



# EN MI LOCAL FUNCIONA

TECHNICAL THOUGHTS, STORIES AND IDEAS

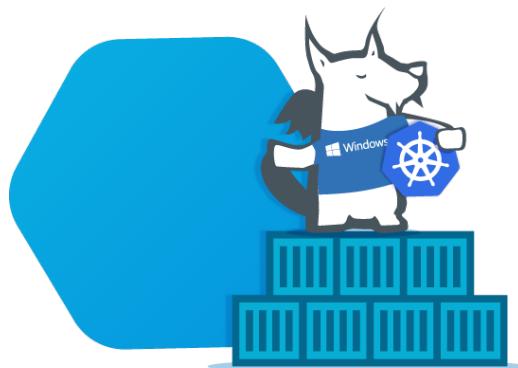


## Instalando y probando Kubernetes en Windows 10

Publicado por [Santi Macias](#) el 11 June 2019

[Microsoft](#) [Docker](#) [Kubernetes](#) [Windows](#) [Contenedores](#)

En un artículo anterior explicamos [como instalar Docker en Windows](#). Desde las últimas versiones de [Docker](#), se ha incluido [Kubernetes](#) de serie, el cual, podemos activar para desplegar nuestras imágenes y contenedores corriendo de forma local sin tener que realizar instalaciones complejas y ninguna otra herramienta en nuestro PC.



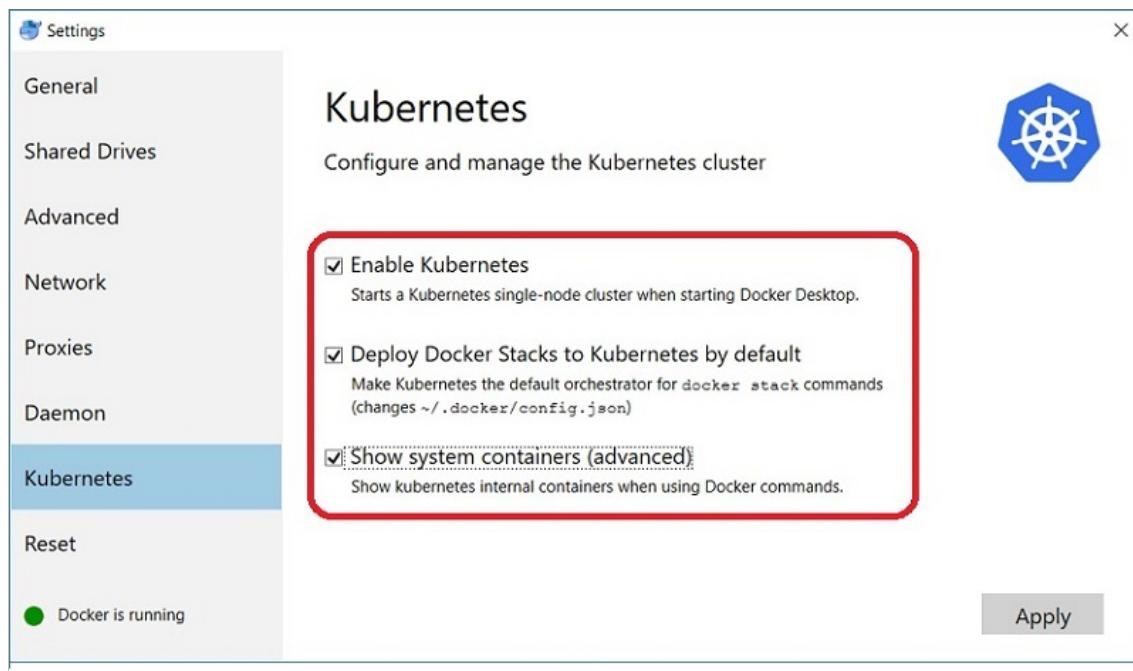
El objetivo de este artículo está pensado para entornos locales de formación, desarrollo y pruebas, donde veremos cómo instalar [Kubernetes en Windows 10](#) y desplegar una [aplicación ASP.NET Core](#) dentro del clúster mediante la línea de comandos. Hay otras alternativas como [Minikube](#), que no veremos en este artículo.

### Requerimientos Previos

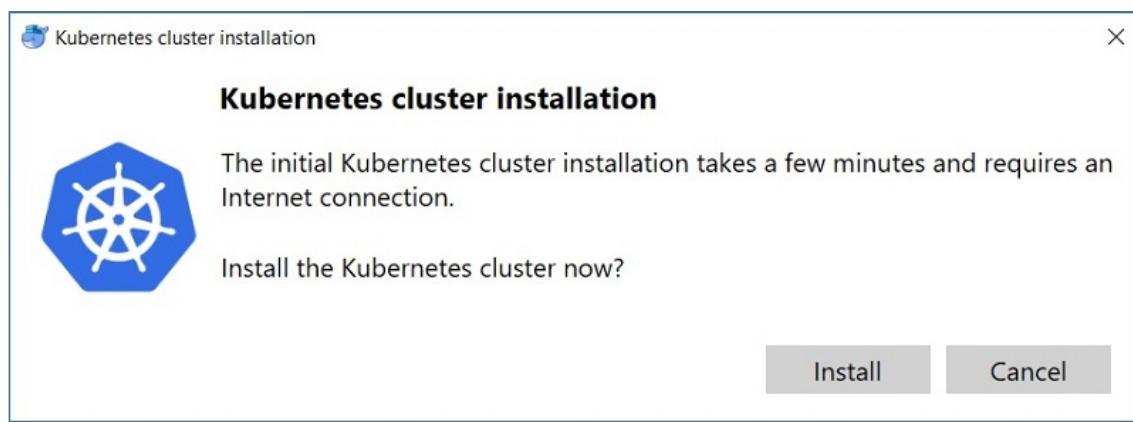
- Windows 10 Professional o superior.
- Hyper-V activado en Windows 10.
- Docker 18.02 en adelante para Windows.
- Conocimientos básicos de PowerShell, Docker y Kubernetes.

### Instalando Kubernetes desde Docker

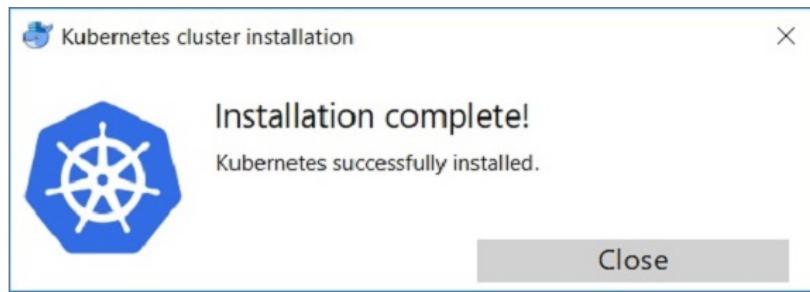
Por defecto, [Kubernetes](#) está desactivado y el proceso de instalación se realiza desde la propia interfaz de configuración de Docker que vemos aquí:



Una vez seleccionamos las opciones, pulsamos **Apply** y nos aparece una ventana de diálogo para confirmar la instalación de Kubernetes en nuestro PC:



Pulsamos **Install** y en unos minutos tendremos la instalación finalizada de un servidor Kubernetes con un clúster de un solo nodo, ya configurado y listo para usar.



## Kubectl y comandos útiles

Para trabajar con Kubernetes, tenemos la herramienta de línea de comandos **kubectl**, que sirve para arrancar, controlar, inspeccionar, gestionar, desplegar y escalar aplicaciones en el clúster y queda instalada de forma automática en el proceso anterior.

Usando **kubectl** desde **PowerShell**, accedemos a todos los recursos de Kubernetes mediante una serie de comandos que debemos conocer previamente. Estos son algunos ejemplos que utilizamos normalmente:

## Obtener información y diagnóstico:

```
kubectl version
kubectl cluster-info
kubectl get componentstatuses
```

## Obtener nodos, pods, deployments y services:

```
kubectl get all
kubectl get nodes -o wide
kubectl get rc -o wide
kubectl get pods -o wide
kubectl get pods -n kube-system
kubectl get deployments -o wide
kubectl get services -o wide
```

Para el resto de comandos aquí tenéis un enlace a cursos de Kubernetes y una super-chuleta ofrecida por la [Linux Academy](#):

Kubernetes Cheat Sheet		Deployments	ReplicaSets
What is Kubernetes?		Services	Roles
Viewing Resource Information		DaemonSets	Secrets
Nodes	\$ kubectl get no \$ kubectl get no -o wide \$ kubectl describe no \$ kubectl get no -o yaml \$ kubectl get node --selector=[label_name] \$ kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'	\$ kubectl get svc \$ kubectl describe svc \$ kubectl get svc -o wide \$ kubectl get svc -o yaml \$ kubectl get svc --show-labels	\$ kubectl get rs \$ kubectl descr \$ kubectl get r \$ kubectl get r
		Events	ConfigMaps
		\$ kubectl get ds \$ kubectl get ds --all-namespaces \$ kubectl describe ds [daemonset_name] -n [namespace_name] \$ kubectl get ds [ds_name] -n [ns_name] -o yaml	\$ kubectl get c \$ kubectl get c \$ kubectl get c
			Ingress

Enlace: <https://linuxacademy.com/blog/containers/kubernetes-cheat-sheet>

## Contexto Docker y Kubernetes

Un punto importante para no tener problemas al usar Kubernetes en nuestro PC, trata sobre configurar el contexto de **“docker-for-desktop”** para nuestro entorno local. Normalmente queda configurado, pero a veces he encontrado escenarios donde hay que hacerlo desde kubectl.

Para ver el nodo de Docker en Kubernetes, ejecutamos desde PowerShell:

```
PS C:\> kubectl get nodes
NAME           STATUS    ROLES      AGE     VERSION
docker-for-desktop   Ready    master    7d      v1.10.11
PS C:\>
```

Para consultar el contexto, ejecutar este comando desde PowerShell:

```
Windows PowerShell
PS C:\> kubectl config get-contexts
CURRENT      NAME          CLUSTER          AUTHINFO
*           docker-for-desktop   docker-for-desktop-cluster  docker-for-desktop

PS C:\>
```

Para configurarlo, ejecutamos este comando desde PowerShell:

```
Windows PowerShell
PS C:\> kubectl config use-context docker-for-desktop
Switched to context "docker-for-desktop".
PS C:\>
```

Una vez configurado correctamente ya podemos continuar.

## Dashboard de Kubernetes

Kubernetes también dispone de un Dashboard muy útil para consultar y gestionar el funcionamiento del clúster, que no viene instalado de forma predeterminada.

Activarlo en Windows 10, necesita algunos pasos, empezando por hacer un deploy desde PowerShell:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml
```

Para confirmar que se ha instalado correctamente veremos esto en la consola:

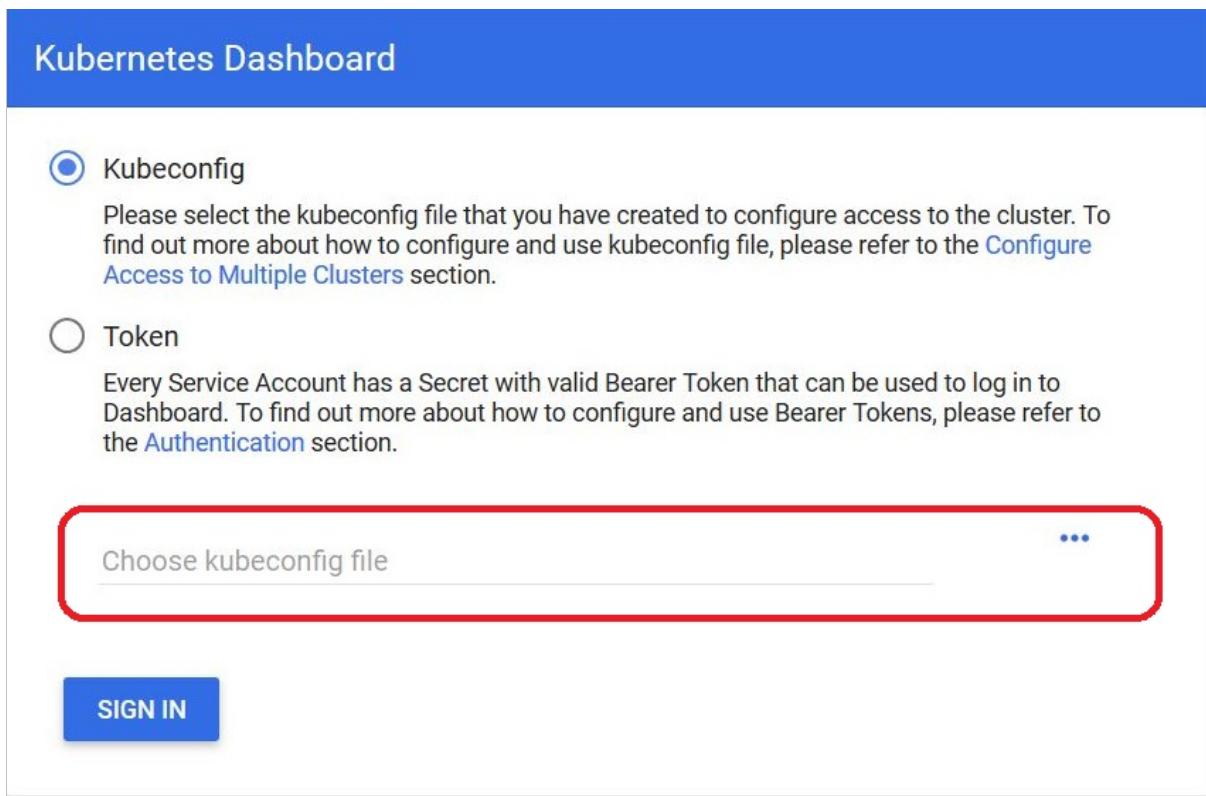
```
Seleccionar Windows PowerShell
PS C:\> kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml
secret "kubernetes-dashboard-certs" created
serviceaccount "kubernetes-dashboard" created
role.rbac.authorization.k8s.io "kubernetes-dashboard-minimal" created
rolebinding.rbac.authorization.k8s.io "kubernetes-dashboard-minimal" created
deployment.apps "kubernetes-dashboard" created
service "kubernetes-dashboard" created
PS C:\>
```

Una vez creado, necesitamos ejecutar el comando: `kubectl proxy`

```
Windows PowerShell
PS C:\> kubectl proxy
Starting to serve on 127.0.0.1:8001
```

Abrimos un navegador, vamos a las URL <http://localhost:8001> y <http://localhost:8001/ui>, pero veremos que **no aparece y devuelve JSON en lugar del Dashboard**. Eso es porque todavía nos falta crear y seleccionar el `kubeconfig` file.

Accedemos a esta URL para activar el Dashboard: <http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy>



Ejecutar estos comandos, **desde otra ventana PowerShell** para nuestro usuario:

```
PS> $TOKEN=((kubectl -n kube-system describe secret default | Select-String "token:") -split " ")[1]
PS> kubectl config set-credentials docker-for-desktop --token="$TOKEN"
```

Veremos que se crea la carpeta ".kube" y el fichero "config", que encontramos dentro de la carpeta c:\users\:

```
Windows PowerShell
PS C:\Users\smacias.rodriguez> $TOKEN=((kubectl -n kube-system describe secret default | Select-String "token:") -split " ")[1]
PS C:\Users\smacias.rodriguez> kubectl config set-credentials docker-for-desktop --token="$TOKEN"
User "docker-for-desktop" set.
PS C:\Users\smacias.rodriguez> DIR

Directorio: C:\Users\smacias.rodriguez

Mode                LastWriteTime         Length Name
----                -              -          -
d----        2018-07-06  12:43            .android
d----        2019-01-25  10:35            .aws
d----        2019-05-07  15:41            .azure
d----        2019-01-18  14:55            .azurerafunctions
d----        2018-07-16  9:30             .cisco
d----        2018-01-12  9:35             .config
d----        2018-06-27  13:02            .dnx
d----        2018-08-08  13:48            .docker
d----        2018-12-18  9:28             .dotnet
d----        2017-11-29  15:38            .groovy
d----        2017-11-29  17:31            .jenkins
d----        2019-05-23  13:01            .kube
d----        2017-09-19  9:47             .nuget
d----        2019-01-25  10:35            .serverless
d----        2018-06-07  12:23            .sonar
d----        2018-01-23  17:15            .sqllops
d----        2018-01-24  17:12            .templateengine
d----        2018-11-16  11:46            .vs
d----        2018-01-09  11:10            .vscode
```

The screenshot shows a Windows PowerShell window with the title "Windows PowerShell". It displays a command to set a token and then lists the contents of the ".kube" directory. The ".kube" directory is highlighted with a red box. Inside, there is a file named "config" which is also highlighted with a red box.

Seleccionamos dentro de ".kube" el fichero "config" en el dialogo de la interfaz web, pulsamos SIGN IN y aparecerá el panel de Kubernetes como se muestra en la imagen.

The screenshot shows the Kubernetes Dashboard interface. On the left, there's a sidebar with 'Cluster' and 'Namespaces' selected under 'Namespaces'. Below that, there are sections for 'Nodes', 'Persistent Volumes', 'Roles', and 'Storage Classes'. Under 'Namespace', 'default' is selected. The main area is titled 'Namespaces' and contains a table with four rows:

Name	Labels	Status	Age
docker	-	Active	7 days
kube-public	-	Active	7 days
kube-system	-	Active	7 days
default	-	Active	7 days

Ahora sí, ya tenemos el Dashboard operativo, y aunque no hemos desplegado nada todavía, nos facilitará información sobre el estado de los recursos del clúster y cualquier error que pueda ocurrir.

## Desplegando Aplicaciones

Como último paso, desplegaremos una aplicación **ASP.NET Core** utilizando las imágenes que ofrece **Microsoft en Docker Hub**: [https://hub.docker.com/\\_/microsoft-dotnet-core](https://hub.docker.com/_/microsoft-dotnet-core)

Primero la ejecutamos desde Docker mediante el siguiente comando:

```
docker run --name aspnetcore_sample --rm -it -p 8000:80 mcr.microsoft.com/dotnet/core/samples:aspnetapp
```

Una vez creado el contenedor, abrimos un navegador en <http://localhost:8000> para verla funcionando:

The screenshot shows the official ASP.NET Core website. At the top, it says "localhost:8000". Below the header, there are navigation links for "aspnetapp", "Home", "About", and "Contact". The main content area features the "ASP.NET Core" logo. To the right of the logo are three navigation links: "Windows", "Linux", and "OSX". Below these are two large arrows pointing left and right. A central banner reads "Learn how to build ASP.NET apps that can run anywhere." with a "Learn More" button. The main content is organized into four sections: "Application uses", "How to", "Overview", and "Run & Deploy", each with a list of bullet points.

Detenemos el contenedor, hacemos lo mismo mediante kubectl para Kubernetes, lanzamos los siguientes comandos para desplegar 3 réplicas y exponemos una dirección IP externa para acceder a nuestra aplicación "aspnetapp" en el clúster:

```
kubectl run aspnetapp --image=mcr.microsoft.com/dotnet/core/samples:aspnetapp --port=80 --replicas=3
kubectl expose deployment aspnetapp --type=NodePort
```

Comprobamos que "aspnetapp" ha sido creado correctamente:

The screenshot shows a Windows PowerShell window. The command "kubectl run aspnetapp --image=mcr.microsoft.com/dotnet/core/samples:aspnetapp --port=80 --replicas=3" is entered and followed by "deployment.apps "aspnetapp" created".

Ya tenemos el despliegue realizado en Kubernetes. Ahora verificamos el resultado mediante el comando: `kubectl get all` para ver que se han creado las 3 réplicas y el puerto expuesto para acceder a nuestra aplicación web:

The screenshot shows a Windows PowerShell window displaying the output of the "kubectl get all" command. It lists three pods: "aspnetapp-79f8fb9564-8mhkv", "aspnetapp-79f8fb9564-qwzbz4", and "aspnetapp-79f8fb9564-zmxf4", all in a "Running" state with 0 restarts and an age of 8 hours. It also lists two services: "service/aspnetapp" of type "NodePort" with an external IP of 10.99.222.24 and port 80 mapped to 31203/TCP, and "service/kubernetes" of type "ClusterIP" with an external IP of 10.96.0.1 and port 443/TCP. Finally, it shows the deployment "deployment.apps/aspnetapp" with 3 desired replicas, 3 current replicas, and an age of 8 hours. The "PORT(S)" column for the service is highlighted with a red box.

Si navegamos a <http://localhost:31203> tendremos la aplicación web funcionando:

The screenshot shows a web browser window with the URL [localhost:31203](http://localhost:31203) in the address bar. The page content is the ASP.NET Core documentation. At the top, there's a navigation bar with links for Windows, Linux, and OSX. Below this is a section with the text "Learn how to build ASP.NET apps that can run anywhere." and a "Learn More" button. The main content is organized into four columns:

- Application uses**
  - Sample pages using ASP.NET Core MVC
  - Theming using Bootstrap
- How to**
  - Add a Controller and View
  - Manage User Secrets using Secret Manager.
  - Use logging to log a message.
  - Add packages using NuGet.
  - Target development, staging or production environment.
- Overview**
  - Conceptual overview of what is ASP.NET Core
  - Fundamentals of ASP.NET Core such as Startup and middleware.
  - Working with Data
  - Security
  - Client side development
  - Develop on different platforms
- Run & Deploy**
  - Run your app
  - Run tools such as EF migrations and more
  - Publish to Microsoft Azure Web Apps

Como punto final, recordaros que este escenario es para entornos de formación, desarrollo y pruebas locales.

## Conclusiones

¡Hemos visto lo sencillo que ha sido instalar un ecosistema local de Kubernetes en nuestro PC local y como desplegar Pods desde imágenes y contenedores Docker en Kubernetes!

A partir de aquí, queda mucho por aprender sobre Kubernetes, cómo funciona, herramientas, addons, despliegues, entornos gestionados como AKS, EKS, GKE y mucho más, pero eso ya son temas que veremos en otros artículos.

Si te ha gustado, ¡síguenos en [Twitter](#) para estar al día de más artículos!



Autor

### SANTI MACIAS

Microsoft Tech Lead en atSistemas, +20 años trabajando con tecnologías Microsoft actualmente centrado sobretodo en Azure, Cloud Native, DevOps, Docker, Kubernetes, Microservicios y Serverless.

### COMPARTE



Please enable JavaScript to view the [comments powered by Disqus.](#)

[Condiciones de Uso](#)

Powered by [atSistemas](#)