Clean up Exited Docker Containers With Docker PS, AWK, and Grep

As we start to mess around more with Docker, one of the things that I have to do regularly is to purge out the containers that are exited and unused. I will put the disclaimer here that this is a way to remove ALL exited containers. In other words, any container that is not running in detached mode will be destroyed. This is very handy for development systems where you test out lots of containers, and we will learn one of the most useful Docker commands out there – docker ps!

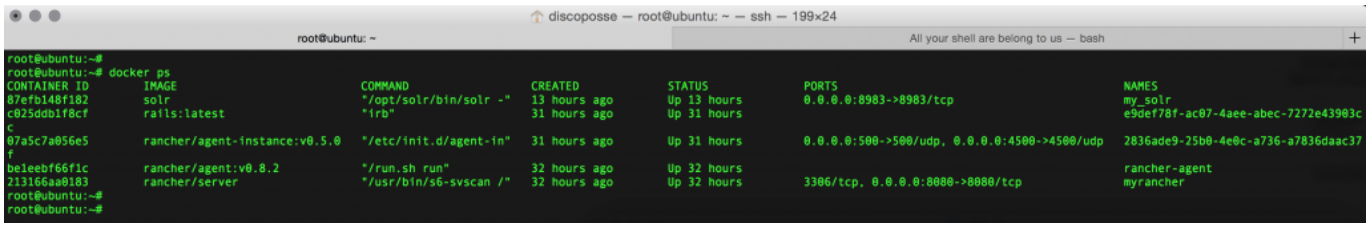
Docker PS and Pattern Matching the Awk-ward Way

I won't dive into the goodness of `awk`, but I do recommend that you get familiar with it as it is a powerful command line tool to do pattern matching and parsing of data. First, we will take a look at how to view containers that are running.

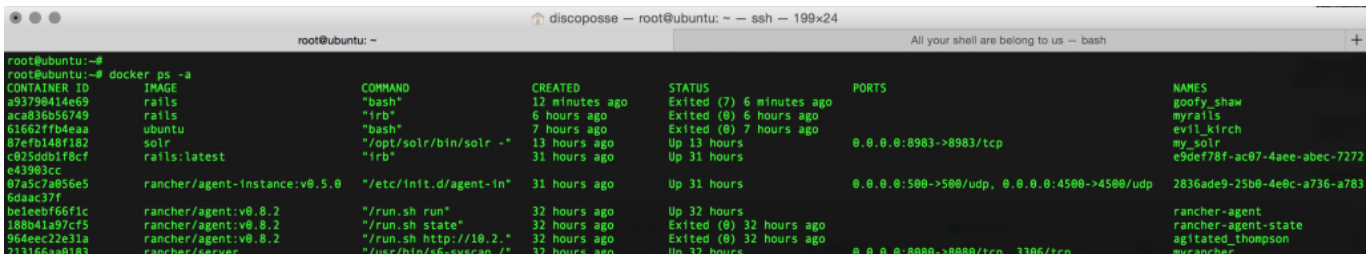
This is done with the `docker ps` command, which simply lists containers, and you can can pass some parameters to it. `docker ps` is one of the most useful commands you will use in docker.

Running the Docker PS Command

By default, `docker ps` shows you the active containers:

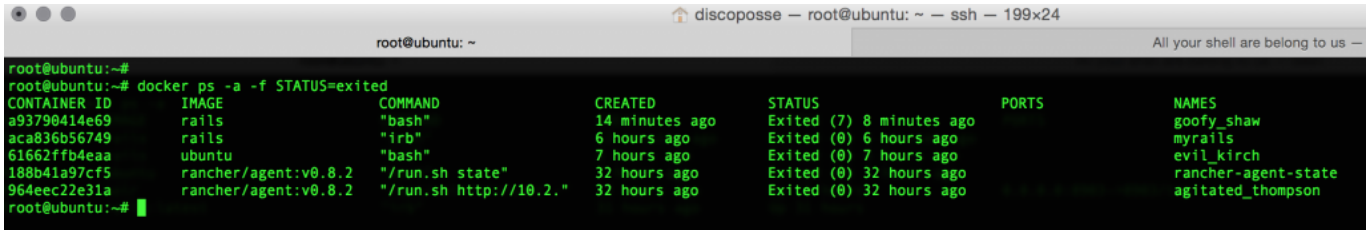


Since we know we have been mucking around with other containers that are not running, we need to run `docker ps -a` to view all containers:



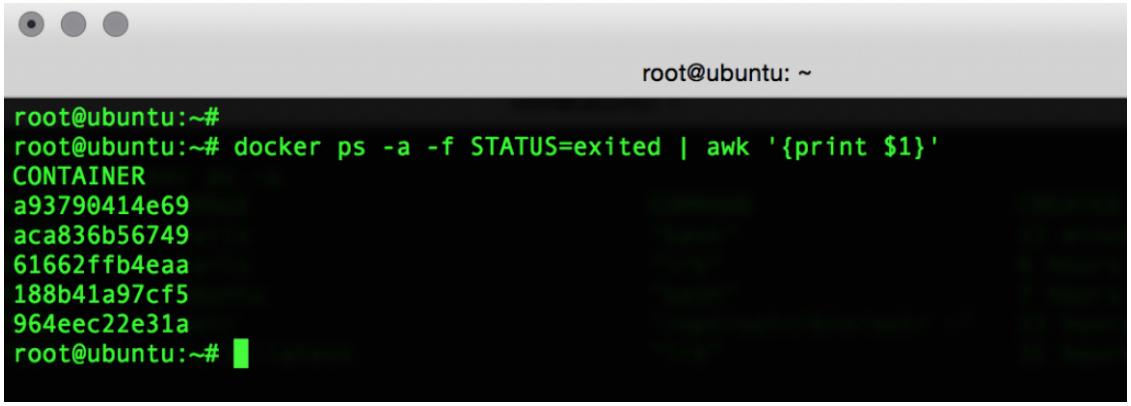
Adding Options to the Docker PS Command

We can also add a filter option to the command by using `docker ps -a -f STATUS=exited` to give os the list of containers which have been exited:



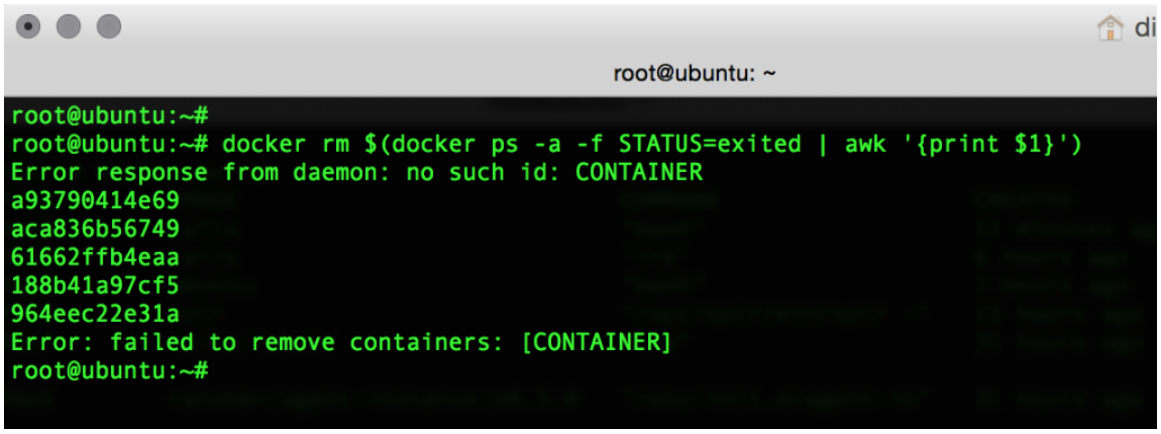
Combining awk and docker ps

Let's refine the results using `awk` to pull only what we need to pass to our `docker rm` command to remove the containers. In this case, we want the container ID which is the first column value. We will use the `docker ps -a -f STATUS=exited | awk '{print $1}'` command to do that:



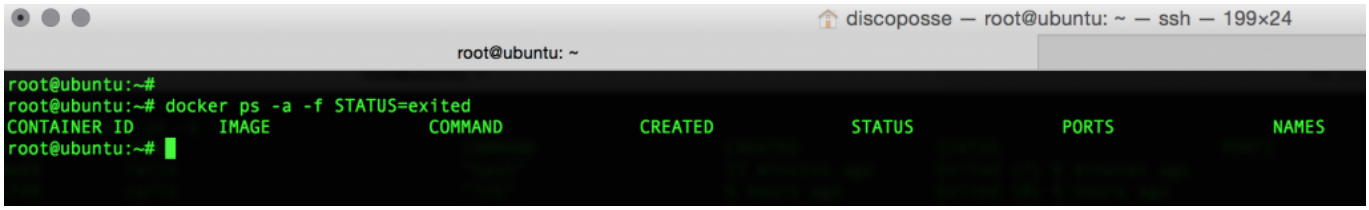
You can use a lower case for the filter parameter, but for some reason I like to use it the way it displays in the table. So, now that we have a command that can give us a list of the container IDs that we want to remove, there is only one step. Let's remove all of these using the `docker rm $(docker ps -a -f STATUS=exited | awk '{print $1}')` command.

This command does a docker remove for all values pulled from the docker ps command:

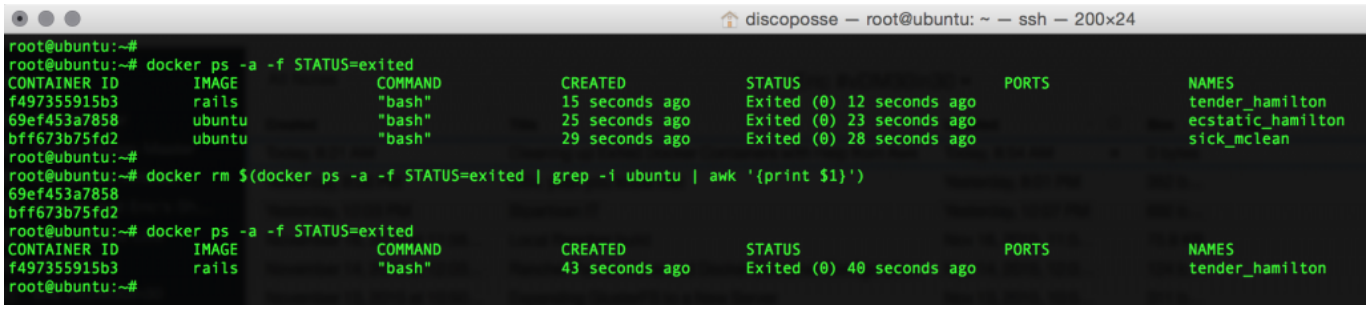


You can see that there will be an error thrown because we pulled the entire first column including the header which said "CONTAINER", so the `docker rm` command tripped on this one.

We can confirm the results by running our `docker ps -a` and you will see that now we only have our running containers in the list:



You could also do further limiting by adding a `grep` command. Perhaps we only want to remove exited containers from the Ubuntu image. That can be done using `docker rm $(docker ps -a -f STATUS=exited | grep -i ubuntu | awk '{print $1}')` as an example:



Between `awk` and `grep` you can do some pretty powerful stuff with a little practice. Hopefully this is a helpful tip for you.

More About docker ps

`docker ps` is one of the most useful and commonly used commands in Docker, since it lists containers. Remember, if you want to list ALL containers, including running and stopped, use the `docker ps -a` command.

Be sure to check out the Docker Documentation for the official and complete `docker ps` reference.