Docker Case Study

Prepared By: Yasasvi Y(IMT2016060)

Requirements:

- Dynamic Allocation of Linux systems for users
- Each user should have independent Linux System
- Specific training environment should be created in Container
- User should not allow to access other containers/images
- User should not allow to access docker command
- Monitor participants containers
- Debug/live demo for the participants if they have any doubts/bug in running applications.
- Automate container creation and deletion.

Container Image Creation:

1. First create a new container from an image.

```
docker create -it --name dockerSample ubuntu /bin/bash
```

2. Then, start and attach to the container

```
docker start dockerSample
docker attach dockerSample
```

3. Install required packages. (don't type 'sudo') For ex:

```
apt update
apt install nano
apt install gcc
apt install vim
```

4. And then, commit the container

```
docker commit -a "Durga Yasasvi" 43a1d0cb378a6 dockerSample
```

Now, the container image is ready.

Containers allocation to users:

1. The shell script createContainers.sh automatically creates a docker container for every user.
   - `users.txt`

   ```
   Yasasvi
   Sumanth
   Puneeth
   Siddu
   Srujan
   ```
   createContainers.sh

   ```
   echo -n "Name of the file(with usernames): "
   read file
   while read user
       do
         docker create -it --name $user dockerSamp /bin/bash
          done < $file
   ```

2. In users.txt fill the entries with usernames and run shell script, this will create a docker container for every username from users.txt.

3. Now, the user can start using the container by following:

   - `docker start <name> # Container gets started`
   - `docker attach <name> # Container gets attached`
   - (OR)
   - `docker start <name> # Container gets started`
   - `docker exec -it <name> /bin/bash`

Containers Monitoring:

The following are the ways to monitor the container:

- Stats of the containers

  ```
  docker stats <user>
  ```

- Logs of a container

  ```
  docker logs -f <user>
  ```

- Attach to container

  ```
  docker attach <user> # On exit, container gets shutdown.
  ```

- To start a new shell

  ```
  docker exec -it <user> bin/bash # On exit the container continues
  to run.
  ```

Deleting the Containers:

- Automate the deletion using the `containers_delete.sh` script.

  - `containers_delete.sh`
  - echo -n “Name of the file (with usernames): “
  - read file
  - while read user
  -         do
  -                 docker stop $user
  -                 docker rm $user
  - done < $file

- You can either delete all users or user by name using the command `containers_delete.sh -x`.