MECAs Application Workshop

1. Static Website

Static Website เป็น เว็บไซต์ ที่ประมวลที่ฝั่งของ Client เพียงอย่างเดียว ประกอบด้วย HTML, JavaScript, CSS ซึ่งเป็นการพัฒนาเว็บไซต์ ทั่วๆ ไป เหมาะกับการแสดงข้อมูล ให้กับสาธารณะ ไม่มีการจะต้องประมวล สำหรับแต่ละการเข้าชม เนื่องจากเทคโนโลยี Web มีการพัฒนามาอย่างยาวนาน ทำให้นักพัฒนาสามารถเริ่มต้นพัฒนาได้จากเครื่องมือง่ายๆ หรือใช้ ซอฟแวร์พัฒนาเว็บไซต์ หรือ จะใช้เทมเพลต ที่มีผู้พัฒนาโครงร่างเอาไว้แล้ว นำมาแก้ไข ได้
ในที่นี้ จะแนะนำเครื่องมือ ที่เรียกว่า Static Web Generator ซึ่งเป็นเครื่องมือในการสร้างเว็บไซต์ จากต้นแบบ โดยหลักการคือ จะแยกส่วนของ โครงร่าง (Layout) ออกจาก ส่วนของข้อมูล ซึ่งเมื่อมีการสร้างเว็บ สำหรับนำไปใช้งาน จะนำส่วนของโครงร่าง และข้อมูลรวมกัน ทำให้เกิดหน้าเว็บ หลายๆ หน้าได้อย่างอัตโนมัติ
กรณีเช่น ต้องการทำหน้าเว็บ Blog สำหรับประชาสัมพันธ์ข้อมูล ผู้พัฒนาสามารถใช้ เครื่องมือ Static Web Generator สร้างเว็บไซต์ สำเร็จรูปออกมาได้เลย โดยไม่จำเป็นจะต้องใช้ Web Application ที่มีฐานข้อมูล โดยทุกอย่าง จะอยู่ในรูปแบบของไฟล์ ช่วยให้ลดความซับซ้อนในการติดตั้ง
ในที่นี้ จะใช้เครื่องมือที่ชื่อว่า Jekyll https://jekyllrb.com/ ในการสร้างเว็บ Blog และจะเน้น เรื่องของการใช้ Docker ร่วมกับการพัฒนา โดยไม่จำเป็นจะต้องติดตั้ง Software อื่นเลย

เริ่มการพัฒนา

การสร้างโปรเจค

```
$ docker run --rm -it -v "${PWD}":/page jekyll/jekyll jekyll new
/page/my-awesome-site
```

หลังจากมีการสร้างโปรเจคแล้ว อาจจะต้องมีการปรับ permission ของ ไฟล์ และโฟลเดอร์ เนื่องจาก ภายใน Docker จะ รันด้วยสิทธิ root

```
$ cd my-awesome-site
```

```
$ docker run --name=jekyll-dev -it -v "${PWD}":/srv/jekyll -p 4000:4000
jekyll/jekyll jekyll serve -H 0.0.0.0
```

จะเห็นว่า มีการตั้งชื่อ ให้กับ container เพื่อสะดวกในการพัฒนา ทำให้คง สภาพแวดล้อมต่างๆ ไว้ เช่น dependency ต่างๆ ซึ่งหากจะกลับมาทำงานอีกครั้ง เราสามารถใช้ docker start <container id> ได้

เปิด Browser ที่ http://localhost:4000 ก็จะได้แบบไซต์สำเร็จรูปขึ้นมา

การสร้าง Blog ทำได้โดยการสร้างไฟล์ ในรูปแบบของ Markdown (อ่านเพิ่ม https://www.markdownguide.org/basic-syntax/) ที่ _post/2019-03-21-my-first-blog.md ตามรูปแบบ YYYY-MM-DD-<name> ตัวอย่างเช่น

```
---
layout: post
title:  "My first blog!"
date:   2019-03-22 11:12:21 +0700
---
# Yeah
```

และเมื่อต้องการนำ เว็บไซต์ที่สร้างขึ้น ไปใช้งานจริง จำเป็นจะต้องมีการ compile ให้อยู่ในรูปแบบของ HTML, JavaScript, CSS ที่เหมาะสม ซึ่งในที่นี้ เราจะใช้ docker exec เพื่อ เป็นการเข้าไป run คำสั่งภายใน docker container ในที่นี้ใช้ command sh เพื่อเข้าในรูปแบบของ shell

```
$ docker ps
CONTAINER ID      IMAGE             COMMAND                  CREATED
      STATUS             PORTS                                    NAMES
5872339a143c      jekyll/jekyll     "/usr/jekyll/bin/ent..."   About an hour
ago   Up About an hour  0.0.0.0:4000->4000/tcp, 35729/tcp   jekyll-dev
$ docker exec -it 587 sh
/srv/jekyll # jekyll build
```

ซึ่งคำสั่ง jekyll build จะเป็นการ build เว็บไซต์สำเร็จรูปอยู่ที่ _site

การสร้าง Docker Image

หลังจากที่มีการพัฒนาโปรแกรมแล้ว จะต้องทำให้โปรแกรมนั้นอยู่ในรูปแบบของ Docker Image เพื่อให้พร้อมสำหรับการนำไปติดตั้งบน production environment
ทำการสร้าง Dockerfile

```
FROM nginx:alpine
COPY _site /usr/share/nginx/html
```

```
$ docker build -t my-awesome-site .
$ docker tag my-awesome-site <username>/my-awesome-site:v1
$ docker push <username>/my-awesome-site:v1
```

Build และทำการ push docker image ขึ้น Docker hub repository

Deploy on MECAs

หลังจากที่มีการเตรียม Docker Image เรียบร้อยแล้ว เราสามารถนำ Docker Image นั้น ไป Deploy บน MECAs ซึ่งเป็น บริการ Container as a Service ได้

โดยทุกครั้ง จะต้องมีการวางแผนการ Deploy Application ก่อน ซึ่งในที่นี้รูปแบบของ Application จะเป็น 1 deployment และ 1 service

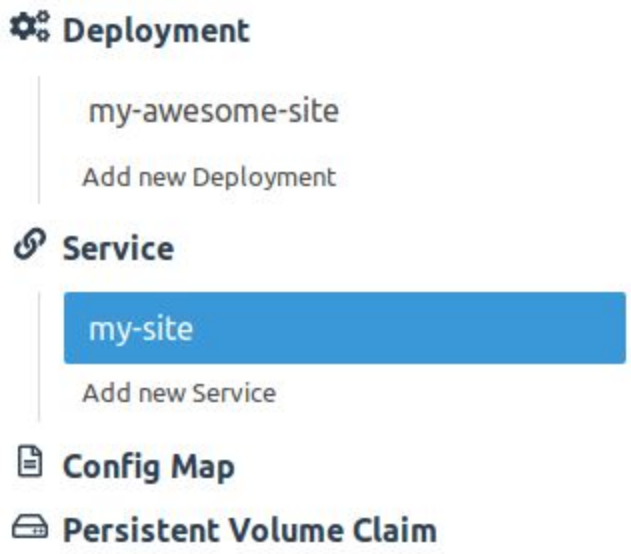


Image list:

- ridnarong/my-awesome-site:v1

การ Deploy my-awesome-site

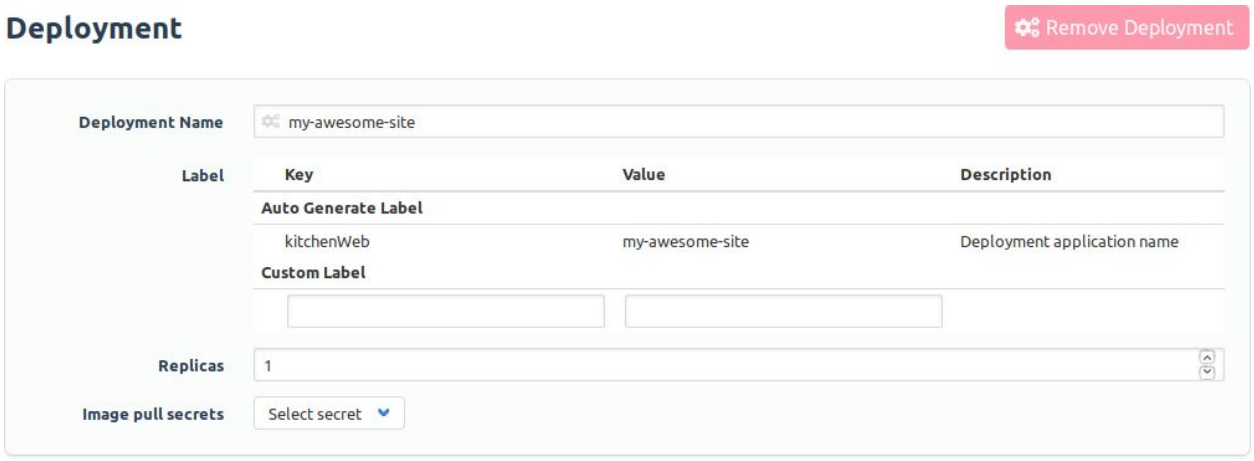
- **Deployment**

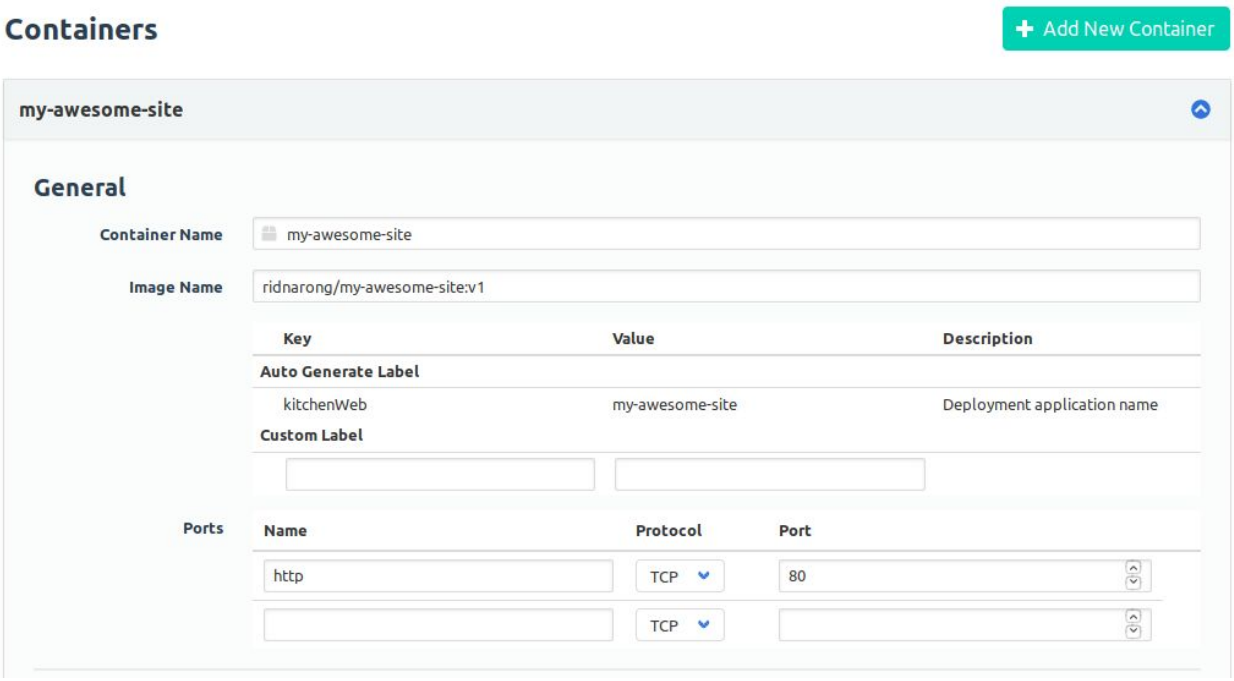
Deployment Name	my-awesome-site

- **Containers**

Container	my-awesome-site
Image Name	ridnarong/my-awesome-site:v1
Ports	
Name	**Port**
http	80

กรอกข้อมูลลงไปในแบบฟอร์ม ในส่วนของ Deployment



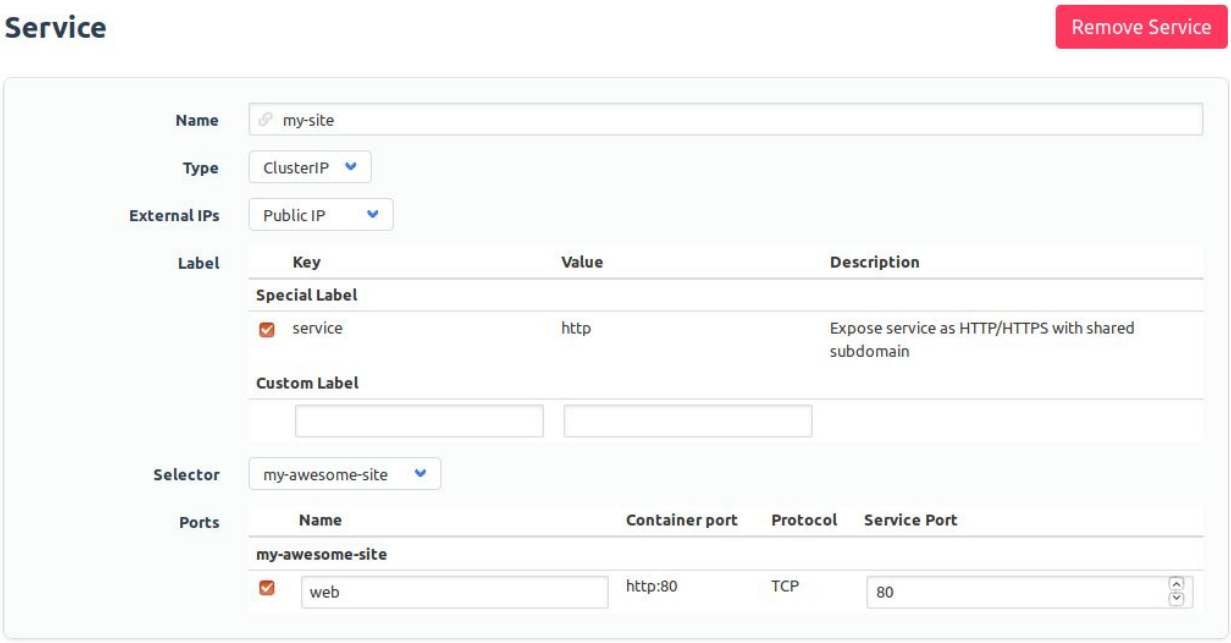


กรอกแบบฟอร์ม Service

Service Name	my-site
Selector	my-awesome-site
Port	
Name	**Service Port**
web	80

เริ่มต้นกรอกข้อมูลของ Service สำหรับ mysite

กรอกข้อมูลลงไปในแบบฟอร์ม ในส่วนของ Service โดยบริการของ MECAs มีบริการสร้าง subdomain <service name>.<namespace>.web.meca.in.th พร้อมทั้ง SSL Certificate (Let's encrypt) ให้อัตโนมัติ สำหรับการใช้งาน Protocol HTTP/HTTPS ผ่าน port 80/443 เพียงเลือก Label service:http ซึ่งผู้ใช้ จะต้องระบุ Service Port เป็น 80 ด้วย เพื่อการทำงานอย่างถูกต้อง



<u>หมายเหตุ</u>

รองรับ WS และ WSS สำหรับ Websocket เช่นเดียวกัน

การใช้งาน subdomain อาจจะใช้เวลา 1 - 2 นาที ในการพร้อมใช้งาน

การให้บริการ Protocol HTTP/HTTPS จะไม่ได้มีการ redirect HTTP ไปยัง HTTPS อัตโนมัติ

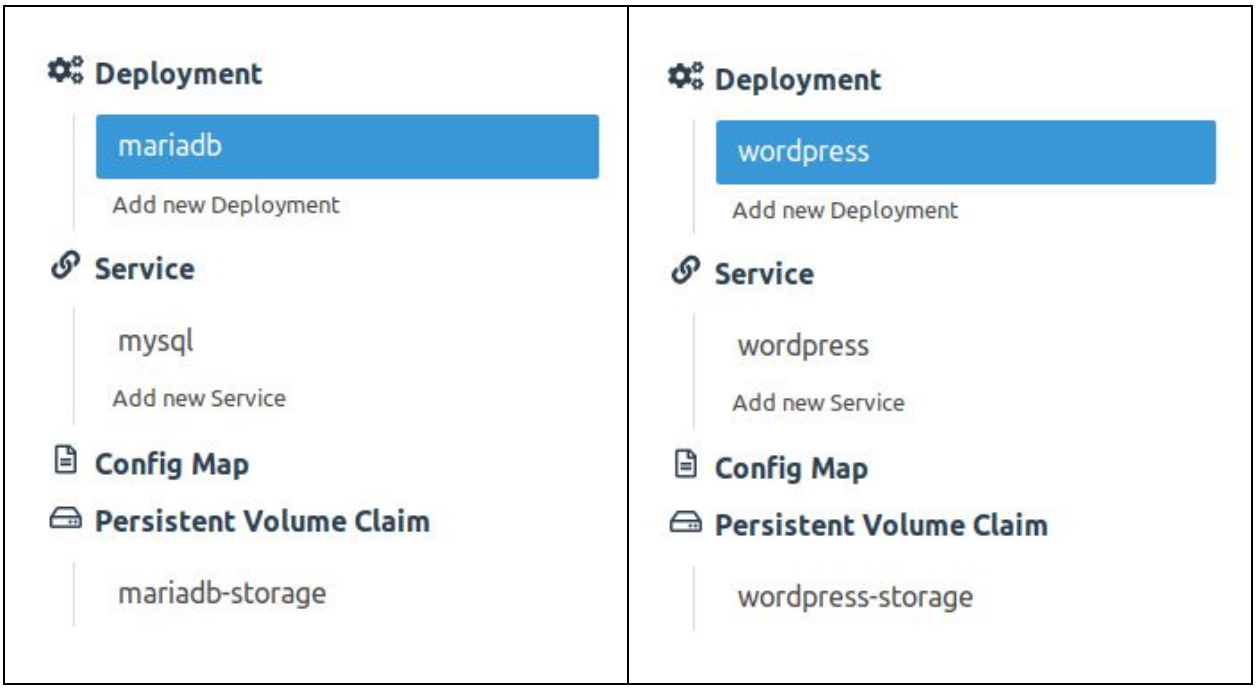
ผู้ใช้สามารถใช้ Custom Domain ได้ โดยมีการตั้งค่า DNS มายังระบบ สามารถใช้งานได้เช่นเดียวกัน แต่ในขณะนี้ ยังไม่มีบริการ Custom Certificate สำหรับผู้ใช้ที่ต้องการใช้ SSL Certificate ของตัวเอง ระบบจะสร้าง SSL Certificate ผ่าน Let’s encrypt ให้สำหรับ Custom domain

2. Web application with database

รูปแบบ Appication ที่เป็นที่นิยม คือ Web Application ที่มีการประมวลทางฝั่ง server และมีการใช้งานฐานข้อมูลโดยภาษาโปรแกรม ที่อยู่ในรูปแบบของ Docker แล้วก็มีหลากหลาย ในที่นี้จะแนะนำ การติดตั้ง และใช้งาน Wordpress
Wordpress เป็น Web blog ถูกเขียนขึ้นโดยภาษา PHP และใช้ฐานข้อมูล SQL ในการจัดเก็บข้อมูล ซึ่งได้รับความนิยมอย่างแพร่หลาย ซึ่งใน Docker Hub มี Wordpress อยู่แล้ว สามารถนำมาใช้งานได้ทันที

การ Deploy บน MECAs

โดยรูปแบบของการ deploy จะประกอบด้วย



2.1 การ Deploy MariaDB

Deployment

Deployment Name	mariadb

Containers

Container	mariadb
Image Name	mariadb:10.3.13-bionic

Ports	
Name	**Port**
mysql	3306

Environment Variable

Key	**Value**
MYSQL_ROOT_PASSWORD	my-secret-pw

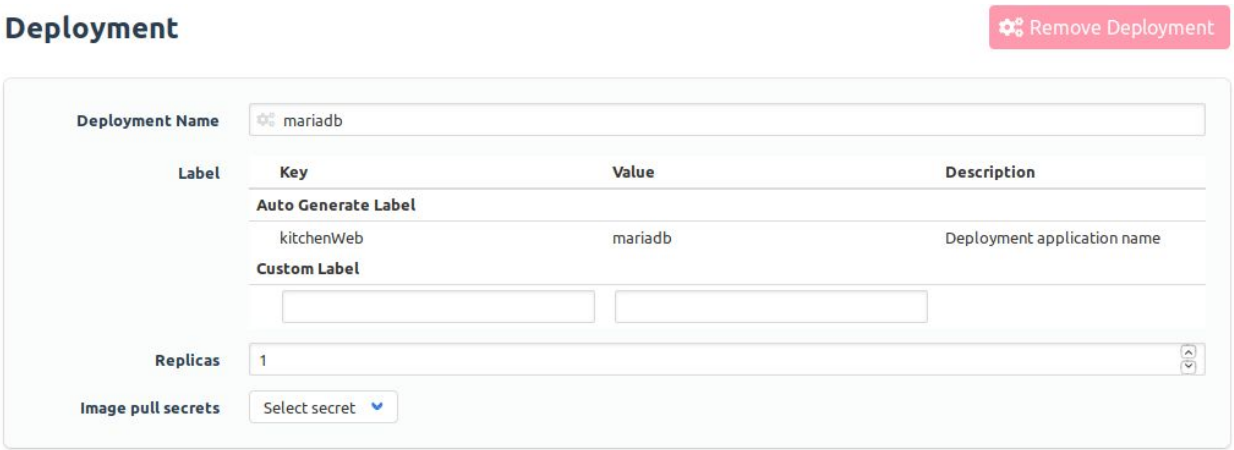
Volume Mounts

Mount Point	/var/lib/mysql

Volumes

Volume Name	wordpress-db-data
PVC Name	wordpress-db-storage
Storage Class	rbd-r2
PVC Size	2 GiB

กรอกแบบฟอร์ม Deployment



Containers

+ Add New Container

mariadb

General

Container Name: mariadb

Image Name: mariadb:10.3.13-bionic

Key	Value	Description
Auto Generate Label		
kitchenWeb	mariadb	Deployment application name
Custom Label		

Ports

Name	Protocol	Port
mysql	TCP	3306
	TCP	

Environment

Arguments:

Command: (Optional)

Working directory: (Optional)

Environment Variable

Key	Value
MYSQL_ROOT_PASSWORD	my-secret-pw
Key	Value

Volume Mounts

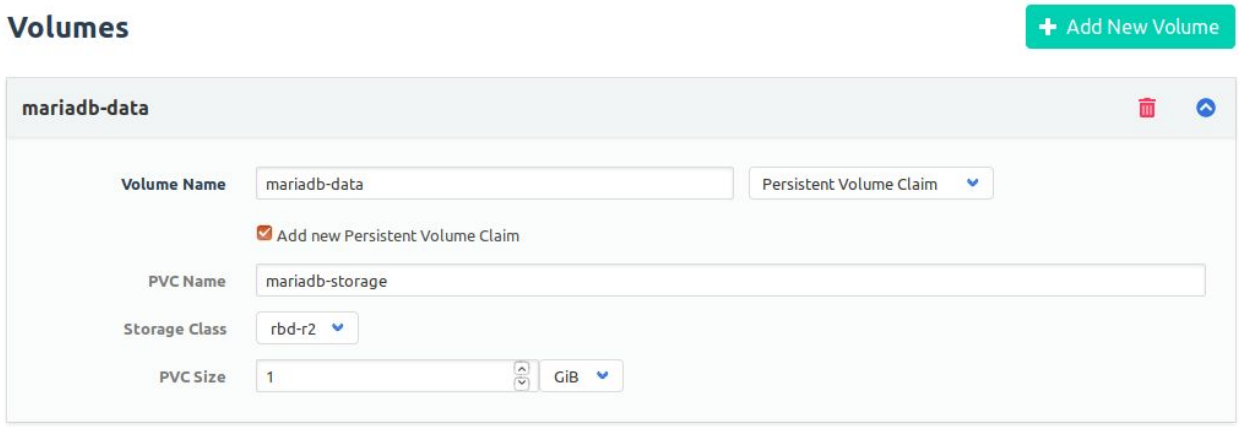
+ Add New Volume Mount

Volume Name: mariadb-data

Mount Point: /var/lib/mysql

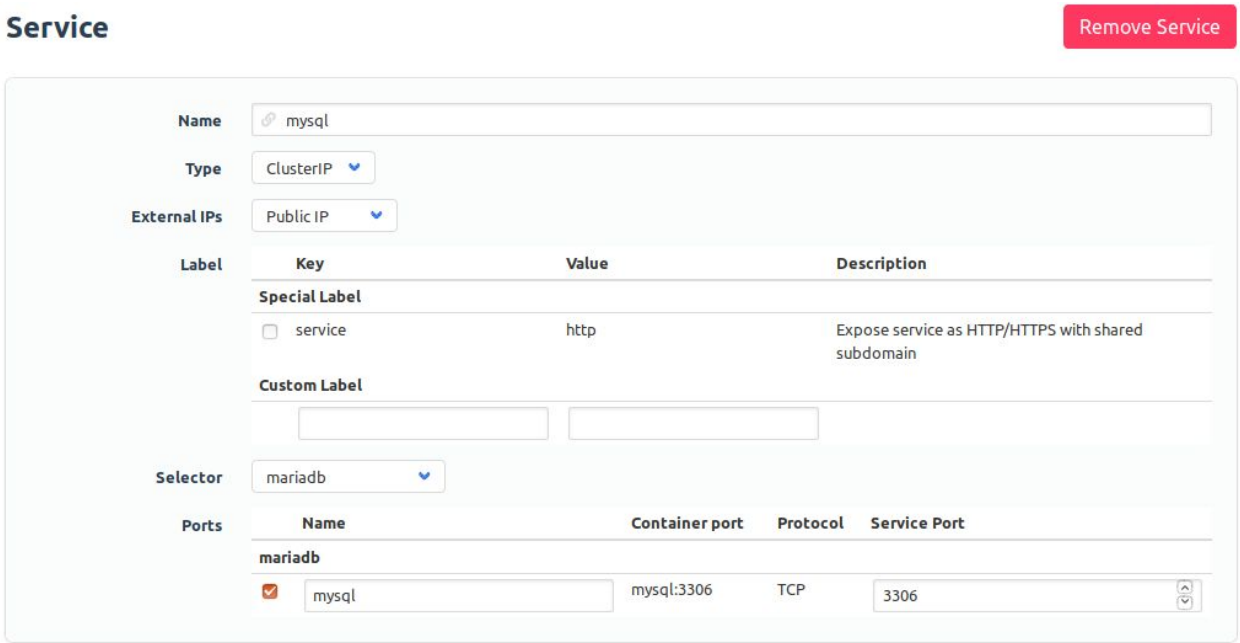
Sub Path in Volume:

Read-Only: ☐



กรอกแบบฟอร์ม Service

Service Name	mysql
Selector	mariadb
Port	
Name	**Service Port**
mysql	3306



2.2 การ Deploy Wordpress

- **Deployment**

Deployment Name	wordpress

- **Containers**

Container Name	wordpress
Image Name	wordpress:5.1.1-apache
Ports	
Name	**Port**
http	80

- **Environment Variable**

Key	**Value**
WORDPRESS_DB_USER	root
WORDPRESS_DB_PASSWORD	my-secret-pw
WORDPRESS_DB_HOST	mysql

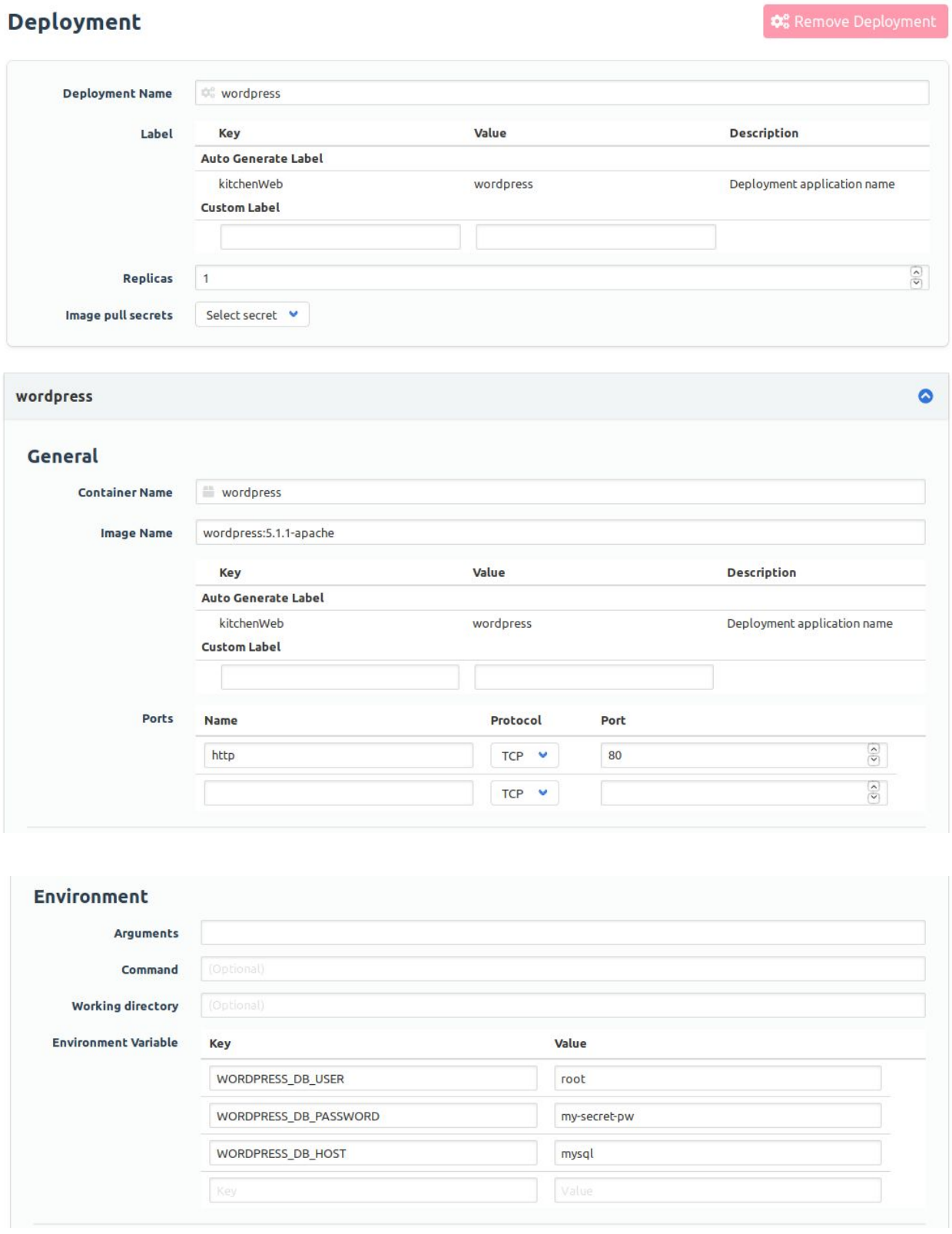
- **Volume Mounts**

Mount Point	/var/www/html/wp-content/

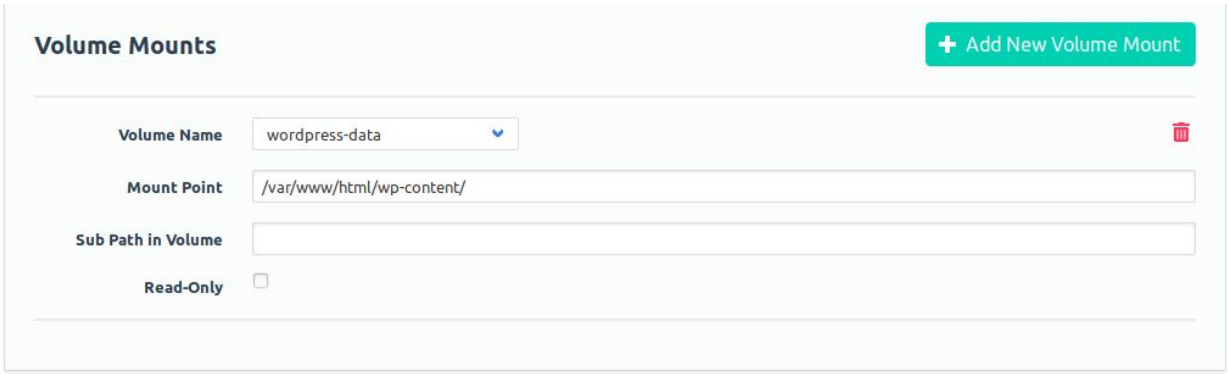
- **Volumes**

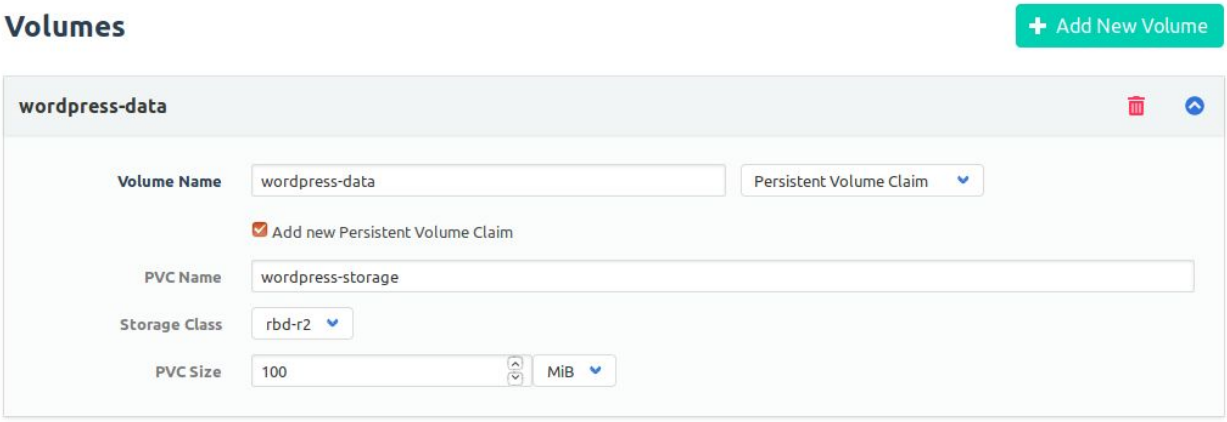
Volume Name	wordpress-data
PVC Name	wordpress-storage
Storage Class	rbd-r2
PVC Size	100 MiB

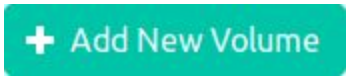
กรอกแบบฟอร์ม Deployment

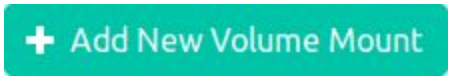


กำหนดค่า Environment Variable สำหรับใช้ใน Container ผ่านแบบฟอร์ม โดย Key คือ ชื่อของ Environment Variable และ Value คือ ค่า
สำหรับ Wordpress จะต้องมีการตั้งค่าเกี่ยวกับฐานข้อมูล ซึ่งในที่นี้จะต้อง ตรงกับที่ตั้งค่าไว้กับ ฐานข้อมูล MariaDB และสำหรับ WORDPRESS_DB_HOST จะเป็นชื่อ service ของ MariaDB





การสร้าง Volume Mount จะต้องมีการสร้าง Volume ขึ้นมาก่อน โดยเพิ่ม Volume ใหม่ สำหรับ Deployment นั้น ผ่าน 

และทำการตั้งชื่อ Volume สำหรับใช้ใน Deployment สามารถเลือกได้ 3 ประเภท คือ ConfigMap, Persistent Volume Claim และ Empty Directory โดยสามารถเลือกจากรายการที่มีอยู่แล้ว หรือจะสร้างใหม่ พร้อมกับการสร้าง Deployment ก็ได้ ซึ่งในที่นี้ จะสร้าง Persistent Volume Claim ขึ้นมาใหม่ ขนาด 100 MiB จาก Storage Class rbd-r2 หลังจากที่มีการสร้าง Volume แล้ว สามารถนำ Volume นี้ Mount เข้าไปใน Container ได้ ผ่าน 

โดยเลือกจาก Volume Name และกำหนด Mount Point ตามต้องการ

อธิบายเพิ่ม

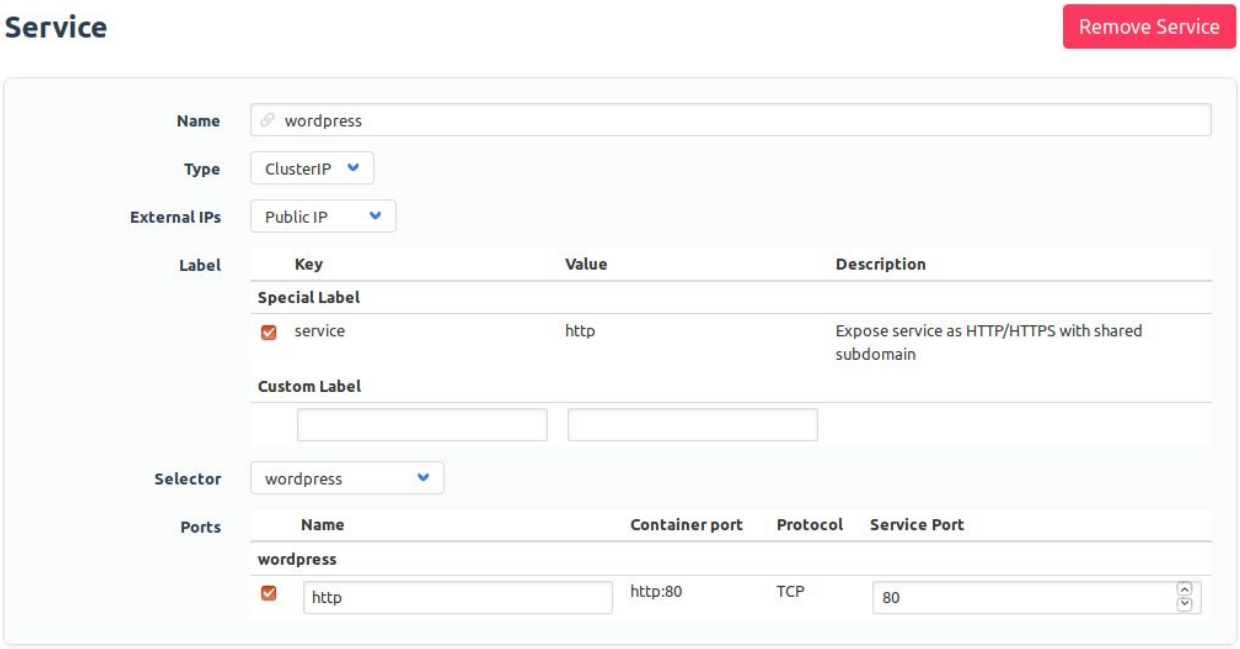
การใช้งาน ประเภทของ Storage
ConfigMap เหมาะกับการเก็บข้อมูลขนาดไม่ใหญ่มาก ในรูปแบบของ Text เช่นการตั้งค่าต่างๆ, ไฟล์ต่างๆ
Persistent Volume Claim เหมาะกับการจัดเก็บข้อมูล ขนาดใหญ่ โดยใช้งานในรูปแบบของ Block stroage
Empty Directory เป็นการสร้างพื้นที่เก็บข้อมูลชั่วคราว ในขณะที่มีการสร้าง Deployment

Storage Class เป็นรูปแบบการให้บริการ Persistent Volume ซึ่ง MECAs ให้บริการในรูปแบบของ Block Storage ซึ่งมีการทำสำเนาข้อมูล และกระจายข้อมูล ไปทั่วทั้ง Cluster
rbd-r2 คือ Ceph Block Storage ที่มีการทำสำเนาข้อมูล สองชุด
rbd-r3 คือ Ceph Block Storage ที่มีการทำสำเนาข้อมูล สามชุด

ในตัวอย่างนี้ มีการสร้าง 2 Service โดย
wordpress จะเป็น service สำหรับเข้าใช้งานผ่าน http จึงมีการเลือก label เป็น service:web
wordpress-db จะเป็น service สำหรับการใช้งานฐานข้อมูล ซึ่งหากไม่ได้กำหนด External IP service จะสามารถเข้าถึงได้เฉพาะภายใน cluster เท่านั้น ซึ่ง service จะสามารถถูก resolve ได้ภายใน เป็นหมายเลข IP ที่เรียกว่า ClusterIP ซึ่งจะไม่ซ้ำ กับ Service อื่นๆ จึงสามารถ Expose port ผ่าน Service Port ได้ โดยไม่ต้องกังวลเกี่ยวกับการใช้ port

กรอกแบบฟอร์ม Service

Service Name	wordpress
Selector	wordpress
Ports	
Name	**Service Port**
http	80



อธิบายเพิ่ม

หมายเลข IP ภายใน ประกอบไปด้วย

- Pod IP เป็นหมายเลข IP ของแต่ละ Pod โดยจะถูกสร้าง และกำหนดขึ้นอัตโนมัติ ตามวัฏจักรของ Pod ซึ่งมีความเป็นพลวัตสูง จึงไม่เหมาะกับการไปใช้งาน เพราะ Pod อาจจะสร้าง หรือลบได้ตลอดเวลา ตามการจัดสรรของ ระบบ
- Cluster IP เป็นหมายเลข IP ของแต่ละ Service ที่จะถูกสร้าง และกำหนดขึ้นอัตโนมัติ ทำหน้าที่คล้าย Proxy/Load Balance โดยจะเป็นอิสระแยกจาก Pod และมีความคงที่ ตราบเท่าที่ Service นั้นยังคงอยู่ และสามารถถูก resolve ได้ภายใน ผ่านชื่อ service ในรูปแบบ <ชื่อ service> หรือ <ชื่อ service>.<namespace> โดยที่ Service จะทำหน้าที่ เป็นตัวแทนของ Pod ผ่านการใช้ Selector ให้สามารถ เข้าถึง Pod ของ Deployment แต่ละอันได้ และนอกจากนี้ ยังสามารถ ปรับการใช้ Port โดย service port ไม่จำเป็นจะต้องเป็น port เดียวกับใน container port ก็ได้

3. IoT Stack

Internet of Thing เป็นตัวอย่างของระบบ ที่มีความซับซ้อน ในการ deploy เนื่องจาก ประกอบข้อจาก Application หลายตัว แต่โดยทั่วไปแล้ว IoT Stack จะประกอบไปด้วย สามส่วน คือ

- Ingress คือ ส่วนที่นำข้อมูลเข้า ซึ่งจะติดต่อกับ Protocol ที่ Thing ใช้ในการสื่อสาร เพื่อรับข้อมูลของ Thing เข้าระบบ
- Data store คือ ส่วนที่เก็บข้อมูลที่ได้รับจาก Thing อาจจะเป็น ฐานข้อมูล หรือรูปแบบอื่นๆ แล้วแต่ Application ที่จะต้องนำข้อมูลไปใช้งาน
- Visualization คือ ส่วนที่แสดงผล ข้อมูลของ Thing ในรูปแบบต่างๆ เช่น ตาราง กราฟ หรือ JSON ที่เปิดให้บุคคลที่สามเข้ามาดึงข้อมูล

โดยในที่นี้ ระบบที่ออกแบบจะประกอบด้วย

- RabbitMQ (https://www.rabbitmq.com/) ซึ่งเป็น Application สำหรับจัดการข้อความที่ Program ใช้สื่อสารระหว่างกัน (Messaging Software) โดย RabbitMQ จะทำหน้าที่ เป็น MQTT Broker และส่งข้อมูลให้กับ Ingress ในลำดับถัดไป สาเหตุที่เลือก RabbitMQ เนื่องจาก มีหน้า GUI สำหรับ จัดการ, มีระบบ Messaging ที่ครอบคลุมการใช้งานรูปแบบต่างๆ, มี MQTT Plugin เพื่อรองรับข้อมูลจากฝั่งของ Thing ผ่าน MQTT และ ใช้ AMQP กับฝั่งของ Telegraf ซึ่งเป็น Ingress ที่เลือกใช้
- Telegraf (https://www.influxdata.com/time-series-platform/telegraf/) เป็น Application ที่ทำหน้าที่ รับ/ดึง ข้อมูลในรูปแบบต่างๆ โดยใช้ Input plugin ที่หลากหลาย แปลงข้อมูล และนำข้อมูลออกไปยังแหล่งต่างๆ ผ่าน Output plugin สาเหตุที่เลือกใช้ เนื่องจาก สามารถทำงานได้ทันที ไม่ต้องมีการเขียนโปรแกรม ใช้การ Config pipeline และ มี Plugin ครอบคลุมงานที่จะทำ
- Graphite (https://graphiteapp.org/) เป็น Application ที่ทำหน้าที่ รับ และจัดเก็บข้อมูลในรูปแบบของ metrics ซึ่งใช้มากในการ monitor ระบบคอมพิวเตอร์ แต่ก็สามารถนำมาประยุกต์ใช้เก็บข้อมูลของ Thing ได้เช่นเดียวกัน สาเหตุที่เลือกใช้ เนื่องจาก มีความสามารถครบถ้วนในตัวเอง ตั้งแต่ รับข้อมูล จัดเก็บข้อมูล แสดงผลข้อมูล มีระบบผู้ใช้ มี API รูปแบบต่างๆ และระบบการจัดเก็บข้อมูล จัดเก็บในรูปแบบของไฟล์ ไม่จำเป็นจะต้องใช้ระบบฐานข้อมูล ที่สามารถตั้งค่า Retention Period ได้

การวางแผน Deployment

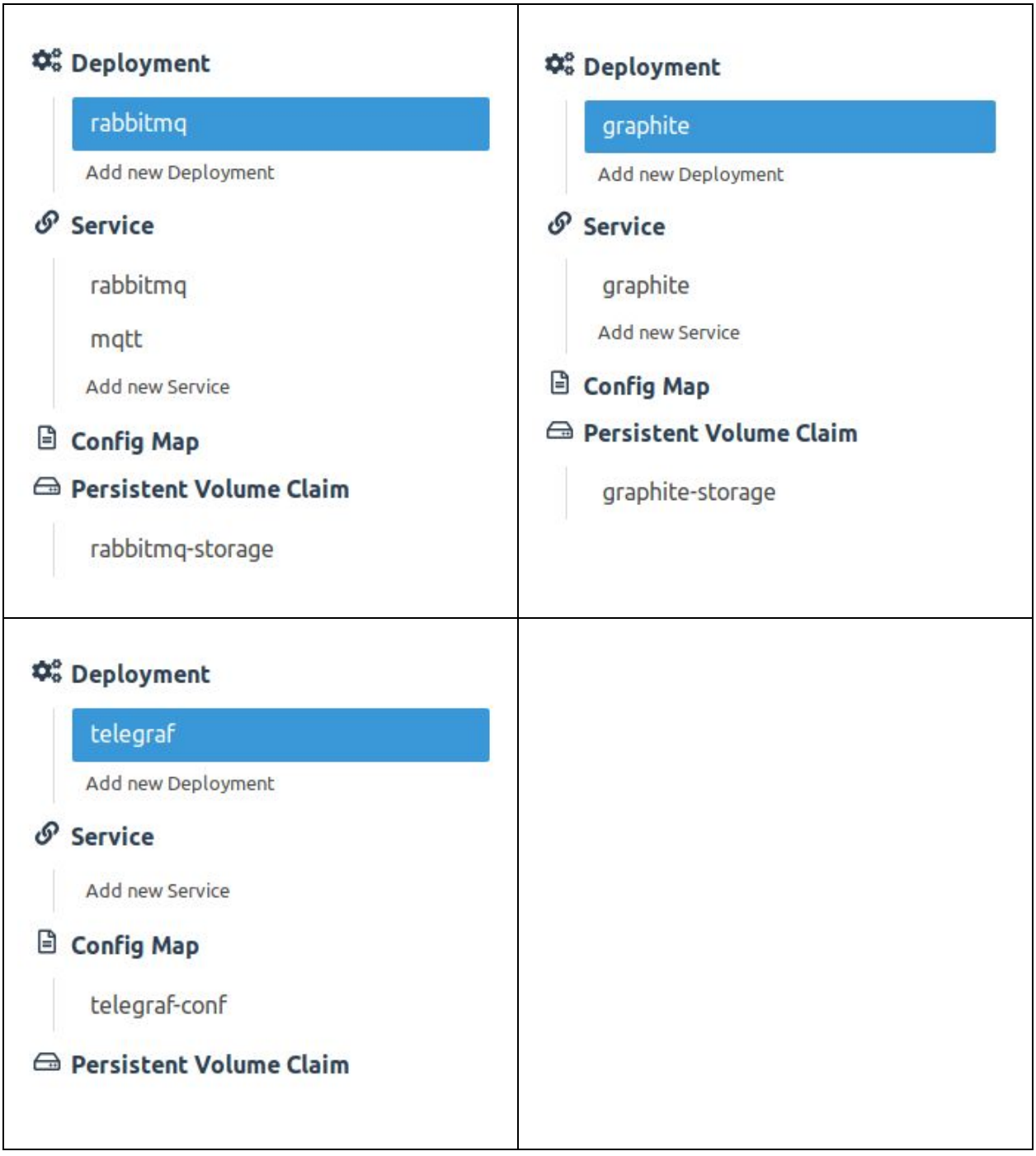


Image list:

- ridnarong/rabbitmq:3.7.13-management

- telegraf:1.10.1
- graphiteapp/graphite-statsd:1.1.5-10

3.1 การ Deploy Rabbitmq

- **Deployment**

Deployment Name	rabbitmq

- **Containers**

Container Name	rabbitmq
Image Name	ridnarong/rabbitmq:3.7.13-management
Ports	
Name	**Port**
amqp	5672
management	15672
mqtt	1883

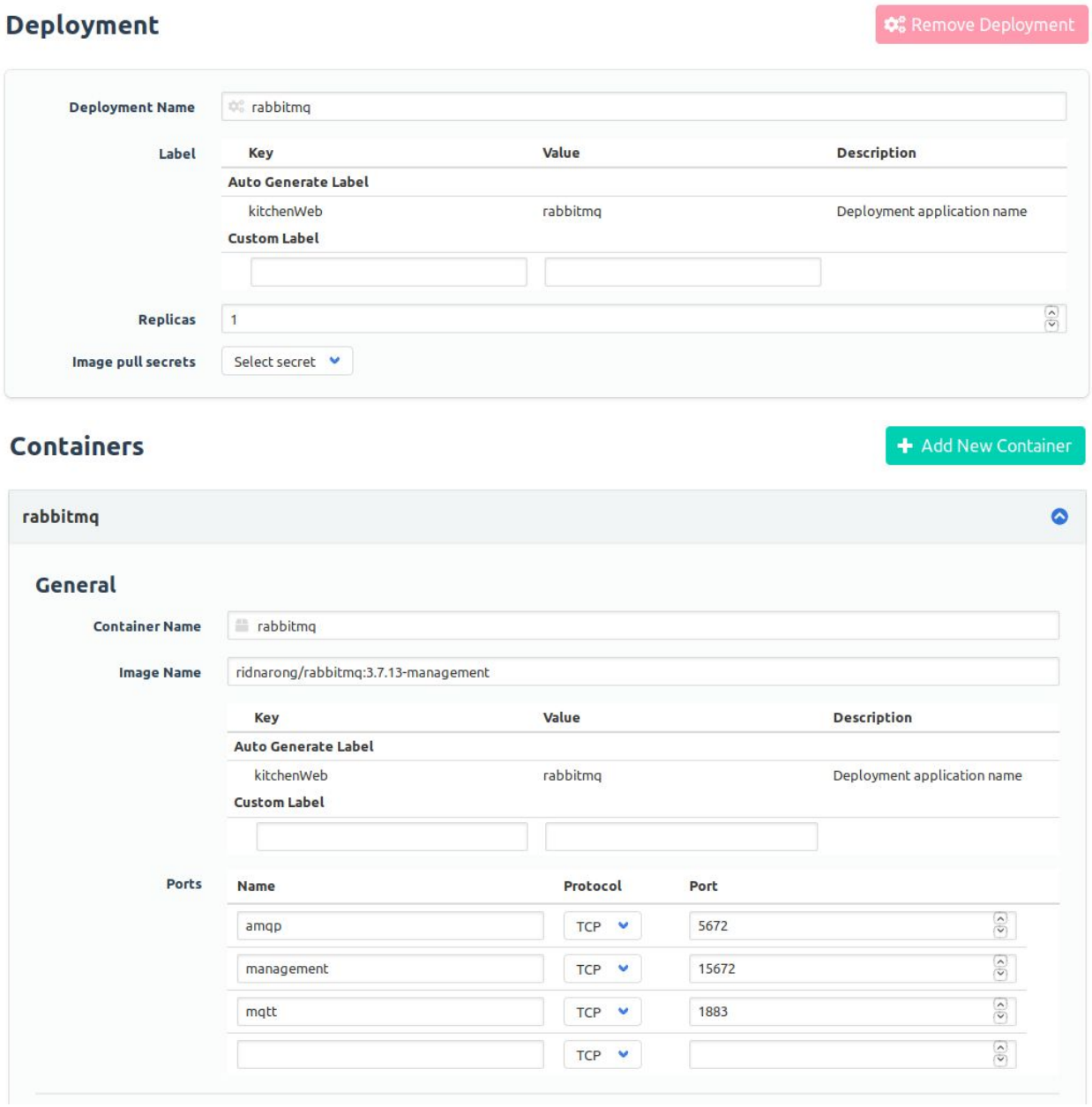
- **Volume Mounts**

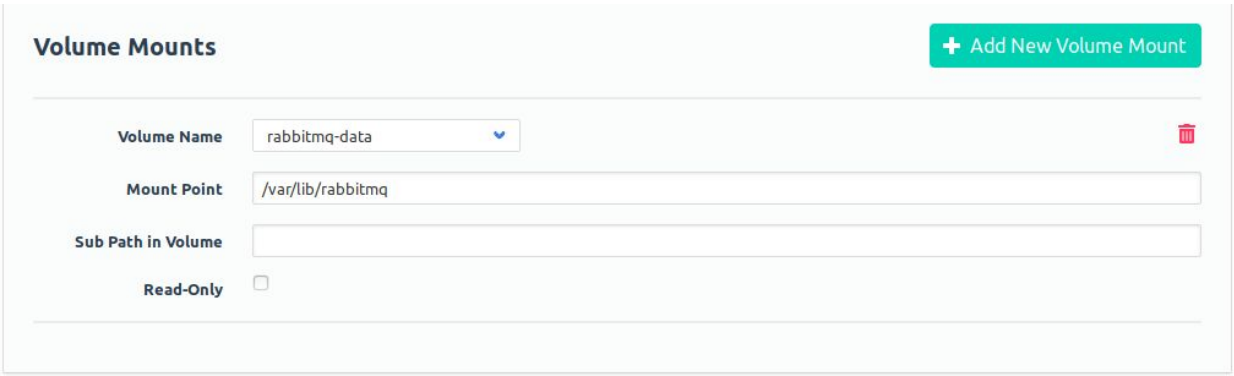
Volume Name	rabbitmq-data
Mount Point	/var/lib/rabbitmq

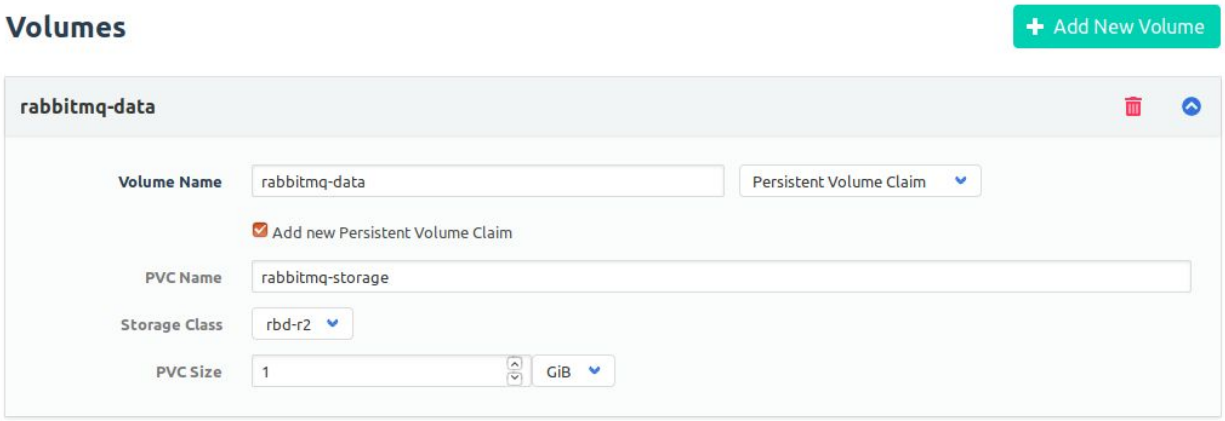
- **Volume**

Volume Name	rabbitmq-data
PVC Name	rabbitmq-storage
Storage Class	rbd-r2
PVC Size	1 GiB

กรอกแบบฟอร์ม Deployment







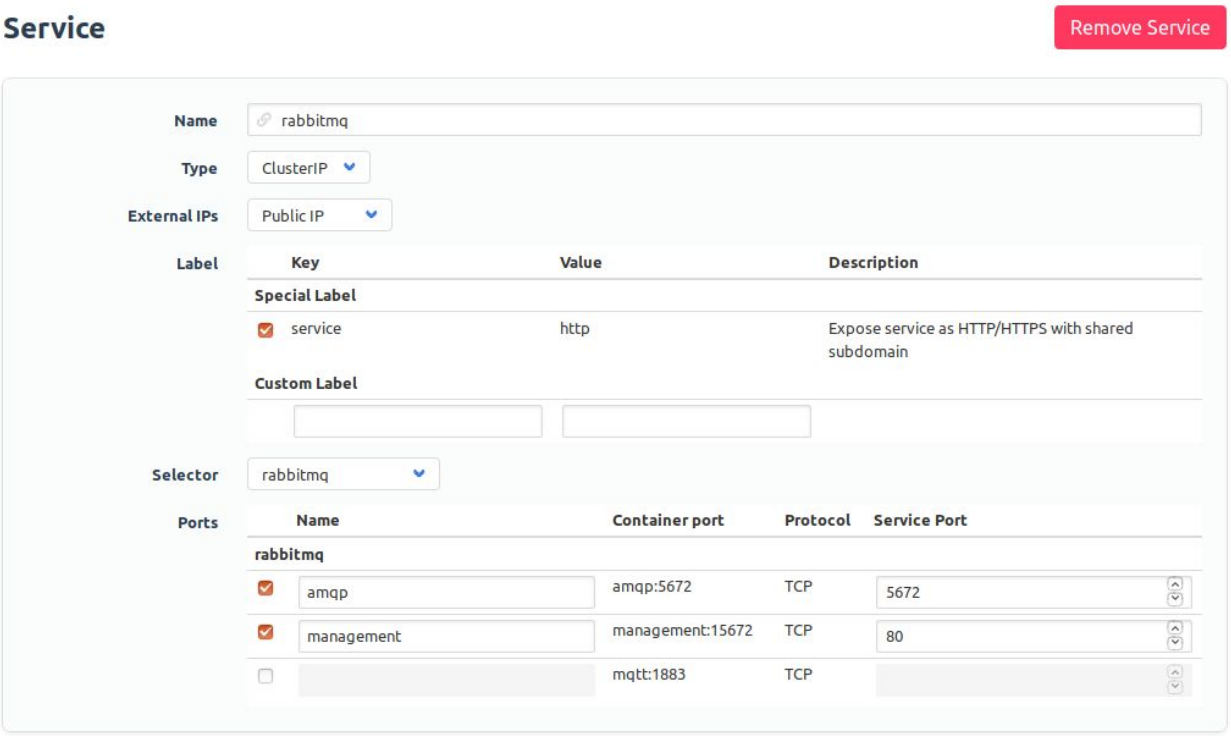
กรอกแบบฟอร์ม Service

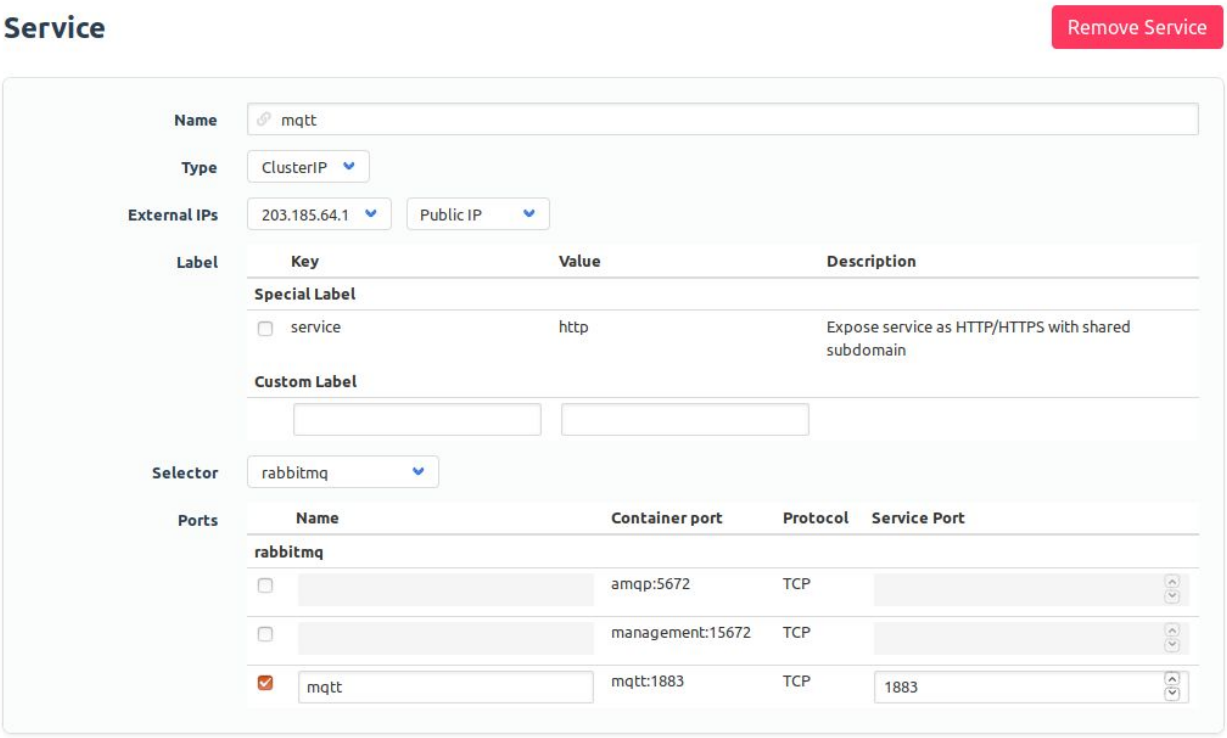
- Rabbitmq

Name	rabbitmq
Selector	rabbitmq
Ports	
Name	**Service Port**
mqtt	5672
management	80

- MQTT

Name	mqtt
Selector	rabbitmq
Ports	
Name	**Service Port**
mqtt	1883





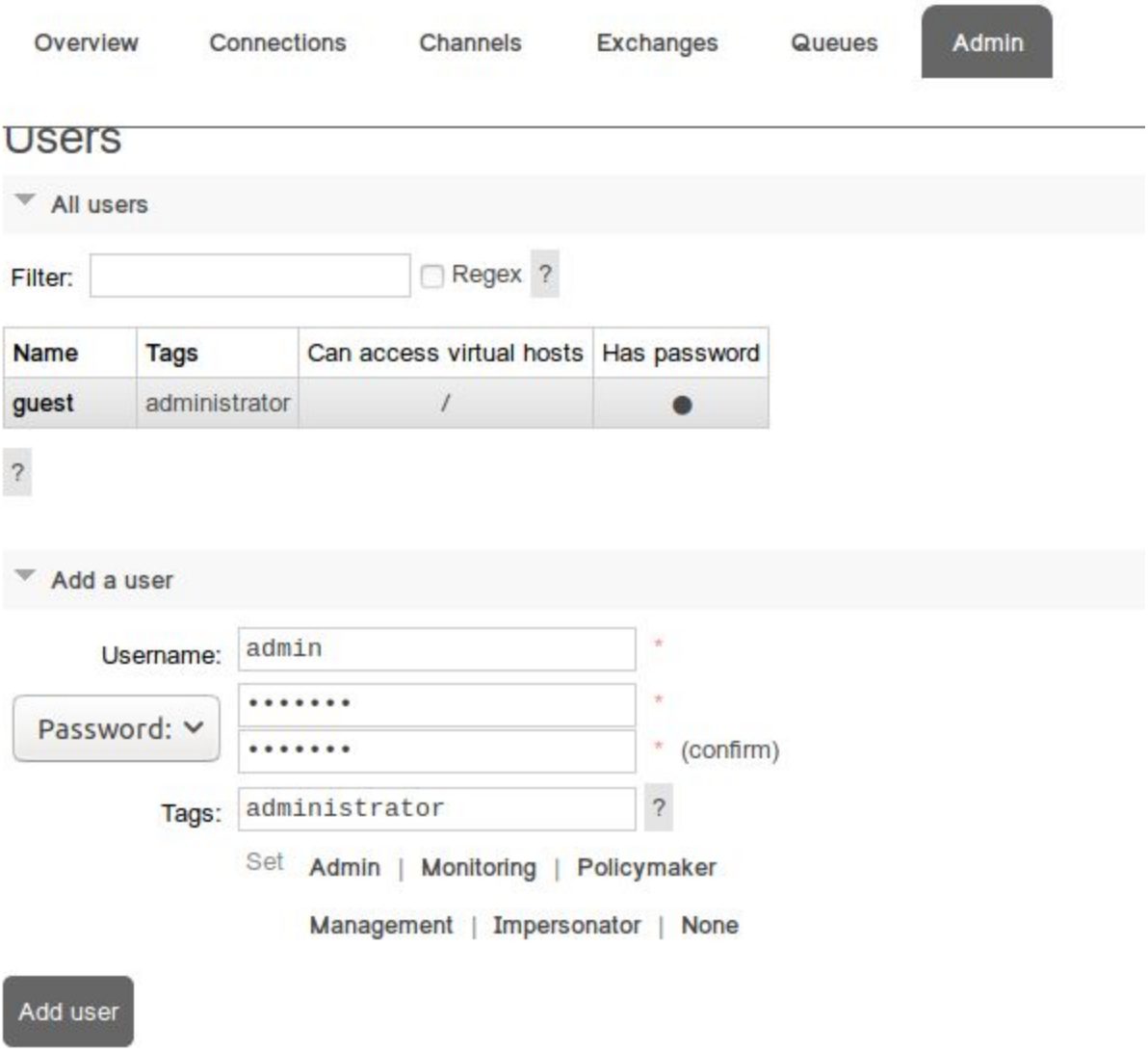
หมายเหตุ เนื่องจากมีการติดตั้ง Plugin สำหรับ MQTT ซึ่งไม่ได้มาพร้อมกับ Image ของ RabbitMQ จึงมีการสร้าง Image ใหม่ที่ติดตั้ง Plugin MQTT ไว้แล้ว

การตั้งค่า RabbitMQ

เปิด Browser เข้าไปที่หน้า Management ของ RabbitMQ ซึ่งได้มีการทำ HTTP Service เอาไว้ https://rabbitmq.<namespace>.web.meca.in.th/

Login โดยใช้ Username: guest และ Password: guest

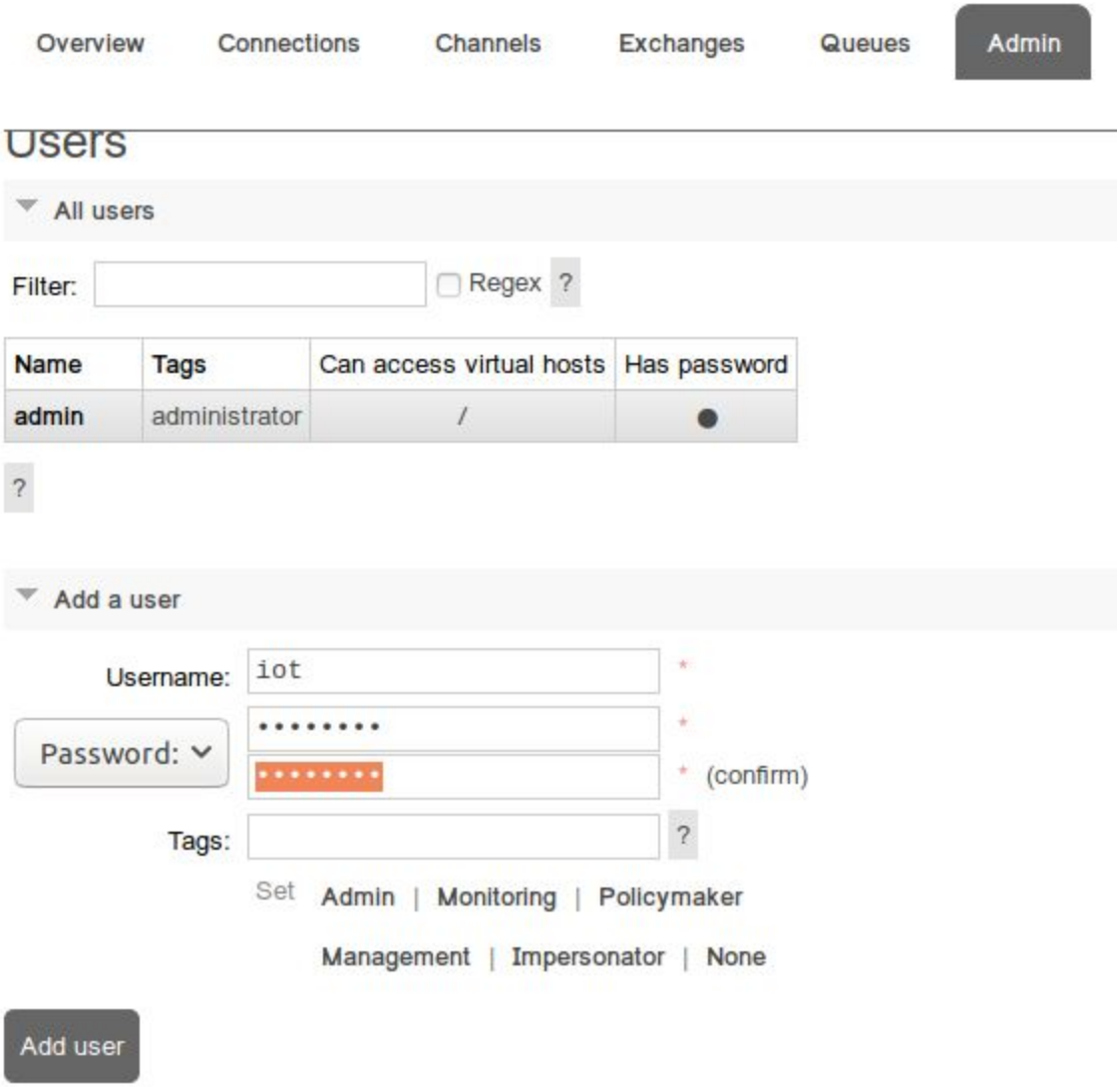
สร้าง User สำหรับผู้ดูแลระบบ ไปที่เมนู Admin > Add a user ตั้งชื่อ และ password ตามต้องการ โดยกำหนด Tags เป็น administrator



คลิกที่ชื่อ User และทำการ set permission

และลบ User: guest ออกเพื่อความปลอดภัย โดยคลิกที่ชื่อ User: guest > Delete this user > Delete แล้วทำการ Login ใหม่ โดยใช้ user ที่สร้างขึ้น

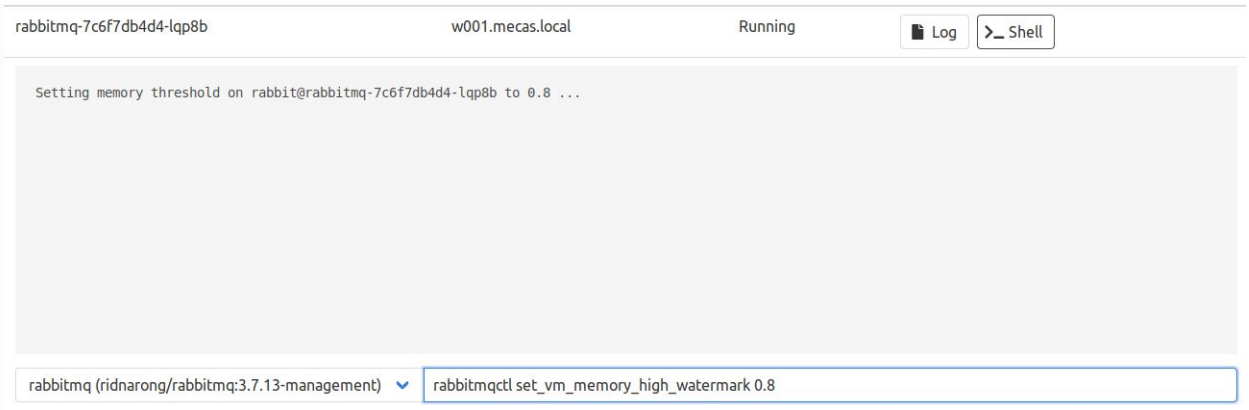
สร้าง User สำหรับส่งข้อมูล จากอุปกรณ์ IoT
ไปที่ เมนู Admin และ Add a user สร้าง Username: iot และ Password: iotthing



และทำการ set permission

เพิ่มอัตราส่วน Memory ผ่านเมนู Shell

rabbitmqctl set_vm_memory_high_watermark 0.8



3.2 การ Deploy Graphite

- **Deployment**

Deployment Name	graphite

- **Container**

Container Name	**graphite**
Image name	graphiteapp/graphite-statsd:1.1.5-10
Ports	
Name	**Port**
gui	80
carbon	2003

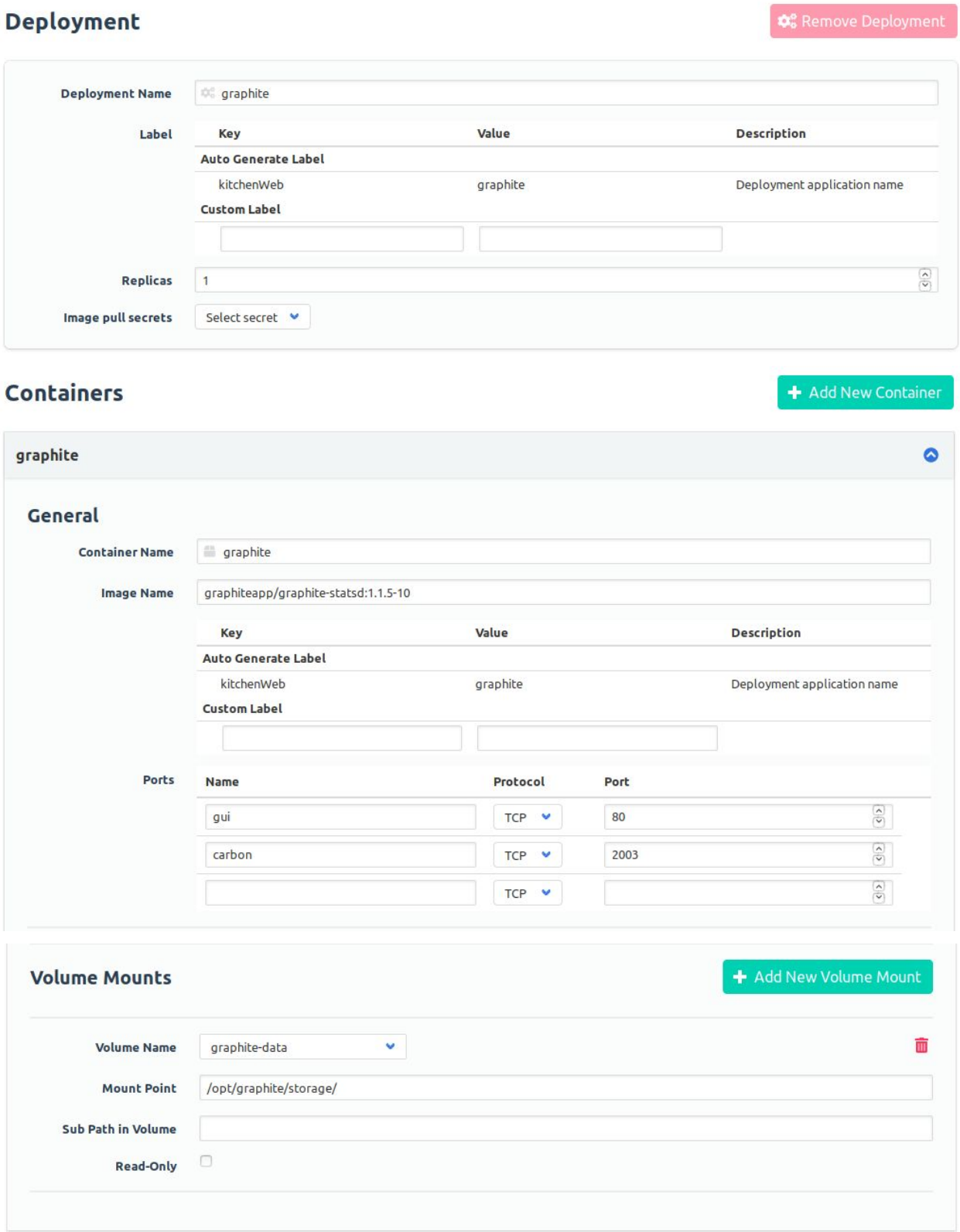
- **Volume Mounts**

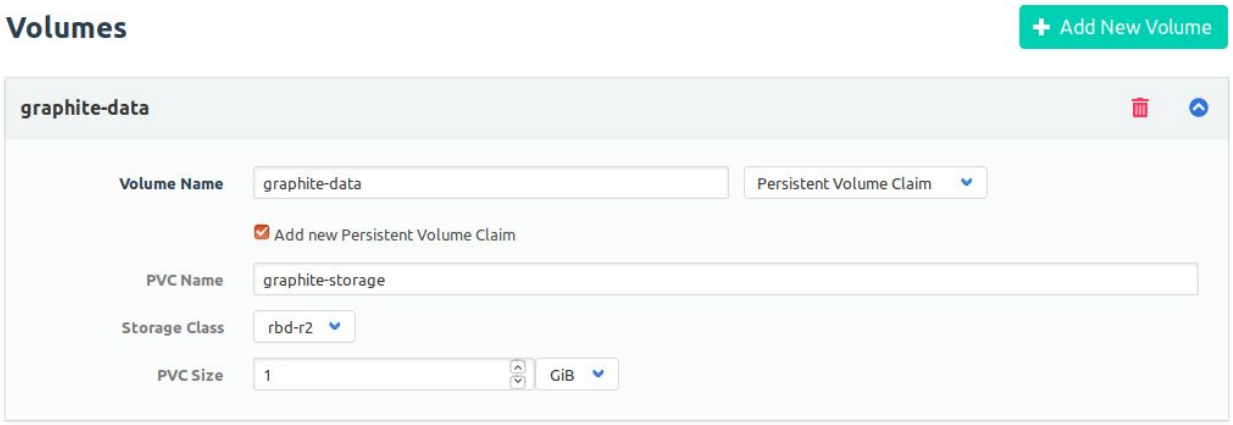
Volume Name	graphite-data
Mount Point	/opt/graphite/storage/

- **Volume**

Volume Name	graphite-data
PVC Name	graphite-storage
Storage Class	rbd-r2
PVC Size	1 GiB

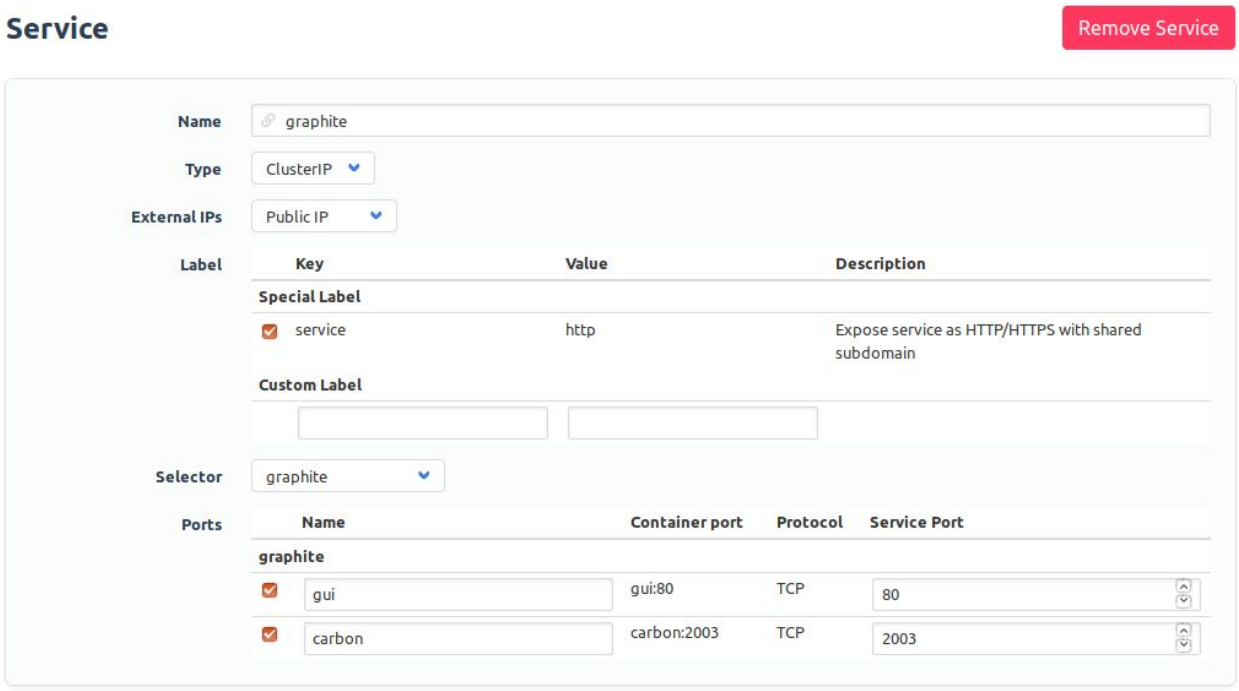
กรอกแบบฟอร์ม Deployment





กรอกแบบฟอร์ม Service

Name	graphite
Selector	graphite
Ports	
Name	**Service Port**
gui	80
carbon	2003



3.2 การ Deploy Telegraf

- **Deployment**

Deployment Name	telegraf

- **Container**

Container Name	telegraf
Image Name	telegraf:1.10.1

- **Volume Mounts**

Volume Name	telegraf-conf
Mount Point	/etc/telegraf/

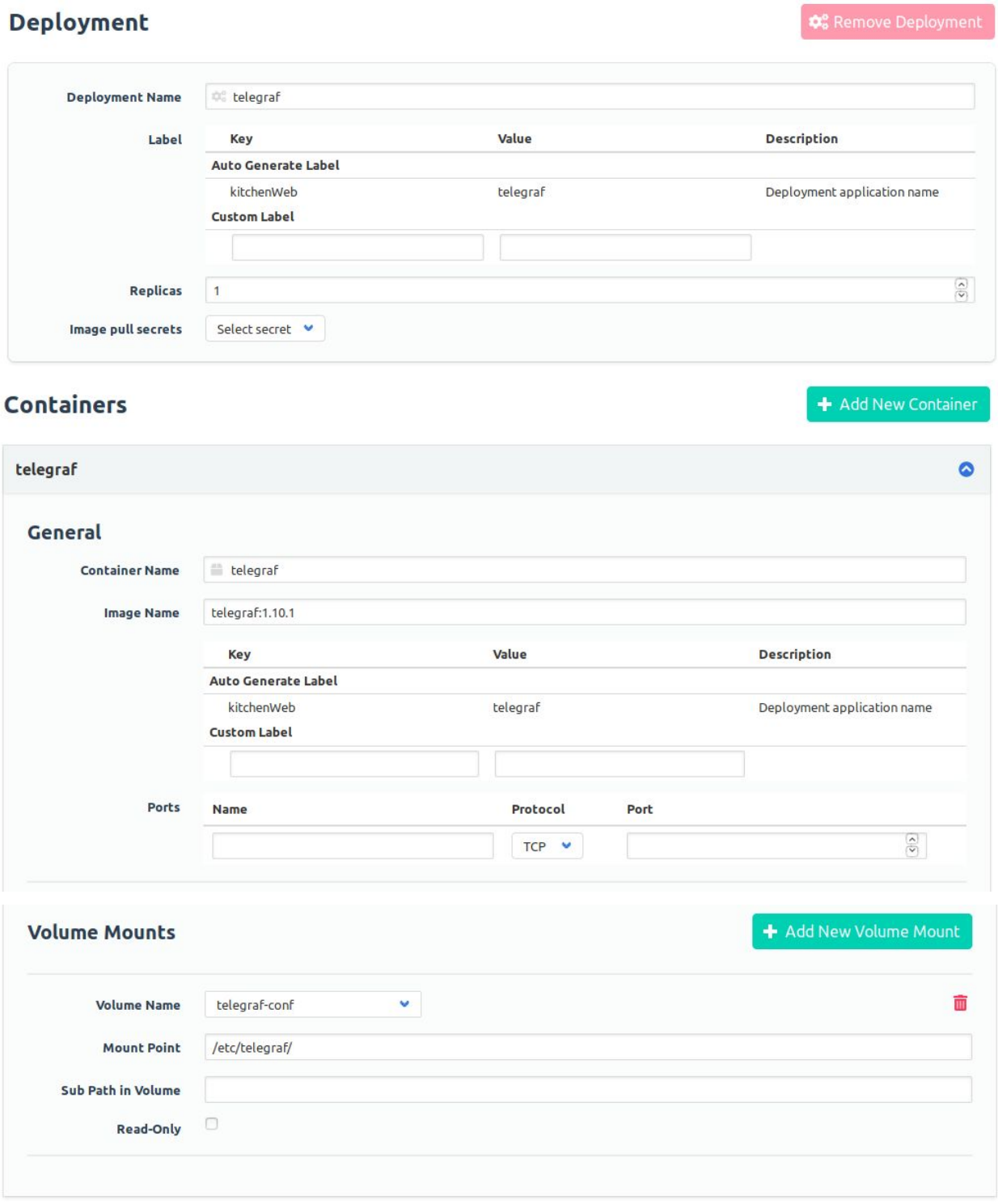
- **Volume**

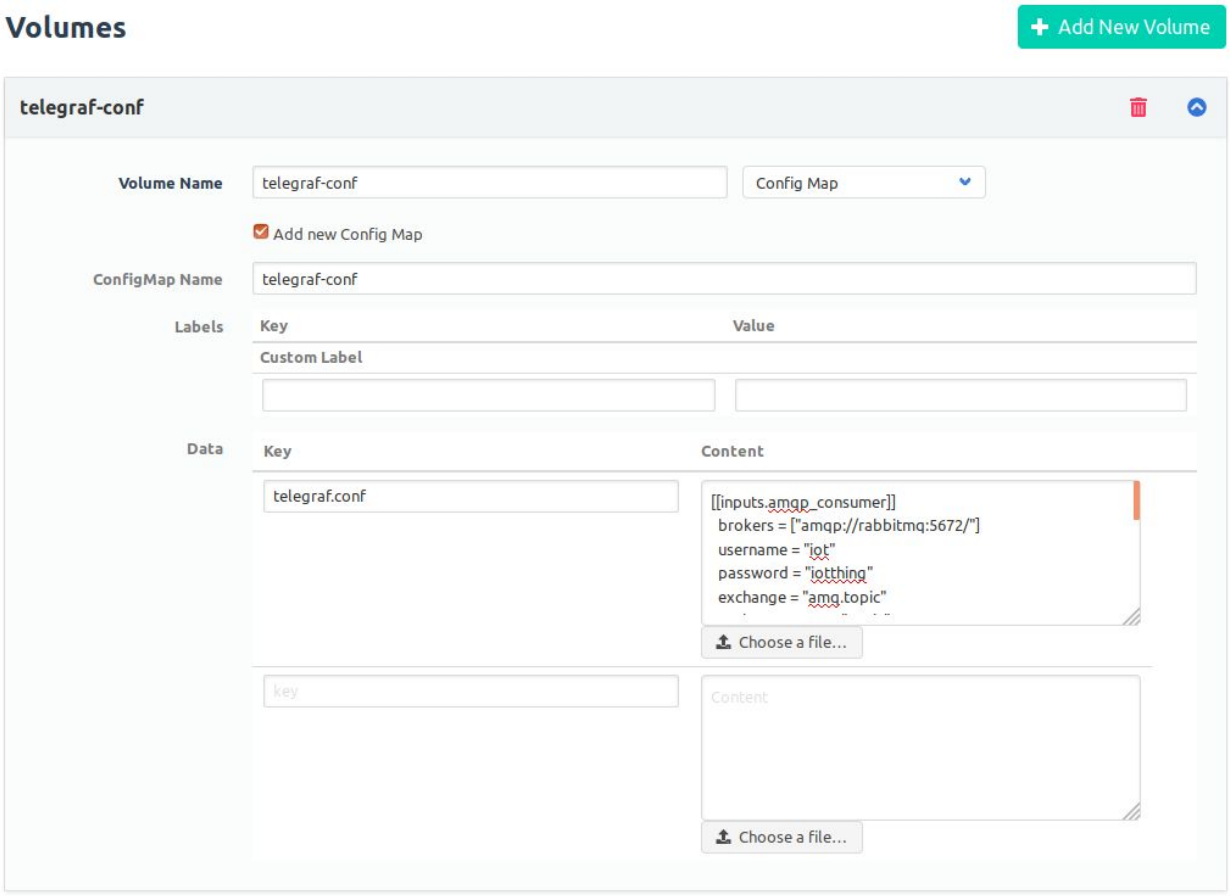
Volume Name	telegraf-conf

ConfigMap Name	telegraf-conf
Data	
Key	**Content**

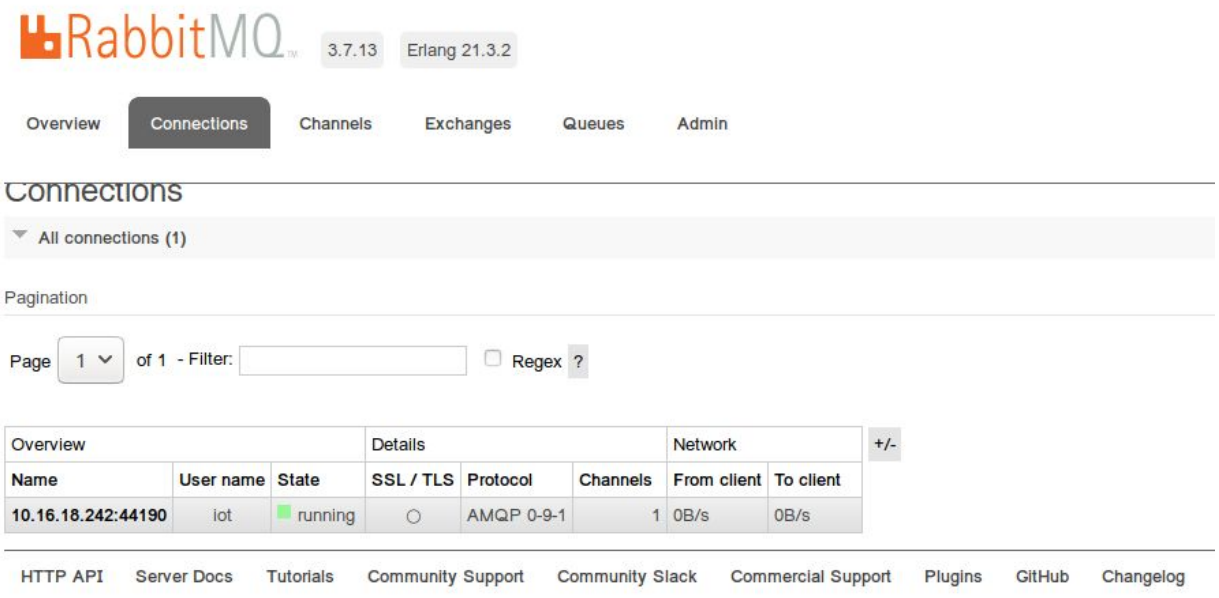
telegraf.conf	[agent]
	  omit_hostname = true
	[[inputs.amqp_consumer]]
	  name_override = "iot"
	  brokers = ["amqp://ridnarong-rabbitmq:5672/"]
	  username = "iot"
	  password = "iotthing"
	  exchange = "amq.topic"
	  exchange_type = "topic"
	  queue = "telegraf"
	  queue_durability = "durable"
	  binding_key = "iot"
	  tag_keys = ["id"]
	  data_format = "json"
	  json_time_key = "dt"
	  json_time_format = "unix"
	
	[[outputs.graphite]]
	  servers = ["ridnarong-graphite:2003"]
	  prefix = ""
	  template = "measurement.tags.field"
	  timeout = 2

กรอกแบบฟอร์ม Deployment





ตรวจสอบการการเชื่อมต่อของ telegraf ที่ RabbitMQ Management



ทดสอบส่งข้อมูล ด้วย mqtt

```
$ docker run -it --rm -e
OPENWEATHER_API_KEY=f47eeabecaa43594079886ab0fcd1508 -e
MQTT_HOST=203.185.64.1 -e MQTT_PORT=1883 -e MQTT_USERNAME=iot -e
MQTT_PASSWORD=iotthing ridnarong/owa-puller-mqtt:v1
```