«Cómo usar un servidor remoto Docker para agilizar vuestro flujo de trabajo» por Kamal Nasser

- **En castellano:** Este artículo es una traducción del inglés al castellano, publicado bajo licencia (en idioma inglés) «Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) » escrito por Kamal Nasser, publicado en línea por la empresa de alojamiento de máquinas virtuales DigitalOcean. El título original en idioma inglés es «***How to Use a Remote Docker Server to Speed Up Your Workflow***» y hemos creado una copia en Wayback Machine para prevenir un posible enlace roto a futuro.

Introducción

La creación de ficheros de imágenes y ficheros binarios en uso intensivo de CPU es un proceso muy lento, que consume mucho tiempo y que puede, a veces, convertir su ordenador portátil en un calefactor. "Subir" las imágenes de Docker con una conexión lenta a Internet también lleva mucho tiempo. Afortunadamente hay una solución fácil para estos problemas. Docker le permite descargar todas esas tareas a un servidor remoto para que su máquina local no tenga que hacer ese difícil trabajo.

Esta característica fue introducida en la versión de Docker número 18.09. Ofrece soporte para conectarse a un anfitrión Docker de forma remota a través de SSH. Requiere muy poca configuración en el lado del cliente, y solo necesita un servidor Docker normal y corriente, sin ninguna configuración especial, ejecutándose en una máquina remota. Antes de Docker 18.09, tenía se tenía que utilizar *Docker Machine* para crear un servidor Docker remoto y, a continuación, configurar el entorno Docker local para utilizarlo. Este nuevo método elimina esa complejidad adicional.

En este tutorial, creará una *Droplet* para alojar el servidor Docker remoto y configurar el comando **docker** en su ordenador local para usar dicho servidor virtual remoto.

Requisitos previos

Para realizar este tutorial usted necesitará:

- Una cuenta de usuario en DigitalOcean. Puede crear una cuenta si aún no tiene alguna.
- El software Docker instalado en su máquina local o servidor con entorno de programación. Si está trabajando con Ubuntu 18.04 siga los pasos 1 y 2 (en español) del tutorial «Cómo

> instalar y usar Docker en Ubuntu 18.04»; en cualquier otro caso siga (en inglés) la documentación oficial para instalarlo en otros sistemas operativos. asegúrese de agregar su usuario sin privilegios de *superusuario* al grupo **docker** como se indica en el paso número dos del tutorial en español.

Paso 1 — Creando el anfitrión Docker

Para comenzar, contrate un *droplet* con una cantidad decente de poder de procesamiento de cómputo. Los planes «CPU Optimizados» son perfectos para este propósito, pero los planes normales funcionan igual de bien. Si va a compilar programas con uso intensivo de recursos, los planes «CPU Optimizados» proporcionan núcleos de CPU dedicados que permiten compilaciones más rápidas. Por otra parte, los planes normales ofrecen una relación más equilibrada entre CPU y RAM.

Las imágenes Docker de "un solo clic" se encargan de toda la configuración por nosotros. Siga este enlace para crear un *droplet* con un plan «CPU Optimizado» (16 gigabytes RAM y 8 núcleos virtuales) con Docker instalado.

Otra alternativa es que usted puede utilizar **doctl** para crear el *droplet* desde su línea de comandos local. Para instalarlo siga las instrucciones del fichero al respecto publicado en GitHub.

El siguiente comando crea un nuevo *droplet* «CPU Optimizado» con 16GB/8vCPU en la región de Francia ("FRA1") basado en la imagen Docker "de un solo clic":

```
$ doctl compute droplet create docker-host \
$      --image docker-18-04 \
$      --region fra1 \
$      --size c-8 \
$      --wait \
$      --ssh-keys $(doctl compute ssh-key list --format ID --no-
header | sed 's/$/,/' | tr -d '\n' | sed 's/,$//')
```

El comando **doctl** utiliza el valor **ssh-keys** para especificar qué claves SSH debe aplicar a su nuevo *droplet*. Usamos una sublínea de comandos en la sexta línea para llamar al comando **doctl** y sus parámetros **compute ssh-key-list** para recuperar las claves SSH asociadas con vuestra cuenta de usuario en DigitalOcean, para luego analizar los resultados usando los comandos **sed** y **tr** y colocar los datos en el formato de lectura correcto. Este comando incluye todas las claves SSH almacenadas en su cuenta, pero puede reemplazar el subcomando resaltado por la huella

digital específica de cualquier clave que tenga en su cuenta.

Una vez la máquina virtual esté creada usted obtendrá la dirección IP, entre otros detalles:

```
Salida:
 ID           Name           Public IPv4        Private IPv4    Public IP
v6    Memory    VCPUs    Disk    Region    Image
        Status    Tags    Features    Volumes 148681562    docker-
host    vuestra_direccion_ip
                                     16384     8        100     fra1
Ubuntu Docker 5:18.09.6~3 on 18.04    active
```

Usted puede aprender más sobre el uso del comando **doctl** en el tutorial (en inglés) «Cómo usar **doctl**, el cliente de línea de comando oficial de DigitalOcean».

Cuando el *droplet* haya sido creado, usted estará listo para comenzar a usar el nuevo servidor Docker. Por razones de seguridad crearemos un usuario normal Linux en vez de utilizar el *superusuario* (**root**).

Primero conecte usted al *droplet* por medio de SSH como *superusuario:*

```
$ ssh root@vuestra_direccion_ip
```

una vez conectado, agregamos un nuevo usuario. Este comando agrega uno llamado **sammy**:

adduser sammy

Entonces agregue usted el usuario nuevo al grupo **docker** para darle permiso para correr los comandos en el anfitrión Docker.

```
# sudo usermod -aG docker sammy
```

Finalmente, salga del servidor remoto por medio del comando **exit**.

Paso 2 — Configurando Docker para usar el anfitrión remoto

Para utilizar el anfitrión remoto como su anfitrión Docker en lugar de su máquina local, configure la variable de entorno **DOCKER_HOST** para que apunte al anfitrión remoto. Esta variable indicará al cliente de línea de comando Docker que se conecte al servidor remoto.

```
$ export DOCKER_HOST=ssh://sammy@vuestra_direccion_ip
```

Ahora cualquier comando Docker que ordene se ejecutará en el *droplet*. Por ejemplo, si inicia un contenedor de servidor web y expone un puerto, se ejecutará en el *droplet* y será accesible a través del puerto que expuso en la dirección IP de la *droplet*.

Para verificar que estás accediendo al *droplet* como anfitrión del Docker, ejecute el comando **docker info**.

```
$ docker info
```

Usted verá que el nombre de anfitrión del *droplet* listado en el campo **Name**, algo similar a esto:

```
Salida:
...
Name: docker-anfitrion
...
```

Una cosa a tener en cuenta es que cuando usted ejecute un comando de **docker build**, el contexto de compilación (todos los archivos y carpetas accesibles desde el **doockerfile**) será enviado al anfitrión y luego se ejecutará el proceso de compilación. Dependiendo del tamaño del contexto de compilación y de la cantidad de archivos, puede llevar más tiempo en comparación con la compilación de la imagen en un equipo local. Una solución sería crear un nuevo directorio dedicado a la imagen del Docker y copiar o enlazar sólo los archivos que se utilizarán en la imagen para que no se "suban" ficheros innecesarios sin intención.

Conclusión

Ha creado un anfitrión Docker remoto y se ha conectado a él de manera local. La próxima vez que

la batería de tu portátil se esté agotando o necesite crear una imagen de Docker grande, utilice su flamante servidor Docker remoto en lugar de la máquina local.

También usted puede estar interesado en aprender «Cómo optimizar imágenes Docker para producción» (en idioma portugués o en idioma inglés) o «Cómo construir contenedores optimizados para Kubernetes» (en idioma inglés o en idioma portugués).

Autoría

Escrito por el señor Kamal Nasser y editado por el señor Brian Hogan.

Workflow» and we created a copy at Wayback Machine for prevent in future a broken link. This work is licensed under the mencioned license but, of course, in castilian language (AKA *spanish*): «Atribución-NoComercial-CompartirIgual 4.0 Internacional (CC BY-NC-SA 4.0) ».

-

- Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional.
- **En castellano:** Este artículo es una traducción del inglés al castellano, publicado bajo licencia (en idioma inglés) «Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) » escrito por Kamal Nasser, publicado en línea por la empresa de alojamiento de máquinas virtuales DigitalOcean. El título original en idioma inglés es «***How to Use a Remote Docker Server to Speed Up Your Workflow***» y hemos creado una copia en Wayback Machine para prevenir un posible enlace roto a futuro.

Recomendaciones del traductor

- Para saber rápida y exactamente a cuál servidor Docker estamos conectados (local o remoto) recomendamos este comando que nos retribuye única y exclusivamente el nombre del servidor Docker:

```
$ docker info | grep Name
```

- Aunque no está directamente relacionado con este artículo, de todos modos si aparece el mensaje «WARNING: No swap limit support» deberemos editar el fichero **/etc/default/grub** para luego modificar o agregar la línea que contenga **GRUB_CMDLINE_LINUX** y establecer su valor a "cgroup_enable=memory swapaccount=1", luego ejecutar **sudo**

update-grub y luego reiniciar el equipo, ya sea local o remoto (tomad precauciones acerca de reiniciar un *servidor en producción*).