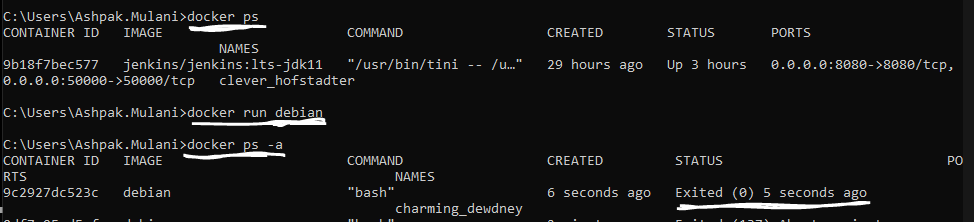
Docker Containers

# Basics of running container

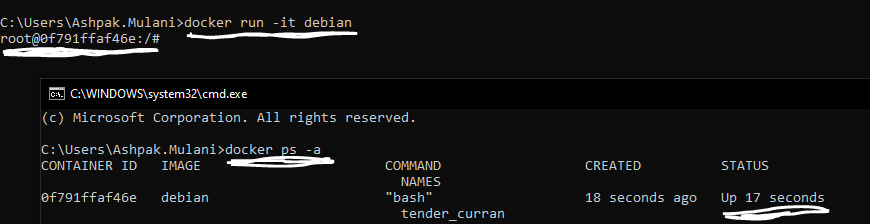
Default behaviour of docker container is to start-> perform given commands-> stop

Docker container stops immediately after finishing work by default until we ask them to keep running in background with some long running command.

Ex. in below screenshot after checking no Debian container is running, we ran the Debian container using docker run debian and when checked all running containers including recently stopped using docker ps -a command we can see Debian container was ran for a second and stopped/exited automatically.

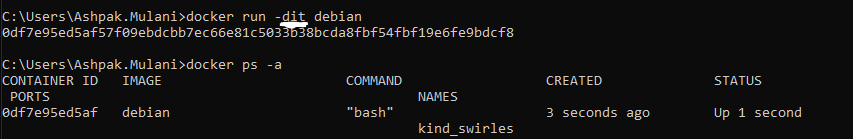


If we have to keep container running then we need to keep service running in container. In below query we are running Debian container in **i**nteractive **t**erminal mode using -it which opened the terminal inside container and kept it running for us in interactive mode. We can see in second command prompt thar container is running. If we type ‘exit’ in terminal from container then it will kill terminal and container will stop running.



In above scenarios if we don’t want container to keep running in ‘foreground’ giving us output from inside container on same window then we can choose to run it in ‘background’ using detached mode.

docker run -dit debian here -d option make sure container is running in detached mode in background and return us only container ID rather than output and details on what is running inside the container. We can connect/ssh to container separately if we need to interact



# Naming container

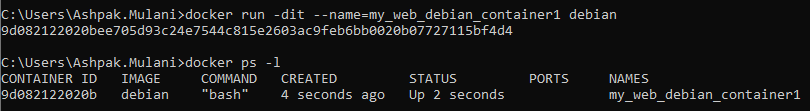
While running container we can name the running instance for easy management. To name a container we can use –name=<<name>> command tag.

docker run -dit --name=my\_web\_debian\_container1 debian

this will start new Debian container in detached interactive terminal mode with name my\_web\_debian\_container1

trick to check last latest running/stopped container is using -l option with ps instead of -a (which shows all container in all states)

docker ps -l as shown below gives our latest one running/stopped container



# Stop and delete container

Commands:

Stop: docker stop <<<container\_name>>>

Remove: docker rm <<<container\_name>>>

Before deleting container, we need to stop it. We can stop container using container ID or name.

In below sequence of commands, we are first running a container in dit mode.

Confirming last actioned container using ‘docker ps -l’ command

Try removing running container using docker rm which throws error

Stop container using ‘docker stop’ command

Remove container using ‘docker rm’ command

Confirm latest actioned container is not the one which we deleted since its gone now so there should not be any traces of it.



We can also use kill command if we get error while stopping container. Kill option forcefully kills container without graceful exit so it might corrupt container so it is not really recommended to use kill command..Docker kill <<container\_name>>

Docker kill my\_web\_debian\_container9

# Start restart container

Commands:

Start: docker start <<<container\_name>>>

Auto restart: docker run -dit –name=my\_web\_debian\_container9 –restart=always debian

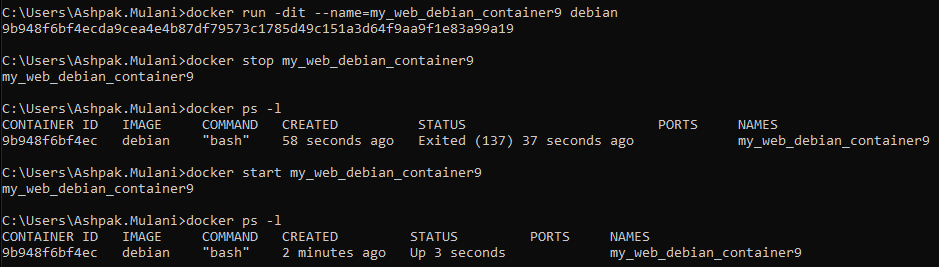
In below sequence we first ran the container using docker -dit command

stopped container using docker stop command

checked latest actioned container which returned stopped container in last command

started the container using ‘docker start <<container\_name>>’ command

checked last actioned container which returned started container in last command



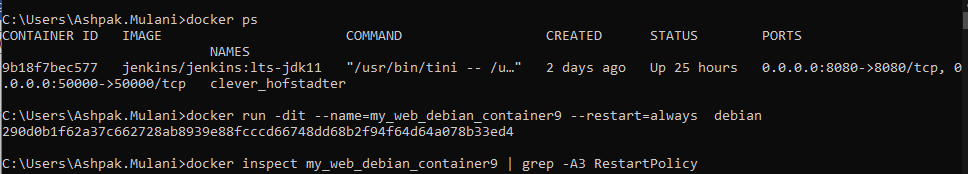
If our container is crashed or host machine restarts then all of our containers will stop, but there is a way to auto restart individual container when host machine comes up using –restart=always option

docker run -dit –name=my\_web\_debian\_container9 –restart=always Debian

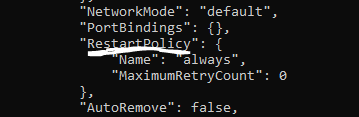
using –restart= always option applies policy to that container. We can check container details including policy using docker inspect <<docker\_name>> command

docker inspect my\_web\_debian\_container9 | grep -A3 RestartPolicy

in this command we ran the inspect command which gives a very big json output containing network config, disc storage details, policy details etc..but we want to only check restart policy hence we have used grep -A3 RestartPolicy command to filter output. -A3 option gives use 3 lines from matching line which contains ‘RestartPolicy’ word.



(GREP doesn’t work in windows so following screenshot is only for relevant part from big output)



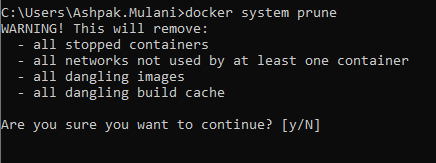
# Container clean-up

When container does its job and stops it stays there as container which is not running if we are automatically starting large number of containers and they are stopping immediately after their job is done the we might end up having big list of not running container consuming disk space.

Docker provide way to clean-up things by removing non running container using

Docker system prune

This command does clean-up for things mentioned in following screenshot



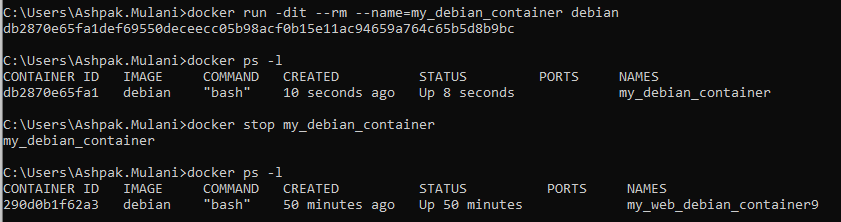
We can also use –rm option while running docker itself to remove container once its job is done.

In following sequence, we started new container using –rm option

Listed latest running container to make sure its running

Stopped container

Tried listing latest container which is completely different container now since our started container with –rm option is deleted automatically once execution is ended.



# Docker Log

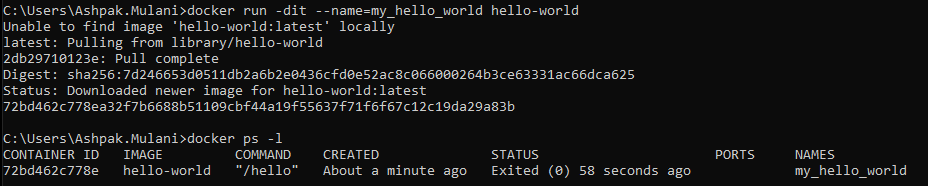
When container runs and something is printed in console then we can see it when we are running thing in container terminal interactively but when we run container in detached mode then we can’t see terminal all the time and messages logged in it until we connect to it.

Docker provides way to see details logged in terminal of a container running in background

Docker logs -t <<containerID> gives us details and logged info for troubleshooting

-t option prints timestamp as well when that info was logged

Let’s create a new container using hello-world image provide by docker and make sure it is running in detached mode in background as show in sequence below.

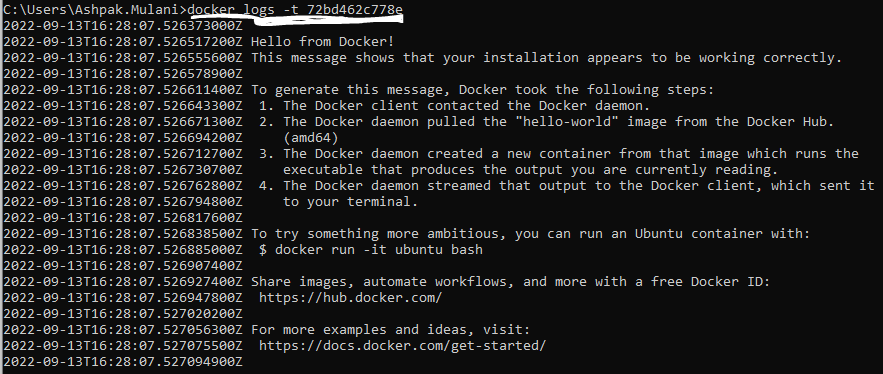


At this point we don’t know what is happening inside container and what all things are logged in terminal since it is running in background in detached mode.

To check the details from terminal we can run below command

docker log -t 72bd462c778e

which can give us results from container terminal with timestamp



To continue displaying console log from inside docker container rather than just printing current things and exiting, we can use -f to follow logs which keeps showing current logs and doesn’t exit the command.

Example: docker logs -tf 9b18f7bec577

Shows all log from console and stays there and keep showing any new logs generating until we press CTRL+C.