

Instruction for deploying Wordpress with MySQL on virtual server

For example, in this instruction i will use Microsoft Azure virtual server with 1 CPU and 1 GB of RAM which will run on the Ubuntu Server 16.04 LTS. All actions are performed from an account that has superuser privileges.

First of all, we will update the packages and install the “nano” text editor:

```
sudo apt-get update && sudo apt-get install -y nano
```

After that, we start install Docker CE - shell script will make all automatically

```
wget -qO- https://get.docker.com/ | incl
```

Also, you need to install Docker Compose - a tool by which we can manage multiple configured containers

```
sudo curl -L
```

```
https://github.com/docker/compose/releases/download/1.20.1/docker-compose-`  
uname -s` - `uname -m` -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

If there were no problems during the installation, check the version of the installed software:

```
rmstr@testVM:~$ docker -v  
Docker version 18.03.0-ce, build 0520e24  
rmstr@testVM:~$ docker-compose -v  
docker-compose version 1.20.1, build 5d8c71b
```

The working directory will be considered /home/rmstr, all commands will be executed from this folder.

Create several matching configuration files and folders:

```
mkdir db-data  
mkdir logs/mysql  
mkdir logs/mysql/log  
mkdir logs/mysql/var  
mkdir logs/nginx  
mkdir nginx  
mkdir wordpress  
touch docker-compose.yml  
touch nginx/default.conf  
touch nginx/wordpress.conf
```

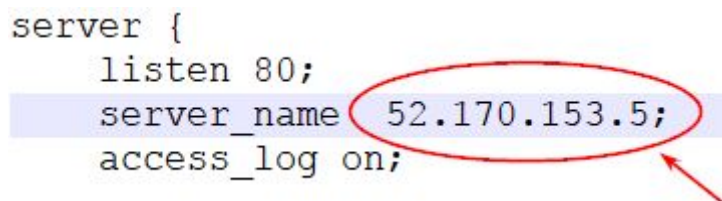
Edit and save wordpress.conf and default.conf (in the GitHub repository called nginx.conf) files using the commands:

```
sudo nano nginx / wordpress.conf
```

```
sudo nano nginx / default.conf
```

Additionally, you need to change the contents of the variable server_name to IP address of your server:

```
server {  
    listen 80;  
    server_name 52.170.153.5;  
    access_log on;
```



Open docker-compose.yml by command

```
sudo nano docker-compose.yml
```

and fill out its contents according to the requirements - the latest versions of mysql and nginx, WordPress 4.9.3.

If everything is configured, files and folders are in right place - we will launch a setup of our system using the command

```
sudo docker-compose up-d
```

The -d key means that the setup will take place in the background, and after successful installation we have 3 containers:

```
Creating rmstr_mysql_1 ... done  
Creating rmstr_wordpress_1 ... done  
Creating rmstr_nginx_1 ... done
```

Also, 3 docker images was downloaded:

```
rmstr@testVM:~$ sudo docker images
```

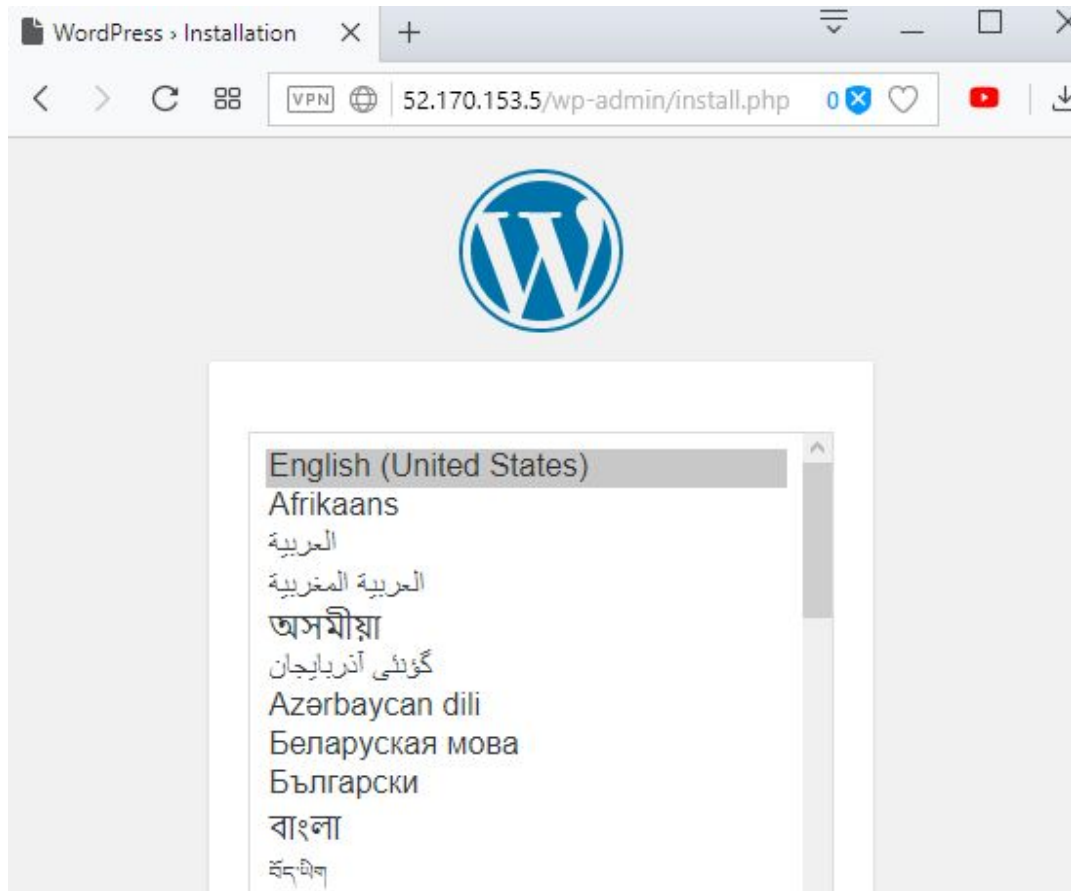
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	7f70b30f2cc6	4 days ago	109MB
mysql	latest	5195076672a7	12 days ago	371MB
wordpress	4.9.3	02d7de3efc5a	6 weeks ago	410MB

As we can see, the containers started automatically:

```
rmstr@testVM:~$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
ae6274e4aab0	nginx:latest	"nginx -g 'daemon of..."	7 minutes ago	Up 7 minutes
bfc29c44526	wordpress:4.9.3	"docker-entrypoint.s..."	7 minutes ago	Up 4 minutes
f5171c829707	mysql:latest	"docker-entrypoint.s..."	7 minutes ago	Up 7 minutes

After we enter the IP address of our server IP into browser, we will see the installation page of Wordpress, so we can continue installing by entering the name of the blog, the administrator's login, his mail and password:



Finally, Wordpress is installed.

Now we can configure MySQL database logs.

Use the following commands to enter the container and enable logs:

```
sudo docker exec -it rmstr_mysql_1 bash
```

```
mysql -u root -p
```

```
SELECT @@global.general_log_file;
```

```
SET GLOBAL general_log:=1;
```

```
SET GLOBAL log_output := 'FILE';
```

```
FLUSH LOGS;
```

So, we found the location of the MySQL log:

```
mysql> SELECT @@global.general_log_file;
+-----+
| @@global.general_log_file |
+-----+
| /var/lib/mysql/994cce443a93.log |
+-----+
1 row in set (0.00 sec)
```

Since the /var/lib/mysql folder is accessible by /home/rmstr/logs/mysql/lib, it is more convenient to work with it.

So, let's set up sending logs to Papertrailapp

Download the agent with the command:

wget

https://github.com/papertrail/remote_syslog2/releases/download/v0.20/remote-syslog2_0.20_amd64.deb

Now install it:

sudo dpkg -i remote-syslog2_0.20_amd64.deb

We will save the agent's configuration file by the path /etc/log_files.yml. That file has paths to all the necessary configuration files and settings for connecting to the PapertrailApp server.

```
files:
  - /home/rmstr/logs/nginx/access.log
  - /home/rmstr/logs/nginx/error.log
  - /home/rmstr/logs/mysql/lib/994cce443a93.log
destination:
  host: logs3.papertrailapp.com
  port: 20273
  protocol: tls
pid_file: /var/run/remote_syslog.pid
```

After that, we run the agent with the `sudo remote_syslog` command, and in a few minutes we can see the logs which coming from MySQL and NGINX:

```
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.573103Z 36 Query SELECT YEAR(post_date) AS `year`, MONTH(post_date) AS `mo
WHERE post_type = 'post' AND post_status = 'publish' GROUP BY YEAR(post_date), MONTH(post_date) ORDER BY post_date DESC
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.573955Z 36 Query SELECT t.*, tt.* FROM wp_terms AS t INNER JOIN wp_term_
tt.term_id WHERE tt.taxonomy IN ('category') AND tt.count > 0 ORDER BY t.name ASC
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.574489Z 36 Query SELECT term_id, meta_key, meta_value FROM wp_termmeta WHE
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.576622Z 36 Query SELECT option_value FROM wp_options WHERE option_name = '
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.577497Z 36 Query SELECT option_value FROM wp_options WHERE option_name = '
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.578028Z 36 Query SELECT option_value FROM wp_options WHERE option_name = '
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.578647Z 36 Query SELECT comment_approved, COUNT( * ) AS total
FROM wp_comments
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.582818Z 36 Query SELECT comment_approved
FROM wp_comments GROUP BY comment_approved
Mar 26 16:35:16 testVM 994cce443a93.log: 2018-03-26T23:35:16.582818Z 36 Quit
Mar 26 16:35:16 testVM access.log: 46.211.156.204 - - [26/Mar/2018:23:35:16 +0000] "GET / HTTP/1.1" 200 19750 "http://52.170.153.5/wp-admin/updat
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36 OPR/51.0.2830.55"
Mar 26 16:35:17 testVM access.log: 46.211.156.204 - - [26/Mar/2018:23:35:17 +0000] "GET /favicon.ico HTTP/1.1" 200 0 "http://52.170.153.5/" "Mozil
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36 OPR/51.0.2830.55"
Mar 26 16:37:40 testVM access.log: 46.211.156.204 - - [26/Mar/2018:23:37:40 +0000] "GET /nginx_status/ HTTP/1.1" 200 100 "-" "Mozilla/5.0 (Window
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36 OPR/51.0.2830.55"
Mar 26 16:37:40 testVM access.log: 46.211.156.204 - - [26/Mar/2018:23:37:40 +0000] "GET /favicon.ico HTTP/1.1" 200 0 "http://52.170.153.5/nginx_s
```

Let's start setting up the metrics to Datadoghq:

Let's set up agent with command:

```
DD_API_KEY=a5f7c913580707ad4fef29b471d00014 bash -c "$(curl -L
https://raw.githubusercontent.com/DataDog/datadog-agent/master/cmd/agent/ins
tall_script.sh)"
```

After setting up the agent, set up sending metrics. You need to create the configuration files one by one for Nginx, MySQL, Docker:

For the Docker file `/etc/datadog-agent/conf.d/docker.d/docker_daemon.yaml`:

```
init_config:

instances:
  - url: "unix:///var/run/docker.sock"
    new_tag_names: true
```

For the nginx - file `/etc/datadog-agent/conf.d/nginx.d/conf.yaml` (in the variable `nginx_status_url`, you must insert the url to your server's nginx status page, in my configuration file its `/nginx_status`):

```
init_config:

instances:
  # For every instance, you have an `nginx_status_url` and (optionally)
  # a list of tags.

  - nginx_status_url: http://52.170.153.5/nginx_status/
```

For MySQL - file /etc/datadog-agent/conf.d/mysql.d/conf.yaml:

```
init_config:

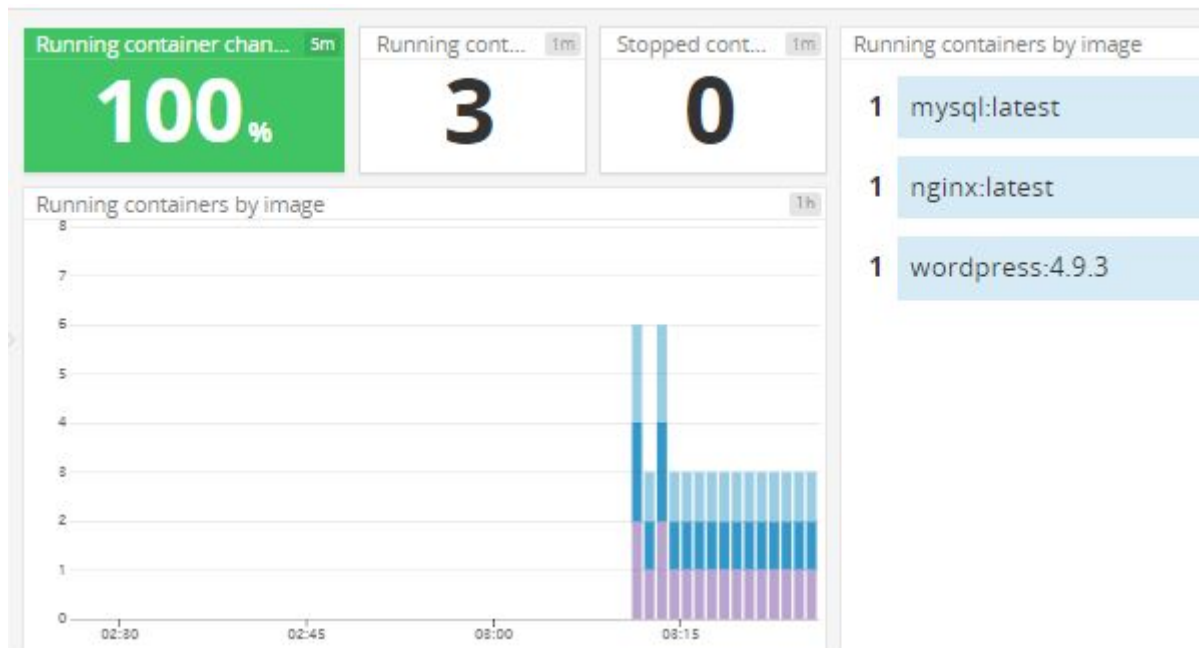
instances:
  - server: localhost
    user: root
    pass: qwel23

options:
  replication: 0
  galera_cluster: 1
```

After files creating, install the relevant products on the site:



Restart the agent with the `sudo service datadog-agent restart` command and in a few minutes the metric begins to appear.



NGINX

Requests

Overall requests per second



Requests: reading, writing, waiting



Requests per second by host



Green : low number of requests per second
Orange : high number of requests per second

Connections

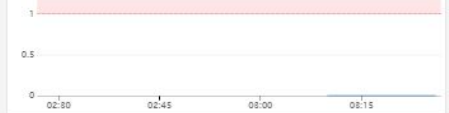
Dropped connections, last 15m

0 conns

Dropped connections, last 1d

0 conns

Dropped connections per second



Active connections per second



MySQL CPU time (per sec)

