



# ROYAL UNIVERSITY OF PHNOM PENH

## DATABASE II

**Deploy MySQL and phpmyadmin in AWS EC2  
using Docker**

Lecturer: Mr. Chhim Bunchhun,



[chhim.bunchhun@rupp.edu.kh](mailto:chhim.bunchhun@rupp.edu.kh), 093 222 380





# Create sub-domain

- Go to namecheap.com
- Select domain "iteg7.com" ->Advance DNS
- Create A Record to point to AWS EC2 public IP

Format of subdomain: **m[no]g[no].[member\_name]**

Example: m3g1.reaksa

=>We will get: m3g1.reaksa.iteg7.com point to that  
public IP

The screenshot shows the Namecheap DNS management interface for the domain 'iteg7.com'. The 'Advanced DNS' tab is selected. Under the 'HOST RECORDS' section, there is a table listing five A Records:

Type	Host	Value	TTL	Action
A Record	bc	18.141.173.205	Automatic	
A Record	bunchhun	18.141.173.205	Automatic	
A Record	g2m2	172.20.10.8	Automatic	
A Record	g4m2	13.215.183.248	Automatic	
A Record	m2g6solkhem.iteg7	13.215.161.178	Automatic	



# Install Docker in Server

- Remote to AWS EC2 server via SSH
- Follow guideline on :<https://docs.docker.com/engine/install/debian/>
- To check if docker installed:

```
docker --version
```

```
root@ip-172-31-26-131:/home/bunchhun# docker --version
Docker version 20.10.18, build b40c2f6
```

# Register free account at DockerHub



- Link: <https://hub.docker.com>

Docker login

->Provide username and password

```
root@ip-172-31-26-131:/home/bunchhun# docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-26-131:/home/bunchhun#
```

Get Started Today for Free

Already have an account? [Sign In](#)

Username

Email

Password

Send me occasional product updates and announcements.

I agree to the [Subscription Service Agreement](#), [Privacy Policy](#), and [Data Processing Terms](#).

I'm not a robot reCAPTCHA  
Privacy · Terms

**Sign Up**

# Pull MySQL and phpmyadmin images from DockerHub



```
docker pull mysql
```

```
docker pull phpmyadmin
```

The screenshot shows the Docker Hub interface. In the search bar at the top, 'phpmyadmin' is typed. Below the search bar, there are navigation links: 'Explore', 'Official Images', and 'phpmyadmin'. The main content area displays the 'phpmyadmin' image card. The card includes the 'phpMyAdmin' logo, the image name 'phpmyadmin', a badge indicating it's a 'DOCKER OFFICIAL IMAGE', a pull count of '10M+', and a star count of '645'. Below the card, there are two tabs: 'Overview' (which is selected) and 'Tags'. On the far right of the card, there is a button labeled 'Pull command copied' with a copy icon.

- To check images that exist:

```
docker image ls
```

or

```
docker images
```

```
root@ip-172-31-26-131:/home/bunchhun# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
Digest: sha256:b9532b1edea72b6cee12d9f5a78547bd3812ea5db842566e17f8b33291ed2921
Status: Image is up to date for mysql:latest
docker.io/library/mysql:latest

root@ip-172-31-26-131:/home/bunchhun# docker pull phpmyadmin
Using default tag: latest
latest: Pulling from library/phpmyadmin
Digest: sha256:ea1339b5d1d43d4170eefeeec36ab8ac40d076797072e00b1a9625c4d8acbadf5
Status: Image is up to date for phpmyadmin:latest
docker.io/library/phpmyadmin:latest
```



# Create Docker Network “mynet”

```
docker network create mynet
```

- To list all existing network

```
docker netwrok ls
```

```
root@ip-172-31-26-131:/home/bunchhun# docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
40b966647bfa   bridge    bridge    local
b816835d4867   host      host      local
161bd70ece68   mynet    bridge    local
1abe650b327a   none      null      local
root@ip-172-31-26-131:/home/bunchhun#
```



# Run MySQL Server Container

- To run MySQL Server Container

```
docker run -d \
    --name mysql \
    -p 3306:3306 \
    -e MYSQL_ROOT_PASSWORD=Rupp4ever! \
    --network mynet \
    --restart=always \
    -it mysql:latest
```

- To check running container: `docker container ls`

```
root@ip-172-31-26-131:/home/bunchhun# docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
e023ae157c54        mysql              "docker-entrypoint.s..."   10 hours ago      Up 10 hours       0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql
```



# Allow port 3306 in AWS EC2 Firewall

- Go to instant->select instant->Security->Select id in Security groups

The screenshot shows the AWS EC2 instance configuration page for an instance named 'instant'. The 'Security' tab is selected. In the 'Security details' section, there is a 'Security groups' field containing 'sg-000b49c0bfa07ea2a (launch-wizard-5)'. A large blue arrow points downwards from the text 'Inbound rules' towards the 'Filter rules' search bar. The 'Subnet ID' is listed as 'subnet-05becb6eac0251ff4'.

IAM Role	Subnet ID
-	<input type="checkbox"/> subnet-05becb6eac0251ff4

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
---------	----------	------------	---------	---------------	------------	------

**▼ Security details**

IAM Role	Owner ID
-	<input type="checkbox"/> 670858105406

Security groups  
 sg-000b49c0bfa07ea2a (launch-wizard-5)

**▼ Inbound rules**



# Allow port 3306 in AWS EC2 Firewall

- Click "Edit inbound rules"

The screenshot shows the AWS Security Groups Inbound Rules page. At the top, there are tabs for "Inbound rules" (which is selected), "Outbound rules", and "Tags". A message bar at the top says "You can now check network connectivity with Reachability Analyzer" with a "Run Reachability Analyzer" button. Below this, the "Inbound rules (5)" section is displayed. It includes a search bar labeled "Filter security group rules", a toolbar with "Edit inbound rules" and other buttons, and a table with columns: Name, Security group rule..., IP version, Type, Protocol, and Port range. The table shows five rows of rules.

- Add rule to allow port 3306:

The screenshot shows the AWS Security Groups Inbound Rules page with two existing rules listed and a new rule being configured. The first rule, "sgr-00a2e98dca6411f26", is for HTTP (Protocol TCP, Port range 80) from 0.0.0.0/0. The second rule, "sgr-06f16ae9c277272b9", is for MySQL/Aurora (Protocol TCP, Port range 3306) from Anywhere. A new rule is being added below them, with "Type" set to "Custom", "Protocol" set to "TCP", "Port range" set to "3306", and "Source" set to "0.0.0.0/0".

# Connect MySQL WorkBench to MySQL Server in AWS EC2



MySQL Workbench

mysqlrapp

# Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

[Browse Documentation >](#) [Read the Blog >](#) [Discuss on the Forums >](#)

**MySQL Connections** [+](#) [-](#)

Filter connections

Connection	User	Host
MySQL_RUPP	root	13.215.229.194:3306
mysqlrapp	root	bc.iteg7.com:3306
niccodb	root	13.215.31.151:3306
orkas	root	13.214.86.84:3306



# Run phpmyadmin Container

- To run phpmyadmin container:

```
docker run -d \
    --name phpmyadmin \
    -p 8080:80 \
    -e PMA_HOST=mysql \
    -e PMA_USER=root \
    -e PMA_PASSWORD=Rupp4ever! \
    --network mynet \
    --restart=always \
    -it phpmyadmin:latest
```

- To check running container: `docker container ls`

CONTAINER ID	IMAGE NAMES	COMMAND	CREATED	STATUS	PORTS
53c4210b4f82	phpmyadmin:latest phpmyadmin	"/docker-entrypoint...."	11 seconds ago	Up 10 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp
e023ae157c54	mysql mysql	"docker-entrypoint.s..."	10 hours ago	Up 10 hours	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp



# Allow port 80 and 443 in AWS EC2 Firewall

- Go to instant->select instant->Security->Select id in Security groups

The screenshot shows the AWS EC2 instance configuration page for an instance named 'instant'. The 'Security' tab is selected. In the 'Security details' section, there is a 'Security groups' field containing 'sg-000b49c0bfa07ea2a (launch-wizard-5)'. A large blue arrow points downwards from the text 'Security groups' towards the 'sg...' part of the text. Below this section is a 'Filter rules' search bar.

IAM Role	Subnet ID
-	<input type="checkbox"/> subnet-05becb6eac0251ff4

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
---------	----------	------------	---------	---------------	------------	------

**▼ Security details**

IAM Role	Owner ID
-	<input type="checkbox"/> 670858105406

Security groups  
 sg-000b49c0bfa07ea2a (launch-wizard-5)

**▼ Inbound rules**

Filter rules



# Allow port 80 and 443 in AWS EC2 Firewall

- Click "Edit inbound rules"

The screenshot shows the AWS Security Groups Inbound Rules page. At the top, there are tabs for "Inbound rules" (which is selected), "Outbound rules", and "Tags". A message bar at the top says "You can now check network connectivity with Reachability Analyzer" with a "Run Reachability Analyzer" button. Below this, the "Inbound rules (5)" section is displayed. It includes a search bar labeled "Filter security group rules", a toolbar with "C" (Create), "Manage tags", and "Edit inbound rules" buttons, and a pagination control showing page 1 of 1. The main table has columns for Name, Security group rule..., IP version, Type, Protocol, and Port range. One row is visible in the table.

- Add rule to allow port 80:

The screenshot shows the AWS Security Groups Inbound Rules page with two existing rules listed. The first rule, "sgr-00a2e98dca6411f26", is for HTTP (TCP port 80) from 0.0.0.0/0. The second rule, "sgr-06f16ae9c277272b9", is for MySQL/Aurora (TCP port 3306) from Anywhere. A new rule is being added in the "Add inbound rule" section, which includes fields for Type (HTTP), Protocol (TCP), Port range (80), and Source (0.0.0.0/0). The "Source" field has a dropdown menu open, showing "0.0.0.0/0" with an "X" button next to it.



# Install NGINX and Let's Encrypt

- Install nginx

```
apt install nginx
```

- Install Let's Encrypt for nginx

```
apt install certbot python3-certbot-nginx
```



# Configure NGINX with phpmyadmin

- Create new file in /etc/nginx/sites-available

```
nano bc.iteg7.com
```

- Add below content as on the right

```
bunchhun@ip-172-31-26-131:~$ sudo su
[sudo] password for bunchhun:
root@ip-172-31-26-131:/home/bunchhun# cd /etc/nginx/sites-available/
root@ip-172-31-26-131:/etc/nginx/sites-available# nano bc.iteg7.com
root@ip-172-31-26-131:/etc/nginx/sites-available#
```

- Activate bc.iteg7.com configuration

```
ln -s /etc/nginx/sites-available/bc.iteg7.com /etc/nginx/sites-enabled
```

- Restart nginx

```
Service nginx restart
```

```
server
{
    listen 80;
    listen [::]:80;
    server_name bc.iteg7.com;
    client_max_body_size 128M;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;
    location ~ /.well-known
    {
        allow all;
    }

    location /
    {
        proxy_pass http://127.0.0.1:8080/;
        proxy_http_version 1.1;
        proxy_cache_bypass $http_upgrade;

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Port $server_port;

        proxy_set_header X-Scheme $scheme;
    }
}
```



# Get SSL Certificate through Let's Encrypt

```
certbot --nginx -d bc.iteg7.com
```

-> enter email address like :bunchhun@iteg7.com

-> y

⇒ <https://bc.iteg7.com>

working with https

The screenshot shows the phpMyAdmin interface for a MySQL server named 'mysql'. The 'General settings' section displays the server connection collation as 'utf8mb4\_unicode\_ci'. The 'Database server' section provides detailed information about the MySQL server, including its version (8.0.30), protocol version (10), and character set (UTF-8 Unicode). The 'Web server' section shows Apache 2.4.54 (Debian) and PHP 8.0.24. The bottom right corner of the interface displays the 'phpMyAdmin' logo.



**THANK YOU  
for your attention !**

