* Inspect container processe:
* Docker top container-name gives all proceses
* Docker stats container-name gives the statics of the container
* We can remove running container using docker rm -f container name
* We can remove running containers by stopping docker using systemctl stop docker command and go to /var/lib/docker/containers directory and delete the container then restart the docker then docker recognize that the container is no more in the containers directory.
* Docker rename old container-name new container name: to change the container name
* Docker events: to see the events
* Commad to see docker events using time stamp:Docker events --since ‘ 1h ‘
* Docker events --filter event=(mant filters =attach, container,image,label,type,volume,network,daemon)
* Can give many events like: docker events --filter event=attach --filter event=die --filter event=stop
* Docker images saving and loading: we save the commited containers will be saved as images
* Docker save centos:latest > centos.latest.tar (or) docker save -o centos.latest.tar centos:latest (or) docker save --output centos.latest.tar centos:latest
* How to pull the image from the saved tar file : docker load --input centos.latest.tar
* Docker load --input centos.latest.tar.qz
* Docker history image name : command to know the history of the particular image
* Docker history --quiet image name
* Docker tags: docker tag image id required image name
* PUSHING TO DOCKER HUB:

Docker login :command ask you for username and password of ur docker hub

(or) docker login --username=ofdockerhub [--email=kkkkkkkkk@gmail.com](mailto:--email=kkkkkkkkk@gmail.com)

Docker push image name: it push image to docker hub in the created repository

**Building a web farm for development and test:**

If we want start a service without rebooting or restart

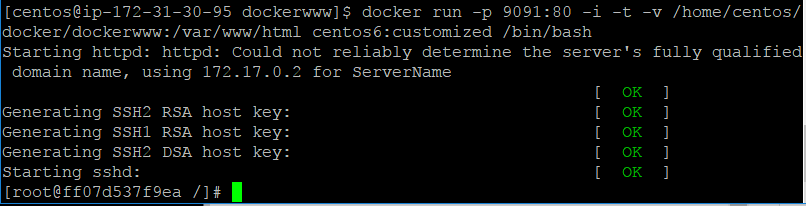
Just add services to the bashrc file

Vi .bashrc

Add service path ex: /sbin/service httpd start

Using the base image setup the container with required services and commit the container save the image with services required.

We configured the container with required services to development and test.

Run the container using customized image with the following command mount the volume with code or html code and give the port 

We can connect to the container using ssh connection: add a user in the container using command:adduser username and add password to the user : passwd username : then exit back to the terminal then hit : ssh username@ip address of the container.

Running two containers and adding nginx using as proxy server and load balancer adding in the nginx directory.

Run two containers:

Docker run -p 9091:80 -d -i -t -v /home/centos/docker/dockerwww:/var/www/html centos6:customized /bin/bash

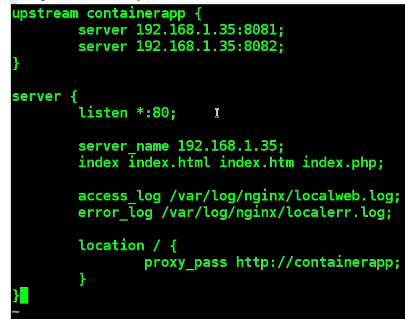
Two containers are created with same html page

To add nginx as load balancer to the containers

Install nginx on the local machine

Go to vi /etc/nginx/sites-available/default.conf

Add the below script in the default.conf and customize according to your ip’s.



Integrating custom network in your docker container:

-for ubuntu

Service docker.io stop for centos service docker stop

* Ip link add br10 (bridge name) type bridge
* Ip addr add 10.10.100.1/24 (custome ip ) dev br10
* Ip link set br10 up
* To start docker with created custome ip bridge
* Docker -d -b br10 &
* Docker run -I -t centos:latest /bin/bash gets ip address of the custome ip bridge
* To set this created bridge as the default for docker
* Go to /etc/network : vim interfaces
* Add the below script
* 