

# Docker Case Study - Automating Infra. Allocation

## Problem Statement

Creating individual workspaces for users in IT training support classes.

## Requirements

1. Dynamic Allocation of Linux systems to users.
2. Each user should have an independent Linux System.
3. Specific training environment should be created in the container.
4. Users should not be able to access other containers or images or even the docker command.
5. Monitor users' containers.
6. Automate container creation and deletion.

## Creating the container image

1. Create a new container from a base image of your choice (we'll be using ubuntu image) by running the following command-

```
sudo docker create -it --name baseCont ubuntu /bin/bash
```

2. Start and Attach to the container-

```
sudo docker start baseCont
```

```
sudo docker attach baseCont
```

3. Install all the required packages. Ex- for creating a training environment for C programming, we need to install a text editor, compiler, and manual pages-

```
apt update
```

```
apt install vim
```

```
apt install gcc
```

```
apt install manpages-dev
```

```
apt install manpages-posix-dev
```

4. Exit the container by exit command

5. Commit the changes to the container by the following command-

```
sudo docker commit -a "nimisha" 6d696a4f1fd7 baseCont_image
```

Now, our training container image is ready.

## Allocating containers to Users

1. The shell script create\_containers.sh will automatically create a container for every user present in the users.txt file.

```
s-NimishaGarg/Docker — Atom
create_containers.sh
1 echo -n "Enter name of file with usernames: "
2 read file
3 while read user
4 do
5     docker create -it --name $user baseCont_image /bin/bash
6 done < $file
7
```

```
users.txt (~/.sem5/se/assignments-NimishaGarg)
Open [+]
Nimisha
Shreya
Mudita
Akshi
Apoorva|
```

2. Fill in the entries in the file users.txt with usernames and run the shell script create\_containers.sh by the following command-

sh create\_containers.sh -x

This will create a container corresponding to each user in users.txt.

3. The user can start using the designated container by the following command-

sudo docker start <name>

sudo docker attach <name>

## Monitoring the Containers

The instructor can monitor the users' containers in the following ways-

1. For seeing the usage stats of containers-

sudo docker stats <name>

2. For seeing the logs of containers-

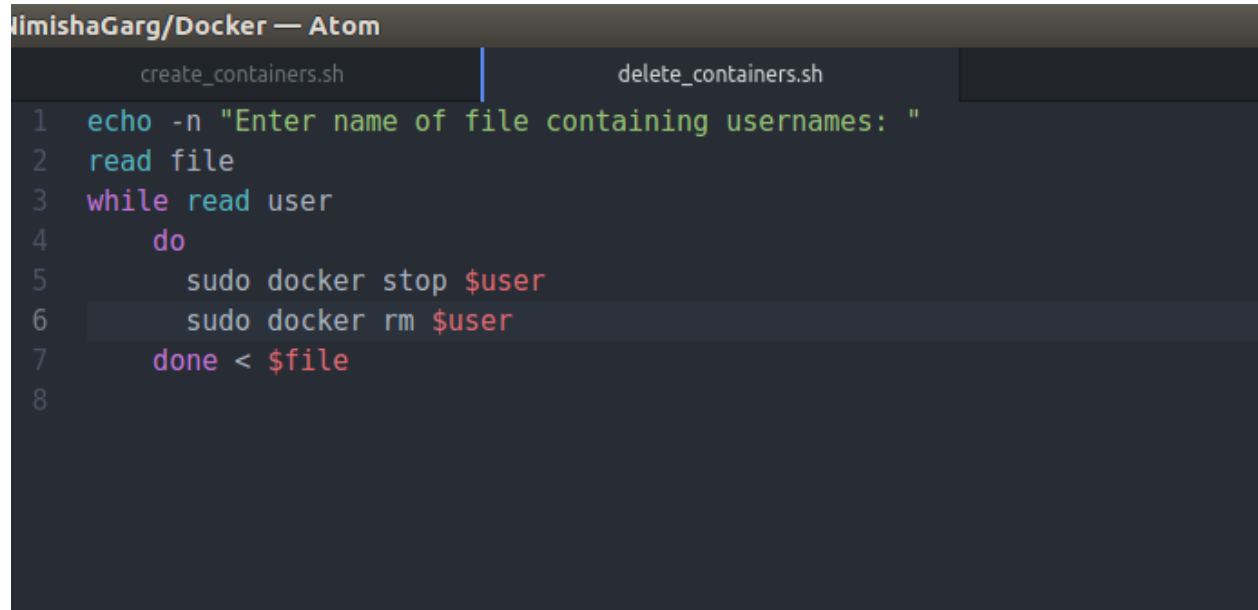
sudo docker logs -f <name>

3. Instructor can also attach to a container-

```
sudo docker attach <name>
```

## Deleting the containers

For deleting all/some containers we can use the shell script `delete_containers.sh`. Enter the names of users whose containers you want to delete in a separate file.

A screenshot of a code editor window titled "imishaGarg/Docker — Atom". The editor has two tabs: "create\_containers.sh" and "delete\_containers.sh", with the latter being the active tab. The code in the active tab is a shell script for deleting Docker containers. It consists of eight lines: line 1 is an echo statement for user input; line 2 reads the input into a variable 'file'; line 3 starts a 'while' loop reading from 'file' into 'user'; line 4 starts a 'do' block; line 5 runs 'sudo docker stop \$user'; line 6 runs 'sudo docker rm \$user'; line 7 ends the 'do' block with 'done < \$file'; and line 8 is an empty line.

```
imishaGarg/Docker — Atom
create_containers.sh  delete_containers.sh
1 echo -n "Enter name of file containing usernames: "
2 read file
3 while read user
4 do
5     sudo docker stop $user
6     sudo docker rm $user
7 done < $file
8
```