**1. Install Docker in Linux**

**=============================**

**# if the docker repo is added just run the below command to install docker**

**#yum install docker -y**

**2.Once you are done installing Docker, test your Docker installation by running the following:**

**===============================**

**# docker run hello-world**

**we are running the commands as root, check the user id using the command #id, it should show root as the user**

**Hello from Docker.**

**This message shows that your installation appears to be working correctly.**

**3. Playing with Busybox**

**===============================**

**Run a simple busybox container**

**# docker pull busybox**

**The pull command fetches the busybox image from the Docker registry and saves it to our system. You can use the docker images command to see a list of all images on your system.**

**#docker images**

**REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE**

**busybox latest c51f86c28340 4 weeks ago 1.109 MB**

**4. Docker Run (Running a container)**

**===============================**

**Great! Let's now run a Docker container based on this image. To do that we are going to use the docker run command.**

**# docker run busybox (there will be no output)**

**Wait, nothing happened! Is that a bug? Well, no. Behind the scenes, a lot of stuff happened. When you call run, the Docker client finds the image (busybox in this case), loads up the container and then runs a command in that container. When we run docker run busybox, we didn't provide a command, so the container booted up, ran an empty command and then exited. Let's try something more exciting.**

**# docker run busybox echo "hello from busybox"**

**hello from busybox**

**You will see the message "hello from busybox" appearing in the screen. In this case, the Docker client dutifully ran the echo command in our busybox container and then exited it. If you've noticed, all of that happened pretty quickly. Imagine booting up a virtual machine, running a command and then killing it. Now you know why they say containers are fast! Ok, now it's time to see the docker ps command. The docker ps command shows you all containers that are currently running.**

**# docker ps**

**CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES**

**Since no containers are running, we see a blank line. Let's try a more useful variant: docker ps -a**

**# docker ps -a**

**CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES**

**305297d7a235 busybox "uptime" 11 minutes ago Exited (0) 11 minutes ago distracted\_goldstine**

**ff0a5c3750b9 busybox "sh" 12 minutes ago Exited (0) 12 minutes ago**

**So what we see above is a list of all containers that we ran. Do notice that the STATUS column shows that these containers exited a few minutes ago.**

**You're probably wondering if there is a way to run more than just one command in a container. Let's try that now:**

**# docker run -it busybox /bin/bash ==> now you are inside the docker container. We can start executing commands here, i = interactive and t = terminal. So we are interactivly talking to the terminal of the container. Try with /bin/sh if the container does not have /bin/bash**

**#hostname ==> you will see the container id number**

**# ls**

**#pwd**

**#uptime**

**Running the run command with the -it flags attaches us to an interactive tty in the container. Now we can run as many commands in the container as we want. Take some time to run your favorite linux commands commands.**

**Docker run is a very important command. You should be comfortable and confident about it.**

**To find out more about run, use docker run --help to see a list of all flags it supports. As we proceed further, we'll see a few more variants of docker run.**

**# docker run --help**

**Before we move ahead though, let's quickly talk about deleting containers. We saw above that we can still see remnants of the container even after we've exited by running docker ps -a. Throughout this tutorial, you'll run docker run multiple times and leaving stray containers will eat up disk space. Hence, as a rule of thumb, you clean up containers once you are done with them. To do that, you can run the docker rm command. Just copy the container IDs from above and paste them alongside the command.**

**# docker rm 305297d7a235 ff0a5c3750b9**

**On deletion, you should see the IDs echoed back to you. If you have a bunch of containers to delete in one go, copy-pasting IDs can be tedious. In that case, you can simply run -**

**# docker rm $(docker ps -a -q -f status=exited)**

**This command deletes all containers that have a status of exited.**

**5. Deleting images**

**===========================**

**You can delete an image just like a container by using the command rmi - remove image**

**# docker rmi < image id >**