Comenzar, Parte 4: Enjambres

Tiempo estimado de lectura: 14 minutos

1: Orientación (https://docs.docker.com/get-started/) | 2: Contenedores (https://docs.docker.com/get-started/part2/)
3: Servicios (https://docs.docker.com/get-started/part3/) | 4: Enjambre (https://docs.docker.com/get-started/part4/)
5: Pilas (https://docs.docker.com/get-started/part5/) | 6: implementar la aplicación (https://docs.docker.com/get-started/part6/)

Requisitos previos

- Instale Docker versión 1.13 o superior (https://docs.docker.com/engine/installation/) .
- Obtenga Docker Compose (https://docs.docker.com/compose/overview/) como se describe en los requisitos previos de la Parte 3 (https://docs.docker.com/get-started/part3/#prerequisites) .
- Get Docker Machine (https://docs.docker.com/machine/overview/) , que está preinstalado con Docker para Mac (https://docs.docker.com/docker-for-mac/) y Docker para Windows (https://docs.docker.com/docker-for-windows/) , pero en sistemas Linux es necesario instalarlo directamente (https://docs.docker.com/machine/install-machine/#installing-machine-directly) . 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/` .
- Tener una copia de la `docker-compose.yml` de la parte 3 (https://docs.docker.com/get-started/part3/) práctico.

Introducción

En la parte 3 (https://docs.docker.com/get-started/part3/) , tomó una aplicación que escribió en la parte 2 (https://docs.docker.com/get-started/part2/) y definió cómo debería ejecutarse en la producción convirtiéndola en un servicio, ampliándola 5 veces en el proceso.

En la parte 4, implementa esta aplicación en un clúster, ejecutándola en varias máquinas. Las aplicaciones de múltiples contenedores y multi-máquinas se hacen posibles uniendo múltiples máquinas en un cluster "Dockerized" llamado **enjambre** .

Entendiendo los grupos de Swarm

Un enjambre es un grupo de máquinas que ejecutan Docker y se unen en un clúster. Después de que ha sucedido, continúa ejecutando los comandos de Docker que está acostumbrado, pero ahora son ejecutados en un clúster por un **administrador de enjambre** . Las máquinas en un enjambre pueden ser físicas o virtuales. Después de unirse a un enjambre, se les conoce como **nodos** .

Los administradores de enjambre pueden utilizar varias estrategias para ejecutar contenedores, como "nodo más vacío", que llena las máquinas menos utilizadas con contenedores. O "global", lo que garantiza que cada máquina obtenga exactamente una instancia del contenedor especificado. Enseña al gestor de enjambre que utilice estas estrategias en el archivo Compose, como el que ya has estado utilizando.

Los administradores de enjambre son las únicas máquinas en un enjambre que pueden ejecutar sus comandos, o autorizar otras máquinas para unirse al enjambre como **trabajadores** . Los trabajadores están ahí para proporcionar capacidad y no tienen autoridad para decirle a ninguna otra máquina lo que puede y no puede hacer.

Hasta ahora, ha estado utilizando Docker en un modo de host único en su máquina local. Pero Docker también se puede cambiar al **modo de enjambre** , y eso es lo que permite el uso de enjambres. Habilitar el modo de enjambre hace que la máquina actual sea un administrador de enjambre. A partir de ese momento, Docker ejecutará los comandos que ejecute en el enjambre que está administrando, en lugar de sólo en la máquina actual.

Configurar tu enjambre

Un enjambre se compone de nodos múltiples, que pueden ser máquinas físicas o virtuales. El concepto básico es bastante simple: ejecutar `docker swarm init` para habilitar el modo de enjambre y hacer que su máquina actual un administrador de enjambre, a continuación, ejecutar `docker swarm join` en otras máquinas para que se unan al enjambre como trabajadores. Elija una pestaña a continuación para ver cómo se reproduce en varios contextos. Utilizaremos máquinas virtuales para crear rápidamente un clúster de dos máquinas y convertirlo en un enjambre.

Crear un clúster

VM locales (Mac, Linux, Windows 7 y 8) (/get-started/part4/#local) | VM locales (Windows 10 / Hyper-V) (/get-started/part4/#localwin)

VMS EN SU MÁQUINA LOCAL (MAC, LINUX, WINDOWS 7 Y 8)

En primer lugar, necesitará un hipervisor que pueda crear VM, así que instale VirtualBox (https://www.virtualbox.org/wiki/Downloads) para el sistema operativo de su máquina.

> **Nota** : Si está en un sistema Windows con Hyper-V instalado, como Windows 10, no es necesario instalar VirtualBox y debe utilizar Hyper-V en su lugar. Consulte las instrucciones para los sistemas Hyper-V haciendo clic en la pestaña Hyper-V anterior.

Ahora, cree un par de máquinas virtuales usando `docker-machine` , usando el controlador VirtualBox:

```
$ docker-machine create --driver virtualbox myvm1
$ docker-machine create --driver virtualbox myvm2
```

Ahora tiene dos máquinas virtuales creadas, nombradas `myvm1` y `myvm2` (como se `docker-machine ls` muestra). El primero actuará como el gerente, que ejecuta `docker` órdenes y autentica a los trabajadores para unirse al enjambre, y el segundo será un trabajador.

Puede enviar comandos a sus máquinas virtuales usando `docker-machine ssh` . Instruye `myvm1` a convertirse en un administrador de enjambre con `docker swarm init` y verá la salida como este:

```
$ docker-machine ssh myvm1 "docker swarm init"
Swarm initialized: current node <node ID> is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join \
  --token <token> \
  <ip>:<port>
```

> **¿Tienes un error sobre la necesidad de usar `--advertise-addr` ?**
>
> Copie la dirección IP para `myvm1` ejecutar `docker-machine ls` , a continuación, ejecute el `docker swarm init` comando de nuevo, utilizando ese IP y especificando el puerto `2377` (el puerto para el enjambre se une) con `--advertise-addr` . Por ejemplo:
>
> ```
> docker-machine ssh myvm1 "docker swarm init --advertise-addr 192.168.99.100:2377"
> ```

Como puede ver, la respuesta a `docker swarm init` contiene un `docker swarm join` comando preconfigurado para que se ejecute en los nodos que desee agregar. Copie este comando y envíelo a `myvm2` via `docker-machine ssh` para tener `myvm2` unirse a su nuevo enjambre como un trabajador:

```
$ docker-machine ssh myvm2 "docker swarm join \
--token <token> \
<ip>:<port>"

This node joined a swarm as a worker.
```

Enhorabuena, usted ha creado su primer enjambre.

> **Nota** : También puede ejecutar `docker-machine ssh myvm2` sin ningún comando adjunto para abrir una sesión de terminal en esa VM. Escriba `exit` cuando esté listo para volver al indicador de shell de host. Puede ser más fácil pegar el comando de combinación de esa manera.

Use `ssh` para conectarse al (`docker-machine ssh myvm1`) y ejecute `docker node ls` para ver los nodos de este enjambre:

```
docker@myvm1:~$ docker node ls
ID                           HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS
brtu9urxwfd5j0zrmkubhpkbd     myvm2               Ready               Active
rihwohkh3ph38fhillhhb84sk *   myvm1               Ready               Active              Leader
```

Escriba `exit` para salir de esa máquina.

Como alternativa, envuelva comandos `docker-machine ssh` para no tener que iniciar sesión y salir directamente. Por ejemplo:

```
docker-machine ssh myvm1 "docker node ls"
```

Implementar la aplicación en un clúster

La parte dificil ya paso. Ahora solo repite el proceso que usó en la parte 3 (https://docs.docker.com/get-started/part3/) para desplegarlo en su nuevo enjambre. Sólo recuerde que sólo los gestores de enjambre como `myvm1` ejecutar comandos de Docker; Los trabajadores son sólo para la capacidad.

Copie el archivo `docker-compose.yml` que creó en la parte 3 en el directorio principal `myvm1` del administrador de enjambre (alias:) `~` usando el `docker-machine scp` comando:

```
docker-machine scp docker-compose.yml myvm1:~
```

Ahora `myvm1` usa sus poderes como gestor de enjambre para desplegar tu aplicación, enviando el mismo `docker stack deploy` comando que usaste en la parte 3 para `myvm1` usar `docker-machine ssh` :

```
docker-machine ssh myvm1 "docker stack deploy -c docker-compose.yml getstartedlab"
```

Y eso es todo, la aplicación se despliega en un clúster.

Envuelva todos los comandos que utilizó en la parte 3 en una llamada `docker-machine ssh` y funcionarán como esperaba. Sólo que esta vez, se verá que los contenedores se han distribuido entre ambos `myvm1` y `myvm2` .

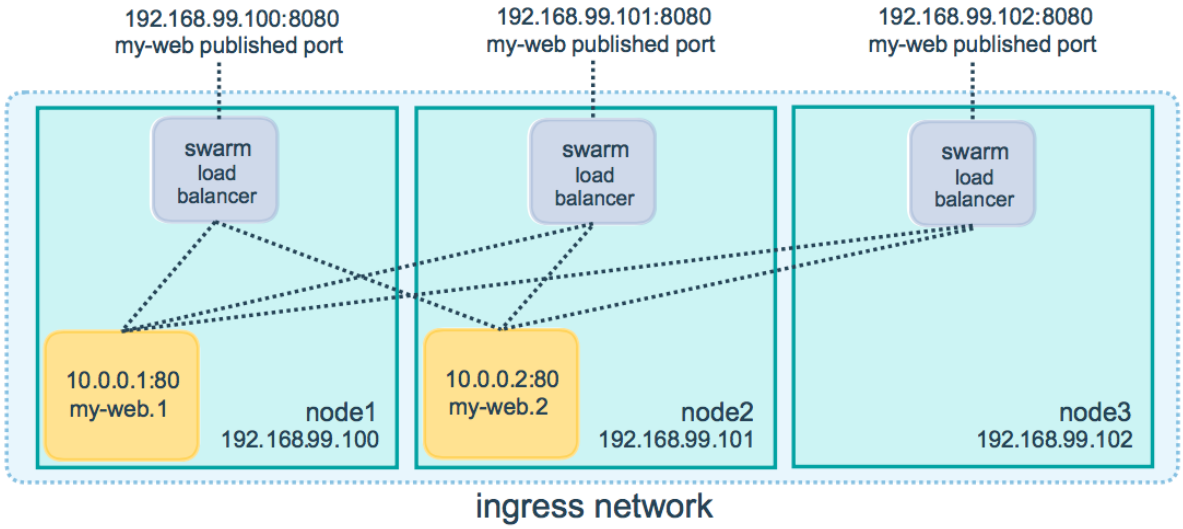
```
$ docker-machine ssh myvm1 "docker stack ps getstartedlab"

ID            NAME        IMAGE              NODE   DESIRED STATE
jq2g3qp8nzwx  test_web.1  username/repo:tag  myvm1  Running
88wgshobzoxl  test_web.2  username/repo:tag  myvm2  Running
vbb1qbkb0o2z  test_web.3  username/repo:tag  myvm2  Running
ghii74p9budx  test_web.4  username/repo:tag  myvm1  Running
0prmarhavs87  test_web.5  username/repo:tag  myvm2  Running
```

Acceso al clúster

Puede acceder a su aplicación desde la dirección IP de **cualquiera** `myvm1` o `myvm2` . La red que creó se comparte entre ellos y el equilibrio de carga. Ejecutar `docker-machine ls` para obtener las direcciones IP de sus máquinas virtuales y visitar cualquiera de ellos en un navegador, golpeando refrescar (o sólo `curl` ellos). Verá cinco identificadores de contenedores posibles que se completan de forma aleatoria, lo que demuestra el equilibrio de carga.

La razón por la que ambas direcciones IP funcionan es que los nodos de un enjambre participan en una **malla de enrutamiento de** entrada . Esto garantiza que un servicio desplegado en un determinado puerto dentro de su enjambre siempre tenga ese puerto reservado a sí mismo, sin importar qué nodo esté ejecutando realmente el contenedor. He aquí un diagrama de cómo una malla de enrutamiento para un servicio llamado `my-web` publicado en el puerto `8080` en un enjambre de tres nodos se vería:



¿Tiene problemas de conectividad?

Tenga en cuenta que para poder utilizar la red de entrada en el enjambre, es necesario tener los siguientes puertos abiertos entre los nodos de enjambre antes de habilitar el modo de enjambre:

- Puerto 7946 TCP / UDP para la detección de redes de contenedores.
- Puerto 4789 UDP para la red de entrada de contenedores.

Cómo iterar y escalar tu aplicación

Desde aquí puedes hacer todo lo que aprendiste en la parte 3.

Escale la aplicación cambiando el `docker-compose.yml` archivo.

Cambie el comportamiento de la aplicación editando el código.

En cualquier caso, simplemente `docker stack deploy` vuelva a ejecutar para implementar estos cambios.

Puedes unirte a cualquier máquina, física o virtual, a este enjambre, usando el mismo `docker swarm join` comando en el que usaste `myvm2` y la capacidad se agregará al clúster. Simplemente corre `docker stack deploy` después, y tu aplicación aprovechará los nuevos recursos.

Limpiar

Puedes derribar la pila con `docker stack rm` . Por ejemplo:

```
docker-machine ssh myvm1 "docker stack rm getstartedlab"
```

¿Mantener el enjambre o eliminarlo?

En algún momento más tarde, puedes eliminar este enjambre si quieres hacerlo con `docker-machine ssh myvm2 "docker swarm leave"` el trabajador y `docker-machine ssh myvm1 "docker swarm leave --force"` el administrador, pero *necesitarás este enjambre para la parte 5, así que manténgalo por ahora* .

Hacia la Parte 5 » (https://docs.docker.com/get-started/part5/)

Recapitulación y trucos (opcional)

Esta es una grabación terminal de lo que estaba cubierto en esta página (https://asciinema.org/a/113837) :

```
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: do
bash-3.2$ docker-machine ssh myvm1 "docker swarm init"
Error response from daemon: could not choose an IP address to advertise since this system has multiple
1) - specify one with --advertise-addr
exit status 1
bash-3.2$ docker-machine ls
NAME    ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER        ERRORS
myvm1   -        virtualbox   Running   tcp:/     168.99.104:2376           v17.04.0-ce
myvm2   -        virtualbox   Running   tcp:/        .99.105:2376           v17.04.0-ce
bash-3.2$ docker-machine ssh myvm1 "docker sw        --advertise-addr 192.168.99.104:2377"
Swarm initialized: current node (x500bs7lrwe     g6xq2ybd) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-197eaghawf5wqunblowkmgwjojv38ugtmscs943xrrz0jk6bpc-4uor1vdi4cleb5kqq2ri7s17g \
    192.168.99.104:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

bash-3.2$ docker-machine ssh myvm2
```

00:00

Powered by asciinema

En la parte 4 aprendiste lo que es un enjambre, cómo nodos en enjambres pueden ser gerentes o trabajadores, creó un enjambre y desplegó una aplicación en él. Viste que los comandos básicos de Docker no cambiaban desde la parte 3, solo tenían que ser dirigidos para ejecutarse en un maestro de enjambre. Usted también vio el poder de la red de Docker en acción, que mantuvo las solicitudes de equilibrio de carga a través de los contenedores, a pesar de que se ejecutan en diferentes máquinas. Por último, aprendió a iterar y escalar la aplicación en un clúster.

Estos son algunos comandos que te gustaría ejecutar para interactuar con tu enjambre un poco:

```
docker-machine create --driver virtualbox myvm1 # Create a VM (Mac, Win7, Linux)
docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" myvm1 # Win10
docker-machine env myvm1                # View basic information about your node
docker-machine ssh myvm1 "docker node ls"         # List the nodes in your swarm
docker-machine ssh myvm1 "docker node inspect <node ID>"        # Inspect a node
docker-machine ssh myvm1 "docker swarm join-token -q worker"   # View join token
docker-machine ssh myvm1   # Open an SSH session with the VM; type "exit" to end
docker-machine ssh myvm2 "docker swarm leave"  # Make the worker leave the swarm
docker-machine ssh myvm1 "docker swarm leave -f" # Make master leave, kill swarm
docker-machine start myvm1            # Start a VM that is currently not running
docker-machine stop $(docker-machine ls -q)               # Stop all running VMs
docker-machine rm $(docker-machine ls -q) # Delete all VMs and their disk images
docker-machine scp docker-compose.yml myvm1:~     # Copy file to node's home dir
docker-machine ssh myvm1 "docker stack deploy -c <file> <app>"   # Deploy an app
```

Enjambre (https://docs.docker.com/glossary/?term=swarm) , escala (https://docs.docker.com/glossary/?term=scale) , racimo (https://docs.docker.com/glossary/?term=cluster) , máquina (https://docs.docker.com/glossary/?term=machine) , vm (https://docs.docker.com/glossary/?term=vm) , gerente (https://docs.docker.com/glossary/?term=manager) , trabajador (https://docs.docker.com/glossary/?term=worker) , desplegar (https://docs.docker.com/glossary/?term=deploy) , ssh (https://docs.docker.com/glossary/?term=ssh) , orquestación (https://docs.docker.com/glossary/?term=orchestration)

Califica esta página:
158 47