

## Deployment of WordPress Application on Kubernetes

Descripción:

A través de este documento se listará y dará solución a aquellas actividades necesarias para el despliegue automatizado de una aplicación WordPress en Kubernetes.

Pre-requisitos

- Jenkins
- Docker
- Github
- Linux (Ubuntu)
- Kubernetes

Actividades

- Configuración de Kubernetes
- Configuración de Jenkins para correr Pipelines de Kubernetes
- Repositorio con Dockerfile
- Creación del Pipeline para el despliegue de WordPress sobre Kubernetes
- Validación en Kubernetes

## Configuración de Kubernetes

Antes de empezar, configuramos nuestro Cluster de Kubernetes para que Jenkins pueda desplegar de manera automatizada, nuestra aplicación de WordPress.

1. Reseteamos la configuración de Kubernetes para evitar malas configuraciones y hacer una configuración limpia, desde cero. `kubeadm reset`
2. Borramos el archivo de configuración de kubernetes `sudo rm ~/.kube/config`
3. Iniciamos el cluster de Kubernetes `kubeadm init`
4. Una vez finalizado la inicialización, realizamos los pasos que nos indica la inicialización.  
`mkdir -p $HOME/.kube`  
`sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config`  
`sudo chown $(id -u):$(id -g) $HOME/.kube/config`
5. Instalamos un Add-on Pod Network que nos permitirá la comunicación entre pods, por lo tanto, podremos tener el cluster solo con el nodo maestro.  
`kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"`
6. Si listamos nuestros nodos con el comando `kubectl get nodes` veremos nuestro nodo maestro con status ready.

7. Ejecutamos el siguiente comando `sudo kubectl taint nodes --all node-role.kubernetes.io/master-` para permitir que nuestro nodo maestro al ser nuestro único maestro, corra nuestro servicio.

8. Ahora es tiempo de asignarle permisos al usuario Jenkins para que pueda consumir la API de kubernetes y ejecutar nuestro despliegue de Wordpress.

```
kubectl -n default create sa jenkins
```

```
kubectl create clusterrolebinding jenkins --clusterrole cluster-admin --serviceaccount=default:jenkins
```

```
kubectl get -n default sa/jenkins --template='{{range .secrets}}{{ .name }}{{end}}' | xargs -n 1 kubectl -n <your-namespace> get secret --template='{{if .data.token}}{{ .data.token }}{{end}}' | head -n 1 | base64 -d -
```

9. La salida del último comando anterior es una clave de credenciales para que Jenkins pueda acceder a Kubernetes, por lo tanto, la guardaremos en Jenkins como una Secret Text --- Vamos a la página de Jenkins → Credentials → System → Global credentials → Add credentials →

Kind: Secret Text y

ID: SecretoKubernetes

oprimimos en 'OK' ... Esta credencial la utilizaremos después.

## Configuración de Jenkins para correr Pipelines de Kubernetes

Para la correcta configuración y ejecución de un Pipeline de Kubernetes en Jenkins, necesitamos primero instalar una serie de Plugins para Jenkins que nos hará más fácil la configuración del Pipeline.

- ☐ Kubernetes::Pipeline::Kubernetes Steps
- ☐ Kubernetes Credentials
- ☐ Kubernetes Cli
- ☐ Kubernetes
- ☐ ElasticBox Jenkins Kubenertes CI/CD

Vamos a la página de Jenkins → Administrar Jenkins → Administrar Plugins

Buscamos los Plugins anteriormente listados y finalmente oprimimos el boton 'Descargar ahora e instalar después de reiniciar'

Una vez descargado los Plugins, reiniciamos Jenkins 'systemctl restart jenkins' para la correcta instalación de los plugins.

Lanzamos Jenkins y vamos a → Administrar Jenkins → Configurar Sistema, hacemos scroll hasta la opción 'Añadir una nueva Nube' llenamos los siguientes parametros:

name: kubernetes

Kubernetes URL: <https://localhost:6443>

Disable https certificate check: check

Kubernetes Namespace: default

credentials: seleccionamos la credencial que creamos anteriormente 'secretoKubernetes'

Hacemos Click en 'Test connection' y deberíamos de recibir el siguiente mensaje

Connection test successful

## Repositorio con Dockerfile


Hacemos un Fork del repositorio <https://github.com/Simplilearn-Edu/DevOps-Wordpress-kubernetes-deployment>, en este repositorio tendremos nuestro archivo Dockerfile en el cual declaramos la imagen de Wordpress, por lo tanto, si nosotros le hacemos una modificación a la Imagen, haremos commit de esos de cambios al repositorio y desplegaremos la nueva imagen con Jenkins.


## Creación del Pipeline para el despliegue de WordPress sobre Kubernetes

Vamos a la página de Jenkins → Nueva Tarea

Ingresamos un nombre a la tarea 'Wordpress-deployment' y seleccionamos la opción 'crear un proyecto de estilo libre'

**Enter an item name**  
  
» Una tarea con el nombre 'wordpress-deployment' ya existe

**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**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Seleccionamos la opción 'GIT' en la sección de 'Configurar el Origen del Código fuente' y en el parámetro 'Repository URL' pegamos la URL de nuestro repositorio con el Dockerfile.

### Configurar el origen del código fuente

☐ Ninguno  
☒ Git

Repositories

Repository URL:

Credentials:  Add

Avanzado...

Add Repository

Vamos a la sección de 'Entorno de Ejecución' y chequeamos el atributo 'Configure Kubernetes CLI (kubectl)' ... agregamos los siguientes valores.

credentials: seleccionamos la credencial creada anteriormente

Kubernetes server endpoint: <https://localhost:6443>

Cluster name: kubernetes

Context name: kubernetes-admin@kubernetes

Namespace: default

☒ Configure Kubernetes CLI (kubectl)

Credentials:  Add

Kubernetes server endpoint:

Cluster name:

Context name:

Namespace:

Certificate of certificate authority:

Vamos a la sección de 'Ejecutar' → 'Añadir un nuevo paso' → 'Ejecutar líneas de comando shell' y Agregamos los siguientes comandos:

```
docker build -t wordpress .
docker images | grep wordpress
```

Agregamos otro nuevo paso de líneas de comando shell y agregamos los siguientes comandos:

```
kubectl run wordpress --image=wordpress:latest --port=80 --image-pull-policy=Never
kubectl expose deployment/wordpress --port=80 --target-port=80 --type=NodePort
kubectl describe services wordpress | grep Endpoints
kubectl get pods -o wide
```

Ahora es tiempo de ejecutar nuestro Pipeline 'Construir ahora' y ver la salida.

```

+ docker build -t wordpress .
Sending build context to Docker daemon 58.88kB

Step 1/2 : FROM wordpress:php7.1-apache
--> f0985bcc2fffb
Step 2/2 : COPY . /usr/src/wordpress/
--> c9ebaddf7bc0
Successfully built c9ebaddf7bc0
Successfully tagged wordpress:latest
+ docker images
+ grep wordpress
wordpress                latest                c9ebaddf7bc0         1 second ago         530MB
wordpress                php7.1-apache        f0985bcc2fffb        3 weeks ago          530MB
[wordpress-deployment] $ /bin/sh -xe /tmp/jenkins1441342099151211301.sh
+ kubectl run wordpress --image=wordpress:latest --port=80 --image-pull-policy=Never
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run
--generator=run-pod/v1 or kubectl create instead.
deployment.apps/wordpress created
+ kubectl expose deployment/wordpress --port=80 --target-port=80 --type=NodePort
service/wordpress exposed
+ kubectl describe services wordpress
+ grep -i port
Type:                NodePort
Port:                <unset> 80/TCP
TargetPort:          80/TCP
NodePort:            <unset> 30397/TCP
+ kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINESS GATES
wordpress-74cbb8ff59-6bhjr 0/1     Pending   0           1s    <none>      <none>        <none>           <none>
kubectl configuration cleaned up
Finished: SUCCESS

```

Después de la exitosa ejecución de nuestro Pipeline, vemos que se ha creado nuestro servicio de wordpress en Kubernetes y con el comando en nuestro pipeline 'kubectl describe services wordpress | grep Endpoints' obtuvimos el endpoint para acceder a nuestro servicio.

Al ingresar por el navegador con el endpoint, observamos la configuración inicial de Wordpress.



- English (United States)
- Afrikaans
- العربية
- العربية المغربية
- অসমীয়া
- گۆنئی آذربایجان
- Azərbaycan dili
- Беларуская мова
- Български
- বাংলা
- བོད་སྐད་
- Bosanski
- Català
- Cebuano
- Čeština
- Cymraeg
- Dansk
- Deutsch (Österreich)
- Deutsch (Schweiz, Du)
- Deutsch
- Deutsch (Sie)
- Deutsch (Schweiz)
- தமிழ்
- Ελληνικά
- English (Australia)