**DOCKER**

1. Installing Docker.io

\* Follow Steps on https://www.simplilearn.com/tutorials/docker-tutorial/how-to-install-docker-on-ubuntu

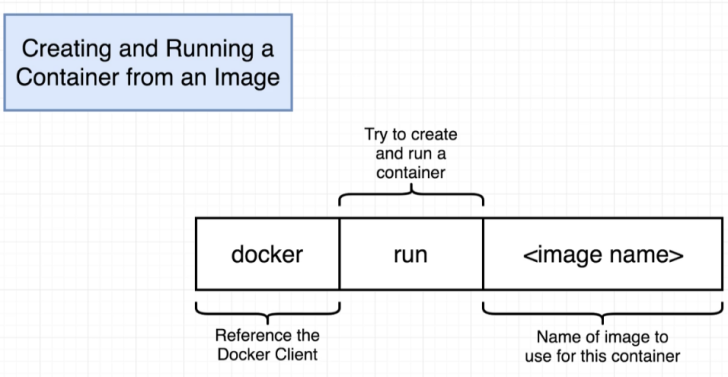
2. Defining Docker

\* It is an Container achitecture uses FS Snapshot (Image) and OverRide Commands ( backend it used linux arch environment)

\* Containers **-->** Its an namespace (space) and control-group (Ram,hdd,cpu **->** RESOURCES) provided on hdd for easy utility.

\* Every **Image** will have its own container and will not share storage to each other

\* How to run an image



**-->** sudo docker run **image\_Name**

\* **How to create and start container manually (run = create+start)**

**-->** sudo docker create **imageName** => will return id of container

**-->** sudo docker start **container\_id** -a => -a will help to provide logs of container after running

\* **What if we skiped -a**

**-->** sudo docker logs container\_id

\* **How to check containers process status (ps) only running**

**-->** sudo docker ps

\* **How to check containers all process status running or stopped**

**-->** sudo docker ps --all

\* **How to stop running container**

**-->** sudo docker stop container\_id => after 10sec if container don't stop on (SIGTERM) Signal then will receive (SIGKILL) Signal to kill Container Process

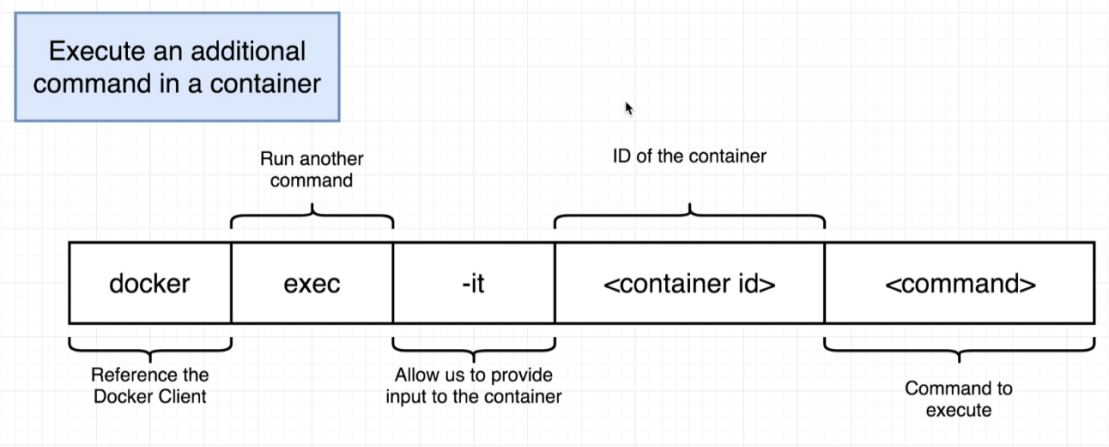
\* **How kill container running manually** => It will receive (SIGKILL) Signal

**-->** sudo kill container\_id

\* **How to remove all containers (hdd,networkServices,cpuUsage,RamUsage) at Once**

**-->** sudo docker system prune

1. EXECUTE MULTI COMMANDS



\* exec with

-i -t or -it is used to provide stdin and stdout of the container

\* Example with redis server

-**->** sudo docker create redis

return **redis\_container\_id**

-**->** sudo docker start **redis\_container\_id** -a

-**->** sudo docker exec -it  **redis\_container\_id** redis-cli

or

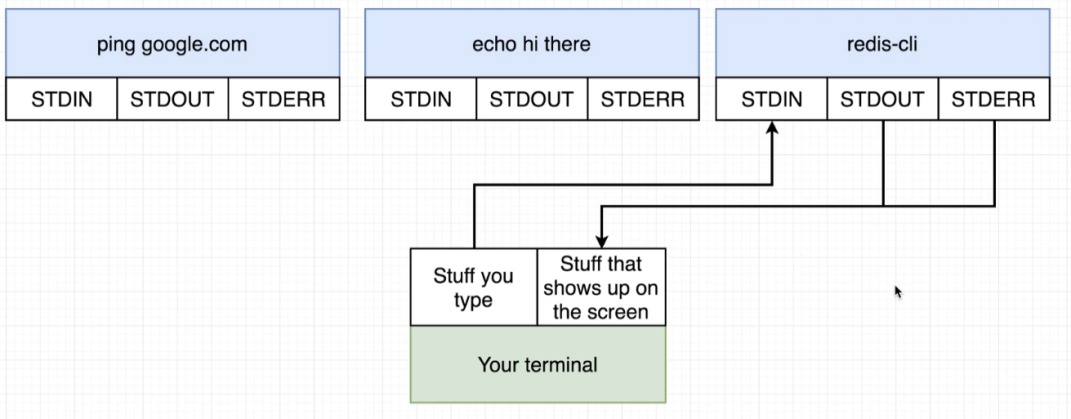
-**->** sudo docker exec -it **redis\_container\_id** sh => sh is shell used to give commands to cmd of linux

-**->** redis-cli

\* How to use commands in container if we use **run**

**-->** sudo docker run -it **redis** redis-cli

\* Why **-it** or **-i -t** ?



**-i** stands for permission to input command using STDIN of container environement

**-t** stands for permission to output using STDOUT from the container environment

**CUSTOM IMAGE**

\* How we can create our own **CUSTOM IMAGE** ?

To create an image we need an OS from **hub.docker.com** like **->** alpine**,** node : alpine

OS must be selected very precisly

Every OS will contains its own set of snapshots of Programs.

**Create a file with name Dockerfile with no extensions**

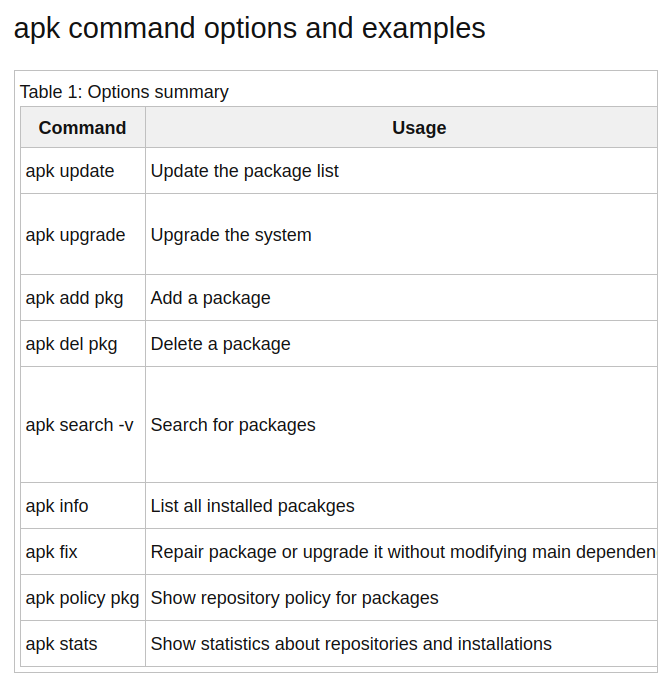
1. **FROM** command for OS Selection from hub

--> **FROM** alpine

**alpine** --> redis, mysql-client etc

**node** --> full sized OS for supported node dependencies

**node : alpine** --> contain compact version of node includes node package manager (npm)



1. **RUN** command to install required programs

--> **RUN** apk add --update redis

APK --> **Alpine Linux Package Keeper**

Detailed Description apk -> [Click Here](https://www.cyberciti.biz/faq/10-alpine-linux-apk-command-examples/)

1. **COPY** for copy data from **parent OS FileSystem** to **child OS FileSystem**

--> **COPY** ./ ./

here 1st ./ means current directory of all files of parent OS

here 2nd ./ means current directory of child OS

for Specific file

--> **COPY** ./package.json ./

This is very important step to follow it avoids folder conflict

with root directory and the customly installed folders directory

of same names

1. To Change Current Working Directory of Container use

--> **WORKDIR** ./usr/app

here **./usr/app** will navigate the current working directory of Container

to folder **app** located in **usr** from root directory

1. **CMD** is used to run commands on command promt of the Container

--> **CMD** [“redis-cli”]

Remember in all Cases

1. if space is required between Commands use **string litrals** anduse **comma** forsperation

eg --> **CMD** [“npm”,”start”]

1. Every addition of command line to the top will force docker container for reinstall all packages, so be carefull and try to add new line of command at the end of the last line before **CMD Command** if necessory

**BUILDING IMAGE**

use files in Microservices\_PracticeData > 2\_UnderStandingDocker > createNodeProject

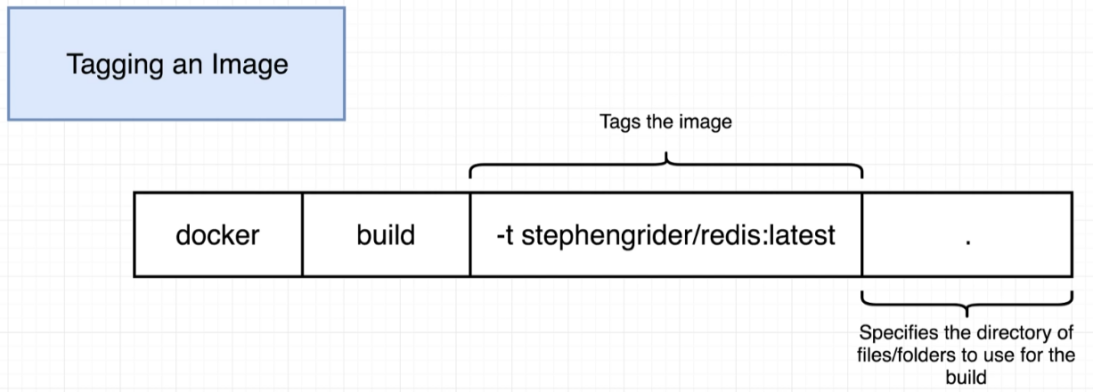
as an refrence

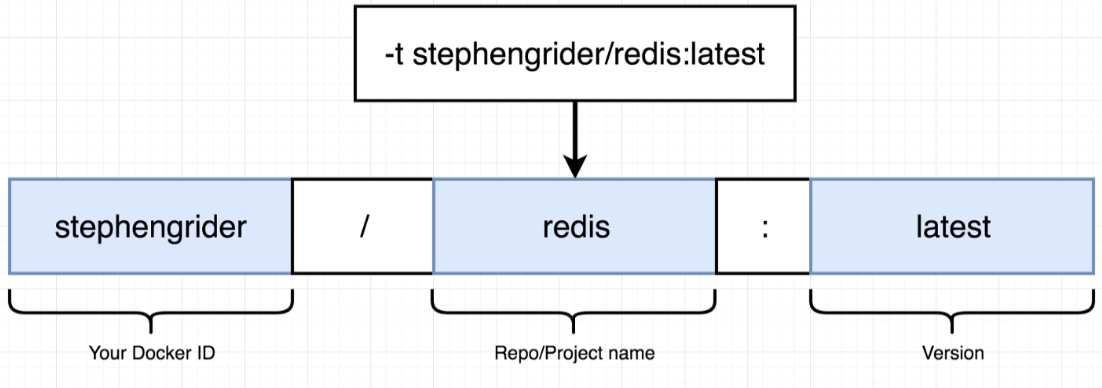
Without Tag

--> sudo docker build .

return container\_id

With TAG -t





--> sudo docker build -t davimehra/nodeserver:latest **.**

build is an command for specifing build

-t is an option represting the tag and image name

tag is latest

imagename or location is -> davimehra/nodeserver

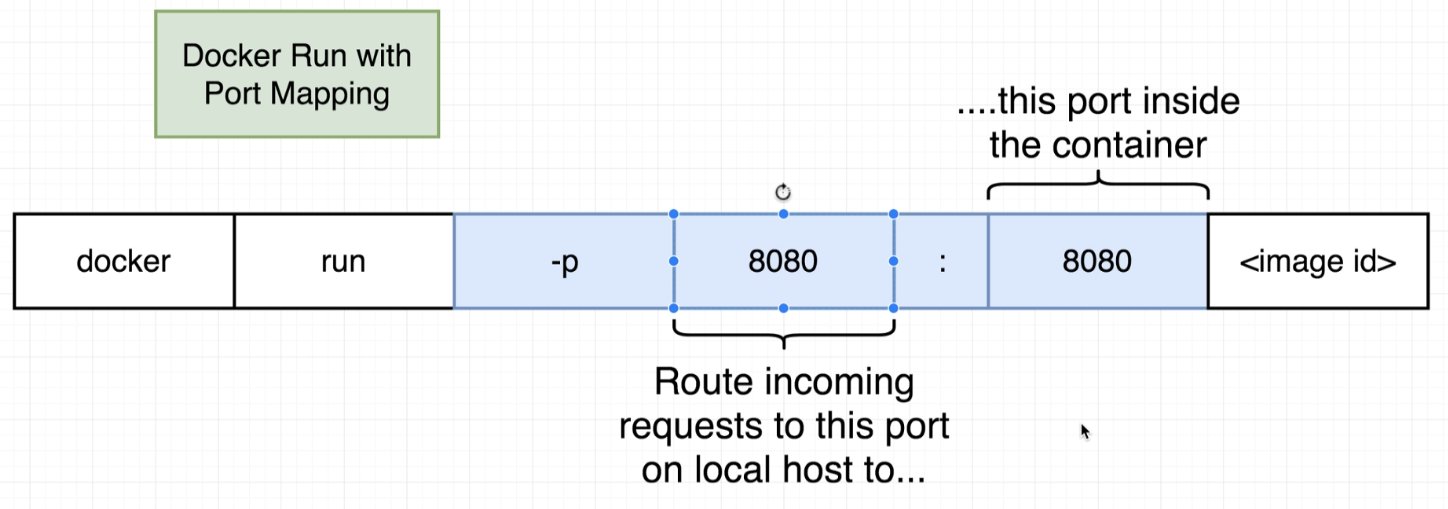
. (dot ) it is an build context or the location for build which is current directory (VERY IMPORTANT)

**\* How to run container with doc tag?**

**-->** sudo docker run davimehra/container\_Name

**\* Does container port and main OS ports are same ?**

--> NO , Container is totaly isolated from the outer world so we need to do PORT FORWARDING so that if any outer world http request comes to the port we can redirect it to container own isolated port



**REMEMBER**

**PORT FORWADING is done while run an Container**

--> sudo docker run -p 4000 : 5000 davimehra/nodeserver

Here

**-p**  [option] stands to port forwarding

**4000**  [PORT] is an port of MAIN OS where World will try to reach out

**5000** [PORT] is an port where to FORWARD the Request in an isolated container