

## Setting up a Docker instance

Type the following command to check the docker version installed on lab:

### docker version

```
root@ip-172-31-17-73:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (18.09.7-0ubuntu1~18.04.3).
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.2-0
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ip-172-31-17-73:~# docker version
Client:
 Version:           18.09.7
 API version:       1.39
 Go version:        go1.10.1
 Git commit:        2d0083d
 Built:             Wed Jul  3 12:13:59 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.09.7
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.1
  Git commit:       2d0083d
  Built:            Mon Jul  1 19:31:12 2019
  OS/Arch:          linux/amd64
  Experimental:     false
root@ip-172-31-17-73:~#
```

Building a custom Docker image to be deployed

- First, clone the Git repository on Docker host using the command below:

```
git clone https://github.com /Docker.git
```

- Run with docker build command to build a custom Docker image

```
cd Docker
```

## `docker build -t phpcode . -f Dockerfile`

```
root@docker:~/Docker# docker build -t phpcode .
Sending build context to Docker daemon 337.9kB
Step 1/14 : FROM ubuntu
---> 93fd78260bd1
Step 2/14 : ENV DEBIAN_FRONTEND=non-interactive
---> Using cache
---> b21eb69f632a
Step 3/14 : RUN apt-get update -y
---> Using cache
---> d2e4866734b9
Step 4/14 : RUN apt-get install -y git curl apache2 php libapache2-mod-php php-mysql
---> Using cache
---> 85f084edfc0b
Step 5/14 : RUN rm -rf /var/www/html/*
---> Using cache
---> b56166da0f16
Step 6/14 : ADD src /var/www/html/
---> Using cache
---> ba9e5c5c651c
Step 7/14 : RUN a2enmod rewrite
---> Using cache
---> cff3e4bb8c42
Step 8/14 : RUN chown -R www-data:www-data /var/www/html
---> Using cache
---> 7a4314c7b69b
Step 9/14 : ENV APACHE_RUN_DIR /var/www/html
---> Using cache
---> 663a68663f90
```

- Once the image is built, check if it is built properly or not. You can see a Docker image entry using Docker images command

```
Removing intermediate container 66720df3cf7e
---> b914fd976a06
Successfully built b914fd976a06
Successfully tagged phpcode:latest
root@ip-172-31-17-73:~/Docker# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
phpcode              latest              b914fd976a06        3 minutes ago      251MB
ubuntu               latest              4c108a37151f        2 weeks ago        64.2MB
root@ip-172-31-17-73:~/Docker#
```

Initializing a Docker swarm cluster and deploying a container to the cluster

- First, we need to initialize Docker swarm using the set of commands given below:

## docker swarm init

## docker node ls

```
root@ip-172-31-17-73:~# docker swarm init
Swarm initialized: current node (rv82bet8lvwyaic8rl8y3mu0n) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3c83c3x6plrnz8f8i89lp34s8lm6lpd2uy7shn3exovkld8a4y-lpeywlF7d4y0Jy283d5cqeazh 172.31.17.73:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@ip-172-31-17-73:~# docker node ls
ID                                HOSTNAME                STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
rv82bet8lvwyaic8rl8y3mu0n *      ip-172-31-17-73        Ready     Active           Leader             18.09.7
root@ip-172-31-17-73:~#
```

- Once the node is configured, deploy the custom Docker image on the Docker swarm cluster following the process shown below

## docker service create -p 80:80 --name webserver phpcode

## docker service ls

## curl localhost

```
root@ip-172-31-17-73:~# docker service create -p 80:80 --name webserver phpcode
image phpcode:latest could not be accessed on a registry to record
its digest. Each node will access phpcode:latest independently,
possibly leading to different nodes running different
versions of the image.

plsows6zdl801fr36ld9533uv
overall progress: 1 out of 1 tasks
1/1: running [=====>]
verify: Service converged
root@ip-172-31-17-73:~# docker service ls
ID                NAME           MODE           REPLICAS    IMAGE           PORTS
plsows6zdl80     webserver      replicated      1/1          phpcode:latest  *:80->80/tcp
root@ip-172-31-17-73:~# curl localhost
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>Simple PHP App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="assets/css/bootstrap.min.css" rel="stylesheet">
    <style>body {margin-top: 40px; background-color: #333;}</style>
    <link href="assets/css/bootstrap-responsive.min.css" rel="stylesheet">
    <!--[if lt IE 9]><script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]-->
  </head>

  <body>
    <div class="container">
      <div class="hero-unit">
        <h1>Simple PHP App</h1>
        <h2>Congratulations</h2>
        <p>Your PHP application is now running on a container in Amazon ECS.</p>
        <p>The Kubernetes Docker container is running PHP version 7.2.19-0ubuntu0.18.04.1.</p>
      </div>
    </div>
  </body>
</html>
root@ip-172-31-17-73:~#
```