**Installation and configuration of Docker on Ubuntu 14.04**

Create one Ubuntu 14.04 instance on AWS environment

**Installing Docker:** Run the following commands on terminal to install the docker.

# sudo apt-get update

# sudo apt-get install apt-transport-https ca-certificates

# sudo apt-get install linux-image-extra-`uname -r`

# sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D

# sudo sh -c "echo 'deb https://apt.dockerproject.org/repo ubuntu-'$(lsb\_release -cs)' main' >> /etc/apt/sources.list.d/docker.list"

# sudo apt-get update

# sudo apt-get install docker-engine

Run following command to verify the successful installation of docker

# sudo docker info

**Fowarding Traffic:** We need to foward all traffic to docker which we be required as we are running apache web server inside docker container.Enable forwarding with UFW(Uncomplicated Firewall):

# sudo view /etc/default/ufw

Replace “DEFAULT\_FORWARD\_POLICY=”DROP”” with “DEFAULT\_FORWARD\_POLICY=”ACCEPT” and save the file.

**Create Private network**

**#** docker network create --subnet=172.19.0.0/16 mynet

**Search Docker Images:** We would need to download Ubuntu image on host Ubuntu system on which we would install Apache web server. Varoius customize docker images are available on Docker repository which could be searched using the following command. This command would search and list all images having Ubuntu string in their name.

# sudo docker search ubuntu

**Download Ubuntu Image:** This command would download Ubuntu version 14.04 image.

# sudo docker pull ubuntu:trusty

We can get list of all available images on our local system using:

# sudo docker images

Docker container would be active until the command we ran is active. We can check the status of the exited/running containers using:

# sudo docker ps

**Create web server docker container and install wordpress**

This commands would create container name “WebServer” and map the ports.

# sudo docker run --name webserver --net mynet --ip 172.19.0.2 -h web1.example.com -p 80:80 -t -i ubuntu:trusty /bin/bash

-p: maps the host’s 80 port with 80 port of container

Now we are inside the container.

**This web server container IP Address: 172.19.0.2**

# sudo apt-get update

Press Ctrl+p and Ctrl+q to come out of the container. This would keep the container running behind. We can verify it using

# sudo docker ps –a

The status WebServer container is “Up” and is running

If we again need to make some changes inside this container, we need to attach to this container using docker attach command.

# sudo docker attach WebServer

Now we are inside the container and need to install apache and PHP

# sudo apt-get -y install apache2 php5-mysql php5 libapache2-mod-php5 php5-mcrypt

Configure Apache

Open ports.conf file and configure

# sudo view /etc/apache2/ports.conf

Find the line that says Listen 80, and add change like bellow.

Listen \*:80

Save and exit.

Restart Apache to put the changes into effect:

# sudo service apache2 restart

# sudo update-rc.d apache2 enable

**Download and Configure Application**

In our example, we are using WordPress as our application.

Download the WordPress archive:

# cd ~

# sudo apt-get install wget mysql-client nmap -y

# wget http://wordpress.org/latest.tar.gz

Extract the WordPress archive:

# tar xvf latest.tar.gz

Change to the extracted directory:

# cd wordpress

WordPress needs a directory to be created for uploads, wp-content/uploads. Let's do that now:

# sudo mkdir wp-content/uploads

We will use the sample WordPress configuration file as a template. Copy it to the proper location:

# cp wp-config-sample.php wp-config.php

Now open the configuration file for editing:

# sudo view wp-config.php

Configure the WordPress database connection by changing the highlighted information in the following lines:

wp-config.php

/\*\* The name of the database for WordPress \*/

define('DB\_NAME', 'app');

/\*\* MySQL database username \*/

define('DB\_USER', 'appuser');

/\*\* MySQL database password \*/

define('DB\_PASSWORD', 'Appuserpw');

/\*\* MySQL hostname \*/

define('DB\_HOST', '172.19.0.3');

Save and exit.

WordPress is now configured, but its files must be copied to the proper location to be served by our web server software.

### Copy Application Files to Document Root

Now that we have our application configured, we need to copy it into Apache's document root, where it can be served to visitors of our website.

The default location of Apache's DocumentRoot is /var/www/html, so we will use that in our example.

First, delete the default index.html file:

# sudo rm /var/www/html/index.html

Then use rsync to copy the WordPress files to /var/www/html, and make www-data (the user that Apache runs as) the owner:

# sudo apt-get install rsync -y

# sudo rsync -avP ~/wordpress/ /var/www/html

# sudo chgrp -R www-data /var/www/html/\*

Our web server is ready with wordpress.

**Create Mysql docker container on this same host**

This commands would create container name “WebServer” and map the ports.

# sudo docker run --name Mysql --net mynet --ip 172.19.0.3 -h mysql.example.com -p 3306:3306 -t -i ubuntu:trusty /bin/bash

-p: maps the host’s 3306 port with 3306 port of container

**This mysql server container IP is: 172.19.0.3**

Now we are inside the container.

# sudo apt-get update

# sudo apt-get -y install mysql-server

Start Mysql server

# sudo /etc/init.d/mysql start

# sudo update-rc.d mysql enable

# sudo mysql\_install\_db

# sudo mysql\_secure\_installation

root@mysql:/# sudo mysql\_secure\_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL

SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current

password for the root user. If you've just installed MySQL, and

you haven't set the root password yet, the password will be blank,

so you should just press enter here.

Enter current password for root (enter for none):

OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL

root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n

... skipping.

By default, a MySQL installation has an anonymous user, allowing anyone

to log into MySQL without having to have a user account created for

them. This is intended only for testing, and to make the installation

go a bit smoother. You should remove them before moving into a

production environment.

Remove anonymous users? [Y/n] y

... Success!

Normally, root should only be allowed to connect from 'localhost'. This

ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y

... Success!

By default, MySQL comes with a database named 'test' that anyone can

access. This is also intended only for testing, and should be removed

before moving into a production environment.

Remove test database and access to it? [Y/n] y

- Dropping test database...

ERROR 1008 (HY000) at line 1: Can't drop database 'test'; database doesn't exist

... Failed! Not critical, keep moving...

- Removing privileges on test database...

... Success!

Reloading the privilege tables will ensure that all changes made so far

will take effect immediately.

Reload privilege tables now? [Y/n] y

... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MySQL

installation should now be secure.

Thanks for using MySQL!

You will have to enter the MySQL administrator's password that you set in the steps above. Afterwards, it will ask if you want to change that password. Type "N" for no if you're happy with your current password. Answer the rest of the questions with the defaults.

### Configure MySQL to Listen on Private Network Interface

Open the MySQL configuration file:

# sudo view /etc/mysql/my.cnf

Find the bind-address setting, and change it to the address of the private network address of your database server:

**/etc/mysql/my.cnf**

bind-address = 172.19.0.3 #this is mysql container IP

Save and exit.

Restart MySQL:

# sudo service mysql restart

### Set Up Database and Database Users

Now we need to create the database and database users that the application servers will use to connect.

Enter the MySQL console:

# mysql -u root -p

Enter the MySQL root password at the prompt.

At the MySQL prompt, create the database for your application:

mysql> CREATE DATABASE app;

Create a database user, "appuser" in our example, that can be connected to from private network address of your application server.

mysql> CREATE USER 'appuser'@'172.19.0.2' IDENTIFIED BY 'Appuserpw';

Give **appuser** full control over the **app** database:

mysql> GRANT ALL PRIVILEGES ON app.\* TO 'appuser'@'172.19.0.2';

mysql> FLUSH PRIVILEGES;

These relaxed privileges ensure that the application's installer will be able to install the application in the database. If you have more than two application servers, you should create all the necessary database users now.

Now grant permissions to nagios user to get data for monitoring

mysql> GRANT PROCESS, USAGE, SELECT, REPLICATION CLIENT ON \*.\* TO nagios@'%' IDENTIFIED BY 'Nagiospw' WITH GRANT OPTION;

mysql> FLUSH PRIVILEGES;

Exit the MySQL prompt now:

**mysql>** exit

The database server setup is complete.

Now run this URL in browser to install wordpress

**http://52.37.92.91/wp-admin/install.php <Public IP address of docker container>**

This will ask some details like Site Title, Username, Password, Your email,

Once you fill those detail click on “Install Wordpress” button

Username: wpadmin

Password: (6W&Xidq7ZxIxyh5&d

#bWFo9Qb8ma4Wsw6kP

**Create Logrotation on both the containers**

Update apache2 logrotation file with bellow details

#cat /etc/logrotate.d/apache2

/var/log/apache2/\*.log {

daily

missingok

dateext

rotate 7

nocompress

delaycompress

notifempty

create 640 root adm

sharedscripts

postrotate

if /etc/init.d/apache2 status > /dev/null ; then \

/etc/init.d/apache2 reload > /dev/null; \

fi;

endscript

prerotate

if [ -d /etc/logrotate.d/httpd-prerotate ]; then \

run-parts /etc/logrotate.d/httpd-prerotate; \

fi; \

endscript

}

**Copy all rotated logs to s3 bucket**

# view /usr/local/bin/logcopytos3

Add bellow content to this file

==================================================

#!/bin/bash

#send logs to S3 bucket

date=$(date +%Y%m%d)

/usr/bin/s3cmd put /var/log/apache2/\*.log-${date} s3://logsmysqlwordpress/weblogs

==================================================

# sudo chmod +x /usr/local/bin/logcopytos3

Create Cron job for log rotation and sending these logs to s3 daily

# sudo crontab –e

**Add below details**

======================================

#Logrotation at 6.55 PM everyday

55 18 \* \* \* /usr/sbin/logrotate /etc/logrotate.d/apache2 > /dev/null 2>&1

#Transfer logs to S3 everyday at 7PM

0 19 \* \* \* /usr/local/bin/logcopytos3 > /dev/null 2>&1

Save and exit

**We can check installed cronjobs using bellow command**

# sudo crontab –l

**Backup apache2 folder to S3**

# /usr/bin/s3cmd -r put /etc/apache2 s3://logsmysqlwordpress/

# sudo crontab –e

Add this in last

=================================================

#Cronjob to sync apache2 folder everyday at 5AM

0 5 \* \* \* /usr/bin/s3cmd -r sync /etc/apache2 s3://logsmysqlwordpress/ > /dev/null 2>&1

=================================================

Save & Exit

Login to Mysql container

# docker attach Mysql

Create database backup script

# mkdir /etc/mysqlbackups/

# view /usr/local/bin/database\_backup

Add bellow content to this file

=====================================================

#!/bin/bash

ydate=$(date --date="1 days ago" +%Y%m%d)

date=$(date +%Y%m%d)

file="/etc/mysqlbackups/wordpress\_databases.sql-${ydate}"

DIRECTORY="/etc/mysqlbackups/"

if [ -d "$DIRECTORY" ]; then

/usr/bin/mysqldump -u root --events --ignore-table=mysql.event --all-databases > /etc/mysqlbackups/wordpress\_databases.sql-${date}

else

echo "/etc/mysqlbackups directory does not exist please create"

fi

if [ -f "$file" ]

then

/bin/rm -rf /etc/mysqlbackups/wordpress\_databases.sql-${ydate}

else

echo "wordpress\_databases.sql-${ydate} not found"

fi

/usr/bin/s3cmd put /etc/mysqlbackups/wordpress\_databases.sql-${date} s3://logsmysqlwordpress/

===============================================================

# sudo chmod +x /usr/local/bin/database\_backup

**Provide mysql root password from different file**

# sudo view ~/.my.cnf

Add bellow content to this file

=================================

[mysqldump]

user = root

password = Wordpress

=================================

Save and exit

# chmod 600 ~/.my.cnf

Create database backup and send it to s3 daily at 6 AM using below cron job

# sudo crontab –e

Add below content at end

#Create database backup and send it to s3 daily at 6 AM

0 6 \* \* \* /bin/sh /usr/local/bin/database\_backup > /dev/null 2>&1