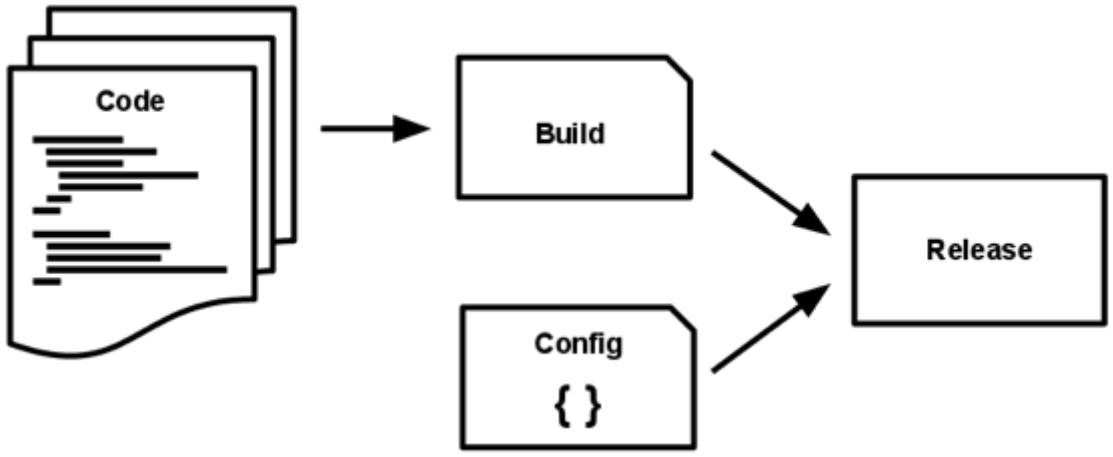
Hand on: Docker build

จากบัญญัติข้อที่ 5: Build, release, run ของ [The Twelve-Factor App]

ขั้นตอนการ Deployment จะมีสาม ระยะ คือ

1. Build คือการสร้าง Package ในลักษณะพร้อม Run ซึ่งในที่นี้คือ Docker image
2. Release คือการ รวม Package ที่ Build แล้วเข้ากับ Config
3. Run



1.ขั้นตอนการทำ Docker build to Docker Hub

1.1) สมัคร Docker hub (URL: https://hub.docker.com/) และ เข้าสู่ระบบ docker ตามที่ได้สมัครเอาไว้ > docker login (กรณีไม่สามารถเชื่อมต่อเครือข่ายอินเทอร์เน็ตได้ ให้ทำการ รีสตาร์ท Docker ใหม่)

```
$ docker login
```

1.2) เริ่มเขียน mysite/Dockerfile โดยสร้างไฟล์ text โดยตั้งชื่อเป็น “Dockerfile” และเขียนคำสั่งลงไปเป็นขั้นตอน 7 step ดังต่อไปนี้

```
FROM python:3
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 8000
CMD [ "python", "manage.py", "runserver", "0.0.0.0:8000"]
```

1.3) จากนั้นก็ทำการสร้างอิมเมจขึ้นมาด้วยคำสั่ง docker build -t mysite . (กรณีไม่สามารถเชื่อมต่อเครือข่ายอินเทอร์เน็ตได้ ให้ทำการ รีสตาร์ท Docker ใหม่)

```
$ docker build -t mysite .
```

1.4) ทำการตรวจสอบไฟล์อิมเมจที่ได้ถูกสร้างขึ้นด้วยคำสั่ง>docker images

```
$ docker images
REPOSITORY        TAG          IMAGE ID          CREATED           SIZE
mysite          latest         df287d7afbc2      2 minutes ago     1.01GB
```

ทดลอง run บน ubuntu

```
$ docker run --link some-mariadb:mysql -e MYSQL_DATABASE=mysite -e
```

```
MYSQL_USER=user -e MYSQL_PASSWORD=password -e MYSQL_HOST=mysql -e
MYSQL_PORT=3306 -p 8000:8000 -it --rm mysite
```

เปิดผ่าน browser http://localhost:8000

ทดลอง run บน windows

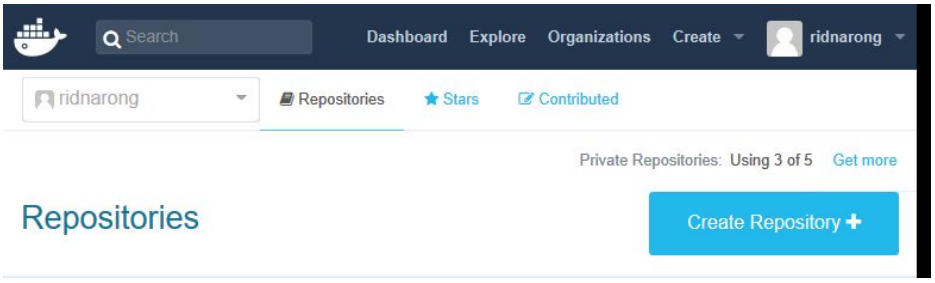
```
$ docker run --net host -e MYSQL_DATABASE=mysite -e MYSQL_USER=user -e
MYSQL_PASSWORD=password -e MYSQL_HOST=127.0.0.1 -e MYSQL_PORT=3306 -it
--rm mysite
```

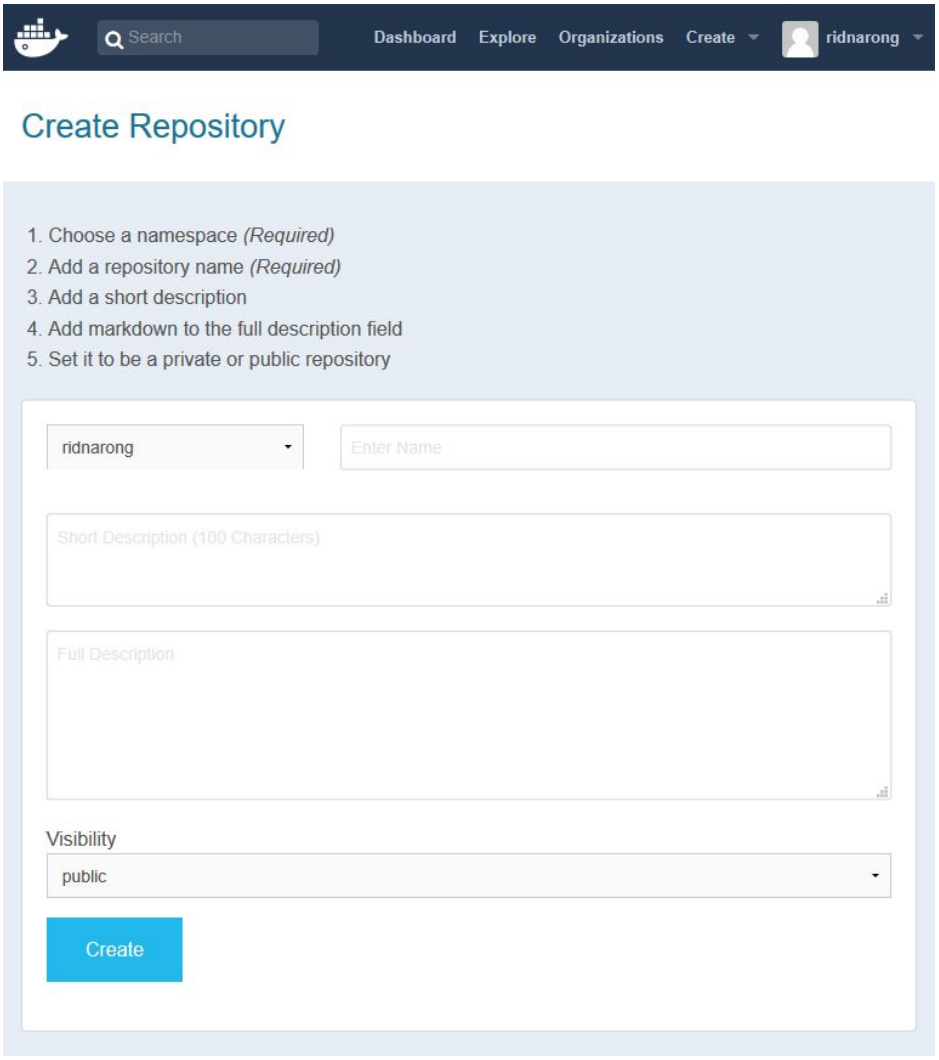
เปิดผ่าน browser http://192.168.99.100:8000/

1.5) Commit change to Code base

```
$ git status
$ git add Dockerfile
$ git commit -m "Add Dockerfile"
$ git push origin master
```

1.6) ทำการเผยแพร่สาธารณะสู่ Docker Hub โดยการสร้าง Repository ขึ้นมา

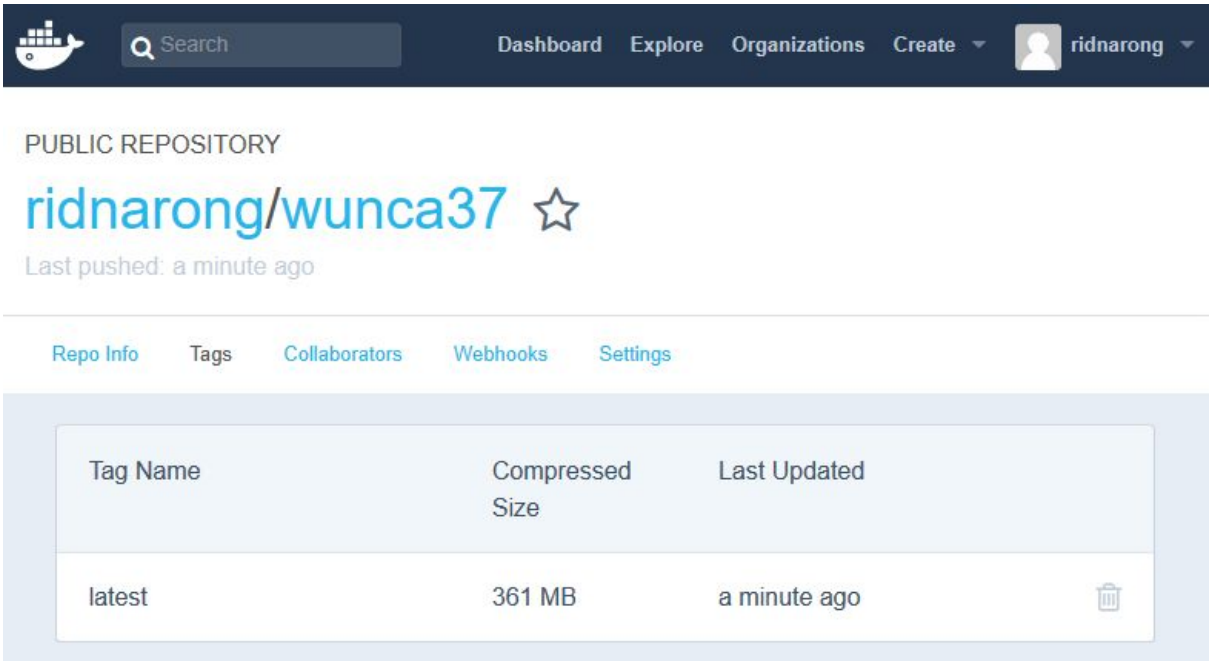




1.7) นำไฟล์อิมเมจลงสู่ Repository ที่ได้สร้างขึ้นมาเพื่อเข้าสู่ docker hub :Alias Docker image name and push to docker hub

```
$ docker tag mysite ridnarong/wunca37
$ docker push ridnarong/wunca37
```

1.8) ตรวจสอบไฟล์อิมเมจใน docker hub: Check on docker hub



2 การทำ Docker compose เพื่อควบคุมการทำงานด้วยชุดคำสั่ง

Docker นั้นสามารถใช้ Docker Engine จัดการ Container แต่ละตัวโดยใช้คำสั่ง docker run | start | stop | rm แต่หากโปรเจคของผู้ใช้มีความประสงค์ใช้งานหลายๆ containers เพื่อที่ทำงานร่วมกัน ผู้ใช้ก็เพียงแค่เขียนโค๊ดคำสั่งให้ Containers สามารถทำงานได้ทีเดียว โดยการใช้ Docker Compose สั่ง start | stop | restart | log | build โดยใช้คำสั่งเพิ่งแค่บรรทัดเดียว ในจัดการหลายๆ containers พร้อมๆ กันได้ ผ่านการเขียนคำสั่งต่างๆ เอาไว้ในไฟล์ docker-compose.yml ซึ่งปัจจุบันได้พัฒนามาอยู่ใน Version 3 แล้วสั่งใช้งานผ่านคำสั่ง docker-compose

```
version: '3'

services:
  db:
       image: mariadb
       environment:
       - MYSQL_ROOT_PASSWORD=my-secret-pw
       - MYSQL_DATABASE=mysite
       - MYSQL_USER=user
       - MYSQL_PASSWORD=password
  web:
       image: ridnarong/wunca37
       command: python3 manage.py runserver 0.0.0.0:8000
       ports:
       - "8000:8000"
       environment:
       - MYSQL_HOST=db
       - MYSQL_PORT=3306
       - MYSQL_DATABASE=mysite
       - MYSQL_USER=user
       - MYSQL_PASSWORD=password
```

```
    depends_on:
    - db
```

```
$ docker-compose run web python manage.py migrate
```

```
$ docker-compose up
```

```
$ docker-compose down
```

จากตัวอย่าง เนื่องจาก เราได้สร้าง container ของ mariadb ใหม่ โดยที่ไม่ได้ทำการ mount volume เข้าไป จึงจำเป็นจะต้องทำการ migrate database อีกครั้ง

และเพื่อให้ เป็นไปตาม **บัญญัติข้อที่ 10. Dev/prod parity** จึงแนะนำให้ใช้การพัฒนาบนสภาพแวดล้อม เดียวกับที่ทำงานบน Production ซึ่ง container ที่เราพัฒนาขึ้น จะต้องถูกทำงานบน Linux server โดยใช้ Docker image เพราะฉะนั้น ขณะที่เราพัฒนาจึงควรพัฒนา บนสภาพแวดล้อม ของ container ด้วย โดยเราสามารถ mount ในส่วนของ source code ที่เรากำลังพัฒนา เข้าไปใน Contaienr เพื่อให้เราสามารถ พัฒนาได้อย่างง่ายดาย และลดขั้นตอนของการทำ docker image

บน ubuntu

```
$ docker run --link some-mariadb:mysql -e MYSQL_DATABASE=mysite -e
MYSQL_USER=user -e MYSQL_PASSWORD=password -e MYSQL_HOST=mysql -e
MYSQL_PORT=3306 -p 8000:8000 -it --name django_dev -v
"${PWD}/mysite":/usr/src/app mysite
```

เปิดผ่าน browser http://localhost:8000

บน windows

```
$ docker run --net host -e MYSQL_DATABASE=mysite -e MYSQL_USER=user -e
MYSQL_PASSWORD=password -e MYSQL_HOST=127.0.0.1 -e MYSQL_PORT=3306 -it
--name django_dev -v /c/Users/Nectec/mysite:/usr/src/app mysite
```

เปิดผ่าน browser http://192.168.99.100:8000/

ผ่าน docker-compose

```
version: '3'

services:
  db:
      image: mariadb
      environment:
      - MYSQL_ROOT_PASSWORD=my-secret-pw
      - MYSQL_DATABASE=mysite
      - MYSQL_USER=user
```

```
      - MYSQL_PASSWORD=password
  web:
      image: ridnarong/wunca37
      command: python3 manage.py runserver 0.0.0.0:8000
      ports:
      - "8000:8000"
      volumes:
      - "/c/Users/Nectec/mysite:/usr/src/app"
      environment:
      - MYSQL_HOST=db
      - MYSQL_PORT=3306
      - MYSQL_DATABASE=mysite
      - MYSQL_USER=user
      - MYSQL_PASSWORD=password
      depends_on:
      - db
```

```
$ docker-compose up
```

```
$ docker-compose run web python manage.py migrate
```

```
$ docker-compose stop
```

```
$ docker-compose start
```

การ Execute command ใน container

```
$ docker-compose exec web pip freeze
```

เมื่อพัฒนา และทดสอบจนเรียบร้อยแล้ว สามารถ docker build อีกครั้ง และ push ขึ้นไปเก็บยัง repository เพื่อนำไป deploy ต่อไป