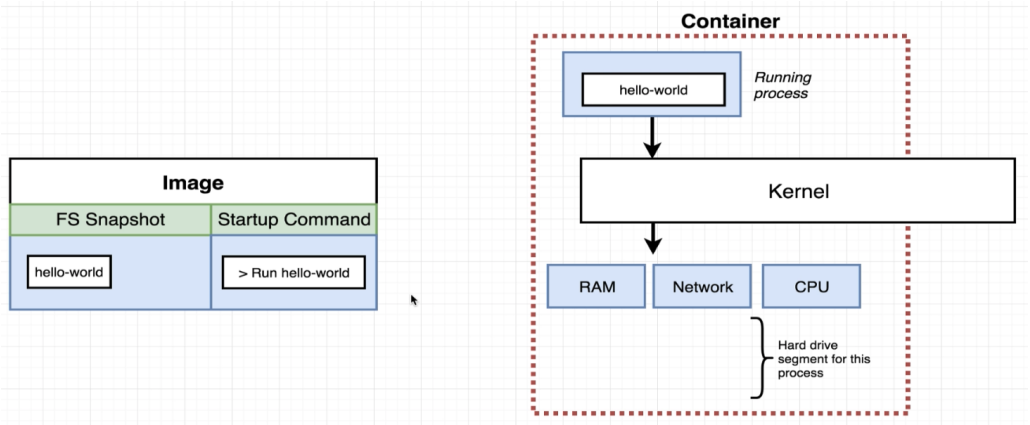
Docker Client – Ciclo de vida de los contenedores

En la última lección, ejecutamos el comando *docker ps* y vimos una variación de él, que era *docker ps --all.* Cuando ejecutamos *--all,* obtenemos una lista de todos los contenedores que hemos ejecutado en nuestro ordenador, lo cual es interesante porque plantea una serie de preguntas como *¿cuándo se cierra deja de ejecutar realmente un contenedor? ¿por qué se deja de ejecutar? ¿qué pasa cuando se deja de ejecutar?*

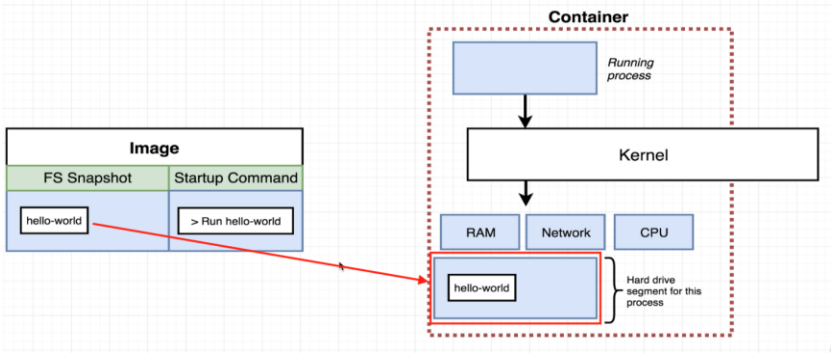
Cuando iniciamos un contenedor con *docker run*, realmente lo que ocurre son dos procesos distintos, la creación y la ejecución. Dicho esto, existen dos comandos adicionales que podemos utilizar para iniciar un nuevo contenedor:

Ejecutar el comando *docker run* es lo mismo que ejecutar otros dos comandos por separado, primero *docker create* y luego *docker start*. El comando *docker create* se utiliza para crear un contenedor en base a una imagen y, a continuación, *docker start* para iniciar el contenedor.

Pero ¿cuál es la diferencia entre crear un contenedor e iniciarlo? Regresemos a nuestro diagrama de antes para entender la diferencia entre estos dos comandos:

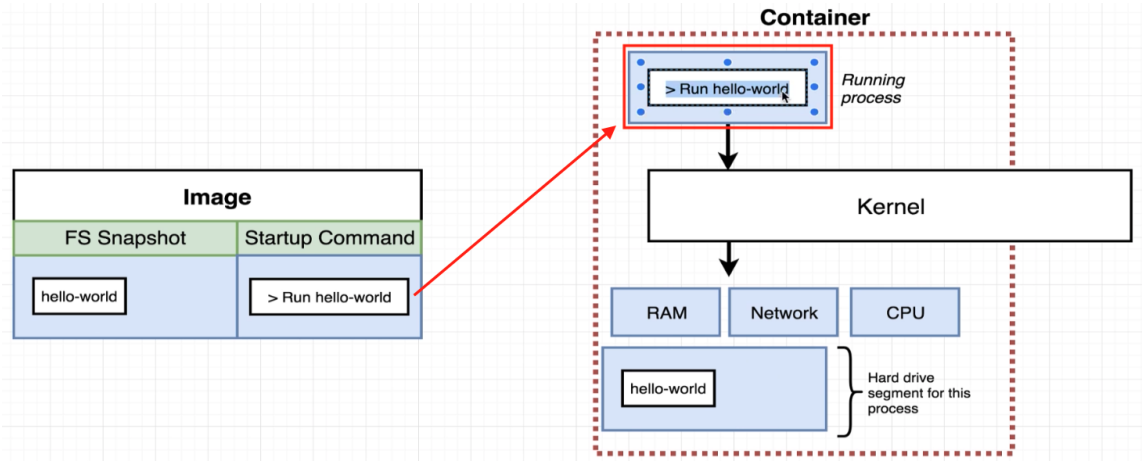


En nuestro diagrama, tenemos nuestra imagen *hello-world* con una snapshot del sistema de archivos, la cual contiene el comando de inicio "*run hello-world*". El proceso de creación de un contenedor es donde tomamos el sistema de archivos y lo preparamos para su uso en este nuevo contenedor:



Cuando creamos el contenedor, estamos hablando de preparar o configurar esta snapshot del sistema de archivos que se utilizará para crear el contenedor.

Para iniciar el contenedor, es cuando ejecutamos este comando de inicio *run hello-world:*



El comando que se ejecuta en el proceso de iniciar el contenedor puede ser también el que enviamos como override, por ejemplo, cuando ejecutamos el docker run busybox echo hi there, el comando que se ejecuta por separado en el segundo paso es echo hi there.

Así que la creación del contenedor trata sobre el sistema de archivos, iniciar el contenedor trata sobre ejecutar el comando de inicio.

Vamos a ejecutar primero *docker create hello-world*:

```
jorge@MacBook-Pro-de-Jorge ~ % docker create hello-world
ad3c9b36bd55291e93f2e0527c563dd0d285149c3fb823095faafd6095177590
jorge@MacBook-Pro-de-Jorge ~ %
```

Vemos que se imprime una larga cadena de caracteres. Esta cadena es el ID del contenedor que se acaba de crear. Ahora podemos ejecutar la imagen *hello-world* dentro de este contenedor ejecutando *docker start* seguido del argumento *–a* y luego agregamos el ID del contenedor que se acaba de imprimir:

```
jorge@MacBook-Pro-de-Jorge ~ % docker start -a ad3c9b36bd55291e93f2e0527c563dd0d285149c3fb823095faafd6095177590

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

jorge@MacBook-Pro-de-Jorge ~ %
```

Cuando ejecutamos este comando veremos que se imprimir el familiar mensaje que hemos visto antes.

Lo que hemos hecho con estos dos comandos, es que en primer lugar preparamos el contenedor con el sistema de archivos que necesitamos, después ejecutamos el comando *docker run* dentro de este contenedor que hemos preparado usando como referencia su ID.

Pero ¿qué hay del argumento –a que pasamos al domando docker run? Para mostrar lo que hace este argumento, vamos a ejecutar de nuevo el comando docker start pero sin el argumento –a:

```
jorge@MacBook-Pro-de-Jorge ~ % docker start ad3c9b36bd55291e93f2e0527c563dd0d285149c3fb823095faafd6095177590
ad3c9b36bd55291e93f2e0527c563dd0d285149c3fb823095faafd6095177590
jorge@MacBook-Pro-de-Jorge ~ %
```

Cuando ejecutamos docker start, sólo podemos ver el ID, esta vez no vemos el mensaje que imprime la imagen hello-world cuando se ejecuta el contenedor. El argumento –a en realidad busca las salidas del contenedor y las imprime en nuestra terminal. Así que el argumento –a significa específicamente “hacer un attach al contenedor, escuchar las salidas que se producen en el e imprimirlas en nuestra terminal”.