EBook Gratis

APRENDIZAJE docker-compose

Free unaffiliated eBook created from
Stack Overflow contributors.

#docker-compose

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: docker-compose

It is an unofficial and free docker-compose ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official docker-compose.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con docker-compose

Observaciones

Compose es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. Con Compose, utiliza un archivo Compose para configurar los servicios de su aplicación. Luego, utilizando un solo comando, creará e iniciará todos los servicios desde su configuración. Para obtener más información sobre todas las funciones de Compose, consulte la lista de funciones.

Usar Compose es básicamente un proceso de tres pasos.

1. Defina el entorno de su aplicación con un `Dockerfile` para que pueda reproducirse en cualquier lugar.
2. Defina los servicios que conforman su aplicación en `docker-compose.yml` para que puedan ejecutarse juntos en un entorno aislado.
3. Por último, ejecute `docker-compose up` y Compose se iniciará y ejecutará toda la aplicación.

Examples

Instalación

Si está ejecutando Docker en OS X o Windows, Docker-compose debe incluirse en su instalación de Docker para Windows o Docker Toolbox.

En Linux, puede obtener los últimos binarios directamente desde la página de lanzamiento de GitHub: https://github.com/docker/compose/releases

Puede instalar la versión específica con los siguientes comandos:

```
curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Para más información consulte la página de documentación.

Crear una aplicación sencilla.

Este ejemplo proviene del documento oficial. Supongamos que tienes una aplicación python usando redis como backend. Después de escribir `Dockerfile` , crea un archivo `docker-compose.yml` como este:

```
version: '2'
services:
  web:
    build: .
    ports:
```

```
      - "5000:5000"
    volumes:
      - .:/code
    depends_on:
      - redis
  redis:
    image: redis
```

Luego, ejecute `docker-compose up` y la configuración completa de la aplicación incluye: aplicación de Python y redis.

- `version: '2'` es la versión de la sintaxis de archivos de la ventana acoplable
- `services:` es una sección que describe los servicios a ejecutar.
- `web:` y `redis:` son los nombres de los servicios para comenzar, sus contenidos describen cómo la ventana acoplable debe iniciar los contenedores para esos servicios
- `depends_on` implica una dependencia de web para redis y, por lo tanto, docker-compose primero inicia el contenedor `redis` y luego el contenedor web. Sin embargo, `docker-compose` no espera hasta que el contenedor `redis` esté listo antes de iniciar el contenedor `web` . Para lograr esto, debe usar un script que demore el inicio del servidor de aplicaciones o lo que sea hasta que el contenedor `redis` pueda realizar solicitudes.

También se puede agregar una sección de volúmenes y redes. El uso de la sección de volúmenes permite un volumen desconectado que puede vivir independientemente de la sección de servicios de composición de la ventana acoplable. La sección de redes tiene un resultado similar.

La sección redis de servicios debería ajustarse así:

```
redis:
  image: redis
  volumes:
    - redis-data:/code
  networks:
    -back-tier
```

A continuación, agregue las siguientes secciones a la parte inferior de la ventana acoplable compose el archivo de la versión 2.

```
volumes:
  # Named volume
  redis-data:
    driver: local
networks:
  back-tier:
    driver: bridge
```

`redis-data` proporciona una etiqueta accesible desde la sección de servicios. `driver:local` configura el volumen en el sistema de archivos local.

`back-tier` establece que la etiqueta de la sección de redes sea accesible en la sección de servicios como puente.

Ejecutar comando en el servicio de composición de ventana acoplable

`docker-compose run service-name command`

Si, por ejemplo, quisiera ejecutar `rake db:create` en su servicio `web` , usaría el siguiente comando:

`docker-compose run web rake db:create`

Instalar Docker Componer

1. Instalar el motor Docker .

   Si obtiene un error de `Permission denied` , ejecute `sudo -i` antes de los dos comandos a continuación, luego salga.

2. Tire de Docker Compose a `/usr/local/bin/docker-compose` .

```
curl -L https://github.com/docker/compose/releases/download/1.7.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

   Puede cambiar la versión `1.7.1` para que coincida con su versión deseada. Intente obtener la versión de https://github.com/docker/compose/releases

3. Aplicar permisos ejecutables al binario.

```
chmod +x /usr/local/bin/docker-compose
```

4. Probar la instalación.

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

   Versión de docker-compose esperada 1.7.1, compilación 0a9ab35

Docker componer hola mundo

Un `docker-compose.yml` muy básico tiene este aspecto:

```
version: '2'
services:
  hello_world:
    image: ubuntu
    command: [/bin/echo, 'Hello world']
```

Este archivo lo está haciendo para que exista un servicio `hello_world` , que se inicializa desde `ubuntu:latest` imagen y que, cuando se ejecuta, solo ejecuta `echo 'Hello world'`

Si está en el directorio de la `folder` (y contiene este archivo `docker-compose.yml`), puede hacer la función `docker-compose up` y debería ver

```
Creating folder_hello_world_1
```

```
Attaching to folder_hello_world_1
hello_world_1 | Hello world
folder_hello_world_1 exited with code 0
```

Esto creó el contenedor desde la imagen de ubuntu y ejecutó el comando que se especificó en el `docker-compose.yml`

`Docker-Compose` usa el nombre de la carpeta como el nombre del proyecto para prefijar contenedores y redes. Para establecer otro nombre de proyecto, puede llamar a `docker-compose --project-name NAME {up|down|...}` o colocar un archivo llamado `.env` junto a su `docker-compose.yml` y escribir `COMPOSE_PROJECT_NAME=name` in eso. Es mejor evitar los nombres largos de proyectos con guiones (-), ya que la ventana acoplable compone bahaves extraños con este tipo de nombres.

Nota: docker-compose le permite ejecutar múltiples contenedores de ventana acoplable en un solo host. Si desea ejecutar varios contenedores en más de un nodo, consulte una solución como swarm / kubernetes.

Ruby on Rails con docker-compose

Si desea utilizar la aplicación docker for rails y la base de datos, debe saber que todos los datos del contenedor de la ventana acoplable se destruirán (a menos que configure el contenedor específicamente para guardar datos). A veces, debe crear un contenedor de la ventana acoplable con una aplicación y adjúntela a un contenedor antiguo con una base de datos.

Como ejemplo de aplicación de rieles, utilicé una aplicación simple. Puedes crearlo desde el comando:

```
rails new compose-app --database=postgresql
```

Por supuesto, necesita instalar rieles, rubí, etc. de antemano.

Luego, cree Dockerfile en su proyecto y establezca estos datos en él:

```
FROM ruby:2.3.1
RUN apt-get update -qq && apt-get install -y build-essential libpq-dev nodejs
RUN mkdir /compose-app
WORKDIR /compose-app
ADD Gemfile /compose-app/Gemfile
ADD Gemfile.lock /compose-app/Gemfile.lock
RUN bundle install
ADD . /compose-app
```

Siguiente paso: crear docker-compose.yml con los datos:

```
version: '2'
services:
  db:
    image: postgres
  web:
    build: .
    command: bundle exec rails s -e development -p 80 -b '0.0.0.0'
```

```
    volumes:
      - .:/compose-app
    ports:
      - "80:80"
    depends_on:
      - db
```

Puede reemplazar el puerto 80 (-p 80) con otro.

La sección de desarrollo de la configuración de database.yml se debe cambiar a:

```
development: &default
  adapter: postgresql
  encoding: unicode
  database: postgres
  pool: 5
  username: postgres
  password:
  host: db
```

Ahora puedes construir imágenes desde el comando:

```
docker-compose build
```

(Ejecutar esto en el directorio del proyecto)

Y empieza todo desde

```
docker-compose up
```

Si todo se hace correctamente, podrá ver los registros de los rieles en la consola.

Cerrar la consola. Va a estar funcionando

Si desea eliminar solo el contenedor con la aplicación de rieles sin la base de datos, debe ejecutar en el directorio del proyecto:

```
docker-compose stop web
docker-compose build web
docker-compose up -d --no-deps web
```

Se creará y lanzará un nuevo contenedor con la aplicación de rieles.

Lea Empezando con docker-compose en línea: https://riptutorial.com/es/docker-compose/topic/1266/empezando-con-docker-compose

Capítulo 2: Ejemplo de Docker-compose multi-contenedor con red predeterminada

Observaciones

Por defecto, Compose configura una sola red para su aplicación. Cada contenedor para un servicio se une a la red predeterminada y es accesible tanto por otros contenedores en esa red como por un nombre de host idéntico al nombre del contenedor.

Los enlaces le permiten definir alias adicionales mediante los cuales se puede acceder a un servicio desde otro servicio. No están obligados a permitir que los servicios se comuniquen; de forma predeterminada, cualquier servicio puede llegar a cualquier otro servicio a su nombre.

https://docs.docker.com/compose/networking/

Examples

Cómo crear un entorno LAMP básico con redes por defecto

docker-compose.yml

```
version: '2'
services:
  php:
     image: phpmyadmin/phpmyadmin
     links:
       - mysql:db
     depends_on:
       - mysql

  mysql:
    image: k0st/alpine-mariadb
    volumes:
      - ./data/mysql:/var/lib/mysql
    environment:
      - MYSQL_DATABASE=mydb
      - MYSQL_USER=myuser
      - MYSQL_PASSWORD=mypass

  nginx:
    image: nginx:stable-alpine
    ports:
      - "81:80"
    volumes:
      - ./nginx/log:/var/log/nginx
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - php
```

nginx / nginx.conf

```
worker_processes  1;
events {
  worker_connections  1024;
}
http {
  sendfile  off;
  server {
    listen 80;

    location / {
      proxy_pass  http://php;
      proxy_set_header Host $host;
      proxy_redirect     off;
    }
  }
}
```

Tenga en cuenta que la configuración de nginx está simplificada, pero lo anterior debería funcionar para las pruebas; básicamente, todo lo que está haciendo es usar la aplicación php. Se asigna al puerto 81 para evitar conflictos en el host; ajústelos según sea necesario.

En cuanto a la vinculación, se puede ver que si se ejecuta: `docker-compose exec mysql ping -c2 nginx` hacer ping *desde* el contenedor de MySQL *al* contenedor nginx, usted tendrá éxito a pesar de que no *hay enlaces especificados entre estos contenedores.* Docker Compose mantendrá esos enlaces en la red predeterminada para usted.

Lea Ejemplo de Docker-compose multi-contenedor con red predeterminada en línea: https://riptutorial.com/es/docker-compose/topic/3226/ejemplo-de-docker-compose-multi-contenedor-con-red-predeterminada

Capítulo 3: Variables de entorno archivo externo

Introducción

Existen varias formas de incluir variables de entorno en la aplicación de la ventana acoplable. Aquí hay unos ejemplos:

Examples

A través de archivo externo

docker-composer.yml

```
  web:
    ...
    env_file:
     - ./filename
```

nombre del archivo

```
variable=value
```

dentro de la ventana acoplable se compone

```
app:
  ...
  environment:
    - var=value
```

Lea Variables de entorno archivo externo en línea: https://riptutorial.com/es/docker-compose/topic/10598/variables-de-entorno-archivo-externo

Creditos

S. No	Capítulos	Contributors
1	Empezando con docker-compose	cizixs, Community, Flamine, g3rv4, granthbr, ItayB, katopz, Kevin Wittek, Nadya Knyazeva, Nate Todd, Ohmen, Thomasleveil, Wolfgang
2	Ejemplo de Docker-compose multi-contenedor con red predeterminada	ldg
3	Variables de entorno archivo externo	Simon Tsang