מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

Real-Time Group

Docker

benny@rt-ed.co.il

by Vladimir Levintovich

רח` מרדכי רוז'נסקי 14, ראשל"צ טל. 7067057 -077

Introduction to Docker

Docker Introduction

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

And importantly, Docker is open source. This means that anyone can contribute to Docker and extend it to meet their own needs if they need additional features that aren't available out of the box.

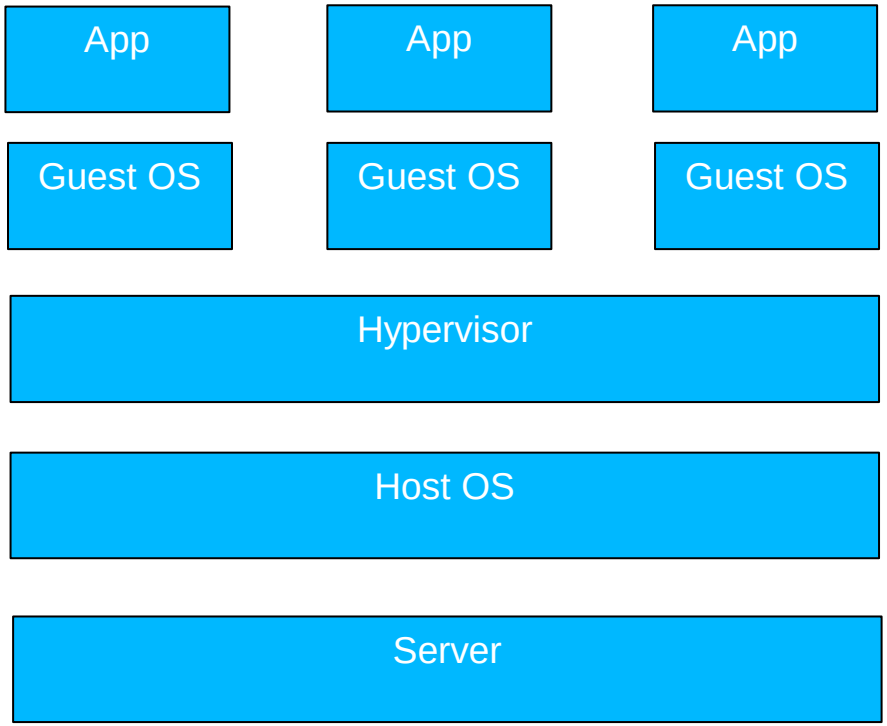
What is Docker for?

Docker is a tool that is designed to benefit both developers and system administrators, making it a part of many DevOps (developers + operations) toolchains. For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on. It also allows them to get a head start by using one of thousands of programs already designed to run in a Docker container as a part of their application. For operations staff, Docker gives flexibility and potentially reduces the number of systems needed because of its small footprint and lower overhead.

Docker Architecture

Virtualization architecture

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

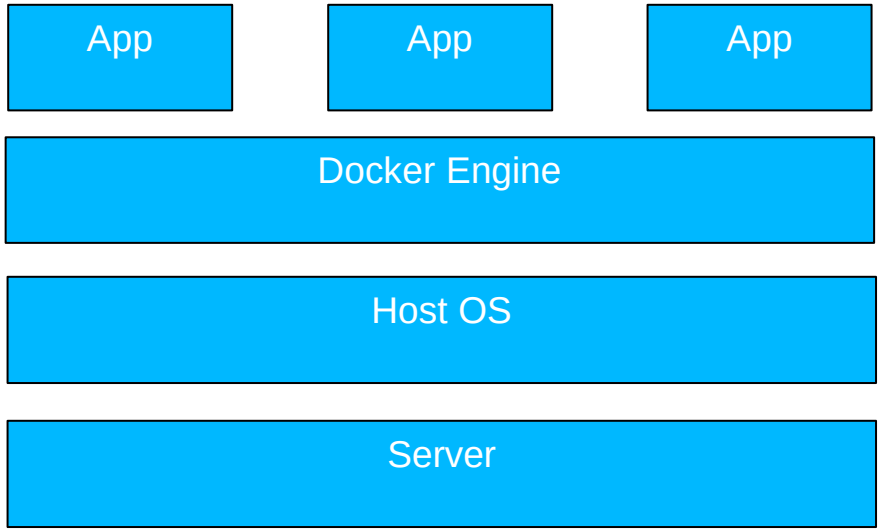


Virtualization architecture

- **Server** - is the physical server that is used to host multiple virtual machines.
- **Host OS** - is the base machine such as Linux or Windows.
- **Hypervisor** - is either VMWare or VirtualBox that is used to host virtual machines.
- **Guest OS** - you can install multiple operating systems as virtual machines.
- **App** - an application running on your Guest OS.

Docker architecture

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק



- **Docker Engine** - is used to run the operating system which earlier used to be virtual machines as Docker containers.
- **All of the Apps** now run as Docker containers.

Docker installation

Docker installation

- Update Local Database

  $ sudo apt-get update

- Download Dependencies

  $ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

- **apt-transport-https** - Allows the package manager to transfer files and data over https
- **ca-certificates** - Allows the system (and web browser) to check security certificates
- **curl** - This is a tool for transferring data
- **software-properties-common** - Adds scripts for managing software

Docker installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Add Docker's GPG Key

  $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

```
[redacted]@slaveDocker:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg
| sudo apt-key add -
OK
```

- Install the Docker Repository

  $ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

- Update Repositories

  $ sudo apt-get update

Docker installation

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Install Latest Version of Docker

  $ sudo apt-get install docker-ce

- Check the Docker version.

  $ docker version

Having an Issue?

```
vladimir@slaveDocker:~$ docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false
Got permission denied while trying to connect to the Docker daemon socket at uni
x:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/version: d
ial unix /var/run/docker.sock: connect: permission denied
```

Docker installation

- Create the docker group.

  $ sudo groupadd docker

- Add your user to the docker group.

  $ sudo usermod -aG docker ${USER}

- Logout and log back.
- Verify that you can run docker commands without sudo.

  $ docker run hello-world

```
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Running a first container

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- You can check your first downloaded image by the commands

  $ docker images

```
[redacted]@slaveDocker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hello-world         latest              bf756fb1ae65        4 months ago
13.3kB
```

- Run an Ubuntu image container

  $ docker run -it ubuntu bash

```
vladimir@slaveDocker:~$  docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
d51af753c3d3: Pull complete
fc878cd0a91c: Pull complete
6154df8ff988: Pull complete
fee5db0ff82f: Pull complete
Digest: sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7
Status: Downloaded newer image for ubuntu:latest
root@dbb429db8653:/#
```

Docker info

- To see more information on the Docker running on the system, you can issue the following command:

  $ docker info

- It is used to ensure that the Docker command returns the detailed information on the Docker service installed:

  - Number of containers

  - Number of images

  - The storage driver used by Docker

  - The root directory used by Docker

Docker Images

Docker images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- In Docker, everything is based on Images. An image is a combination of a file system and parameters.
- The run command is used to mention that we want to create an instance of an image, which is called a container.
- For example we can use the CentOS docker image to run CentOS on our Ubuntu machine.

$ docker run -it centos /bin/bash

```
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
8a29a15cefae: Pull complete
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700
Status: Downloaded newer image for centos:latest
[root@35b4dfc6d6ef /]#
```

Docker images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

$ docker run -it centos /bin/bash

- **centos** – name of the image. It was downloaded from Docker Hub.
- **it** – running in **interactive mode**
- **/bin/bash -** is used to run the bash shell once CentOS is up and running.

$ docker run -it centos echo “hello”

- To see the list of Docker images on the system

$ docker images

```
[redacted]@slaveDocker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              1d622ef86b13        4 weeks ago         73.9MB
centos              latest              470671670cac        4 months ago        237MB
hello-world         latest              bf756fb1ae65        4 months ago        13.3kB
```

Docker images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

```
[redacted]@slaveDocker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              1d622ef86b13        4 weeks ago         73.9MB
centos              latest              470671670cac        4 months ago        237MB
hello-world         latest              bf756fb1ae65        4 months ago        13.3kB
```

- From the above output, we can see that the server has three images: ubuntu, centos and hello-world. Each image has the following attributes:
- **TAG** – This is used to logically tag images.
- **IMAGE ID** - This is used to uniquely identify the image.
- **CREATED** - The number of days since the image was created.
- **SIZE** – The virtual size of the image.

Docker images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- The images can be downloaded from Docker Hub using the Docker run command

  $ docker run mysql

```
vladimir@slaveDocker:~$ docker run mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
afb6ec6fdc1c: Pull complete
0bdc5971ba40: Pull complete
```

- The Docker images on the system can be removed via the **docker rmi** command.

  $ docker rmi <ImageID>

- To remove **mysql** image:

  $ docker rmi 30f937e841c8

  $ docker rmi –f 30f937e841c8 (force removal of the image)

Docker images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- To return only the Image ID's of the images

  $ docker images -q

```
[redacted]@slaveDocker:~$ docker images -q
1d622ef86b13
470671670cac
bf756fb1ae65
```

- To see the details of an image or container
- $ docker inspect <Repository>
- **Repository – the name of the Image.**

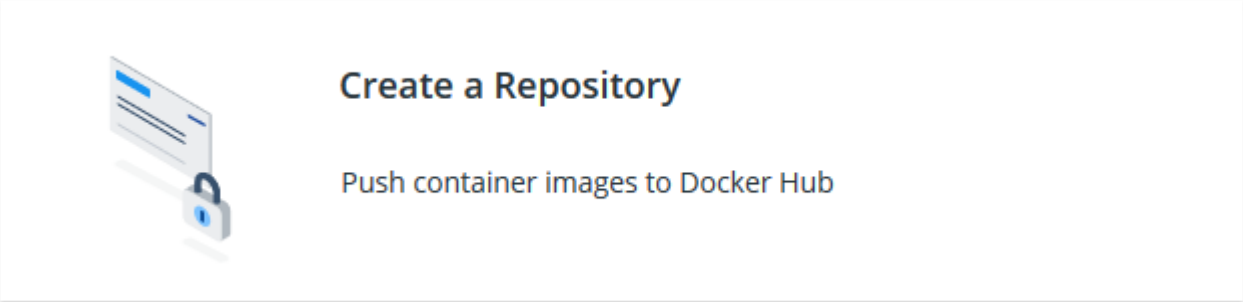
- $ docker inspect centos

Docker Hub

Docker hub

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Next step is to create account in Docker hub
- https://hub.docker.com/
- Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications.
- After you login go to the main page and click on create repository



Docker hub

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

Docker Hub

https://cloud.docker.com/repository/create

docker hub

Explore Repositories Organizations Get Help rlss91

Repositories > Create

Using 0 of 1 private repositories. Get more

Create Repository

rlss91 maven-project

Description

Visibility

Using 0 of 1 private repositories. Get more

Public
Public repositories appear in Docker Hub search results

Private
Only you can view private repositories

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository. Learn More.

Please re-link a GitHub or Bitbucket account

We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. Learn More

Disconnected Disconnected

Cancel Create Create & Build

Pro tip

You may push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

Docker hub

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Login to your Docker Hub

  $ docker login

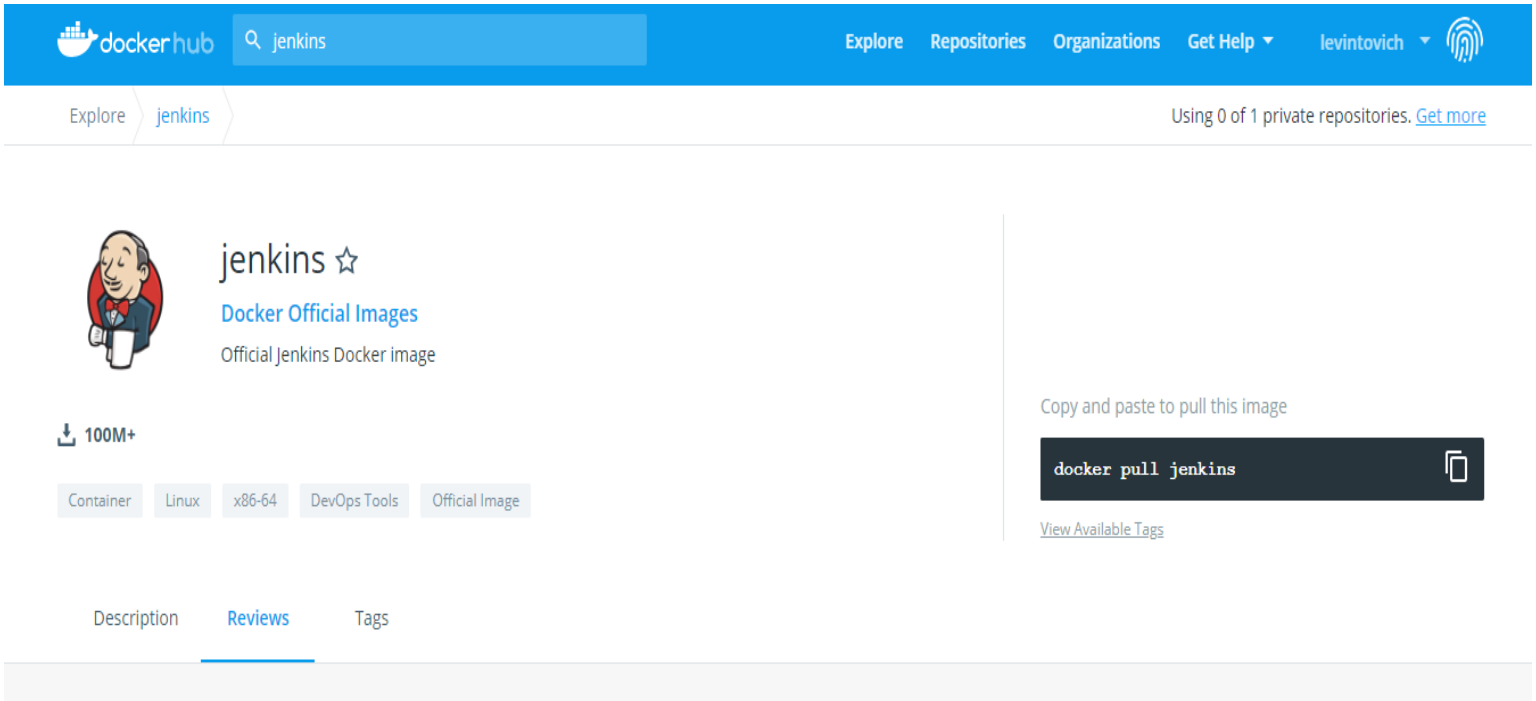
```
[redacted]@slaveDocker:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: [redacted]
Password:
WARNING! Your password will be stored unencrypted in /home/vladimir/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

- After login is successful you are able to push docker images to your docker hub repositories.

To pull images

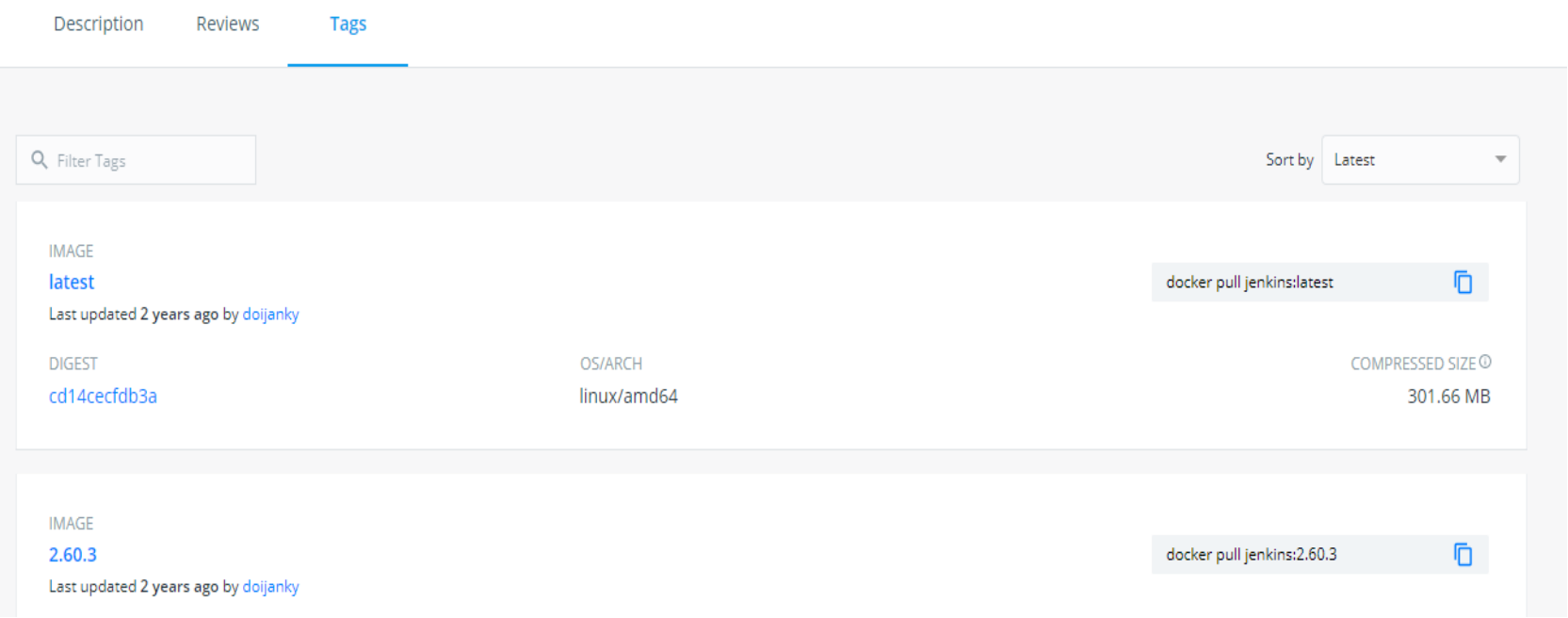
מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Let’s browse and find the Jenkins image



To pull images

- You can choose a particular tag (version) of the docker image and then pull



To pull images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Pull the latest version of Jenkins docker image

  $ docker pull jenkins

- To run Jenkins, you need to run the following command

  $ docker run -p 8080:8080 -p 50000:50000 jenkins

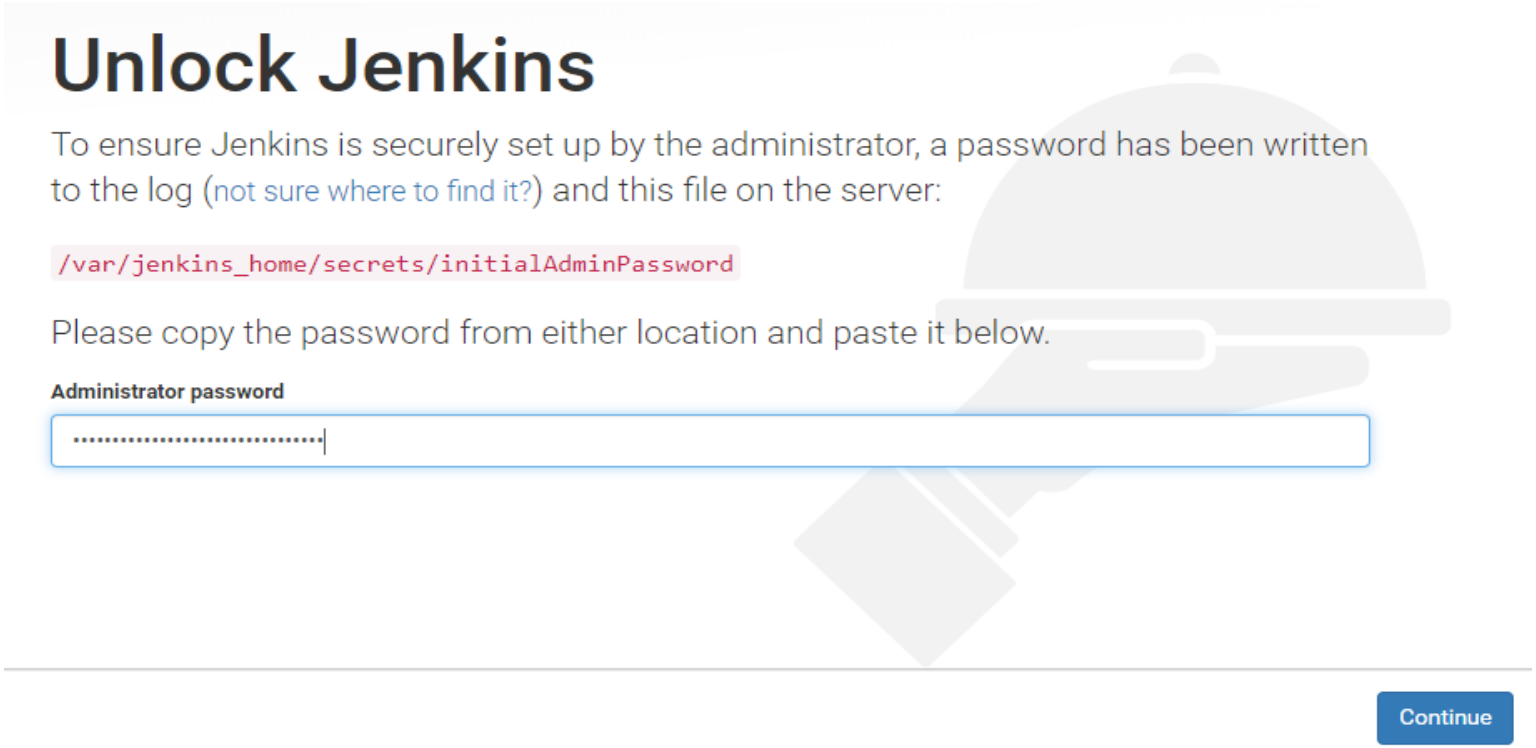
```
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

d73cb7f5667b4985865d111e8dd365e9

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

--> setting agent port for jnlp
--> setting agent port for jnlp... done
May 22, 2020 7:44:10 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
May 22, 2020 7:44:11 PM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
May 22, 2020 7:44:11 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
May 22, 2020 7:44:12 PM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
May 22, 2020 7:44:13 PM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
May 22, 2020 7:44:13 PM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata. 15,119 ms
```

Jenkins on Docker

Jenkins on docker

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Browse your VM IP with port 8080 and run Jenkins



Docker - Containers

Docker Containers

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Run a container in an interactive mode

  $ docker run -it centos /bin/bash

```
@6f601a201e32:/
vladimir@slaveDocker:~$ docker run -it centos /bin/bash
[root@6f601a201e32 /]#
```

- Open another session to your Ubuntu VM and get the list of all running docker containers via the command:

  $ docker ps

```
[redacted]@slaveDocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
6f601a201e32        centos              "/bin/bash"         8 minutes ago       Up 8 minutes                            peaceful_agnesi
```

Docker Containers

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- To list all of the containers on the system

  $ docker ps -a

```
[redacted]@slaveDocker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS                          PORTS                NAMES
6f601a201e32        centos              "/bin/bash"              13 minutes ago
     Up 13 minutes                                            peaceful_agnesi
795fa6c22dd6        jenkins             "/bin/tini -- /usr/l…"   52 minutes ago
     Exited (130) 23 minutes ago                              naughty_keller
fe37eca43cld        ubuntu              "/bin/bash -c 'apt-g…"   3 hours ago
     Exited (0) 3 hours ago                                   priceless_kepler
da8cc05ad050        ubuntu              "/bin/bash -c 'apt-g…"   3 hours ago
     Exited (0) 3 hours ago                                   inspiring_burnell
```

- You can see all the commands that were run with an image via a container

  $ docker history <ImageID>

```
vladimir@slaveDocker:~$ docker history centos
IMAGE               CREATED             CREATED BY
        SIZE                COMMENT
470671670cac        4 months ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]
        0B
<missing>           4 months ago        /bin/sh -c #(nop)  LABEL org.label-schem
a.sc…   0B
<missing>           4 months ago        /bin/sh -c #(nop) ADD file:aa54047c80ba3
0064…   237MB
```

Docker Containers

- To see the top processes within a container

  $ docker top <ContainerID>

```
[redacted]@slaveDocker:~$ docker top 24b56ea12763
UID                 PID                 PPID                C
root                20215               20190               0
```

- To stop a running container

  $ docker stop <ContainerID>

```
[redacted]@slaveDocker:~$ docker stop 24b56ea12763
24b56ea12763
```

Docker Containers

- To show the CPU and Memory utilization of the Container.

  $ docker stats <ContainerID>

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT
  MEM %               NET I/O             BLOCK I/O           PIDS
7419485dc84d        lucid_ganguly       0.00%               1.141MiB / 1.941GiB
  0.06%               3.09kB / 0B         0B / 0B             1
```

- To attach to a running container.

  $ docker attach <ContainerID>

```
[redacted]@slaveDocker:~$ docker attach 4844a06f8e08
[root@4844a06f8e08 /]#
```

Docker Containers

- To pause the processes in a running container.

  $ docker pause <ContainerID>

```
[redacted]@slaveDocker:~$ docker pause 4844a06f8e08
4844a06f8e08
vladimir@slaveDocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                  PORTS               NAMES
4844a06f8e08        centos              "/bin/bash"         7 minutes ago
Up 7 minutes (Paused)                       priceless_agnesi
```

- To unpause the processes in a running container.

  $ docker unpause <ContainerID>

```
[redacted]@slaveDocker:~$ docker unpause 4844a06f8e08
4844a06f8e08
```

Docker Containers

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- To kill the processes in a running container.

  $ docker kill <ContainerID>

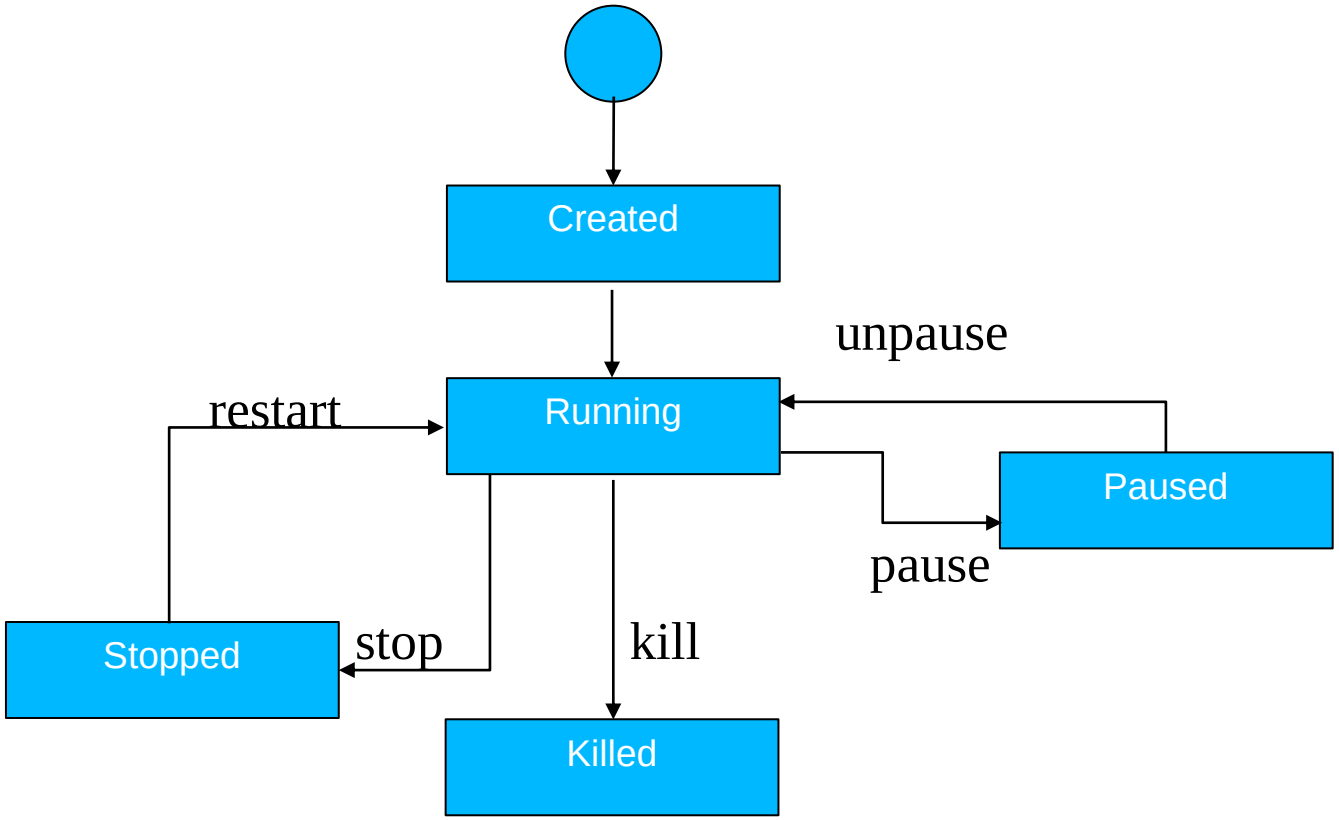
```
[redacted]@slaveDocker:~$ docker kill 0553addf9ee2
0553addf9ee2
```

- To restart the processes in a stopped container.

  $ docker restart <ContainerID>

```
[redacted]@slaveDocker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
      STATUS              PORTS               NAMES
d5be0492ed34        centos              "/bin/bash"         10 seconds ago
      Up 8 seconds                            jovial_lalande
[redacted]@slaveDocker:~$ docker stop d5be0492ed34
d5be0492ed34
vladimir@slaveDocker:~$ docker restart d5be0492ed34
d5be0492ed34
```

Docker Container - Lifecycle



Removing containers

- To remove all unused and exited containers.

  $ docker rm $(docker ps -aq)

- After removing all containers:

```
vladimir@slaveDocker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

Docker Daemon Process

Docker daemon process

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- To stop the Docker daemon process.

  $ sudo service docker stop

- To see the Docker daemon status:

  $ service docker status

```
[redacted]@slaveDocker:~$ service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2020-06-04 19:20:40 IDT; 4min 40s ago
     Docs: https://docs.docker.com
  Process: 1235 ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock (code=exited, status=0/SUCCESS)
 Main PID: 1235 (code=exited, status=0/SUCCESS)

Jun 04 17:36:33 slaveDocker dockerd[1235]: time="2020-06-04T17:36:33.248454409+03:00" level=info msg="Docker daemon" commit=afacb8b7f0 graphdriver(s)=overlay2 version=1
Jun 04 17:36:33 slaveDocker dockerd[1235]: time="2020-06-04T17:36:33.291246082+03:00" level=info msg="Daemon has completed initialization"
Jun 04 17:36:33 slaveDocker systemd[1]: Started Docker Application Container Engine.
Jun 04 17:36:33 slaveDocker dockerd[1235]: time="2020-06-04T17:36:33.744835736+03:00" level=info msg="API listen on /var/run/docker.sock"
Jun 04 19:20:29 slaveDocker systemd[1]: Stopping Docker Application Container Engine...
Jun 04 19:20:29 slaveDocker dockerd[1235]: time="2020-06-04T19:20:29.800328221+03:00" level=info msg="Processing signal 'terminated'"
Jun 04 19:20:39 slaveDocker dockerd[1235]: time="2020-06-04T19:20:39.808689921+03:00" level=info msg="Container 67c7975c2dd8ca3d15c5140594fa255fcd3ed9d79b898c6d1585048f
Jun 04 19:20:39 slaveDocker dockerd[1235]: time="2020-06-04T19:20:39.963503507+03:00" level=info msg="ignoring event" module=libcontainerd namespace=moby topic=/tasks/d
Jun 04 19:20:40 slaveDocker dockerd[1235]: time="2020-06-04T19:20:40.221560973+03:00" level=info msg="Daemon shutdown complete"
Jun 04 19:20:40 slaveDocker systemd[1]: Stopped Docker Application Container Engine.
lines 1-17/17 (END)
```

Docker daemon process

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- To start the Docker daemon process.

  $ sudo service docker start

- To see the Docker daemon status is **Running**:

  $ service docker status

```
[redacted]@slaveDocker:~$ service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-06-04 19:43:57 IDT; 6s ago
     Docs: https://docs.docker.com
 Main PID: 5454 (dockerd)
    Tasks: 9
   CGroup: /system.slice/docker.service
           └─5454 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.008025557+03:00" level=warning msg="Your kernel does not support cgroup rt runtime"
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.008131045+03:00" level=warning msg="Your kernel does not support cgroup blkio weight"
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.008237833+03:00" level=warning msg="Your kernel does not support cgroup blkio weight_device"
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.008476547+03:00" level=info msg="Loading containers: start."
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.160341034+03:00" level=info msg="Default bridge (docker0) is assigned with an IP address 172.17.0.0
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.190107698+03:00" level=info msg="Loading containers: done."
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.325835262+03:00" level=info msg="Docker daemon" commit=afacb8b7f0 graphdriver(s)=overlay2 version=1
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.326521980+03:00" level=info msg="Daemon has completed initialization"
Jun 04 19:43:57 slaveDocker systemd[1]: Started Docker Application Container Engine.
Jun 04 19:43:57 slaveDocker dockerd[5454]: time="2020-06-04T19:43:57.369072991+03:00" level=info msg="API listen on /var/run/docker.sock"
```

Docker - File

Docker - File

- Create a file with name Dockerfile

  $ touch Dockerfile

- Add to the file the following content:

```
#This is a sample Image
FROM ubuntu
MAINTAINER <your mail>

RUN apt-get update
RUN apt-get install -y nginx
CMD ["echo","Image created"]
```

Docker – File

- **#This is a sample Image** - is a comment
- **FROM** - from which base image you want to base your image from. In our example a new image will be created from **ubuntu** image.
- **MAINTAINER** - the person who is going to maintain this image.
- **RUN** - the command is used to run instructions against the image. In our case we update Ubuntu system and then install Nginx on our **ubuntu** image.
- **CMD** - the command is used to display a message to the user.

Docker – Build Image

Docker – Build image

- The method **“docker build”** allows to build our own images.

  $ docker build -t ImageName:TagName dir

- Arguments and flags description:

  -t - a tag to the image

  ImageName – This is the name you want to give to your image.

  TagName – This is the tag you want to give to your image.

  Dir – The directory where the Docker File is present.

  $ docker build -t myfirstimage:1.0.1 .

  repository name must be lowercase!

Docker – Build image

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Now you can find the new created image

  $ docker images

```
[redacted]@slaveDocker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
myfirstimage        1.0.1               a2d50f87c47e        10 seconds ago      155MB
ubuntu              latest              1d622ef86b13        4 weeks ago         73.9MB
centos              latest              470671670cac        4 months ago        237MB
hello-world         latest              bf756fb1ae65        4 months ago        13.3kB
jenkins             latest              cd14cecfdb3a        22 months ago       696MB
```

Docker – Push Images

Docker – Push images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- You can upload the image **myfirstimage:1.0.1** to a public repository. In our case is Docker Hub. Below are steps:

  1) Log in to Docker Hub and create a new repository

  2) Pull the new created repository

  $ docker pull <repositoryName>

  If you see an error it is OK because your repository still empty.

```
[redacted]r@slaveDocker:~/.docker$ docker pull l[redacted]/testrepo
Using default tag: latest
Error response from daemon: manifest for [redacted]/testrepo:latest not found: manifest unknown: manifest unknown
```

  3) Login into the Docker Hub repository from the command prompt.

  $ docker login

  4) We need to tag our image to the new repository created in Docker Hub

  $ docker tag <imageID> <repositoryName>

Docker – Push images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

Here you can find your new taged image **with the same imageID**.

```
[redacted]r@slaveDocker:~/.docker$ docker tag a2d50f87c47e [redacted]/testrepo:1.0.1
[redacted]r@slaveDocker:~/.docker$ docker images | grep a2d50f87c47e
[redacted]/testrepo        1.0.1              a2d50f87c47e        3 hours ago         155MB
myfirstimage               1.0.1              a2d50f87c47e        3 hours ago         155MB
```

5) Now you can push your docker image to Docker Hub.

$ docker push <repositoryname>

```
[redacted]r@slaveDocker:~/.docker$ docker push [redacted]/testrepo:1.0.1
The push refers to repository [docker.io/[redacted]/testrepo]
5773dc7fe390: Pushed
eb2ca6733fd5: Pushed
8891751e0a17: Mounted from [redacted]/docker_course
2a19bd70fcd4: Mounted from [redacted]/docker_course
9e53fd489559: Mounted from [redacted]/docker_course
7789f1a3d4e9: Mounted from [redacted]/docker_course
1.0.1: digest: sha256:8d06b9e4c30ced708cfe660a0bfa582d065fb30fde01c3b885bc2ba2d5f627bb size: 1576
```

Docker – Push images

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

You have published the first docker image to Docker Hub.

[redacted] / testrepo

This repository does not have a description

Last pushed: 7 minutes ago

Tags

This repository contains 1 tag(s).

1.0.1 7 minutes ago

See all

Docker – Managing Ports

Docker – Managing Ports

- In Docker, the containers themselves can have applications running on ports. When you run a container, if you want to access the application in the container via a port number, you need to map the port number of the container to the port number of the Docker host.
- Let's pull the image **Jenkins** from Docker Hub

  $ docker pull Jenkins
- To understand what ports are exposed by the container, you should use the **docker inspect** command to inspect the image.

  $ docker inspect <container / image>

  $ docker inspect jenkins

Docker – Managing Ports

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- The output of the **inspect** command gives a JSON output. Here is a section **ExposedPorts** and you can find all mentioned ports. One is the data port of **8080** and the other is the control port of **50000**.


```
    },
    "DockerVersion": "17.06.2-ce",
    "Author": "",
    "Config": {
        "Hostname": "",
        "Domainname": "",
        "User": "jenkins",
        "AttachStdin": false,
        "AttachStdout": false,
        "AttachStderr": false,
        "ExposedPorts": {
            "50000/tcp": {},
            "8080/tcp": {}
        },
        "Tty": false,
        "OpenStdin": false,
        "StdinOnce": false,
        "Env": [
```

Docker – Managing Ports

- To run Jenkins and to map ports:

  $ docker run -p 8080:8080 -p 50000:50000 jenkins
- **8080:8080** – Data Port Mapping
- **50000:50000** – Control Port Mapping
- Left-hand side: Docker Host port number (your VM)
- Right-hand side: Docker container port number
- When you open the browser and navigate to the Docker host on port 8080, you will see Jenkins up and running.

Docker – Build a Web Server

Build a Web Server

- We will use the Apache Web Server on Ubuntu to build our image. First we need to build our **Dockerfile**

```
FROM ubuntu

ENV TZ=Asia/Jerusalem
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone

RUN apt update
RUN apt install apache2 -y
RUN apt install apache2-utils -y
RUN apt clean
EXPOSE 80
CMD ["apache2ctl", "-D", "FOREGROUND"]
```

- We need to update time zone is required for Apache installation.
- We need to update Ubuntu and then to install Apache2 and Apache2 packages.

Build a Web Server

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- We have to clean any unnecessary files from the system.
- **EXPOSE** - is used to expose port 80 of Apache in the container.
- **CMD** - to run apache2 in the background. **–D** – detached mode
- Run the Docker build command

  $ docker build -t="myfirstwebserver" .

```
Step 9/9 : CMD ["apache2ctl", "-D", "FOREGROUND"]
 ---> Running in 9dd28f1a3423
Removing intermediate container 9dd28f1a3423
 ---> d6081f530007
Successfully built d6081f530007
Successfully tagged myfirstwebserver:latest
```

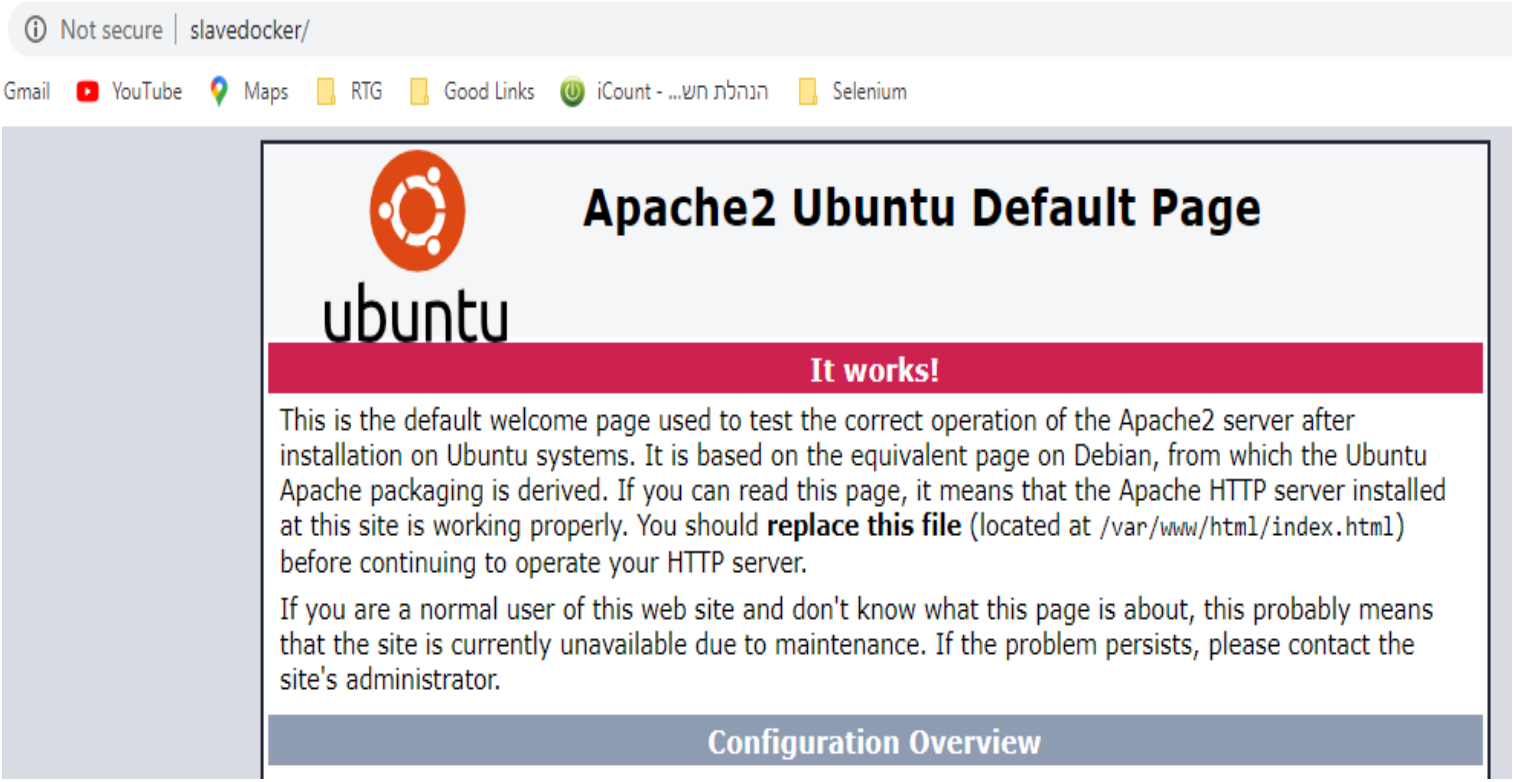
- Create a container from the image.

  $ docker run -d -p 80:80 myfirstwebserver

Build a Web Server

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Browse your VM host with port 80 and see that Apache is up.



Build a Web Server

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Now let's push the image to Docker Hub.

  $ docker tag myfirstwebserver:latest <reponame>/myfirstwebserver:latest

  $ docker push <reponame>/myfirstwebserver:latest

```
[redacted]@slaveDocker:~$ docker push levintovich/myfirstwebserver:latest
The push refers to repository [docker.io/levintovich/myfirstwebserver]
4e472d07f310: Pushed
bcd24cb26719: Pushed
b7e9ac1fe8ea: Pushed
64af10b622ca: Pushed
ad4a7e7214d3: Pushed
889175le0a17: Mounted from levintovich/testrepo
2a19bd70fcd4: Mounted from levintovich/testrepo
9e53fd489559: Mounted from levintovich/testrepo
7789fla3d4e9: Mounted from levintovich/testrepo
latest: digest: sha256:50422b3eb920f7df6447bc98cda9212577c39d47386b69c4bbb497e31f47439c size: 2197
```

levintovich / **myfirstwebserver**
Updated 15 minutes ago

0 1 PUBLIC

Building a Web Site as a container

Build a Website

- Now let's build a new image with our own website.

  $ mkdir –p website

  $ cd website

  $ vim index.html

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Website</h1>

<p>Hello World!</p>

</body>
</html>
```

Build a Website

- Now let's build Dockerfile.

  $ vim Dockerfile

```
FROM levintovich/myfirstwebserver:latest

COPY ./index.html /var/www/html/
```

  $ docker build -t="mywebsite:latest" .

  $ docker run -d -p 80:80 mywebsite:latest

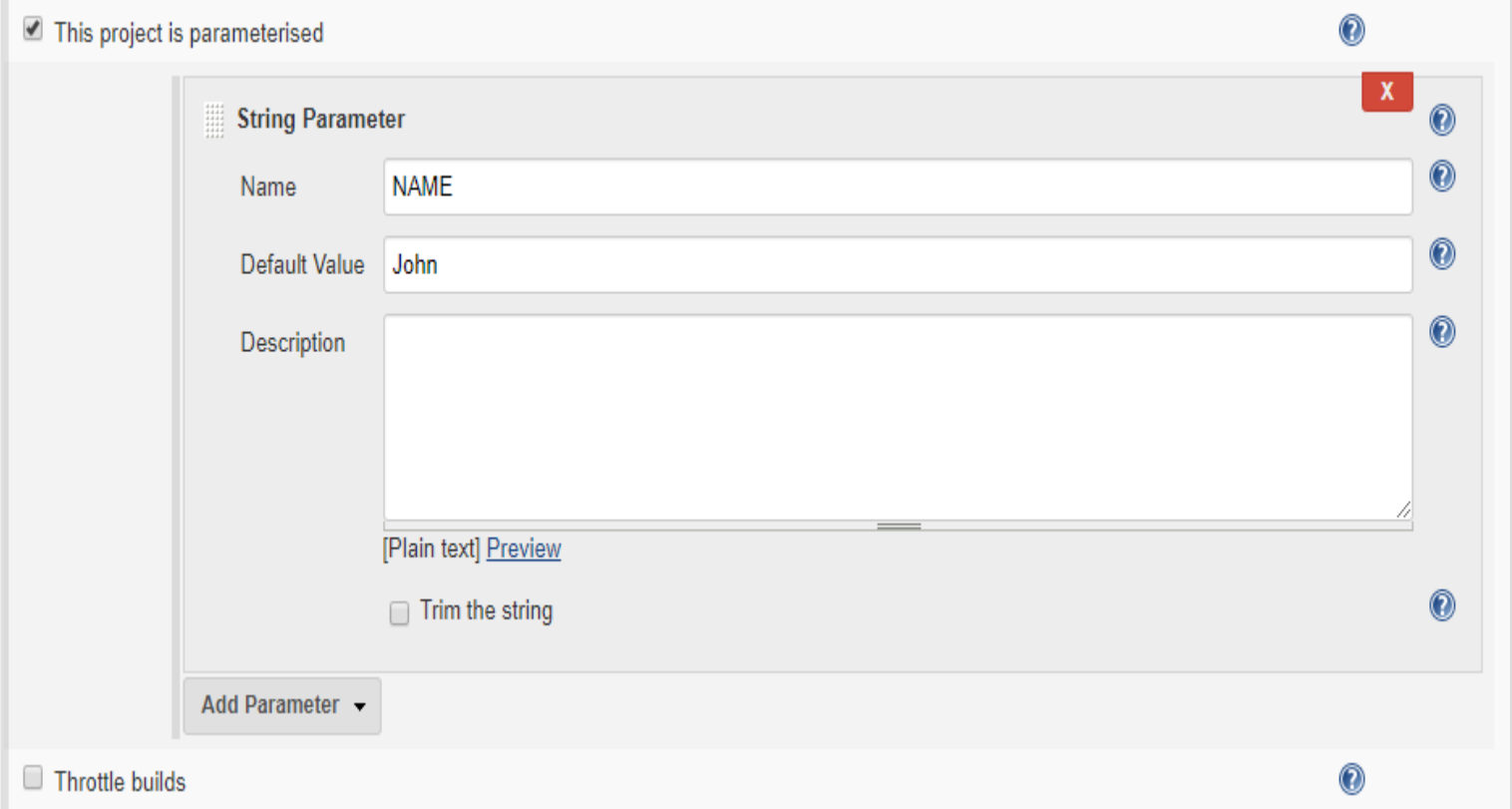
- For debugging Apache webserver with your website use:

  $ docker exec -it <containerID> bash

CI-CD with docker

CI-CD with docker

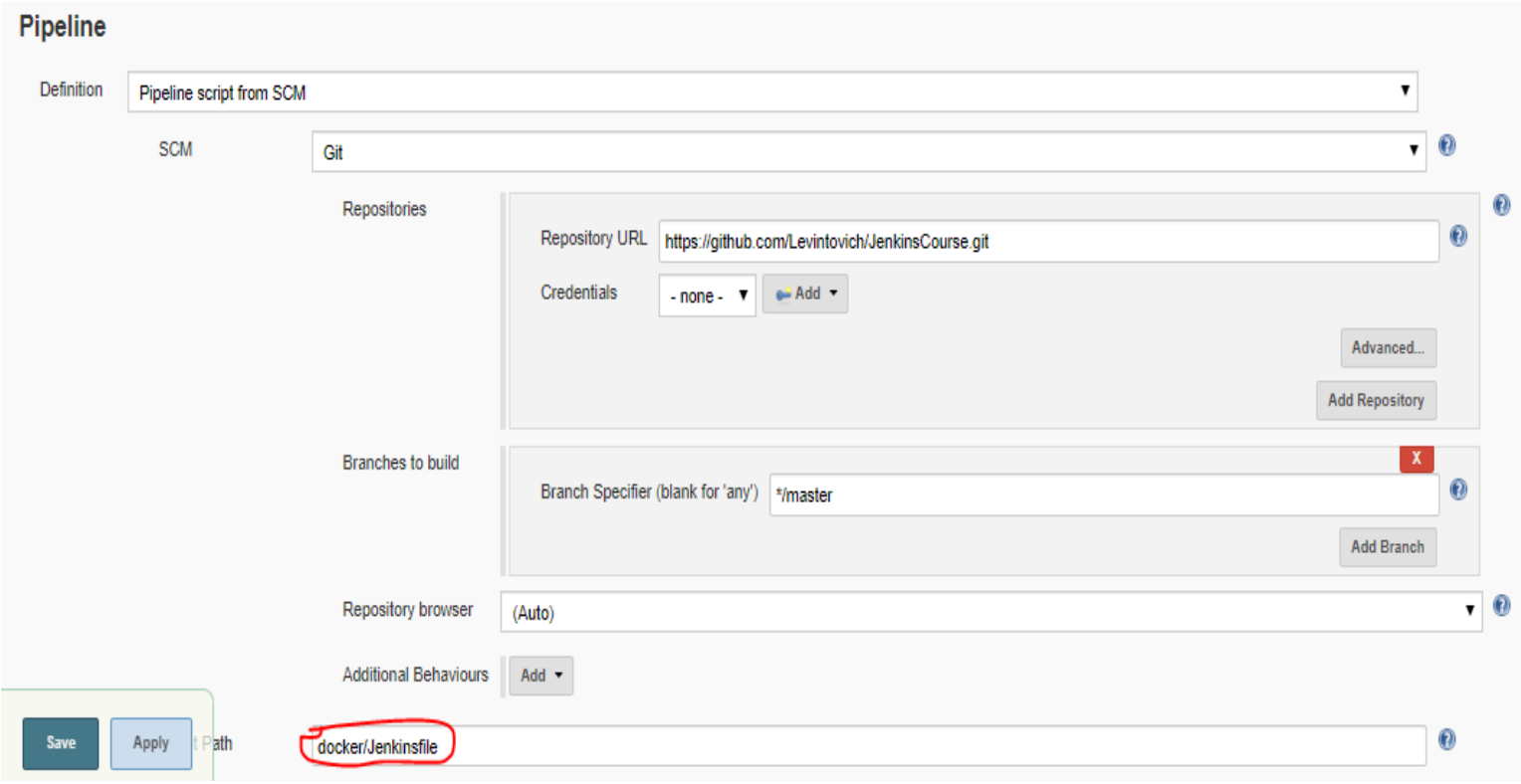
- Go to Jenkins server and create a pipeline



CI-CD with docker

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Add SCM



CI-CD with docker

- Run the build

Stage View

	Declarative: Checkout SCM	Clone Sources	Checking environment	Prepare sources of the website	Build a docker image	Push a docker image	Deploy the website
Average stage times: (Average full run time: ~30s)	2s	924ms	595ms	530ms	531ms	10s	10s
#8 May 30 21:52 No Changes	950ms	957ms	386ms	347ms	608ms	17s	12s

- Find new docker images at the Git Hub repository.

CI-CD with docker

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Find new docker images at the Git Hub repository.

levintovich / mywebsite

This repository does not have a description

Last pushed: a day ago

Tags

This repository contains 6 tag(s).

3		a day ago
7		a day ago

CI-CD Secured

CI-CD Secured

- We need to push docker images automatically without entering a password. Also we don't want to reveal the password to everyone.
- Run the following command and see that your pipeline is failed after.

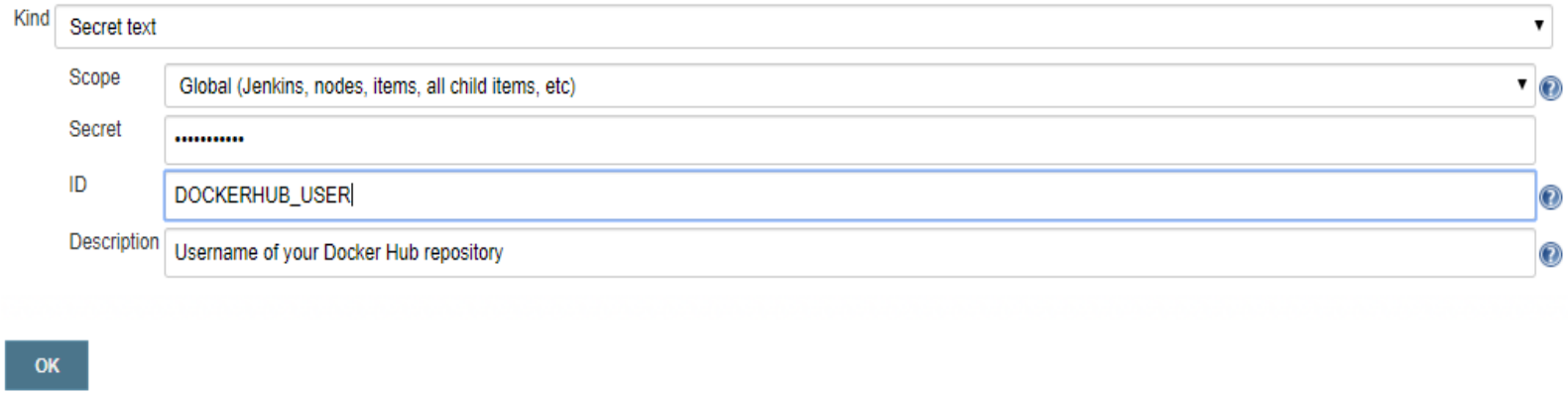
  $ docker logout
- That means you are not able to push docker images without logging in to your Docker Hub repository.
- Go to **Jenkins > Credentials > Global > Add Credentials**
- Create 2 credentials:

  DOCKERHUB_USER

  DOCKERHUB_PASSWORD

CI-CD Secured

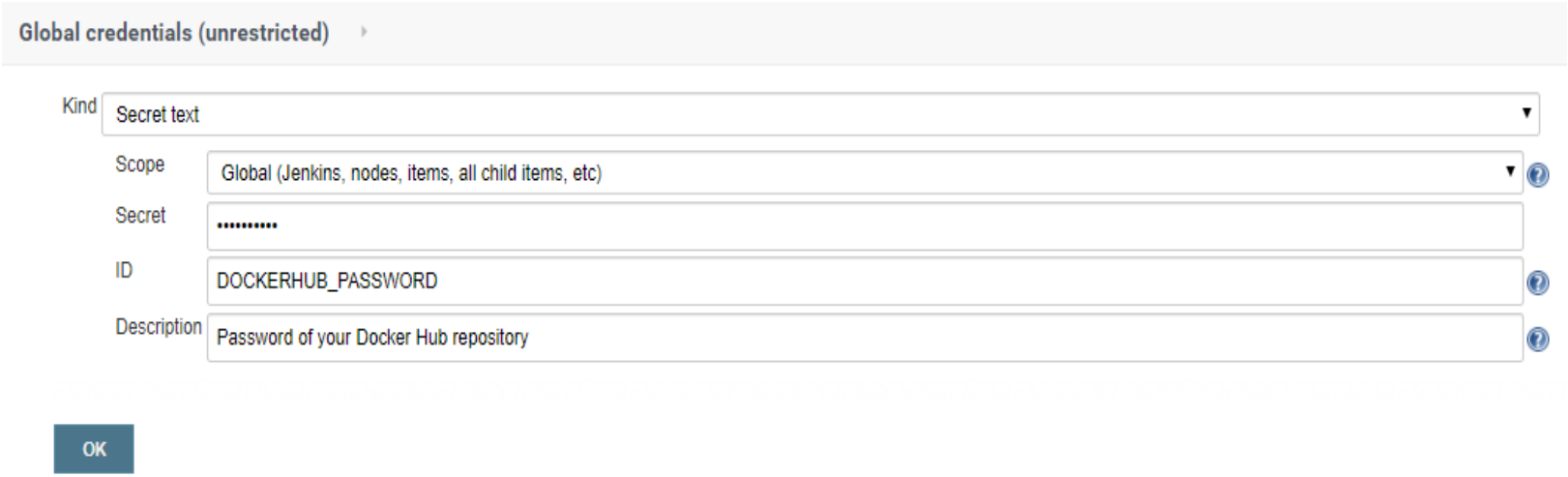
- Create **DOCKERHUB_USER**



CI-CD Secured

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Create **DOCKERHUB_PASSWORD**



Extra – Instruction Commands

Instruction Commands

- Docker has a host of instruction commands. These are commands that are put in the Docker File.
- **CMD** - is used to execute a command at runtime when the container is executed.

  $ CMD [command, param1, param2, …]

  $ CMD ["python", "./checkUserName.py", "John"]
- **WORKDIR** - is used to set the working directory of the container.

  $ WORKDIR <directory_name>

  $ WORKDIR container_scripts
- **ENV** - is used to set environment variables in the container.

  $ ENV <key1=value1> <key2=value2> …..

  $ ENV PLACE=RTG COURSE=DOCKER

Instruction Commands

- Find the file at https://github.com/Levintovich/JenkinsCourse/blob/master/docker/Dockerfile_python

```
FROM ubuntu:18.04

RUN apt update
RUN apt install python -y

WORKDIR container_scripts

COPY scripts/checkUserName.py .

ENV PLACE=RTG COURSE=DOCKER

CMD env
CMD ["python", "./checkUserName.py", "John"]

```

Instruction Commands

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- Also you can build the docker image using another option

  $ git clone https://github.com/Levintovich/JenkinsCourse.git

  $ cd JenkinsCourse

  $ docker build -t pytonapp -f ./docker/Dockerfile_python .

- Run a new Docker container

  $ docker run pytonapp

```
vladimir@slaveDocker:~/Documents/JenkinsCourse$ docker run pytonapp
Access denied
vladimir@slaveDocker:~/Documents/JenkinsCourse$ docker run -it pytonapp bash
root@129b330dd611:/container_scripts# env
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;
32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01
;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.t
z=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab
=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*
.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;3
5:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;
35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:
*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;
36:
PLACE=RTG
HOSTNAME=129b330dd611
COURSE=DOCKER
PWD=/container_scripts
```

Instruction Commands

מרכז להכשרה מקצועית והשמה בתעשיית ההייטק

- **ENTRYPOINT** – is used to execute commands at runtime for the container by entering input arguments.

  ENTRYPOINT [command, param1, param2, …]

  ENTRYPOINT ["python", "./checkUserName.py"]

- Build a new docker image from GitHub.

  $ docker build -t pytonapp_entry_point -f ./docker/Dockerfile_entry_point .

- Run a new docker container with different argument's values.

  $ docker run pytonapp_entry_point John

```
[redacted]@slaveDocker:~/Documents/JenkinsCourse$ docker run pytonapp_entry_point John
Access denied
[redacted]@slaveDocker:~/Documents/JenkinsCourse$ docker run pytonapp_entry_point Jack
Access granted
[redacted]@slaveDocker:~/Documents/JenkinsCourse$ docker run pytonapp_entry_point Jill
Welcome to the system
```

Data Volumes

- **Data Volume** – is the separate volume that can shared across containers. All data exists even if a docker containers will be deleted.
- Create a data volume location on your Docker host.

  $ mkdir -p ${HOME}/Documents/volume

- Pull the image for creation a website and run a new container using **data volume.**

  $ docker pull levintovich/mywebsite:latest

  $ docker run -d -v ${HOME}/Documents/volume:/var/www/html/ -p 80:80 levintovich/mywebsite:latest

- Copy **index.html** to your data volume location.
- Working on your docker host you can modify **index.html** and also add other files and your website will be changed.

Hope you like it ;)