

# Cloud Computing – WS 2017

Exercise 2 : Docker Installation and Configuration

21<sup>st</sup> November 2017

Anshul Jindal

[anshul.jindal@tum.de](mailto:anshul.jindal@tum.de)

# Index

- Exercise 1 Solution
- Exercise 2 Introduction
- Tasks To be Completed
- Submission

# Exercise 1 Solution

## Task 1

Use of exec API to call the commands

Get the Number of users logged in

```
exec("who | wc -l",function (error, stdout, stderr)
{
    var lines = stdout.toString().split('\n');
    exercise_1_Message.numberUsers=lines[0];
    exec("who | cut -d \" \" -f 1",function (error, stdout, stderr)
    {
        var lines = stdout.toString().split('\n');
        lines = lines.filter(Boolean)
        exercise_1_Message.userNames=lines;
        exec("fdisk -l | grep \"^Disk\" | grep \"vd\" | cut -f 1 -d ',' | cut -f 3- -d '/' | wc -l",function (error, stdout, stderr)
        {
```

Get the Names of users logged in

Get the Number of disks attached to VM

## Task 1 Cont..

```
var lines = stdout.toString().split('\n');
lines = lines.filter(Boolean)
exercise_1_Message.numStorageDisks=lines[0];
exec("fdisk -l | grep \"^Disk\" | grep \"vd\" | cut -f 1 -d ',' | cut -f 3- -d '/'",function (error, stdout, stderr)
{
    var lines = stdout.toString().split('\n');
    lines = lines.filter(Boolean)
    exercise_1_Message.storageDisksInfo=lines;
    console.log(exercise_1_Message);
    res.json( exercise_1_Message);
});
});
});
```

Get the Size information of the disks attached

Send Back the filled message

## Task 2

```
/**
 * checks if:
 * auth array exists
 * first value matches the expected username
 * second value the expected password
 */
if (!auth || auth[0] !== 'CCS' || auth[1] !== 'CCS_exercise1_task2') {
  /**
   * any of the tests failed
   * send an Basic Auth request (HTTP Code: 401 Unauthorized)
   */
  res.statusCode = 401;
  res.setHeader('WWW-Authenticate', 'Basic realm="Enter the Username and Passord:"');
  /**
   * this will displayed in the browser when authorization is cancelled
   */
  res.end('Unsuccessful Authentication');
}
else {
  /**
   * Processing can be continued here, user was authenticated
   */
  res.send('Successful Authentication');
}
```

Parse auth and check for the given username and password

If not Successful

If Successful

# Exercise 2

**What are the problems with the deployment method used  
in the first exercise ?**



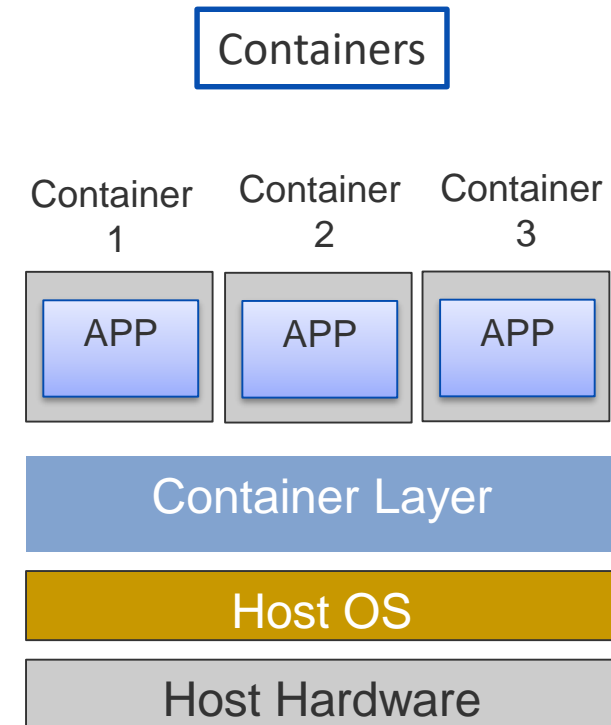
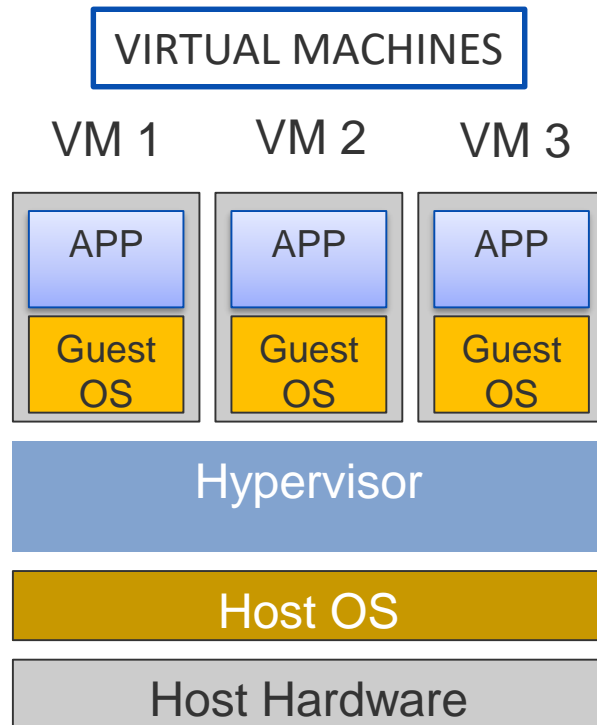
## Problems with the deployment method of First Exercise

- **OS Dependent**
  - Different deployment procedure and requirements for different OS.
- **Not-Scalable**
  - Create more VMs ?
- **Not-Portable**
  - Running the same procedure from starting again on the new machine ?
- And many more....

## Containerization (container-based virtualization)

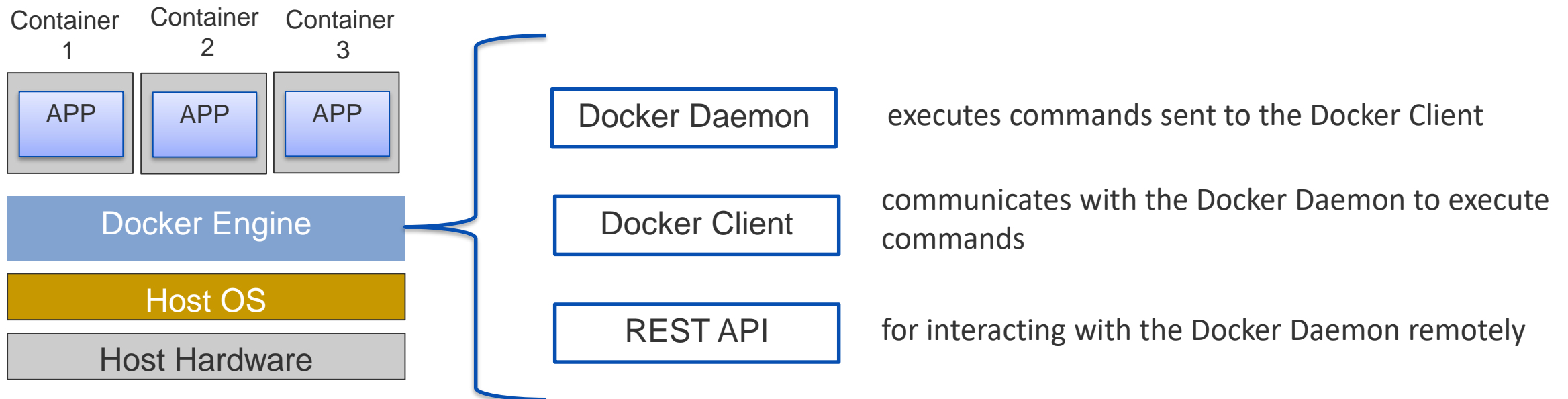
### What?

- is an OS-level virtualization method for deploying and running distributed applications without launching an entire VM for each application.
- share the same OS kernel as the host.

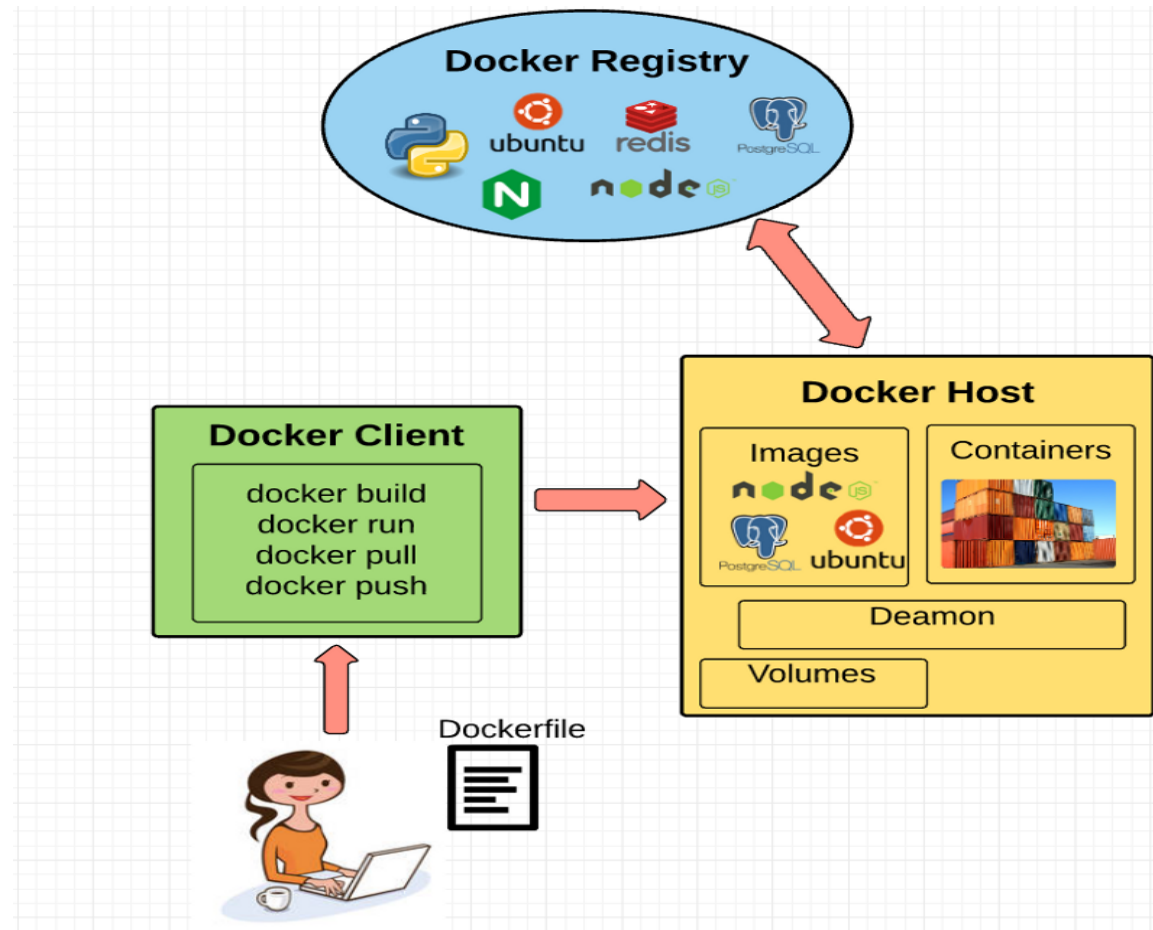


## Docker

- Docker is a tool that packages, provisions and runs containers independent of the OS.



## Docker Cont.



## Why Containers ?

In real World if a country wants to export any luggage to another country, they put them into a container then they ship that container either by road or through a dock to another country.



Within containers, every stock is placed such that it does not get damaged upon small jerks and everything is placed safely regardless of where the container is moving.

## Why Containers ? Cont..

The same above applies to Docker for running and maintaining distributed applications.

- It packs your application in a container with all of your application's bins\libs and dependencies
- makes it fully isolated from external environments regardless of where it is running.
- we can ship that container to any where i.e. To any other OS, to Docker Registry (such as Docker Hub) or to the cloud.

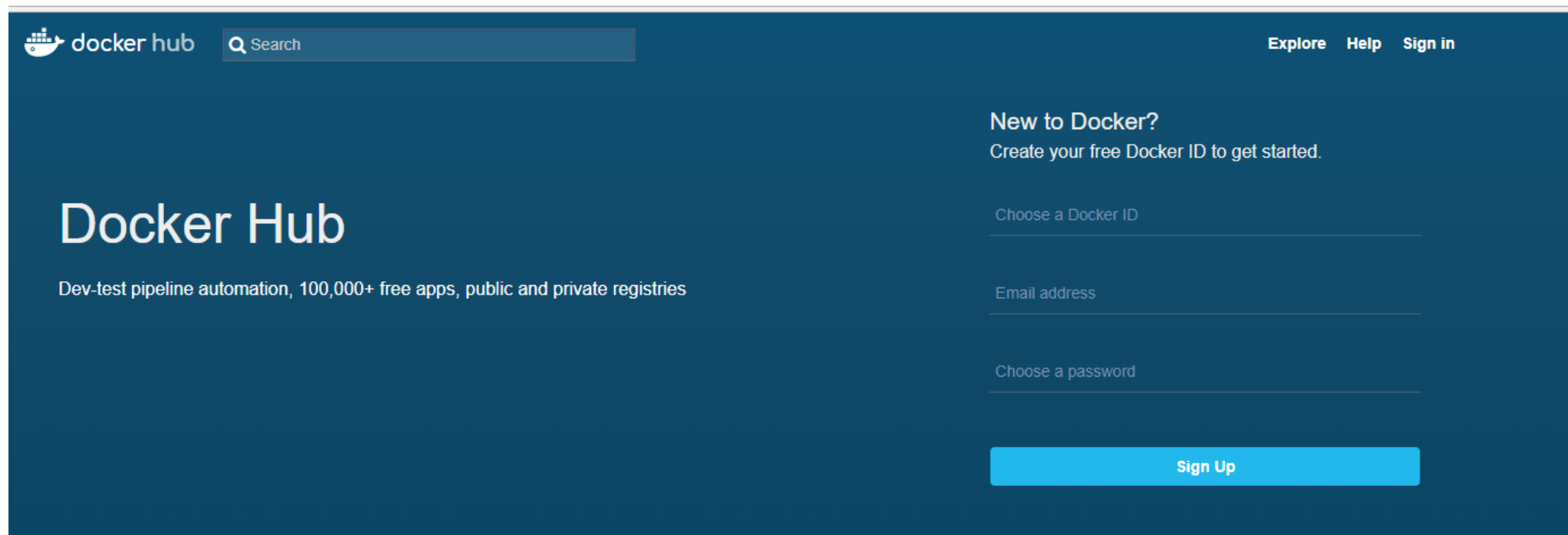
**OS Independent**

**Scalable**

**Portable**

## Docker Hub

- Cloud-based registry service
- Allows you to link to code repositories, build your images, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.
- It provides a centralized resource for container image discovery, distribution and management



The screenshot shows the Docker Hub homepage with a dark blue background. At the top left is the Docker Hub logo and a search bar. At the top right are links for 'Explore', 'Help', and 'Sign in'. The main heading 'Docker Hub' is prominently displayed, followed by the tagline 'Dev-test pipeline automation, 100,000+ free apps, public and private registries'. On the right side, there is a 'New to Docker?' section with the text 'Create your free Docker ID to get started.' Below this are three input fields: 'Choose a Docker ID', 'Email address', and 'Choose a password'. A bright blue 'Sign Up' button is positioned at the bottom of this section.

docker hub

Explore Help Sign in

# Docker Hub

Dev-test pipeline automation, 100,000+ free apps, public and private registries

**New to Docker?**  
Create your free Docker ID to get started.

**Sign Up**

# Short Hands-on Demo



# Tasks To be Completed

## Tasks to be completed

As part of the exercise2, there are five subtasks to be completed:

1. Add an API in your application:
  1. `/exercises/exercise2` : Which sends a message "group 'GroupNumber' application deployed using docker".
2. Create the docker file of your application and then build the image.
3. Create your docker hub account and push your application image to it by following the procedure stated in detail document. **Name of your application image should be as `cloudcomputinggroup'GroupNumber'`**
4. Start a VM and pull this application image on it and run it using docker (you should pull and run the image otherwise exercise would not be able to pass successfully).
5. Enable docker remote API (procedure specified in detail document page 13)

\*Replace GroupNumber with your group number in above tasks.

**Deadline for submission: 27<sup>th</sup> November 2017 23:59**

# Submission

## Submission

To submit your application results you need to follow this :

1. Open the cloud Class server url
2. Login with your provided username and password.
3. After logging in, you will find the button for **exercise2**
4. Click on it and a form will come up where you must provide
  1. VM ip on which your application is running
  2. and the **dockerhub** image path name.

**Example:**

10.0.23.1

dockerHubUserId/myImage

5. Then click submit.
6. You will get the correct submission from server if everything is done correctly.

Remember no cheating and no Hacking 😊

## Important points to Note:

1. Make sure your VM and your application is running after following all the steps mentioned in this manual.
2. We will grade you based upon the number of tasks completed by you.
3. You will get to see, what your application has submitted to the server.
4. You can submit as many times until the deadline of exercise.
5. Multiple submission will overwrite the previous results.

Good Luck and Happy Coding😊

Thank you for your attention! 😊

# Questions?

# Appendix

## Install the node and npm

- Get yourself a more recent version of Node.js by adding a PPA (personal package archive) maintained by NodeSource.

```
curl -sL https://deb.nodesource.com/setup_7.x | sudo -E bash -
```

- You can now install the Node.js package.

```
sudo apt-get install nodejs
```

The nodejs package contains the nodejs binary as well as npm, so you don't need to install npm separately.

- For some npm packages to work (such as those that require building from source), you will need to install the build-essentials package

```
sudo apt-get install build-essential
```

- Test Node: `node -v` (This should print a version number, so you'll see something like this v0.10.35)
- Test NPM: `npm -v` (This should print NPM's version number so you'll see something like this 1.4.28)



## Installation of required modules

1. As part of this application some modules need to be installed. For installing them run the following command from inside the directory of application.

`npm install`

This command will install all the dependent modules mentioned in the package.json file.

2. If you need some other modules you can install them by running the command

`npm install "module name" --save`

This will automatically add that module in the package.json file and now you can use it inside your development file.

## Node.js Client Application Deployment

1. Now your application is ready to be deployed on VM. Run the following command to start the application:

**node clientApplication.js**

After running this, on console you will see

**“Server started and listening on port 8080”**

2. Now your application is deployed on the server. Open your browser on your local machine and enter the address as

[http://IP\\_ADDRESS\\_VM:8080/exercises](http://IP_ADDRESS_VM:8080/exercises)

`'Welcome to Cloud Computing Exercises API`

## Node.js Client Application Deployment : Port unblock

- If your request timed out, your VM probably has some firewall rules in place prevent a user to call your web server from the outside.
- The iptables rules are located in the file **/etc/iptables/rules.v4**. Open this file with your favourite editor:
- After line 9 insert a new line allowing incoming connections on port 8080:

`-A INPUT -p tcp -m tcp --dport 8080 -m state --state NEW -j ACCEPT`

```
# Generated by iptables-save v1.6.0 on Fri May  6 15:32:09 2016
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 4 -j ACCEPT
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8080 -m state --state NEW -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -m limit --limit 3/min -j LOG
COMMIT
# Completed on Fri May  6 15:32:09 2016
~
```

- Save it and to apply the new iptables rules, you need to reload them to your local firewall system.

`iptables-restore < /etc/iptables/rules.v4`