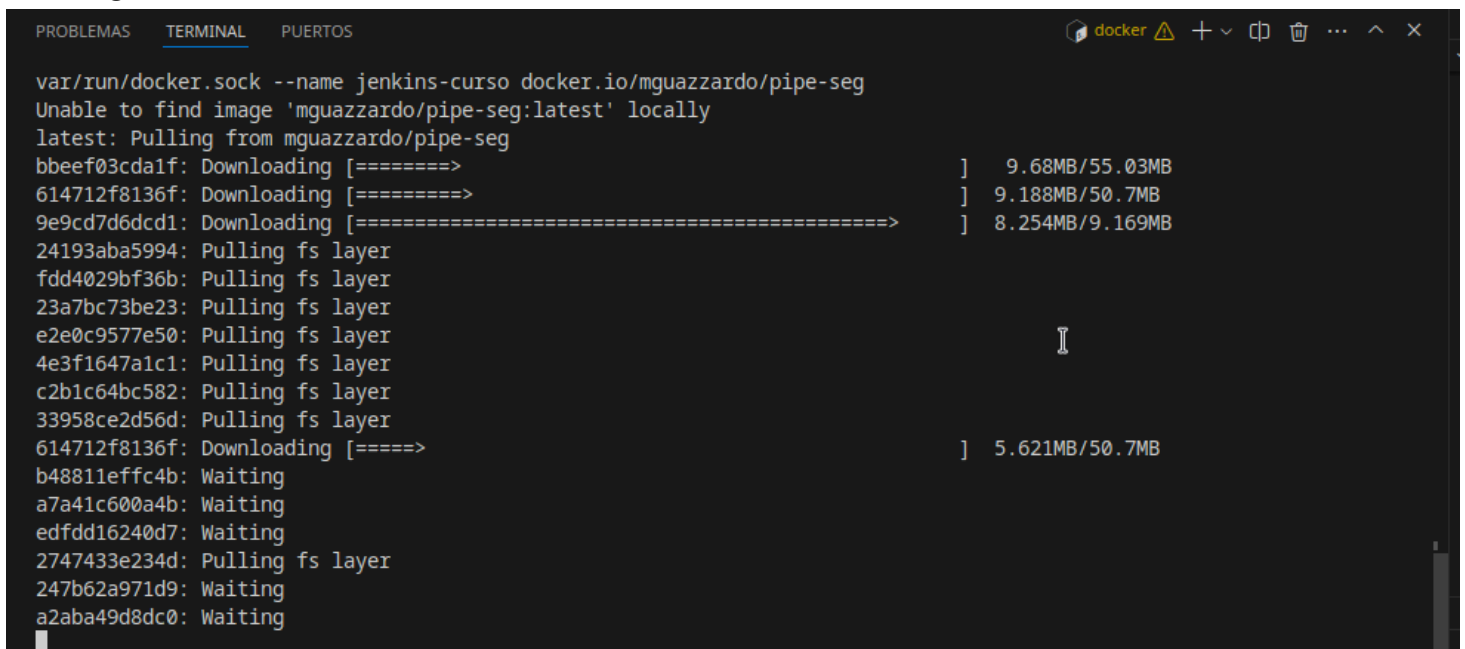
```
docker --version;  
docker compose version;
```

## Lanzar Jenkins

Para lanzar Jenkins se debe ejecutar el siguiente comando:

```
docker run -dit -p 8080:8080 --network=host -v /var/run/docker.sock:/var/run/dock  
--name jenkins-curso docker.io/mguazzardo/pipe-seg
```

### Descargando



```
PROBLEMAS  TERMINAL  PUERTOS  
var/run/docker.sock --name jenkins-curso docker.io/mguazzardo/pipe-seg  
Unable to find image 'mguazzardo/pipe-seg:latest' locally  
latest: Pulling from mguazzardo/pipe-seg  
bbeeef03cda1f: Downloading [=====>] 9.68MB/55.03MB  
614712f8136f: Downloading [=====>] 9.188MB/50.7MB  
9e9cd7d6dcd1: Downloading [=====>] 8.254MB/9.169MB  
24193aba5994: Pulling fs layer  
fdd4029bf36b: Pulling fs layer  
23a7bc73be23: Pulling fs layer  
e2e0c9577e50: Pulling fs layer  
4e3f1647a1c1: Pulling fs layer  
c2b1c64bc582: Pulling fs layer  
33958ce2d56d: Pulling fs layer  
614712f8136f: Downloading [=====>] 5.621MB/50.7MB  
b48811effc4b: Waiting  
a7a41c600a4b: Waiting  
edfdd16240d7: Waiting  
2747433e234d: Pulling fs layer  
247b62a971d9: Waiting  
a2aba49d8dc0: Waiting
```

Lunch - Lanzado Jenkins

Usuario y contraseña: admin

URL: <http://localhost:8080>



**Welcome to Jenkins!**

☒ Keep me signed in

Sign in

# Prácticas de Jenkins

## Práctica 1

Se desea realizar un flujo de trabajo de CI/CD con Jenkins, para ello se debe realizar lo siguiente:

### Crear un nuevo proyecto en Jenkins

1. Seleccionar la opción de "Nueva tarea"
2. Poner nombre a la tarea
3. Seleccionar la opción de "Pipeline"
4. Agregas una descripción
5. Seleccionar la opción de "Advanced Project Options"
6. Tomar la opción de "Pipeline script from SCM" y seleccionar "Git" y agregar la dirección del git Fork a su usuario de GitHub, en mi caso es [github.com/omargo33/PIN1](https://github.com/omargo33/PIN1)

localhost:8080/job/Ejemplo%20de%20Tarea/confi...

de Tarea >

General Build Triggers Advanced Project Options **Pipeline**

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/omargo33/PIN1

Credentials ?

- none -

+ Add

Avanzado...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Guardar Apply

7. Guardar

## Primera ejecución

Volvemos a la pantalla principal de Jenkins y seleccionamos la tarea que acabamos de crear, en la parte izquierda seleccionamos la opción de "Build Now" y esperamos a que se ejecute el flujo de trabajo.

Y esta nos dará el siguiente resultado:

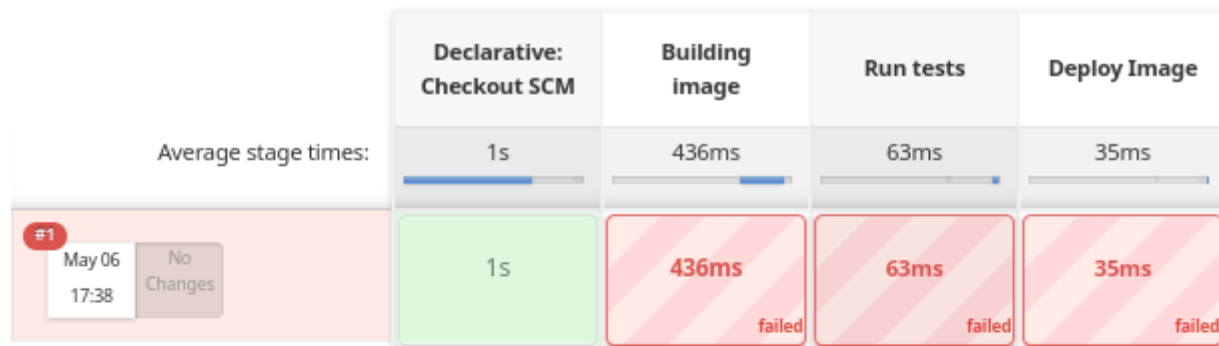
# Pipeline Ejemplo de Tarea

Prueba de Jenkins



[Recent Changes](#)

## Stage View



## Enlaces permanentes

- ["Última ejecución \(#1\) hace 10 Min"](#)
- ["Última ejecución fallida \(#1\) hace 10 Min"](#)
- ["Última ejecución fallida \(#1\) hace 10 Min"](#)
- ["Last completed build \(#1\) hace 10 Min"](#)

Como la corrida fue errónea, se procede a revisar el log.

Sobre la tarea creada, seleccionamos en la columna de estado de ejecución, nube con lluvia, y seleccionamos la opción de "Console Output" para ver el log de la ejecución.

↑ Back to Project

📄 Status

🔗 Changes

📄 Console Output

📄 View as plain text

✍ Edit Build Information

🗑 Delete build '#1'

🔗 Git Build Data

🌊 Open Blue Ocean

🔄 Restart from Stage

🔄 Replay

📋 Pipeline Steps

📁 Workspaces

❌ Salida de consola

Started by user [jenkinsadmin](#)

Obtained Jenkinsfile from git <https://github.com/omargo33/PIN1>

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [Jenkins](#) in /var/jenkins\_home/workspace/Ejemplo de Tarea

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

Cloning the remote Git repository

Cloning repository <https://github.com/omargo33/PIN1>

> git init /var/jenkins\_home/workspace/Ejemplo de Tarea # timeout=10

Fetching upstream changes from <https://github.com/omargo33/PIN1>

> git --version # timeout=10

> git --version # 'git version 2.30.2'

> git fetch --tags --force --progress -- <https://github.com/omargo33/PIN1> +refs/heads/\*:refs/remotes/origin/\* # timeout=10

> git config remote.origin.url <https://github.com/omargo33/PIN1> # timeout=10

> git config --add remote.origin.fetch +refs/heads/\*:refs/remotes/origin/\* # timeout=10

Avoid second fetch

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

Checking out Revision 15221f2a825f72658e117147e604a93fe449ed8b (refs/remotes/origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f 15221f2a825f72658e117147e604a93fe449ed8b # timeout=10

Commit message: "Update Jenkinsfile.all"

First time build. Skipping changelog.

En la que vemos que la carpeta webapp no existe, por lo que se procede a retirarla del archivo Jenkinsfile del repositorio y se vuelve a ejecutar el flujo de trabajo.

## Segunda ejecución

En una segunda ejecución, se procede a ejecutar el flujo de trabajo y se obtiene el siguiente resultado:

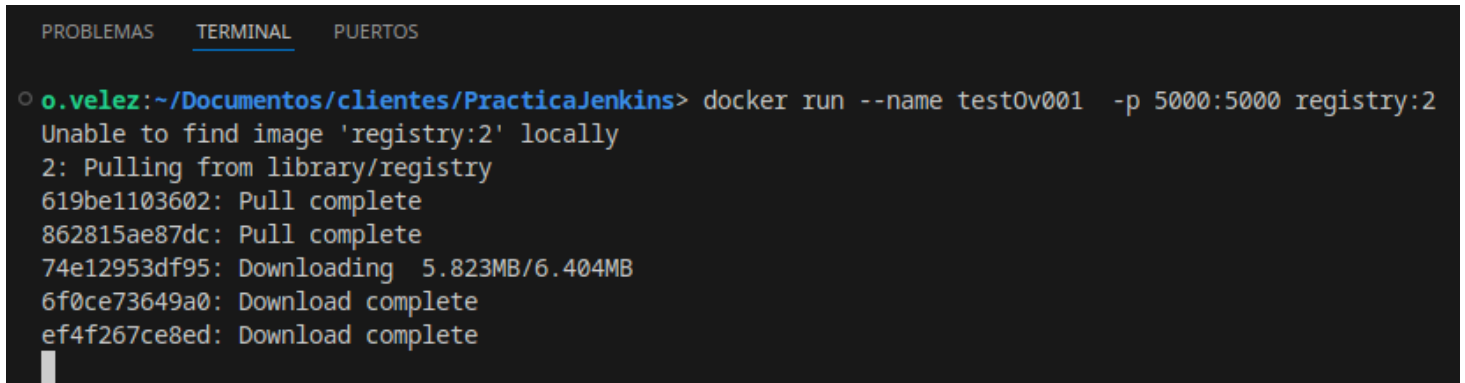
```
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy Image)
[Pipeline] sh
+ docker tag testapp 127.0.0.1:5000/mguazzardo/testapp
+ docker push 127.0.0.1:5000/mguazzardo/testapp
Using default tag: latest
The push refers to repository [127.0.0.1:5000/mguazzardo/testapp]
Get "http://127.0.0.1:5000/v2/": dial tcp 127.0.0.1:5000: connect: connection refused
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

En este se describe que no se puede hacer push a un servidor local sobre el puerto 5000, por lo que se procede a ejecutar el comando docker para levantar dicho servidor.

## Tercera ejecución

En este momento se ve que hace falta levantar un nuevo contenedor docker con el puerto 5000 activos con el siguiente comando:

```
docker run --name testOv001 -p 5000:5000 registry:2
```



```
PROBLEMAS  TERMINAL  PUERTOS

o.velez:~/Documentos/clientes/PracticaJenkins> docker run --name testOv001 -p 5000:5000 registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
619be1103602: Pull complete
862815ae87dc: Pull complete
74e12953df95: Downloading 5.823MB/6.404MB
6f0ce73649a0: Download complete
ef4f267ce8ed: Download complete
```

Y se realiza una nueva ejecución del flujo de trabajo, en la que se obtiene el siguiente resultado:



Panel de Control > Ejemplo de Tarea >

Back to Dashboard

Status

Changes

Construir ahora

Configurar

Borrar Pipeline

Full Stage View

Open Blue Ocean

Rename

Pipeline Syntax

Historia de tareas

Tendencia

Filter builds...

#3

7 may 2024 14:13

#2

6 may 2024 19:00

#1

6 may 2024 17:38

Atom feed Para Todos

Atom feed para los errores

Pipeline Ejemplo de Tarea

Prueba de Jenkins

Recent Changes

Stage View

Average stage times:

(Average full run time: ~19s)

#3

May 07 14:13

No Changes

#2

May 06 19:00

1 commit

#1

May 06 17:38

No Changes

Declarative: Checkout SCM	Building image	Run tests	Deploy Image
991ms	7s	1s	1s
856ms	8s	1s	4s
792ms	13s	2s	334ms failed
1s	436ms failed	63ms failed	35ms failed

Enlaces permanentes

Y con eso se da por terminado el flujo de trabajo.

## Práctica 2

### Crear un nuevo proyecto en Jenkins

1. Seleccionar la opción de "Nueva tarea"
2. Poner nombre a la tarea
3. Seleccionar la opción de "Pipeline"
4. Agregas una descripción
5. Seleccionar la opción de "Advanced Project Options"
6. Tomar la opción de "Pipeline script from SCM" y seleccionar "Git" y agregar la dirección del git Fork a su usuario de GitHub, en mi caso es [github.com/omargo33/PIN1](https://github.com/omargo33/PIN1)

## Enter an item name

Ejemplo Tarea 2

» Required field



### Crear un proyecto de estilo libre

Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.



### Pipeline

Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines (conocidas anteriormente como workflows) y/o para la organización de actividades complejas que no se pueden articular fácilmente con tareas de tipo freestyle.



### Crear un proyecto multi-configuración

Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.



### Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

## 7. Se cambia el nombre del archivo Jenkinsfile.all

Añadir ▾

Script Path ?

jenkinsfile.all

☒ Lightweight checkout ?

[Pipeline Syntax](#)

**Guardar** Apply

## 8. Guardar

# Primera ejecución

Volvemos a la pantalla principal de Jenkins y seleccionamos la tarea que acabamos de crear, en la parte izquierda seleccionamos la opción de "Build Now" y esperamos a que se ejecute el flujo de trabajo.

Y esta nos dará el siguiente resultado:

↑ Back to Dashboard

Status

</> Changes

▶ Construir ahora

⚙️ Configurar

🗑️ Borrar Pipeline

🔍 Full Stage View

🌊 Open Blue Ocean

✎ Rename

? Pipeline Syntax

Historia de tareasTendencia ▾

Q Filter builds...

#17 may 2024 14:25

Atom feed Para Todos

Atom feed para los errores

Pipeline Ejemplo Tarea 2

Tarea dos de pipeline

</>Recent Changes

Stage View

Average stage times:

Declarative: Checkout SCM	Building image	Run tests	Deploy Image	Pass To K8s
1s	1s	6s	586ms	1min 51s
1s	1s	6s	586ms	1min 51s <small>aborted</small>

#1May 07 14:25No Changes

Enlaces permanentes

- "Última ejecución (#1) hace 2 Min 14 Seg"
- "Última ejecución fallida (#1) hace 2 Min 14 Seg"
- "Last completed build (#1) hace 2 Min 14 Seg"

# Problemática

En este punto nos dimos cuenta de que el apartado "Pass To K8s" no se ejecutaba correctamente porque este hacía mención a un servidor de kubernetes que no tenemos aún acceso o instalado por lo que se decidió instalar un servidor de pruebas en nuestro equipo local, y optamos por un servidor minikube. Ver más en [minikube](#)

## Instalación de Minikube

Para lo cual usamos los siguientes comandos:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

```
PROBLEMAS  TERMINAL  PUERTOS

ritten=0
172.19.0.1 - - [07/May/2024:14:25:38 +0000] "PUT /v2/mguazzardo/testapp/manifests/latest HTTP/1.1" 201 0
"" "docker/24.0.5 go/go1.20.6 git-commit/a61e2b4 kernel/5.15.0-71-generic os/linux arch/amd64 UpstreamC
lient(Docker-Client/23.0.1 \\\(linux\\))"
● o.velez:~/Documentos/clientes/PracticaJenkins> curl -LO https://storage.googleapis.com/minikube/releases
/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 91.2M  100 91.2M    0     0  2665k      0  0:00:35  0:00:35 --:--:-- 2790k
● o.velez:~/Documentos/clientes/PracticaJenkins> sudo install minikube-linux-amd64 /usr/local/bin/minikube
&& rm minikube-linux-amd64
[sudo] contraseña para colaborador:
● o.velez:~/Documentos/clientes/PracticaJenkins> sudo install minikube-linux-amd64 /usr/local/bin/minikube
&& rm minikube-linux-amd64
install: no se puede efectuar `stat' sobre 'minikube-linux-amd64': No existe el archivo o el directorio
❖ o.velez:~/Documentos/clientes/PracticaJenkins> minikube start
🐹 minikube v1.33.0 en Ubuntu 20.04
🌟 Controlador docker seleccionado automáticamente. Otras opciones: virtualbox, none, ssh
🚀 Using Docker driver with root privileges
👍 Starting "minikube" primary control-plane node in "minikube" cluster
📡 Pulling base image v0.0.43 ...
📦 Descargando Kubernetes v1.30.0 ...
> preloaded-images-k8s-v18-v1...: 24.39 MiB / 342.90 MiB 7.11% 815.60 KiB
```

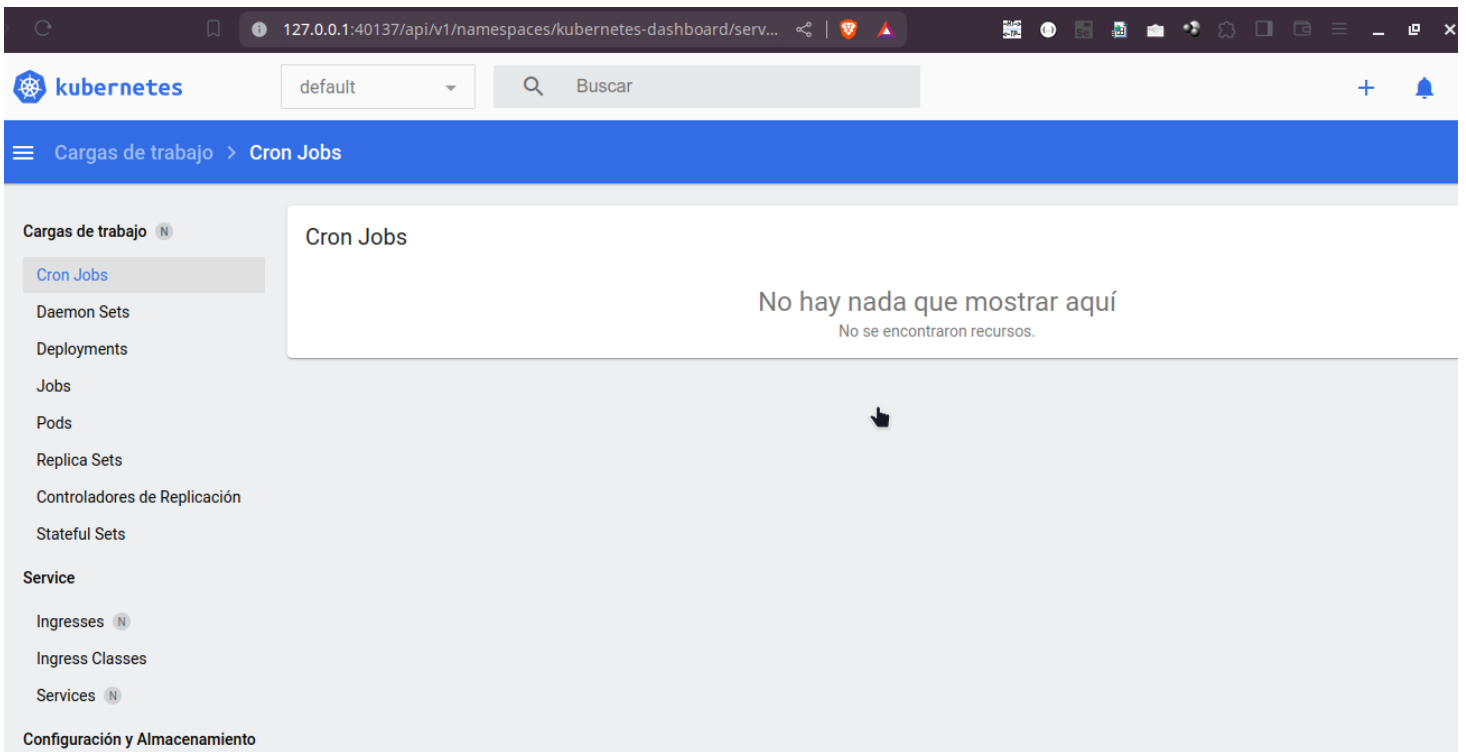
Una vez instalado lo iniciamos con el siguiente comando:

```
minikube start
```

Luego de esto lanzamos su dashboard con el siguiente comando:

```
minikube dashboard
```

Que nos permite visualizar nuestro cluster de kubernets. (de un unico nodo)



## Cambios en la tarea k8s

El archivo original de la tarea k8s es el siguiente:

```
stage('Pass To K8s'){

    steps {

        sh '''
        sshpass -p 'master' ssh 172.17.0.1 -l root -o StrictHostKeyChecking=no "kubect
        echo "Wait"
        sleep 10
        sshpass -p 'master' ssh 172.17.0.1 -l root -o StrictHostKeyChecking=no "kubect
        sshpass -p 'master' ssh 172.17.0.1 -l root -o StrictHostKeyChecking=no "wget h
        sshpass -p 'master' ssh 172.17.0.1 -l root -o StrictHostKeyChecking=no "kubect

        '''
    }
}
```

En la que se puede apreciar que se hace uso de un servidor de kubernetes en la ip 172.17.0.1 con el usuario root, cuya clave de acceso es master y para interactuar con el mismo se usa los comandos de kubectl.

Mientras que para nuestra instancia local de minikube se uso el siguiente comando:

```
minikube kubectl
```

Dando como resultado el siguiente archivo:

```
stage('Pass To K8s'){

    steps {

        sh '''
        sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyC
        echo "Wait"
        sleep 10
        sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyC
        sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyC
        sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyC

        '''

    }

}
```

En la que se puede apreciar que se hace uso de un servidor de kubernetes en la ip 172.18.5.35 y que el puerto ssh es el 21212, con el usuario colaborador, cuya clave de acceso es 12341234s y para interactuar con el mismo se usa los comandos de minikube kubectl.

## Ejecución 18

Luego de el trabajo de configuración del servidor, revisión de servicio ssh y la configuración de la tarea k8s, se procede a ejecutar el flujo de trabajo y se obtiene el siguiente resultado:



Historia de tareas

Tendencia 



#18

7 may 2024 19:33



#17

7 may 2024 19:33

		Declarative: Checkout SCM	Building image	Run tests	Deploy Image	Pass To K8s
Average stage times: (Average <u>full</u> run time: ~21s)		936ms	2s	1s	610ms	17s
#18 May 07 19:33	No Changes	1s	1s	2s	581ms	14s
#17 May 07 19:33	1 commit	1s	1s	2s	1s	1s failed
#16 May 07 19:31	1 commit	1s	3s	1s	580ms	12s

Y; el log de la ejecución:

```
+ sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyChecking=no minikube kubectl -- expose deployment testapp --port=3000
Ubuntu 20.04.6 LTS
service/testapp exposed
+ sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyChecking=no wget https://raw.githubusercontent.com/tercemundo/platzi-
scripts-integracion/master/webapp/nodePort.yml
Ubuntu 20.04.6 LTS
--2024-05-07 14:34:09-- https://raw.githubusercontent.com/tercemundo/platzi-scripts-integracion/master/webapp/nodePort.yml
Resolviendo raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.108.133, ...
Conectando con raw.githubusercontent.com (raw.githubusercontent.com)[185.199.109.133]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 289 [text/plain]
Guardando como: "nodePort.yml"

      OK                               100% 16,4M=0s

2024-05-07 14:34:11 (16,4 MB/s) - "nodePort.yml" guardado [289/289]

+ sshpass -p 12341234s ssh 172.18.5.35 -p 21212 -l colaborador -o StrictHostKeyChecking=no minikube kubectl -- apply -f nodePort.yml
Ubuntu 20.04.6 LTS
Warning: resource services/testapp is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl
apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be
patched automatically.
service/testapp configured
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // timeout
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

## Integrantes

Manuel Tinajero [manuel93tc@hotmail.com](mailto:manuel93tc@hotmail.com)

Sebastián Peña [penasantiago346@gmail.com](mailto:penasantiago346@gmail.com)

Omar Vélez [omargo33@gmail.com](mailto:omargo33@gmail.com)

Susy Robalino [susana.robolino@gmail.com](mailto:susana.robolino@gmail.com)

Joel Cittar [joelcittar@gmail.com](mailto:joelcittar@gmail.com)

## URI del proyecto:

<https://github.com/omargo33/PIN1>