**Docker Lab**

* Create Docker Images, Containers.
* Build a Node.js app on Docker.
* How to build a node server, run app on Docker containers.
* http:// access to the node.js app on docker.
* How to pull Docker images from Docker Hub and Google Container Registry.
* How to push Docker images to Google Container Registry.

**Docker**: Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker.

**Docker Image**: is a read-only template with instructions for creating a Docker container. A docker image is described in text file called a Dockerfile, which has a simple, well-defined syntax. An image does not have states and never changes. Docker Engine provides the core Docker technology that enables images and containers.

**Docker File**: A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession. This page describes the commands you can use in a Dockerfile.

**Docker Container**: is a running instance of an image. You can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. You can also provide configuration for the network and environment variables. Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

**How to create or pull docker image?** So, there is two ways to getting docker image 1st by pulling the image from public repo using “pull” command, and other by modifying the image using Dockerfile and building the image using “build” command. So, let’s have look for both way to get docker image.

**Note:** if we are try to run an image that is not available locally but exist in public repo then by default docker will fetch the image by running docker pull command in background. Let’s try to get an image “hello-world” that is not available locally but present in dockers public repo. By default it will pull the image and we can check the images by command “docker images” and we can see a new image will listed with latest tag.

Text

Description automatically generatedText

Description automatically generated

Let’s try to run the image inside container. To run the image inside container we use “docker run <image name>” with some flag i.e. -**i** for interactive, -**t** for terminal, -**d** for run container in background, -**p** for mapping the external port with the container, -v for mapping external storage to the container etc. we have multiple flags available with run command we also have –name and –tag flag to mention the name and tag to the container. Let’s look the demo of running the image and creating a container.

Note: if we are not providing -d flag during run time then container automatically get terminated if we try to get out our prompt from docker run.

After running the container we can check the running and all processes including dead process by typing commands “docker ps” and “docker ps -a”

Text

Description automatically generatedGraphical user interface, text

Description automatically generated

Now try to create an image using Dockerfile in which we need to specify base image, working directory, some commands etc. and create the image using “docker build -t node-app:0.1 .” where -t is for Tag value. Then we can check the available images by typing “docker images”. Now we can create a container or we can run a image and expose the internal port 80 to the external host using “docker run -it -p 8080:80 –name my-app node-app:0.1”

Now we can check the server running inside the container in our host machine that is assigned to port 8080. We can browse it via browser of host or via curl cmd tool.

Similarly, we are going to change the content of the app.js file and create another image from Dockerfile and run it using same command that we use previously just we need to change the version or tag of the docker file to “node-app:02” and for container “my-app-2”. Again expose the internal server for the host and we can visit the server by specified port number.

Text

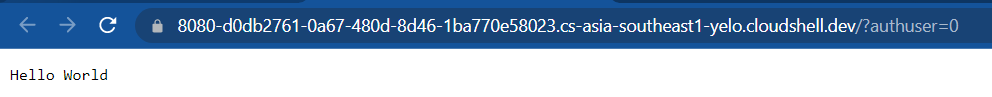
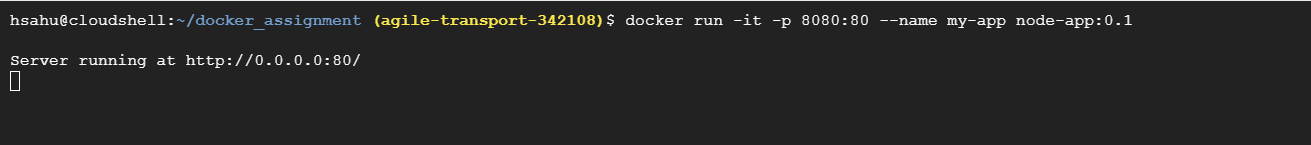
Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

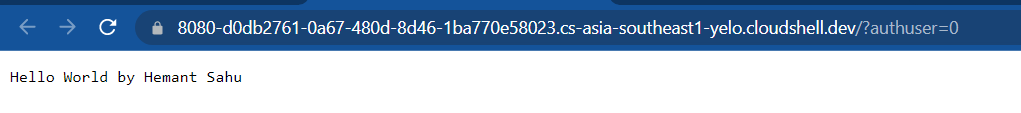
Description automatically generatedText

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Now we have our docker images with us. Before pushing to the GCR or any central repo we need to follow naming format i.e. “repo\_name/project\_id/image\_name”. we must change the tag name of the images by “docker tag old\_tag new\_tag” and then we can list the images available. Now time to push the images to the GCR using “docker push gcr.io/agile-transport-342108/node-app” and we can navigate to our GCR to view the uploaded file by URL http://gcr.io/[project-id]/image\_name/ i.e. <http://gcr.io/agile-transport-342108/node-app/>

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated