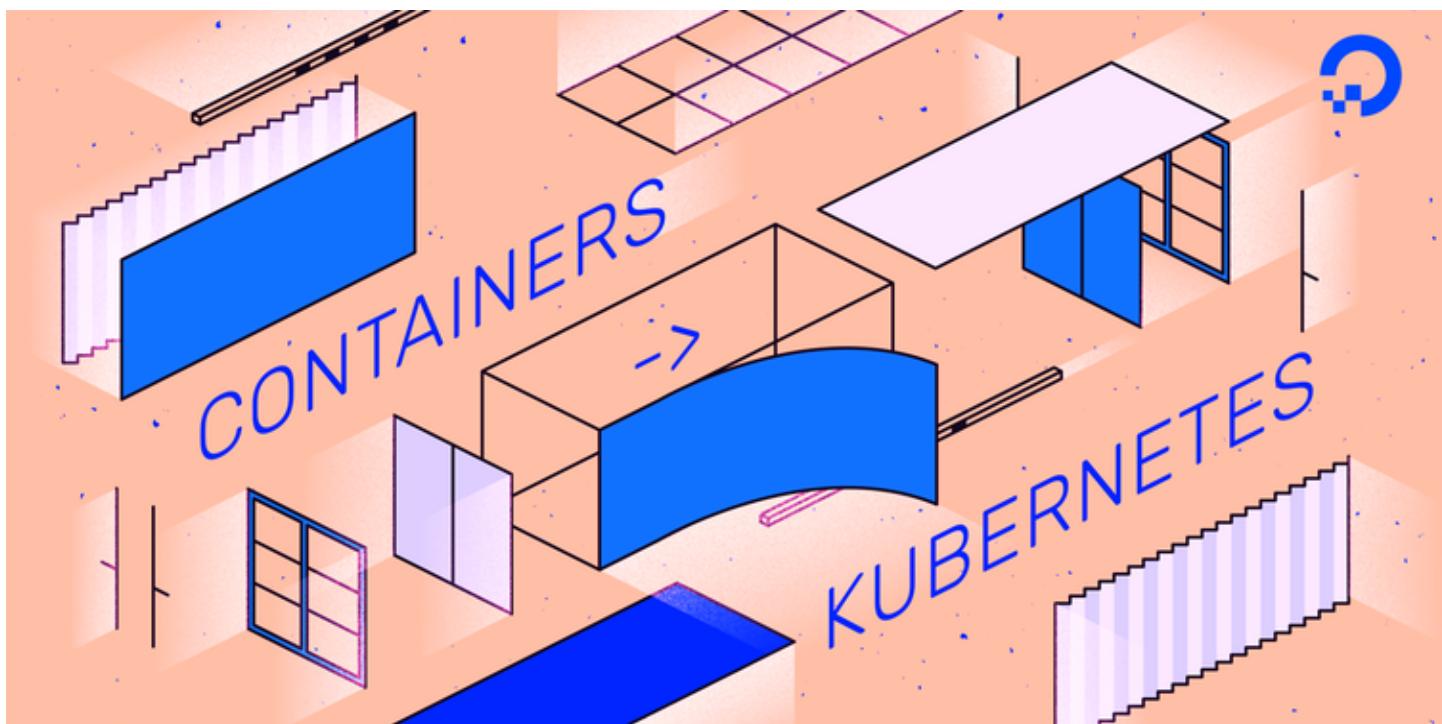


De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾



Community



How To Migrate a Docker Compose Workflow to Kubernetes

Posted April 3, 2019 ⚡53.2k

MONGODB

NODE.JS

DOCKER

KUBERNETES

MICROSERVICES

By [Kathleen Juell](#)

[Become an author](#)

Introduction

When building modern, stateless applications, containerizing your application's components is the first step in deploying and scaling on distributed platforms. If you have used Docker Compose in development, you will have modernized and

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

You will also have written service definitions that specify how your container images should run.

To run your services on a distributed platform like Kubernetes, you will need to translate your Compose service definitions to Kubernetes objects. This will allow you to scale your application with resiliency. One tool that can speed up the translation process to Kubernetes is kompose, a conversion tool that helps developers move Compose workflows to container orchestrators like Kubernetes or OpenShift.

In this tutorial, you will translate Compose services to Kubernetes objects using kompose. You will use the object definitions that kompose provides as a starting point and make adjustments to ensure that your setup will use Secrets, Services, and PersistentVolumeClaims in the way that Kubernetes expects. By the end of the tutorial, you will have a single-instance Node.js application with a MongoDB database running on a Kubernetes cluster. This setup will mirror the functionality of the code described in Containerizing a Node.js Application with Docker Compose and will be a good starting point to build out a production-ready solution that will scale with your needs.

Prerequisites

- A Kubernetes 1.10+ cluster with role-based access control (RBAC) enabled. This setup will use a DigitalOcean Kubernetes cluster, but you are free to create a cluster using another method.
- The `kubectl` command-line tool installed on your local machine or development server and configured to connect to your cluster. You can read more about installing `kubectl` in the official documentation.
- Docker installed on your local machine or development server. If you are working with Ubuntu 18.04, follow Steps 1 and 2 of How To Install and Use Docker on Ubuntu 18.04; otherwise, follow the official documentation for information about installing on other operating systems. Be sure to add your non-root user to the `docker` group, as described in Step 2 of the linked tutorial.
- A Docker Hub account. For an overview of how to set this up, refer to this introduction to Docker Hub.

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

link to the current release (version **1.18.0** as of this writing). Paste this link into the following `curl` command to download the latest version of kompose:

```
$ curl -L https://github.com/kubernetes/kompose/releases/download/v1.18.0/kompose-linux-amd64
```

For details about installing on non-Linux systems, please refer to the [installation instructions](#).

Make the binary executable:

```
$ chmod +x kompose
```

Move it to your PATH:

```
$ sudo mv ./kompose /usr/local/bin/kompose
```

To verify that it has been installed properly, you can do a version check:

```
$ kompose version
```

If the installation was successful, you will see output like the following:

Output

1.18.0 (06a2e56)

With `kompose` installed and ready to use, you can now clone the Node.js project code that you will be translating to Kubernetes.

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.



and

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

setup described in [Containerizing a Node.js Application for Development With Docker Compose](#), which uses a demo Node.js application to demonstrate how to set up a development environment using Docker Compose. You can find more information about the application itself in the series [From Containers to Kubernetes with Node.js](#).

Clone the repository into a directory called `node_project`:

```
$ git clone https://github.com/do-community/node-mongo-docker-dev.git node_project
```

Navigate to the `node_project` directory:

```
$ cd node_project
```

The `node_project` directory contains files and directories for a shark information application that works with user input. It has been modernized to work with containers: sensitive and specific configuration information has been removed from the application code and refactored to be injected at runtime, and the application's state has been offloaded to a MongoDB database.

For more information about designing modern, stateless applications, please see [Architecting Applications for Kubernetes](#) and [Modernizing Applications for Kubernetes](#).

The project directory includes a `Dockerfile` with instructions for building the application image. Let's build the image now so that you can push it to your Docker Hub account and use it in your Kubernetes setup.

Using the `docker build` command, build the image with the `-t` flag, which allows you to tag it with a memorable name. In this case, tag the image with your Docker Hub username and name it `node-kubernetes` or a name of your own choosing:

```
$ docker build -t your_dockerhub_username/node-kubernetes .
```

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.



images:

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js**Cómo migrar un flujo de trabajo d...**

You will see the following output:

Output

REPOSITORY	TAG	IMAGE ID	CREATED
your_dockerhub_username/node-kubernetes	latest	9c6f897e1fbc	3 seconds
node	10-alpine	94f3c8956482	12 days ago

Next, log in to the Docker Hub account you created in the prerequisites:

```
$ docker login -u your_dockerhub_username
```

Cuando se le solicite, ingrese la contraseña de su cuenta de Docker Hub. Iniciar sesión de esta manera creará un `~/.docker/config.json` archivo en el directorio de inicio de su usuario con sus credenciales de Docker Hub.

Empuje la imagen de la aplicación a Docker Hub con el `docker push` comando. Recuerde reemplazar `your_dockerhub_username` con su propio nombre de usuario de Docker Hub:

```
$ docker push your_dockerhub_username/node-kubernetes
```

Ahora tiene una imagen de aplicación que puede extraer para ejecutar su aplicación con Kubernetes. El siguiente paso será traducir las definiciones de servicio de su aplicación a objetos de Kubernetes.

Paso 3: traducción de servicios de composición a objetos de Kubernetes con kompose

Nuestro archivo Docker Compose, aquí llamado `docker-compose.yaml`, presenta las definiciones que ejecutarán nuestros servicios con Compose. Un *servicio* en Compose es un contenedor en ejecución, y *las definiciones de servicio* contienen información sobre cómo se ejecutará cada imagen del contenedor. En este paso, traduciremos estas

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

X
vos. Estos
scriben su

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Implementaciones , que contendrán información sobre el estado deseado de nuestros Pods; un PersistentVolumeClaim para aprovisionar almacenamiento para los datos de nuestra base de datos; un ConfigMap para variables de entorno injectadas en tiempo de ejecución; y un secreto para el usuario y contraseña de la base de datos de nuestra aplicación. Algunas de estas definiciones estarán en los archivos que kompose creará para nosotros, y otras tendremos que crearlas nosotros mismos.

Primero, necesitaremos modificar algunas de las definiciones en nuestro docker-compose.yaml archivo para trabajar con Kubernetes. Incluiremos una referencia a nuestra imagen de aplicación recién creada en nuestra nodejs definición de servicio y eliminaremos los montajes de enlace , los volúmenes y los comandos adicionales que usamos para ejecutar el contenedor de la aplicación en desarrollo con Compose. Además, redefiniremos las políticas de reinicio de ambos contenedores para que estén en línea con el comportamiento que Kubernetes espera .

Abra el archivo con nano o su editor favorito:

```
$ nano docker-compose.yaml
```

La definición actual para el nodejs servicio de aplicación se ve así:

```
~ / node_project / docker-compose.yaml
```

```
...
services:
  nodejs:
    build:
      context: .
      dockerfile: Dockerfile
    image: nodejs
    container_name: nodejs
    restart: unless-stopped
    env_file: .env
    environment:
      - MONGO_USERNAME=$MONGO_USERNAME
```

Regístrate para recibir nuestro boletín. X

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

```
- node_modules:/home/node/app/node_modules  
networks:  
- app-network  
  
command: ./wait-for.sh db:27017 -- /home/node/app/node_modules/.bin/nodemon app.js  
...  
...
```

Realice las siguientes ediciones en la definición de su servicio:

- Usa tu `node-kubernetes` imagen en lugar de la local `Dockerfile`.
- Cambie la `restart` política de contenedor de `unless-stopped` a `always`.
- Eliminar la `volumes` lista y las `command` instrucciones.

La definición del servicio terminado ahora se verá así:

~ / node_project / docker-compose.yaml

```
...  
services:  
nodejs:  
  image: your_dockerhub_username/node-kubernetes  
  container_name: nodejs  
  restart: always  
  env_file: .env  
  environment:  
    - MONGO_USERNAME=$MONGO_USERNAME  
    - MONGO_PASSWORD=$MONGO_PASSWORD  
    - MONGO_HOSTNAME=db  
    - MONGO_PORT=$MONGO_PORT  
    - MONGO_DB=$MONGO_DB  
  ports:  
    - "80:8080"  
  networks:  
    - app-network  
...  
...
```

A continuación desplácese hacia abajo hasta la dh definición del servicio. Aquí, realice Regístrate para recibir nuestro boletín. 

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

~~contenedor de base de datos utilizando el secreto vamos a crear en el paso 4.~~

La db definición del servicio ahora se verá así:

~ / node_project / docker-compose.yaml

...

```
db:
  image: mongo:4.1.8-xenial
  container_name: db
  restart: always
  environment:
    - MONGO_INITDB_ROOT_USERNAME=$MONGO_USERNAME
    - MONGO_INITDB_ROOT_PASSWORD=$MONGO_PASSWORD
  volumes:
    - dbdata:/data/db
  networks:
    - app-network
...
```

Finalmente, en la parte inferior del archivo, elimine los node_modules volúmenes de la volumes clave de nivel superior. La clave ahora se verá así:

~ / node_project / docker-compose.yaml

...

```
volumes:
  dbdata:
```

Guarde y cierre el archivo cuando termine de editar.

Antes de traducir nuestras definiciones de servicio, necesitaremos escribir el .env archivo que kompose usará para crear el ConfigMap con nuestra información no confidencial. Consulte el [Paso 2 de contener una aplicación Node.js para desarrollo con Docker Compose](#) para obtener una explicación más detallada de este archivo.

~~En ese tutorial agregamos .env a nuestro gitignore archivo para asegurarnos de que no Regístrate para recibir nuestro boletín.~~

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

X mos el

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

\$ nano .env

kompose usará este archivo para crear un ConfigMap para nuestra aplicación. Sin embargo, en lugar de asignar todas las variables de la `nodejs` definición del servicio en nuestro archivo Compose, agregaremos solo el `MONGO_DB` nombre de la base de datos y el `MONGO_PORT`. Asignaremos el nombre de usuario y la contraseña de la base de datos por separado cuando creamos manualmente un objeto secreto en el paso 4.

Agregue el siguiente puerto y la información del nombre de la base de datos al `.env` archivo. Siéntase libre de cambiar el nombre de su base de datos si desea:

~ / node_project / .env

`MONGO_PORT=27017`

`MONGO_DB=sharkinfo`

Guarde y cierre el archivo cuando termine de editar.

Ahora está listo para crear los archivos con sus especificaciones de objeto. kompose ofrece múltiples opciones para traducir sus recursos. Usted puede:

- Cree `yaml` archivos basados en las definiciones de servicio en su `docker-compose.yaml` archivo con `kompose convert`.
- Crea objetos Kubernetes directamente con `kompose up`.
- Crea un gráfico de `Helm` con `kompose convert -c`.

Por ahora, convertiremos nuestras definiciones de servicio en `yaml` archivos y luego los agregaremos y revisaremos los archivos que kompose crea.

Convierta sus definiciones de servicio en `yaml` archivos con el siguiente comando:

\$ kompose convert

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

X [Unsubscribe](#)

Ingrese su dirección de correo electrónico

[Regístrate](#)

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Output

```
INFO Kubernetes file "nodejs-service.yaml" created
INFO Kubernetes file "db-deployment.yaml" created
INFO Kubernetes file "dbdata-persistentvolumeclaim.yaml" created
INFO Kubernetes file "nodejs-deployment.yaml" created
INFO Kubernetes file "nodejs-env-configmap.yaml" created
```

Estos incluyen `.yaml` archivos con especificaciones para el Servicio de aplicación de Nodo, Implementación y ConfigMap, así como para la `dbdata` Implementación de la base de datos PersistentVolumeClaim y MongoDB.

Estos archivos son un buen punto de partida, pero para que la funcionalidad de nuestra aplicación coincida con la configuración descrita en [Containerizing a Node.js Application for Development With Docker Compose](#), necesitaremos hacer algunas adiciones y cambios en los archivos que kompose ha generado.

Paso 4 - Creando secretos de Kubernetes

Para que nuestra aplicación funcione de la manera que esperamos, necesitaremos hacer algunas modificaciones en los archivos que kompose ha creado. El primero de estos cambios generará un secreto para el usuario y la contraseña de nuestra base de datos y lo agregará a nuestras implementaciones de aplicaciones y bases de datos. Kubernetes ofrece dos formas de trabajar con variables de entorno: ConfigMaps y Secrets. kompose ya ha creado un ConfigMap con la información no confidencial que incluimos en nuestro `.env` archivo, por lo que ahora crearemos un secreto con nuestra información confidencial: nuestro nombre de usuario y contraseña de la base de datos.

El primer paso para crear manualmente un Secreto será convertir su nombre de usuario y contraseña a [base64](#), un esquema de codificación que le permite transmitir datos de manera uniforme, incluidos los datos binarios.

Convierta su nombre de usuario de la base de datos:

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Luego, convierta su contraseña:

```
$ echo -n 'your_database_password' | base64
```

Tome nota del valor en la salida aquí también.

Abra un archivo para el secreto:

```
$ nano secret.yaml
```

Nota: Los objetos de Kubernetes generalmente se definen usando YAML, que prohíbe estrictamente las pestañas y requiere dos espacios para la sangría. Si desea verificar el formato de cualquiera de sus yaml archivos, puede usar un linter o probar la validez de su sintaxis `kubectl create` con las banderas `--dry-run` y `--validate`:

```
$ kubectl create -f your_yaml_file.yaml --dry-run --validate=true
```

En general, es una buena idea validar su sintaxis antes de crear recursos con `kubectl`.

Agregue el siguiente código al archivo para crear un secreto que definirá su `MONGO_USERNAME` y `MONGO_PASSWORD` utilizando los valores codificados que acaba de crear. Asegúrese de reemplazar los valores ficticios aquí con su nombre de usuario y contraseña **codificados**:

~ / node_project / secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mongo-secret
data:
  MONGO_USERNAME: your_encoded_username
```

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

o desee.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

fueras de control de versiones.

Una vez `secret.yaml` escrito, nuestro próximo paso será asegurarnos de que nuestros Pods de aplicaciones y bases de datos usen los valores que agregamos al archivo. Comencemos agregando referencias al secreto a la implementación de nuestra aplicación.

Abra el archivo llamado `nodejs-deployment.yaml`:

```
$ nano nodejs-deployment.yaml
```

Las especificaciones del contenedor del archivo incluyen las siguientes variables de entorno definidas bajo la `env` clave:

`~ / node_project / nodejs-despliegue.yaml`

```
apiVersion: extensions/v1beta1
kind: Deployment
...
spec:
  containers:
    - env:
        - name: MONGO_DB
          valueFrom:
            configMapKeyRef:
              key: MONGO_DB
              name: nodejs-env
        - name: MONGO_HOSTNAME
          value: db
        - name: MONGO_PASSWORD
        - name: MONGO_PORT
          valueFrom:
            configMapKeyRef:
              key: MONGO_PORT
              name: nodejs-env
        - name: MONGO_USERNAME
```

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Agregue las siguientes referencias secretas a las variables MONGO_USERNAME y MONGO_PASSWORD:

~ / node_project / nodejs-despliegue.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
...
spec:
  containers:
    - env:
        - name: MONGO_DB
          valueFrom:
            configMapKeyRef:
              key: MONGO_DB
              name: nodejs-env
        - name: MONGO_HOSTNAME
          value: db
        - name: MONGO_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: MONGO_PASSWORD
        - name: MONGO_PORT
          valueFrom:
            configMapKeyRef:
              key: MONGO_PORT
              name: nodejs-env
        - name: MONGO_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: MONGO_USERNAME
```

Guarde y cierre el archivo cuando termine de editar.

A continuación, agregaremos los mismos valores al db-deployment.yaml archivo.

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

variables: MONGO_INITDB_ROOT_USERNAME y MONGO_INITDB_ROOT_PASSWORD. La mongo imagen pone a disposición estas variables para que pueda modificar la inicialización de su instancia de base de datos. MONGO_INITDB_ROOT_USERNAME y MONGO_INITDB_ROOT_PASSWORD juntos crean un root usuario en la admin base de datos de autenticación y se aseguran de que la autenticación esté habilitada cuando se inicia el contenedor de la base de datos.

El uso de los valores que establecemos en nuestro Secreto garantiza que tendremos un usuario de la aplicación con root privilegios en la instancia de la base de datos, con acceso a todos los privilegios administrativos y operativos de ese rol. Cuando trabaje en producción, deseará crear un usuario de aplicación dedicado con privilegios de alcance apropiado.

Debajo de las variables MONGO_INITDB_ROOT_USERNAME y MONGO_INITDB_ROOT_PASSWORD , agregue referencias a los valores secretos:

~ / node_project / db-deploy.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
...
spec:
  containers:
    - env:
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: MONGO_PASSWORD
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: MONGO_USERNAME
      image: mongo:4.1.8-xenial
...
```

Regístrate para recibir nuestro boletín. X

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

que esté completamente configurada e inicializada.

Paso 5: creación del servicio de base de datos y un contenedor de inicio de aplicación

Ahora que tenemos nuestro secreto, podemos pasar a crear nuestro Servicio de base de datos y un Contenedor de inicio que sondeará este Servicio para garantizar que nuestra aplicación solo intente conectarse a la base de datos una vez que las tareas de inicio de la base de datos, incluida la creación del `MONGO_INITDB` usuario y contraseña están completos

Para una discusión sobre cómo implementar esta funcionalidad en Compose, consulte el [Paso 4 de Contenedor de una aplicación Node.js para desarrollo con Docker Compose](#).

Abra un archivo para definir las especificaciones para el Servicio de base de datos:

```
$ nano db-service.yaml
```

Agregue el siguiente código al archivo para definir el Servicio:

```
~/node_project/db-service.yaml
```

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    kompose.cmd: kompose convert
    kompose.version: 1.18.0 (06a2e56)
  creationTimestamp: null
  labels:
    io.kompose.service: db
  name: db
spec:
  ports:
    port: 27017
```

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Lo selector que hemos incluido aquí coincidirá con este objeto de Servicio con nuestros Pods de base de datos, que han sido definidos con la etiqueta `io.kompose.service: db` por kompose en el `db-deployment.yaml` archivo. También hemos llamado a este servicio `db`.

Guarde y cierre el archivo cuando termine de editar.

A continuación, agreguemos un campo de contenedor de inicio a la `containers` matriz `nodejs-deployment.yaml`. Esto creará un Contenedor Init que podemos usar para retrasar el inicio de nuestro contenedor de aplicaciones hasta que el `db` Servicio se haya creado con un Pod al que se pueda acceder. Este es uno de los posibles usos para los contenedores Init; Para obtener más información sobre otros casos de uso, consulte la [documentación oficial](#).

Abre el `nodejs-deployment.yaml` archivo:

```
$ nano nodejs-deployment.yaml
```

Dentro de la especificación Pod y junto a la `containers` matriz, vamos a agregar un `initContainers` campo con un contenedor que sondeará el `db` Servicio.

Agregue el siguiente código debajo de las `ports` y `resources` los campos y por encima del `restartPolicy` de la `nodejs` `containers` matriz:

~ / node_project / nodejs-despliegue.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
...
spec:
  containers:
    ...
      name: nodejs
      ports:
```

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

...

Este contenedor de inicialización utiliza la [imagen BusyBox](#), una imagen ligera que incluye muchas utilidades UNIX. En este caso, usaremos la `netcat` utilidad para sondear si el Pod asociado con el `db` Servicio acepta o no conexiones TCP en el puerto `27017`.

Este contenedor `command` replica la funcionalidad del `wait-for` script que eliminamos de nuestro `docker-compose.yaml` archivo en el [Paso 3](#). Para una discusión más larga sobre cómo y por qué nuestra aplicación usó el `wait-for` script al trabajar con Compose, consulte el [Paso 4 de Contenedor de una aplicación Node.js para desarrollo con Docker Compose](#).

Los contenedores `Init` se ejecutan hasta su finalización; en nuestro caso, esto significa que nuestro contenedor de la aplicación Node no se iniciará hasta que el contenedor de la base de datos se esté ejecutando y acepte conexiones en el puerto `27017`. La `db` definición del Servicio nos permite garantizar esta funcionalidad independientemente de la ubicación exacta del contenedor de la base de datos, que es mutable.

Guarde y cierre el archivo cuando termine de editar.

Con su Servicio de base de datos creado y su Contenedor `Init` en su lugar para controlar el orden de inicio de sus contenedores, puede pasar a verificar los requisitos de almacenamiento en su `PersistentVolumeClaim` y exponer su servicio de aplicación utilizando un [LoadBalancer](#).

Paso 6: modificación del reclamo de volumen persistente y exposición de la interfaz de la aplicación

Antes de ejecutar nuestra aplicación, haremos dos cambios finales para asegurarnos de que el almacenamiento de nuestra base de datos se aprovisionará correctamente y que podamos exponer nuestra interfaz de aplicación utilizando un `LoadBalancer`.

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

X m que
nente el

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

trabaja con [digitalocean Kubernetes](#), nuestra Almacenamiento predeterminado provisioner se establece en `dobs.csi.digitalocean.com` - [digitalocean bloque de almacenamiento](#).

Podemos verificar esto escribiendo:

```
$ kubectl get storageclass
```

Si está trabajando con un clúster de DigitalOcean, verá el siguiente resultado:

Output

NAME	PROVISIONER	AGE
do-block-storage (default)	<code>dobs.csi.digitalocean.com</code>	76m

Si no está trabajando con un clúster de DigitalOcean, deberá crear una StorageClass y configurar una provisioner de su elección. Para obtener detalles sobre cómo hacer esto, consulte la [documentación oficial](#).

Cuando se crea kompose `dbdata-persistentvolumeclaim.yaml`, establece storage resource un tamaño que no cumple con los requisitos de tamaño mínimo de nuestro provisioner. Por lo tanto, tendremos que modificar nuestra PersistentVolumeClaim para usar la [unidad mínima de almacenamiento de bloque DigitalOcean viable : 1 GB](#). No dude en modificar esto para cumplir con sus requisitos de almacenamiento.

Abierto `dbdata-persistentvolumeclaim.yaml`:

```
$ nano dbdata-persistentvolumeclaim.yaml
```

Reemplace el `storage` valor con `1Gi`:

~ / node_project / dbdata-persistentvolumeclaim.yaml

`apiVersion: v1`

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

```
accessModes:  
- ReadWriteOnce  
resources:  
requests:  
storage: 1Gi  
status: {}
```

Also note the `accessMode: ReadWriteOnce` means that the volume provisioned as a result of this Claim will be read-write only by a single node. Please see the [documentation](#) for more information about different access modes.

Save and close the file when you are finished.

Next, open `nodejs-service.yaml`:

```
$ nano nodejs-service.yaml
```

We are going to expose this Service externally using a [DigitalOcean Load Balancer](#). If you are not using a DigitalOcean cluster, please consult the relevant documentation from your cloud provider for information about their load balancers. Alternatively, you can follow the official [Kubernetes documentation](#) on setting up a highly available cluster with `kubeadm`, but in this case you will not be able to use `PersistentVolumeClaims` to provision storage.

Within the Service spec, specify `LoadBalancer` as the Service type:

```
~/node_project/nodejs-service.yaml
```

```
apiVersion: v1  
kind: Service  
...  
spec:  
  type: LoadBalancer  
  ports:  
  ...
```

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

ed,

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

With all of our files in place, we are ready to start and test the application.

Step 7 – Starting and Accessing the Application

It's time to create our Kubernetes objects and test that our application is working as expected.

To create the objects we've defined, we'll use `kubectl create` with the `-f` flag, which will allow us to specify the files that kompose created for us, along with the files we wrote. Run the following command to create the Node application and MongoDB database Services and Deployments, along with your Secret, ConfigMap, and PersistentVolumeClaim:

```
$ kubectl create -f nodejs-service.yaml,nodejs-deployment.yaml,nodejs-env-configmap.yaml,d
```

You will see the following output indicating that the objects have been created:

Output

```
service/nodejs created
deployment.extensions/nodejs created
configmap/nodejs-env created
service/db created
deployment.extensions/db created
persistentvolumeclaim/dbdata created
secret/mongo-secret created
```

To check that your Pods are running, type:

```
$ kubectl get pods
```

You don't need to specify a Namespace here, since we have created our objects in the default Namespace. If you're working with multiple Namespaces, be sure to include the [Regístrate para recibir nuestro boletín](#).

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

[Regístrate](#)

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Output

NAME	READY	STATUS	RESTARTS	AGE
db-679d658576-kfpsl	0/1	ContainerCreating	0	10s
nodejs-6b9585dc8b-pnsws	0/1	Init:0/1	0	10s

Once that container has run and your application and database containers have started, you will see this output:

Output

NAME	READY	STATUS	RESTARTS	AGE
db-679d658576-kfpsl	1/1	Running	0	54s
nodejs-6b9585dc8b-pnsws	1/1	Running	0	54s

The `Running` STATUS indicates that your Pods are bound to nodes and that the containers associated with those Pods are running. `READY` indicates how many containers in a Pod are running. For more information, please consult the [documentation on Pod lifecycles](#).

Note:

If you see unexpected phases in the `STATUS` column, remember that you can troubleshoot your Pods with the following commands:

```
$ kubectl describe pods your_pod  
$ kubectl logs your_pod
```

With your containers running, you can now access the application. To get the IP for the LoadBalancer, type:

```
$ kubectl get svc
```

You will see the following output:

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

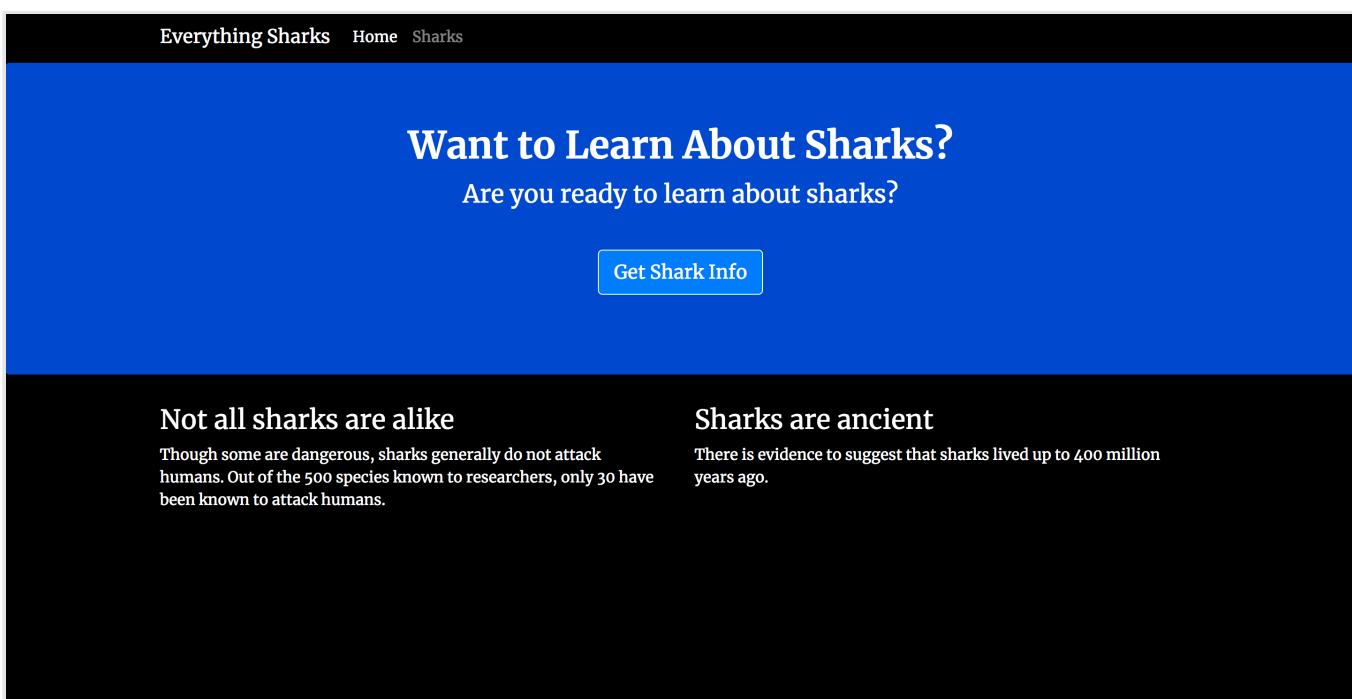
Cómo migrar un flujo de trabajo d... ▾

nodejs	LoadBalancer	10.245.15.56	your_lb_ip	80:30729/TCP	93s
--------	--------------	--------------	------------	--------------	-----

The EXTERNAL_IP associated with the nodejs service is the IP address where you can access the application. If you see a <pending> status in the EXTERNAL_IP column, this means that your load balancer is still being created.

Once you see an IP in that column, navigate to it in your browser: http://your_lb_ip.

You should see the following landing page:



Click on the **Get Shark Info** button. You will see a page with an entry form where you can enter a shark name and a description of that shark's general character:

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Regístrate

De contenedores a Kubernetes con Node.js >

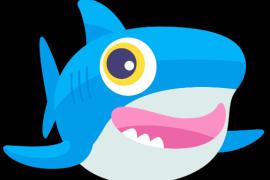
Cómo migrar un flujo de trabajo d... ▾

Shark Info

Some sharks are known to be dangerous to humans, though many more are not. The sawshark, for example, is not considered a threat to humans.



Other sharks are known to be friendly and welcoming!



Enter Your Shark

Shark Name
Shark Character

Submit

In the form, add a shark of your choosing. To demonstrate, we will add **Megalodon Shark** to the **Shark Name** field, and **Ancient** to the **Shark Character** field:

Everything Sharks Home Sharks

Shark Info

Some sharks are known to be dangerous to humans, though many more are not. The sawshark, for example, is not considered a threat to humans.



Other sharks are known to be friendly and welcoming!



Enter Your Shark

Megalodon Shark
Ancient

Submit

Click on the **Submit** button. You will see a page with this shark information displayed back to you:

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

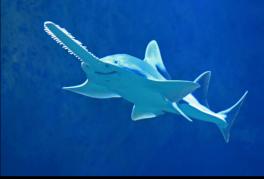
Regístrate

De contenedores a Kubernetes con Node.js >

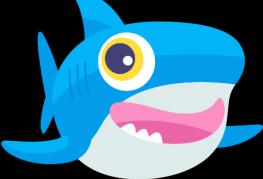
Cómo migrar un flujo de trabajo d... ▾

Shark Info

Some sharks are known to be dangerous to humans, though many more are not. The sawshark, for example, is not considered a threat to humans.



Other sharks are known to be friendly and welcoming!



Your Sharks
Name: Megalodon Shark
Character: Ancient

You now have a single instance setup of a Node.js application with a MongoDB database running on a Kubernetes cluster.

Conclusion

The files you have created in this tutorial are a good starting point to build from as you move toward production. As you develop your application, you can work on implementing the following:

- **Centralized logging and monitoring.** Please see the [relevant discussion in Modernizing Applications for Kubernetes](#) for a general overview. You can also look at [How To Set Up an Elasticsearch, Fluentd and Kibana \(EFK\) Logging Stack on Kubernetes](#) to learn how to set up a logging stack with [Elasticsearch](#), [Fluentd](#), and [Kibana](#). Also check out [An Introduction to Service Meshes](#) for information about how service meshes like [Istio](#) implement this functionality.
- **Ingress Resources to route traffic to your cluster.** This is a good alternative to a LoadBalancer in cases where you are running multiple Services, which each require their own LoadBalancer, or where you would like to implement application-level routing strategies (A/B & canary tests, for example). For more information, check out [How to Set Up an Nginx Ingress with Cert-Manager on DigitalOcean Kubernetes](#) and the [related discussion of routing in the service mesh context in An Introduction to Service Meshes](#).

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

By Kathleen Juell

Was this helpful?

Yes

No



[Report an issue](#)

Tutorial Series

From Containers to Kubernetes with Node.js

In this series, you will build and containerize a Node.js application with a MongoDB database. The series is designed to introduce you to the fundamentals of migrating an application to Kubernetes, including modernizing your app using the 12FA methodology, containerizing it, and deploying it to Kubernetes. The series also includes information on deploying your app with Docker Compose using an Nginx reverse proxy and Let's Encrypt.

[Next in series: How To Scale a Node.js Application with MongoDB on Kubernetes Using Helm →](#)

Related

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Testing is an integral part of software development. With the right test setup, this process can be ...

In this guide, you will take basic steps to secure your DigitalOcean Kubernetes cluster. You will ...

TUTORIAL

How To Automate Your Node.js Production Deployments with Shipit on CentOS 7

Shipit is a universal automation and deployment tool for ...

TUTORIAL

How To Set Up ReadWriteMany (RWX) Persistent Volumes with NFS on DigitalOcean Kubernetes

With the digitalocean-csi, DigitalOcean Block ...

Still looking for an answer?



Ask a question



Search for more help

10 Comments

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

Sign In to Comment

bronzetab April 10, 2019

I was looking for this article without knowing it. A big help. Finding articles like this is one of the many reasons I like DO so much! Thanks.

[Reply](#) [Report](#)

katjuell April 15, 2019

Thank you [@bronzetab](#)! That is great to hear

[Reply](#) [Report](#)

vantrixqa April 15, 2019

There's an error with your instructions.

The correct command is:

```
kubectl create -f nodejs-service.yaml,nodejs-deployment.yaml,nodejs-env-configmap.y
```

You are missing the db-env-configmap.yaml there so when trying to bring up the pods, you'll see a

Error: configmaps "db-env" not found

error.

[Reply](#) [Report](#)

Regístrate para recibir nuestro boletín.

Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

nt. Instead,
in Step 4.
er-

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

with kompose.

[Reply](#) [Report](#)

vantrixqa April 16, 2019

0 Ah. My apologies.

You're correct, I haven't removed `env_file` option from the `db` so the produced `db-deployment.yaml` still expected to get `MONGO_DB` and `MONGO_PORT` from a `configMapKeyRef` to the `db-env`.

I was passing the sensitive information via secrets though.

Thanks for the awesome tutorial!

[Reply](#) [Report](#)

katjuell April 16, 2019

1 Ah great! I've also rewritten the editing instructions for the `db` service definition in Step 3 to be a bulleted list. Hopefully that will pull that instruction out a bit more.

[Reply](#) [Report](#)

noobmaster February 2, 2020

0 Thanks for the tutorial. I am pretty new to this and have a couple questions –

1. Why did we remove volumes relating to NodesJS in docker-compose?
2. Why would Kompose not create `db_service.yaml` already?

Thanks!

[Reply](#) [Report](#)

noobmaster February 2, 2020

0 Also how to support multiple `depends_on` with the `initContainers` using the command?
Thanks

[Reply](#) [Report](#)

Foarsitter February 21, 2020

0 Thanks for your tutorial!

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

error:

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾

nodejs in the corresponding files will solve this.

Also the value of `apiVersion` has changed to `apps/v1`.

Learned a lot, thanks again!

[Reply](#) [Report](#)

^ [viveksysops](#) March 13, 2020

0 Hello

The status of loadbalancer is showing down. How can I change it?

[Reply](#) [Report](#)



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



BECOME A CONTRIBUTOR

You get paid; we donate to tech non-profits

Regístrate para recibir nuestro boletín.



Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

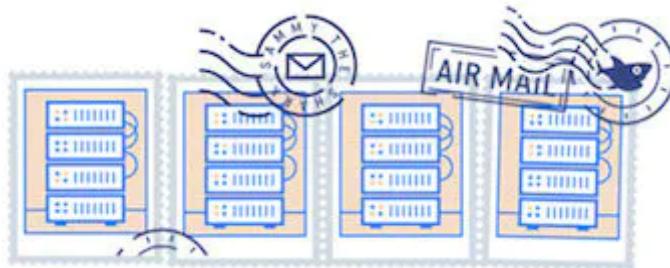
De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾



CONNECT WITH OTHER DEVELOPERS

Find a DigitalOcean Meetup
near you.



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a
Newsletter.

Featured on Community Kubernetes Course Learn Python 3 Machine Learning in Python
Getting started with Go Intro to Kubernetes

DigitalOcean Products Droplets Managed Databases Managed Kubernetes Spaces Object Storage
Marketplace

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the
cloud and scale up as you grow – whether you're
running one virtual machine or ten thousand.

Regístrate para recibir nuestro boletín.



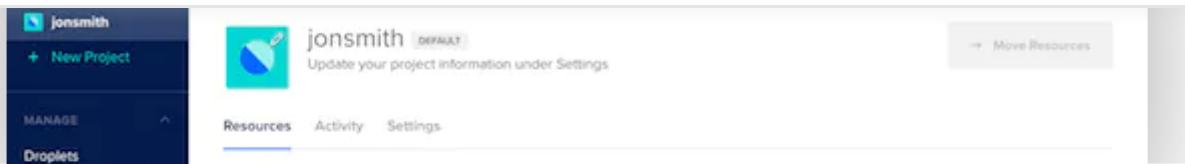
Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate

De contenedores a Kubernetes con Node.js >

Cómo migrar un flujo de trabajo d... ▾



© 2020 DigitalOcean, LLC. All rights reserved.

Company

About
Leadership
Blog
Careers
Partners
Referral Program
Press
Legal & Security

Products

Products Overview
Pricing
Droplets
Kubernetes
Managed Databases
Spaces
Marketplace
Load Balancers
Block Storage
Tools & Integrations
API
Documentation
Release Notes

Community

Tutorials
Q&A
Tools and Integrations
Tags
Product Ideas
Meetups
Write for DOnations
Droplets for Demos
Hatch Startup Program
Shop Swag
Research Program

Contacto

Obtener apoyo
¿Problemas para entrar?
Ventas
Reportar abuso
Estado del sistema

Regístrese para recibir nuestro boletín. X
Obtenga los últimos tutoriales sobre SysAdmin y temas de código abierto.

Ingrese su dirección de correo electrónico

Regístrate