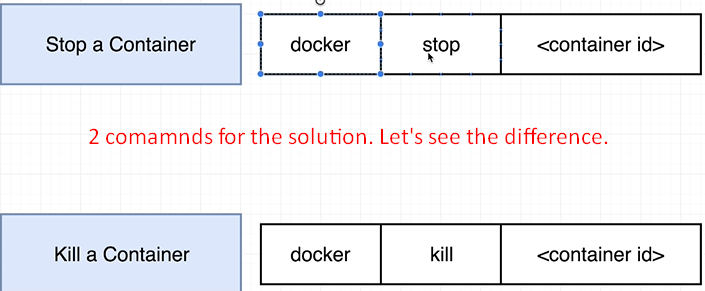
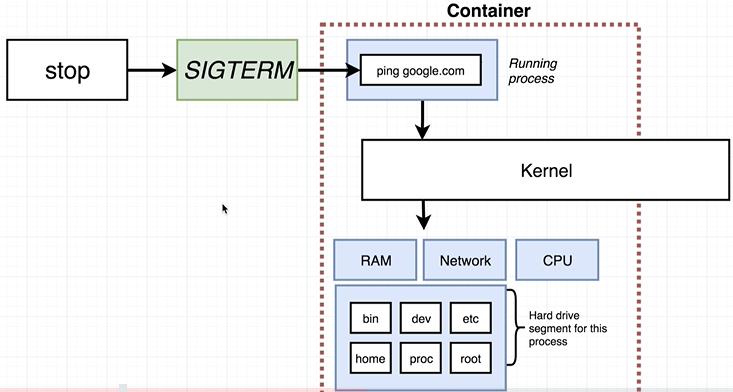
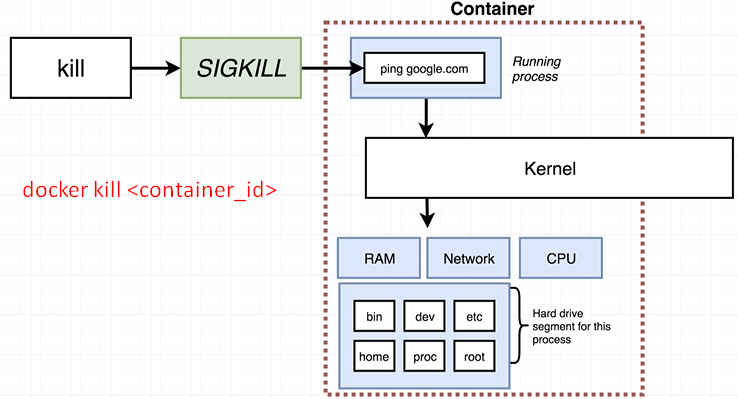
There is one oddity that I want to show you really quickly.

1. When give command 🡺 docker start ping google.com  
   The prompt would come back immediately. Now how to stop the container as it’s running to reach out to google server but you can’t use CTRL + C on command prompt to stop it or to kill it as it’s not running on the command prompt.
2. 
3. **Command** 🡺 docker stop <container\_id>  
   A hardward signal is sent to the primary process inside the container.  
   The signal is SIGTERM. This is the message sent to container’s primary process which stands for **Terminate Signal** telling the process essentially to shut down on its own time.  
   Used to stop a process inside of your container and shut the container down and you want to give that process inside there a little bit of time to shut itself down and do a little bit of cleanup. A lot of different programming languages have the ability for you to listen for these signals inside of your codebase. As soon as you get that signal you could attempt to do a little bit of cleanup or maybe save some file or email some message or something like that.   
   
4. **Command** 🡺 docker kill <container\_id>
   1. This command issues the signal **SIGKILL** to the primary process inside the container. SIGKILL essentially means you have to shut down right now and you don’t get to do any additional work.  
      
5. So ideally we always stop a container with the docker stop command in order to get the running process inside of it a little bit of time to shut itself down.  
   Otherwise, if it feels like the container has locked up and it’s not responding to the “docker stop command” then use “docker kill command”.  
   Oddity: One kind of little oddity or interesting thing about “docker stop”when you issue docker stop to a container, if the container doesn’t automatically stop in 10 seconds then docker is going to automatically fall back to issuing the “docker kill”command.
6. 