Comience, Parte 3: Servicios

Tiempo estimado de lectura: 7 minutos

Requisitos previos

- Instale Docker versión 1.13 o superior (https://docs.docker.com/engine/installation/) .
- Obtener Docker Compose (https://docs.docker.com/compose/overview/) . En Docker para Mac (https://docs.docker.com/docker-for-mac/) y Docker para Windows (https://docs.docker.com/docker-for-windows/) es pre-instalado, por lo que es bueno para ir. En sistemas Linux tendrá que instalarlo directamente (https://github.com/docker/compose/releases) . En sistemas anteriores a Windows 10 *sin Hyper-V* , utilice Docker Toolbox (https://docs.docker.com/toolbox/overview/) .
- Lea la orientación en la Parte 1 (https://docs.docker.com/get-started/) .
- Aprenda a crear contenedores en la Parte 2 (https://docs.docker.com/get-started/part2/) .
- Asegúrese de haber publicado la `friendlyhello` imagen que creó empujándola a un registro (https://docs.docker.com/get-started/part2/#share-your-image) . Usaremos esa imagen compartida aquí.
- Asegúrese de que su imagen funciona como un contenedor implementado. Ejecutar este comando, ranurado en su información de `username` , `repo` y `tag` : `docker run -p 80:80 username/repo:tag` , entonces visita `http://localhost/` .

Introducción

En la parte 3, escalamos nuestra aplicación y habilitamos el balanceo de carga. Para ello, debemos subir un nivel en la jerarquía de una aplicación distribuida: el **servicio** .

- Apilar
- **Servicios** (usted está aquí)
- Contenedor (cubierto en la parte 2 (https://docs.docker.com/get-started/part2/))

Servicios comprensivos

Por ejemplo, si imaginas un sitio de intercambio de videos, probablemente incluya un servicio para almacenar datos de aplicaciones en una base de datos, un servicio para la transcodificación de vídeo en segundo plano después de una aplicación Usuario carga algo, un servicio para el front-end, y así sucesivamente.

Los servicios son realmente sólo "contenedores en producción". Un servicio sólo ejecuta una imagen, pero codifica la forma en que se ejecuta la imagen: qué puertos debe usar, cuántas réplicas del contenedor debe ejecutar para que el servicio tenga la capacidad que necesita y pronto. El escalado de un servicio cambia el número de instancias de contenedor que ejecutan esa pieza de software, asignando más recursos de computación al servicio en el proceso.

Afortunadamente, es muy fácil definir, ejecutar y escalar los servicios con la plataforma Docker - simplemente escriba un `docker-compose.yml` archivo.

Su primer `docker-compose.yml` archivo

Un `docker-compose.yml` archivo es un archivo YAML que define cómo se deben comportar los contenedores Docker en la producción.

`docker-compose.yml`

Guarde este archivo como `docker-compose.yml` si lo desea. Asegúrese de haber insertado la imagen (https://docs.docker.com/get-started/part2/#share-your-image) que creó en la Parte 2 (https://docs.docker.com/get-started/part2/) en un registro y actualizarla `.yml` reemplazando `username/repo:tag` con los detalles de la imagen.

```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: username/repository:tag
    deploy:
      replicas: 5
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "80:80"
    networks:
      - webnet
networks:
  webnet:
```

Este `docker-compose.yml` archivo le dice a Docker que haga lo siguiente:

- Tire de la imagen que subimos en el paso 2 (https://docs.docker.com/get-started/part2/) del registro.
- Ejecute cinco instancias de esa imagen como un servicio llamado `web` , limitando cada uno a utilizar, como máximo, el 10% de la CPU (a través de todos los núcleos) y 50 MB de RAM.
- Inmediatamente reinicie los contenedores si falla.
- Mapa del puerto 80 en el puerto 80 del host `web` .
- Instruya `web` los contenedores para compartir el puerto 80 a través de una red balanceada de carga llamada `webnet` . (Internamente, los propios contenedores se publicarán en `web` el puerto 80 en un puerto efímero).
- Defina la `webnet` red con la configuración predeterminada (que es una red de superposición con equilibrio de carga).

> **¿Te preguntas acerca de las versiones, nombres y comandos del archivo Compose?**
>
> Observe que establecemos el archivo de composición en `version: "3"` . Esto hace esencialmente el modo del enjambre (https://docs.docker.com/engine/swarm/) compatible. Podemos hacer uso de la clave deploy (https://docs.docker.com/compose/compose-file/#deploy) (sólo disponible en los formatos de archivo de Compose versión 3.x (https://docs.docker.com/compose/compose-file/) y superior) y sus subopciones para equilibrar la carga y optimizar el rendimiento de cada servicio (por ejemplo, `web`). Podemos ejecutar el archivo con el `docker stack deploy` comando (también sólo se admite en Componer archivos de la versión 3.xy más). Podrías usar `docker-compose up` para ejecutar archivos de la versión 3 con configuraciones que *no sean de enjambre* , pero nos estamos enfocando en un despliegue de pila ya que estamos construyendo un ejemplo de enjambre.
>
> Usted puede nombrar el archivo de Compose cualquier cosa que usted quiera que sea logicamente significativo para usted; `docker-compose.yml` Es simplemente un nombre estándar. Podríamos haber llamado tan fácilmente este archivo `docker-stack.yml` o algo más específico a nuestro proyecto.

Ejecuta tu nueva aplicación con equilibrio de carga

Antes de poder usar el `docker stack deploy` comando, ejecutaremos primero:

```
docker swarm init
```

> **Nota** : Entraremos en el significado de ese comando en la parte 4 (https://docs.docker.com/get-started/part4/) . Si no se ejecuta `docker swarm init` , obtendrá un error que "este nodo no es un administrador de enjambre".

Ahora vamos a ejecutarlo. Tienes que darle un nombre a tu aplicación. Aquí, se establece en `getstartedlab` :

```
docker stack deploy -c docker-compose.yml getstartedlab
```

Vea una lista de los cinco contenedores que acaba de lanzar:

```
docker stack ps getstartedlab
```

Puede ejecutar `curl http://localhost` varias veces en una fila, o ir a esa URL en su navegador y pulsa actualizar un par de veces. De cualquier manera, verá el cambio de ID de contenedor, demostrando el equilibrio de carga; Con cada solicitud, una de las cinco réplicas se elige, en un round-robin de moda, para responder.

> **Nota** : En esta etapa, los contenedores pueden tardar hasta 30 segundos en responder a las solicitudes HTTP. Esto no es indicativo de Docker o rendimiento de enjambre, sino más bien una dependencia de Redis no satisfecha que abordaremos más adelante en el tutorial.

Escala de la aplicación

Puede escalar la aplicación cambiando el `replicas` valor en `docker-compose.yml` , guardando el cambio y volviendo a ejecutar el `docker stack deploy` comando:

```
docker stack deploy -c docker-compose.yml getstartedlab
```

Docker realizará una actualización in situ, sin necesidad de romper la pila primero ni de matar a ningún contenedor.

Ahora vuelva a ejecutar el `docker stack ps` comando para ver las instancias implementadas reconfiguradas. Por ejemplo, si amplió las réplicas, habrá más contenedores en ejecución.

Derribar la aplicación y el enjambre

Tome la aplicación con `docker stack rm` :

```
docker stack rm getstartedlab
```

Esto elimina la aplicación, pero nuestro enjambre de un nodo sigue funcionando (como se muestra `docker node ls`). Derriba el enjambre con `docker swarm leave --force` .

Es tan fácil como para ponerse de pie y escalar su aplicación con Docker. Has dado un gran paso para aprender a ejecutar contenedores en producción. A continuación, aprenderá a ejecutar esta aplicación como un enjambre de buena fe en un grupo de máquinas Docker.

> **Nota** : Los archivos de este tipo se usan para definir aplicaciones con Docker y se pueden cargar a proveedores de la nube utilizando Docker Cloud (https://docs.docker.com/docker-cloud/) o cualquier proveedor de hardware o cloud que elija con Docker Enterprise Edition (https://www.docker.com/enterprise-edition) .

En "Parte 4" » (https://docs.docker.com/get-started/part4/)

Recapitulación y trucos (opcional)

Esta es una grabación terminal de lo que estaba cubierto en esta página (https://asciinema.org/a/b5gai4rnflh7r0kie01fx6lip) :

```
bash-3.2$ cat docker-compose.yml
version: "3"
services:
  web:
    image: johndmulhausen/get-started:part1
    deploy:
      replicas: 5
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
networks:
  webnet:
bash-3.2$ docker stack deploy -c docker-compose.yml getstartedlab
Creating network getstartedlab_webnet
Creating service getstartedlab_web
bash-3.2$
```

00:00

Powered by asciinema

Para recapitular, mientras que mecanografiar `docker run` es bastante simple, la puesta en práctica verdadera de un envase en la producción lo está funcionando como servicio. Los servicios codifican el comportamiento de un contenedor en un archivo de Compose, y este archivo se puede usar para escalar, limitar y reasignar nuestra aplicación. Los cambios en el servicio se pueden aplicar en su lugar, ya que se ejecuta, utilizando el mismo comando que puso en marcha el servicio: `docker stack deploy` .

Algunos comandos para explorar en esta etapa:

```
docker stack ls              # List all running applications on this Docker host
docker stack deploy -c <composefile> <appname>  # Run the specified Compose file
docker stack services <appname>       # List the services associated with an app
docker stack ps <appname>   # List the running containers associated with an app
docker stack rm <appname>                             # Tear down an application
```

Servicios (https://docs.docker.com/glossary/?term=services) , réplicas (https://docs.docker.com/glossary/?term=replicas) , escala (https://docs.docker.com/glossary/?term=scale) , puertos (https://docs.docker.com/glossary/?term=ports) , componer (https://docs.docker.com/glossary/?term=compose) , componer archivo (https://docs.docker.com/glossary/?term=compose%20file) , pila (https://docs.docker.com/glossary/?term=stack) , redes (https://docs.docker.com/glossary/?term=networking)

Califica esta página:
351 76

Documentación (/)

Aprender (https://www.docker.com/docker)

Blog (https://blog.docker.com)

Formación (https://training.docker.com/)

Apoyo (https://www.docker.com/docker-support-services)

Base de conocimientos (https://success.docker.com/kbase)

Recursos (https://www.docker.com/products/resources)

Comunidad (https://www.docker.com/docker-community)

Fuente abierta (https://www.docker.com/technologies/overview)

Eventos (https://www.docker.com/community/events)

Foros (https://forums.docker.com/)

Capitanes del muelle (https://www.docker.com/community/docker-captains)

Becas (https://www.docker.com/docker-community/scholarships)

Noticias de la comunidad (https://blog.docker.com/curated/)

Estado (http://status.docker.com/) Seguridad (https://www.docker.com/docker-security) Legal (https://www.docker.com/legal)

Contacto (https://www.docker.com/company/contact)

Copyright © 2017 Docker Inc. Todos los derechos reservados.