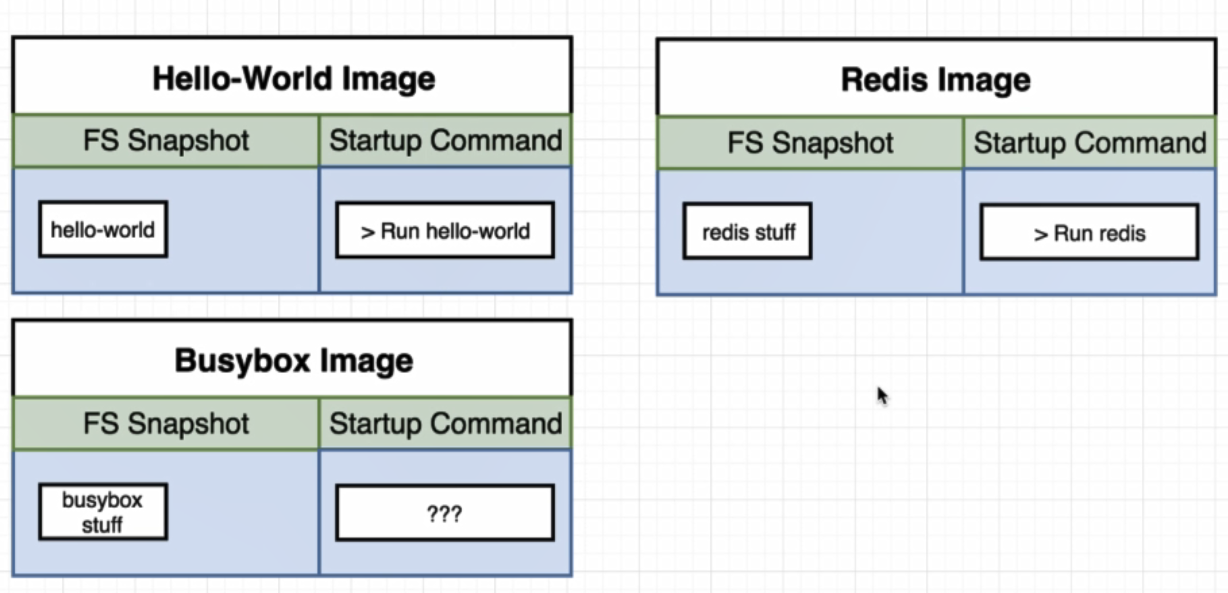
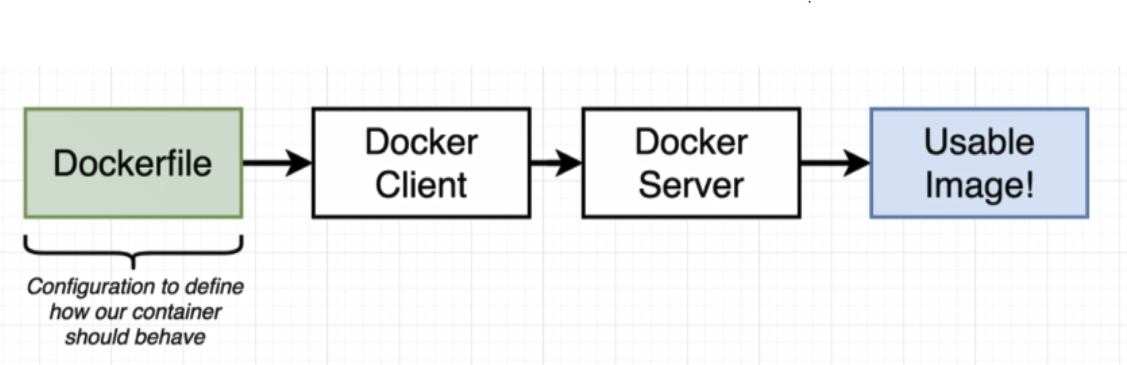
**Building Custom Images Through Docker Server**

In previous section we use images that created by other engineers, we download them into local machine and create container out of them.



Make out own images:

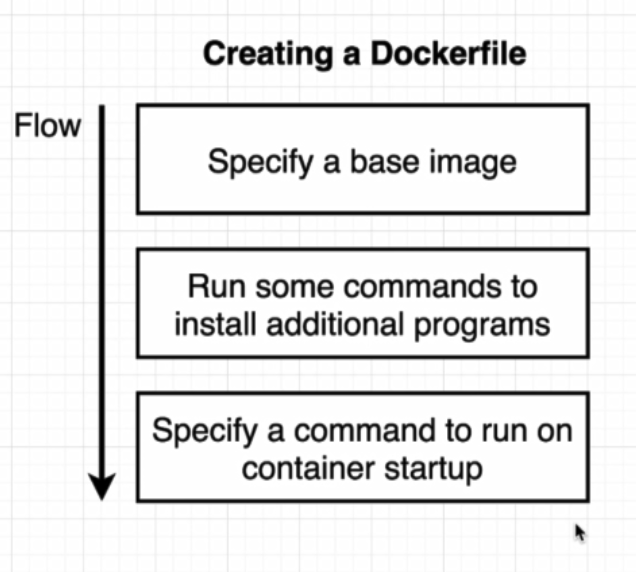


DockerFile: A plain text file that is going to have to couple of lines off configurations placed inside of it. ( define how our container behaves).

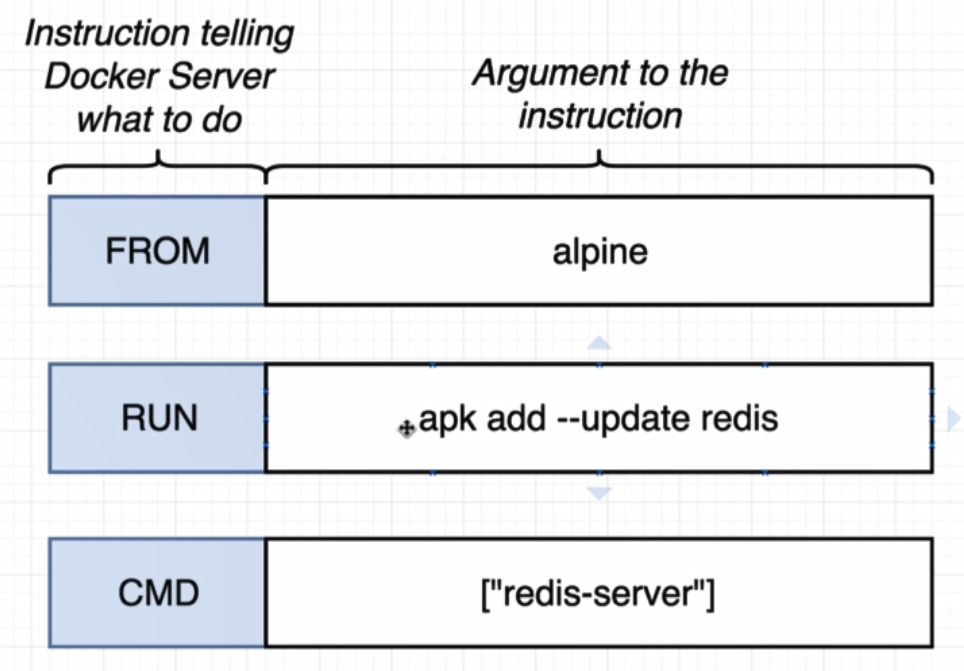
Docker Client: After create dockerfile we pass it off to docker client.

Docker Server: Docker client provide file to docker server, then docker server create usable image that can then be used to start up a new container.

**Create a DockerFile**



* mkdir redis-image
* cd redis-image
* code .
* create Dockerfile
* # use an existing docker image as base
* From alpine
* # download and install dependency
* RUN apk add --update redis
* # Tell the image what to do when it starts as a container
* CMD ["redis-server"]
* Docker build .
  + After docker build it return Successfully built b51f0902ef3f message
* Docker run b51f0902ef3f



FROM:

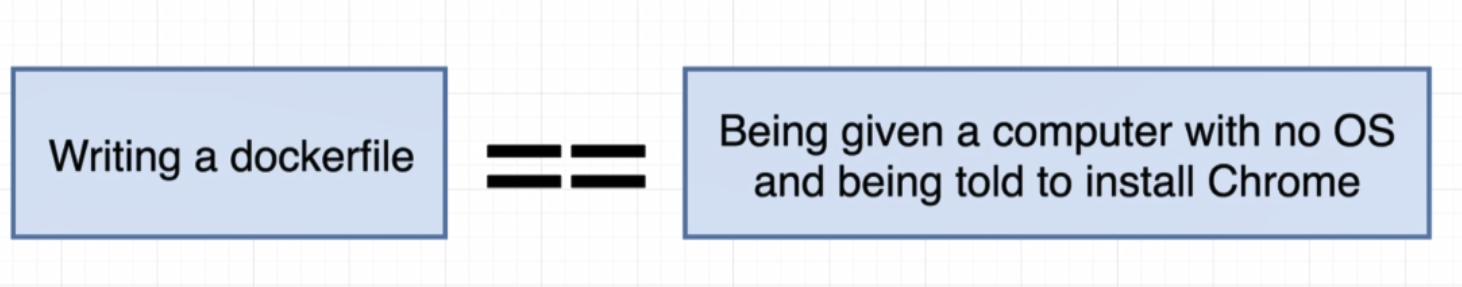
* Specify the docker or image that we want to use as a base. (base image as alpine)
* It is like an installing initial operating system
* It is like initial set of program that we can use to further customize our image

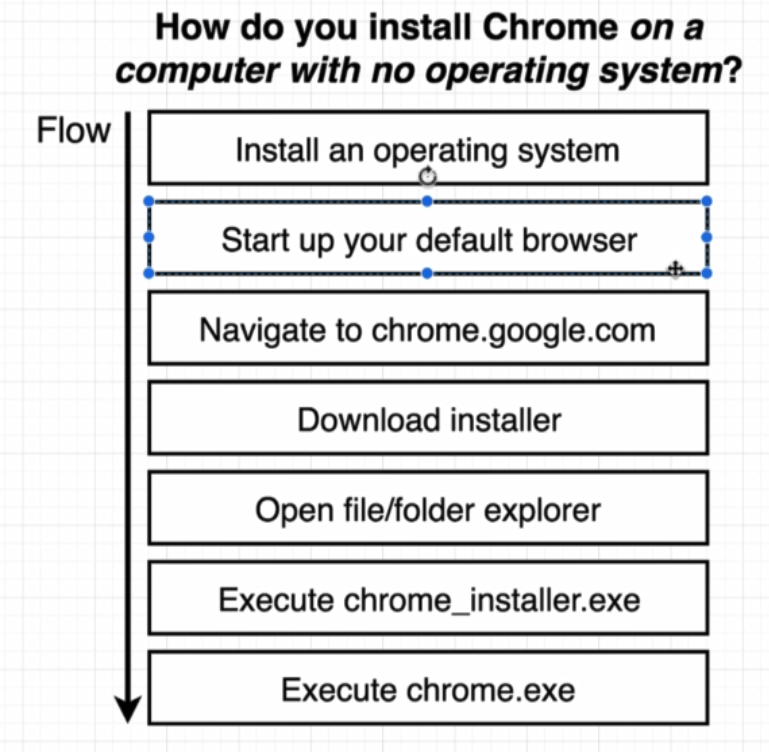
RUN:

* run instruction is used to execute some command while we are preparing our custome image.
* apk is a package manager that pre-installed on the Alpine image

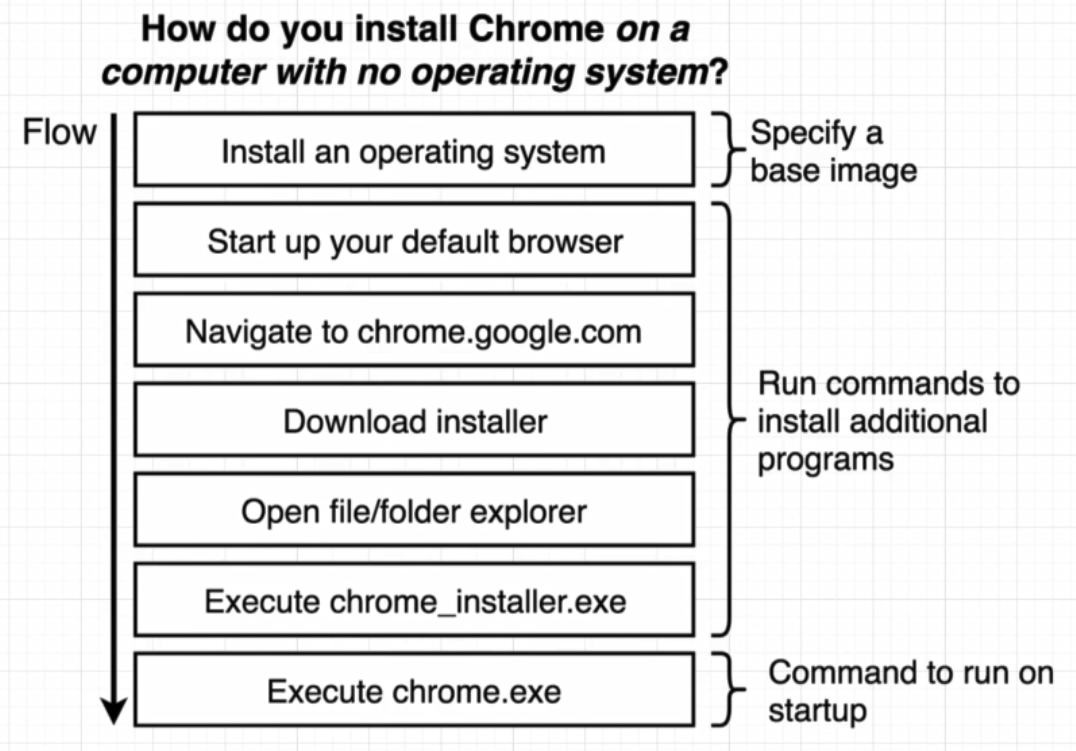
CMD:

* What should be executed when our image used to start up a brand new container

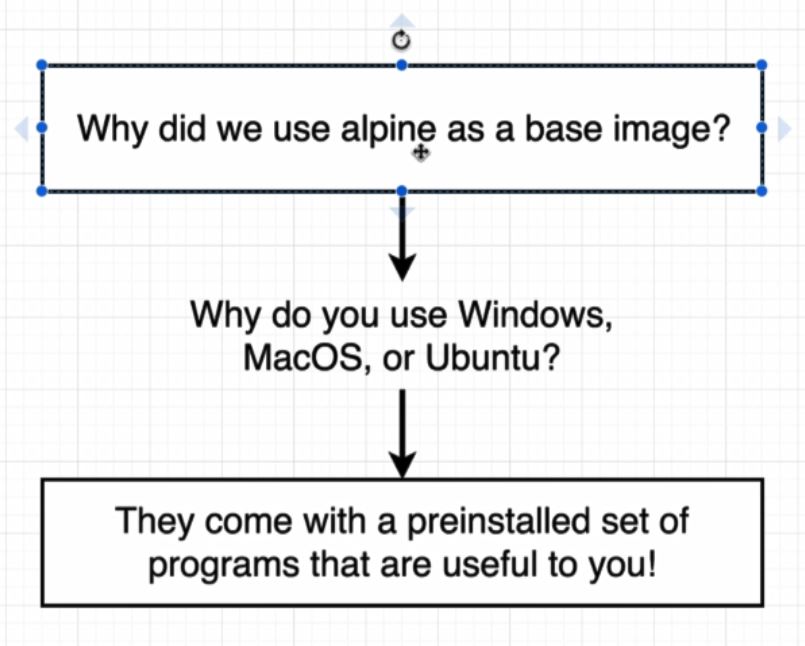




Steps bellow is similar to what we just did inside that DockerFile :

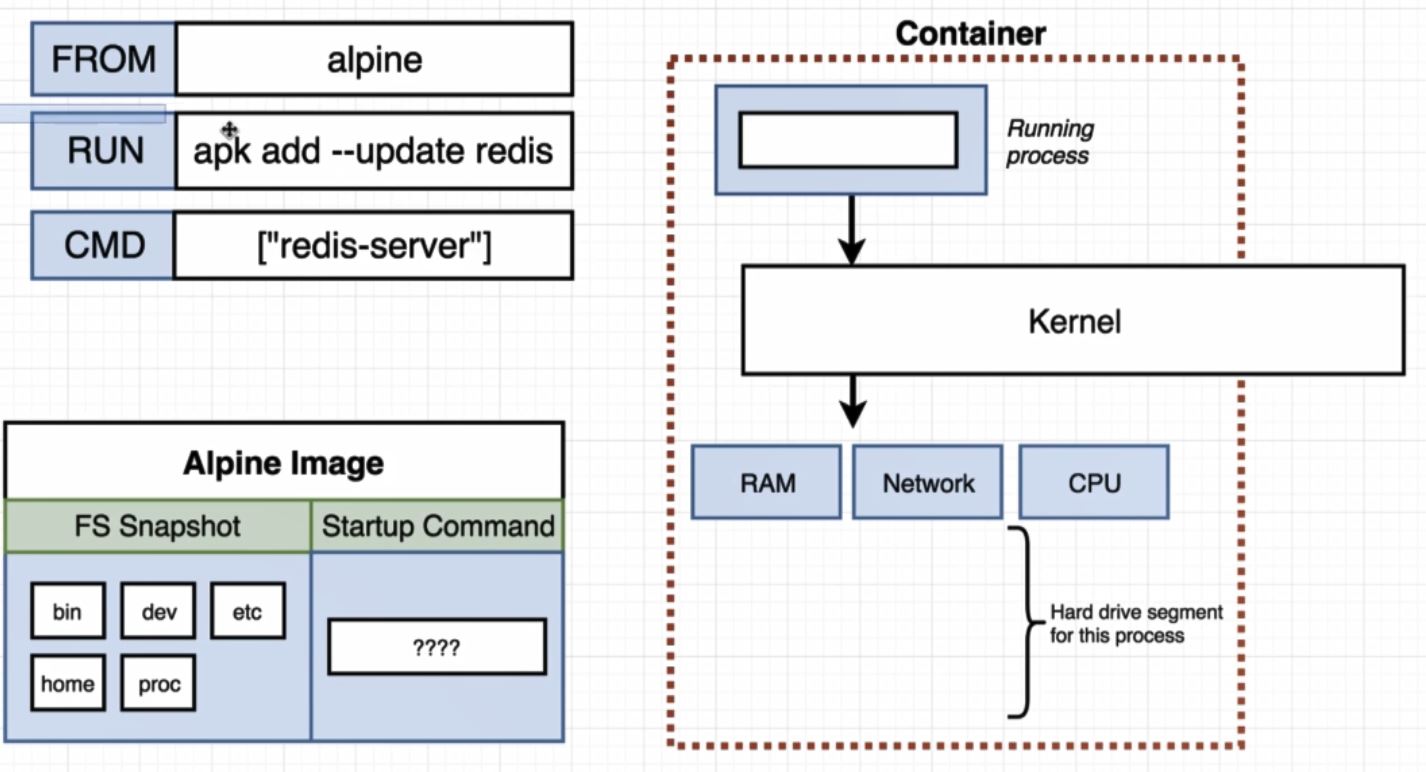


**Why did we use alpine:**

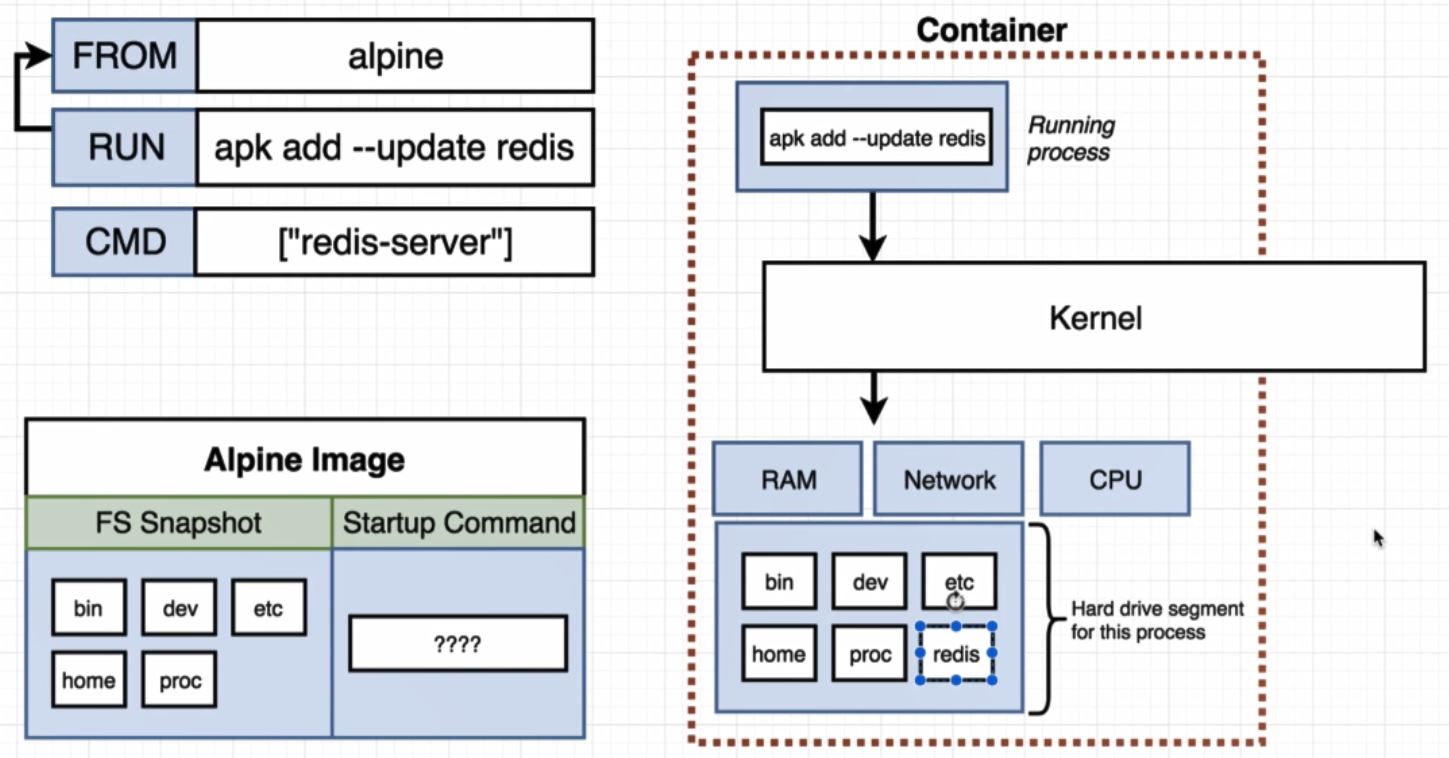


* docker build .
  + take a dockerfile and generate an image out of it.

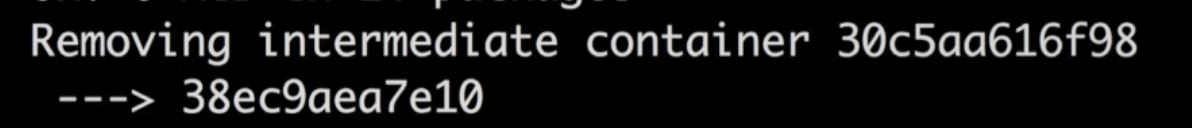
**By running docker build . the images bellow show the steps that accurre:**



After download Alpine Image, for running apk add –update redis, docker create a temporary container to run this command

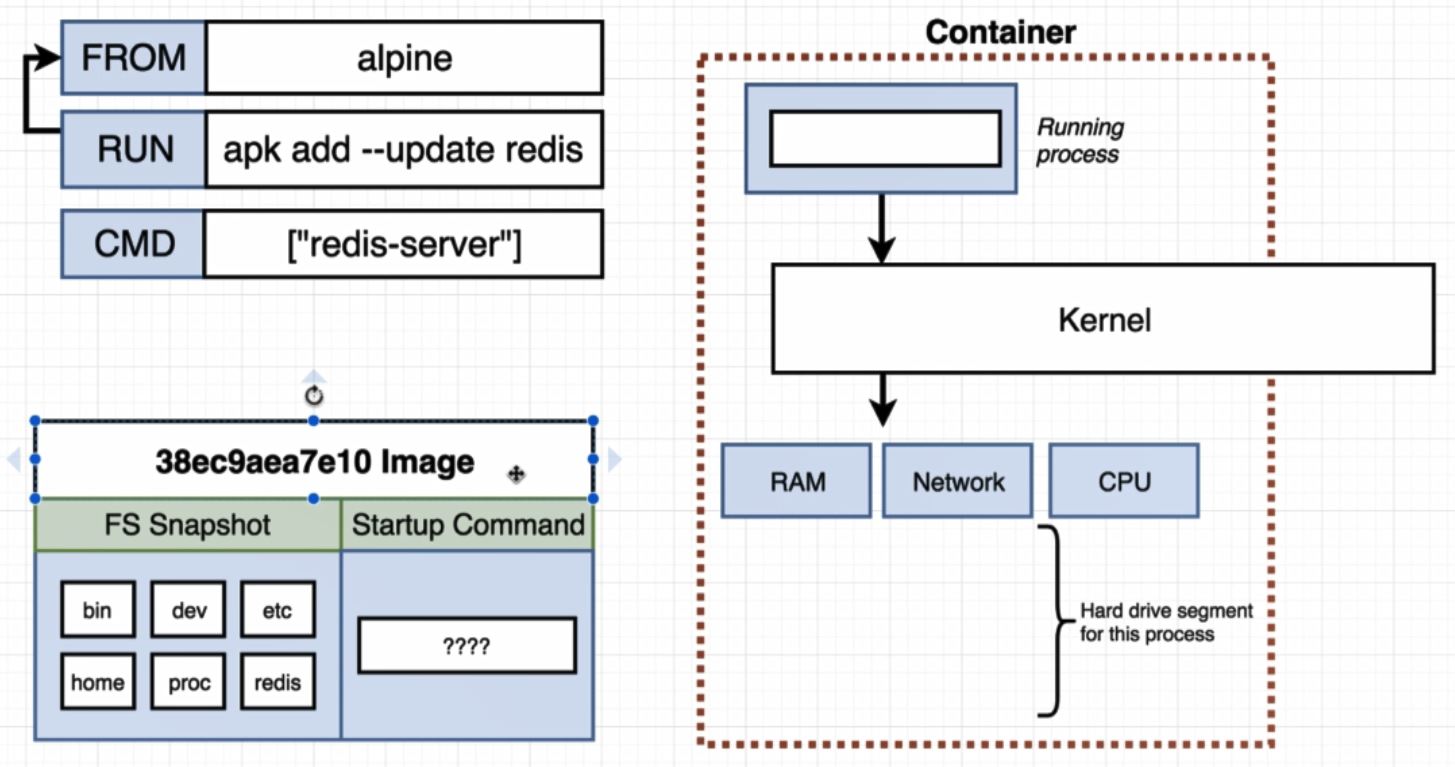


After this steps, docker stop that container and took a file system snapshot of that container and save it as a temporary image with ID



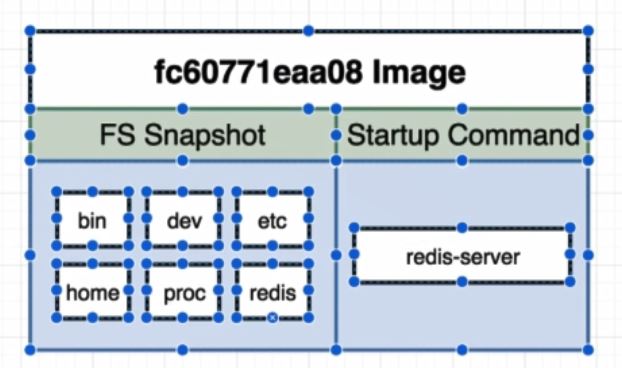
38ec9aea7e10 is the temporary image ID

After that we took prev container file system and save it as a temporary image.



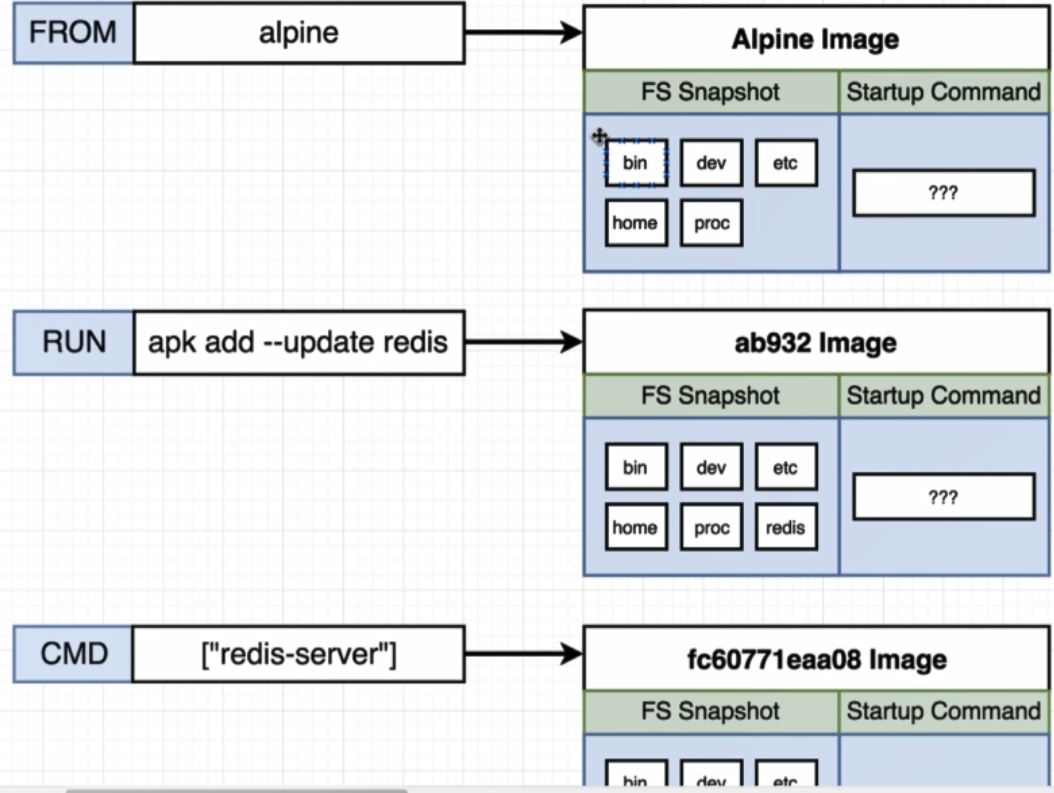
The last step is CMD

* in this step the container does not actually execute redis-server.
* It is just tells the container if you were ever run for real you should be running redis-server as your primary command.
* And the end result is something like image bellow:



**Finished the image creation**

To create image in background docker create temporary image for each step like this:



Inside Dockerfile adding another apk :

* RUN apk add –update gcc (second dependency on dockerfile)

After add this line rebuild Dockerfile

In this case there is no download for redis and use from cache because docker has realize that from previous step nothing has changed from the last time that we run docker build . .

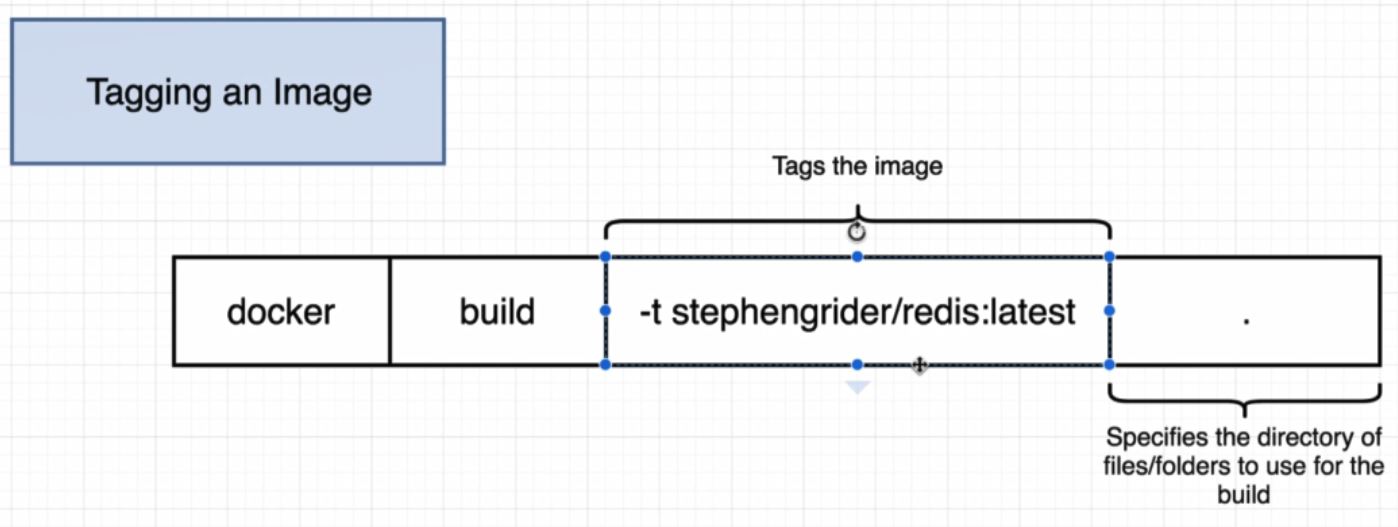
After create an image docker return ID of the created image, now by using this ID we can create container from created image.

* docker run 7dfdfbcf1017 (ID of the latest created costume image)

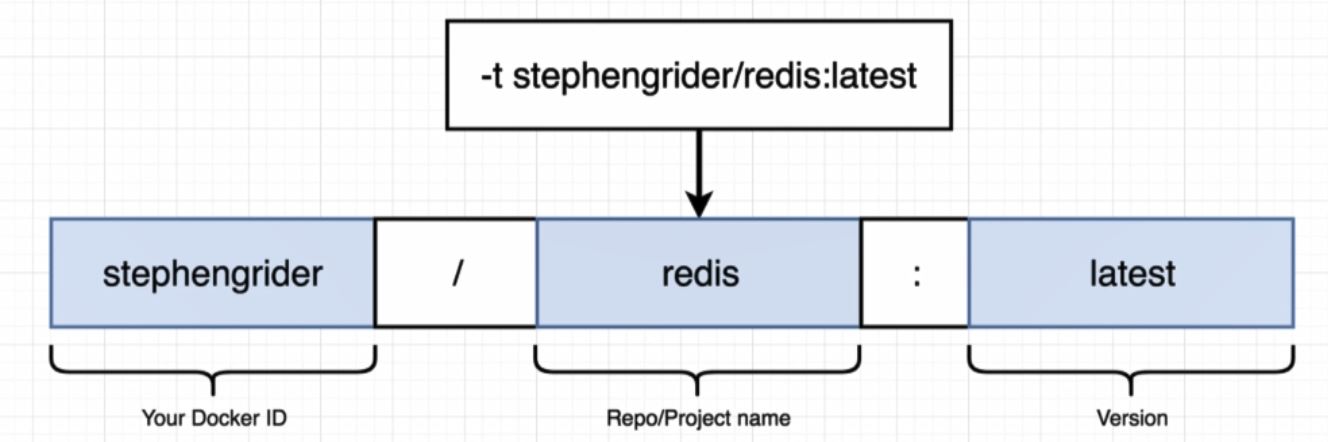
**Tagging an image**

To tag the image that is created we can use command bellow:

This make it a lot easier to refer to it without remember the ID



To give a name to image it is better to use convention bellow:



* docker build –t tohid1987/redis:latest .
* docker run tohid1987/redis