

# Hand on: Create new Django application

Adapt from: <https://docs.djangoproject.com/en/2.0/intro/>

## 1. ลงมือปฏิบัติ: สร้างแอปพลิเคชันใหม่ Django

### 1.1) การสร้าง: Create new App

```
$ python manage.py startapp polls
```

1.2) ทำการสร้าง view แรก (เข้าไปแก้ไขโดยใช้ Text Editor) ตามที่อยู่ polls และไฟล์ views.py โดยการพิมพ์โค้ดคำสั่งดังนี้

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

1.3) ทำการสร้าง Map view with URL โดยการสร้างไฟล์ .py ที่ชื่อ urls ลงตามที่อยู่ polls/urls.py จากนั้นทำการเพิ่มโค้ดคำสั่งดังต่อไปนี้

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

1.4) ทำการสร้าง Map view with URL โดยการเพิ่มโคดลงไปตามที่อยู่ mysite: mysite/urls.py จากนั้นทำการเพิ่มโค้ดคำสั่งดังต่อไปนี้

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

### 1.5) ทำการเช็คสถานะเซิร์ฟเวอร์ Start Django server to check

```
$ python manage.py runserver
```

โดยไปยังบราวเซอร์และเข้าสู่ Url: <http://localhost:8000/polls/>



Hello, world. You're at the polls index.

### 1.6) ตรวจสอบข้อมูล Check DB config in mysite: mysite/settings.py

```
# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

### 1.7) ทำการ Run migration โดยใช้คำสั่ง python manage.py migrate

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

### 1.8) Create model for Polls: polls/models.py

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    opened = models.BooleanField(default=True)

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

### 1.9) Install app to project: mysite/settings.py เพิ่มในส่วนของ polls.apps.PollsConfig

```
INSTALLED_APPS = [  
    'polls.apps.PollsConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

### 1.10) Make Model Migrations

```
$ python manage.py makemigrations polls  
Migrations for 'polls':  
  polls\migrations\0001_initial.py  
    - Create model Choice  
    - Create model Question  
    - Add field question to choice
```

### 1.11) Migrate Model to Database

```
$ python manage.py migrate  
Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, polls, sessions  
Running migrations:  
  Applying polls.0001_initial... OK
```

### 1.12) Django interactive shell

```
$ python manage.py shell  
Python 3.6.6rc1 (v3.6.6rc1:1015e38be4, Jun 12 2018, 08:38:06) [MSC  
v.1900 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
(InteractiveConsole)  
>>> from polls.models import Choice, Question  
>>> Question.objects.all()  
<QuerySet []>  
>>> from django.utils import timezone  
>>> q = Question(question_text="What's new?", pub_date=timezone.now())  
>>> q.save()  
>>> q.id  
1  
>>> q.question_text  
"What's new?"  
>>> q.question_text = "What's up?"  
>>> q.save()  
>>> Question.objects.all()  
<QuerySet [<Question: Question object (1)>]>  
>>> quit()
```

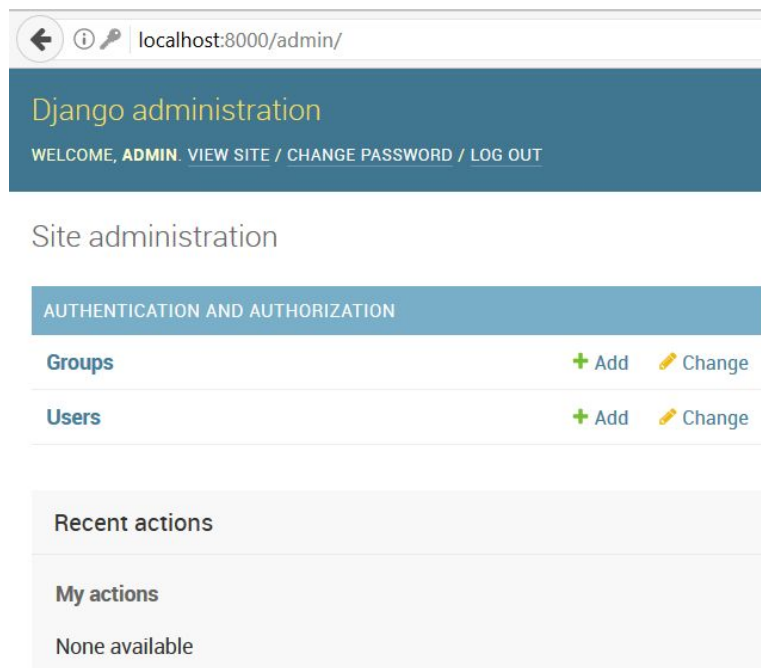
### 1.13) Create super user

```
$ python manage.py createsuperuser
Username (leave blank to use 'nectec'): admin
Email address: admin@example.com
Password:
Password (again):
Superuser created successfully.
```

### 1.14) Start Django server to check

```
$ python manage.py runserver
```

### 1.15) Go to browser and enter: <http://localhost:8000/admin/>



รูปที่ 1 แสดงหน้าต่าง admin

### 1.16) Make Choice, Question visible to admin page: polls/admin.py

```
from django.contrib import admin
from .models import Question, Choice

admin.site.register(Question)
admin.site.register(Choice)
```

### 1.17) Check in terminal Django server will detect change and auto reload

```
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

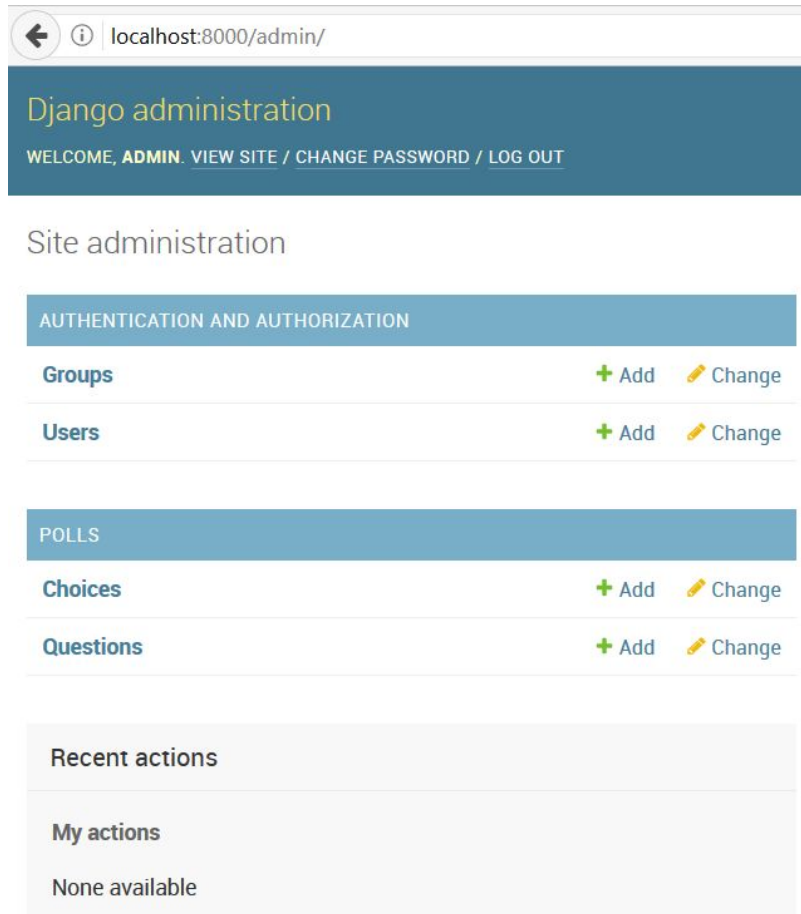
```
June 27, 2018 - 12:50:40
```

```
Django version 2.0.6, using settings 'mysite.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CTRL-BREAK.
```

### 1.18) Refresh browser



The screenshot shows the Django administration interface in a browser. The address bar displays 'localhost:8000/admin/'. The page title is 'Django administration' and the user is logged in as 'ADMIN'. The main content area is titled 'Site administration' and is divided into several sections. The 'AUTHENTICATION AND AUTHORIZATION' section contains 'Groups' and 'Users', each with '+ Add' and 'Change' links. The 'POLLs' section contains 'Choices' and 'Questions', each with '+ Add' and 'Change' links. At the bottom, there are sections for 'Recent actions' and 'My actions', both showing 'None available'.

รูปที่ 2 เมื่อรีเฟรชหน้าจอก็กรอบหัวข้อ POLLs จะแสดงขึ้นมา

## บัญชี 12 ประการ สำหรับช่วยวางโครงการสำหรับการสร้าง Cloud Native Application (Web Application)

### บัญชีข้อที่ 1: Codebase

การจัดการ Version ของ Code แนะนำ Git และให้ฝากไว้บนอินเทอร์เน็ต หรือ Git Server ขององค์กร

สิ่งที่จะต้องเตรียมคือ สมัคร Gitlab และ Github

-Free private git repository Gitlab (แนะนำ URL: <https://about.gitlab.com/>)

-Create .gitignore (<https://github.com/jpadilla/django-project-template/blob/master/.gitignore>)

ขั้นตอนการวางโปรเจกต์ลงสู่ Gitlab มีดังนี้

1.การ Push project ลงสู่ Gitlab ด้วยคำสั่งดังต่อไปนี้

```
$ git config --global user.email "email@example.com"

$ git config --global user.name "Mona Lisa"

$ git t
Initialized empty Git repository in C:/Users/Nectec/mysite/.git/

$ git remote add origin https://gitlab.com/ridnarong/wunca37.git

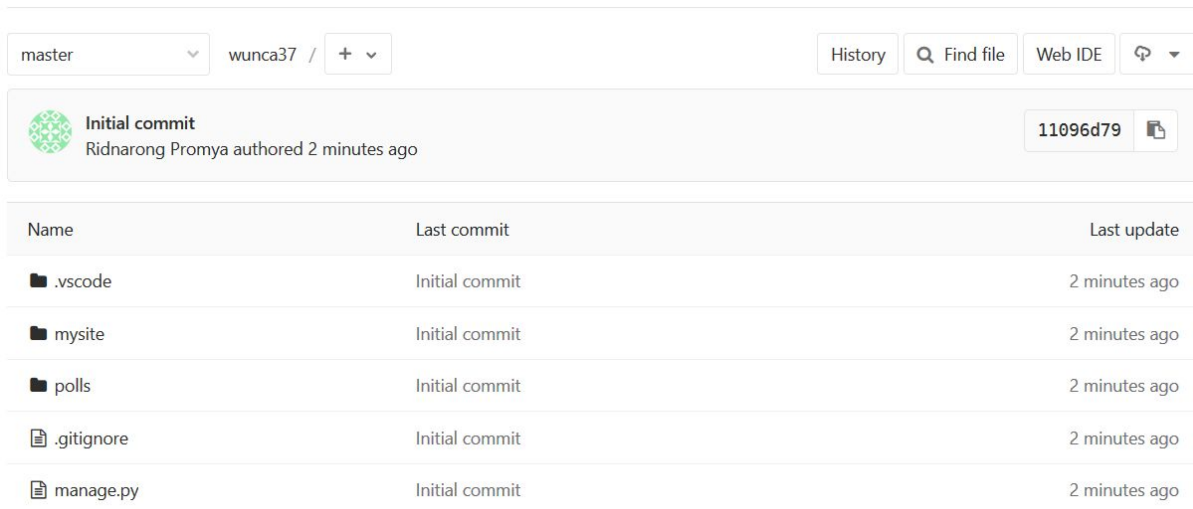
$ git add .

$ git commit -m "Initial commit"
[master (root-commit) 11096d7] Initial commit
16 files changed, 337 insertions(+)
create mode 100644 .gitignore
create mode 100644 .vscode/settings.json
create mode 100644 manage.py
create mode 100644 mysite/__init__.py
create mode 100644 mysite/settings.py
create mode 100644 mysite/urls.py
create mode 100644 mysite/wsgi.py
create mode 100644 polls/__init__.py
create mode 100644 polls/admin.py
create mode 100644 polls/apps.py
create mode 100644 polls/migrations/0001_initial.py
create mode 100644 polls/migrations/__init__.py
create mode 100644 polls/models.py
create mode 100644 polls/tests.py
create mode 100644 polls/urls.py
create mode 100644 polls/views.py

$ git push -u origin master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (18/18), done.
```

```
Writing objects: 100% (20/20), 4.77 KiB | 1.19 MiB/s, done.
Total 20 (delta 0), reused 0 (delta 0)
To https://gitlab.com/ridnarong/wunca37.git
* [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

## 2. ตรวจสอบ Gitlab ว่าไฟล์ได้ถูกสร้างขึ้นแล้ว



Initial commit  
Ridnarong Promya authored 2 minutes ago  
11096d79

Name	Last commit	Last update
📁 .vscode	Initial commit	2 minutes ago
📁 mysite	Initial commit	2 minutes ago
📁 polls	Initial commit	2 minutes ago
📄 .gitignore	Initial commit	2 minutes ago
📄 manage.py	Initial commit	2 minutes ago

รูปที่ 3 ไฟล์ใน mysite ได้ถูก push ลง Gitlab

## ปัญญุดีข้อที่ 2: Dependencies

บันทึก Dependencies ไว้ในรูปแบบของไฟล์

```
$ pip freeze
astroid==1.6.5
colorama==0.pyty==1.3.1
mccabe==0.6.1
mysqlclient==1.3.12
pylint==1.9.2
pytz==2018.4
six==1.11.0
wrapt==1.10.11

$ pip freeze > requirements.txt
```



## ปัญหาที่ข้อที่ 4. Backing services

แยก Service ออกมาต่างหากจาก App

ปรับ Database config ให้ mysql ภายนอก mysite/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydatabase',
        'USER': 'mydatabaseuser',
        'PASSWORD': 'mypassword',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

สร้าง MySQL server จาก Docker

ทำการดึงอิมเมจ mariadb ก่อนโดยใช้คำสั่ง docker pull mariadb

```
(myproject) C:\Users\PKL10012017>docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
1c7fe136a31e: Pull complete
ecb0394ab02f: Pull complete
ece3821473b4: Pull complete
b090b58fe851: Pull complete
fdc67e5246f1: Pull complete
7b0408b2b91f: Pull complete
02c7ce6b6eea: Pull complete
eb3d50d2d52c: Pull complete
800dafb39329: Pull complete
b98513c56899: Pull complete
f7407e397090: Pull complete
Digest: sha256:f2085c2176ba6294cf73033b344a420faa2ddae1b97b6795c101552e86284ba3
Status: Downloaded newer image for mariadb:latest
```

การสร้าง MySQL server บน Ubuntu

```
$ mkdir mysql-datadir
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -e
MYSQL_DATABASE=mysite -e MYSQL_USER=user -e MYSQL_PASSWORD=password -v
/home/test/mysql-datadir:/var/lib/mysql -p 3306:3306 mariadb
```

การสร้าง MySQL server บน Window หรือ PowerShell

```
$ mkdir mysql-datadir
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -e
MYSQL_DATABASE=mysite -e MYSQL_USER=user -e MYSQL_PASSWORD=password -v
/c/Users/Nectec/mysql-datadir:/var/lib/mysql --net host mariadb:5 mysqld
--innodb-flush-method=littlesync --innodb-use-native-aio=OFF
--log_bin=ON --binlog_format=row
```

การ mount folder จากเครื่อง host ที่เป็น windows เข้าไปใน vm ของ docker toolbox จะถูกทำให้เข้าถึงได้ที่ /c/ แต่เนื่องจาก ไฟล์อยู่บนระบบของ windows หากใช้ Docker for Windows สามารถใช้ c:/Users/Nectec/mysql-datadir ได้หากได้มีการ Shared drive ไว้แล้ว ทำให้บางโปรแกรมไม่รองรับ การอ่าน/เขียนไฟล์ได้ หรือต้องมีการตั้งค่าพิเศษ

### ปัญหาข้อที่ 3. Config

เก็บ config ไว้ใน environment

1. ปรับ Database config ให้ Environment Variable: mysite/settings.py

```
import os

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.environ['MYSQL_DATABASE'],
        'USER': os.environ['MYSQL_USER'],
        'PASSWORD': os.environ['MYSQL_PASSWORD'],
        'HOST': os.environ['MYSQL_HOST'],
        'PORT': os.environ['MYSQL_PORT'],
    }
}
```

2. สร้างไฟล์ .env เพื่อเก็บ ข้อมูล config ต่างๆ ต้องระหว่าง อย่าให้ อยู่ใน version control (กำหนดไว้ใน .gitignore)

On Ubuntu .env file

```
export MYSQL_HOST=127.0.0.1
export MYSQL_PORT=3306
export MYSQL_DATABASE=mysite
export MYSQL_USER=user
export MYSQL_PASSWORD=password
```

On Window env.bat file

```
set MYSQL_HOST=127.0.0.1 #For docker for Windows
set MYSQL_HOST=192.168.99.100 #For docker toolbox
set MYSQL_PORT=3306
set MYSQL_DATABASE=mysite
set MYSQL_USER=user
set MYSQL_PASSWORD=password
```

3. เรียก ใช้งาน

On Ubuntu

```
$ . .env
```

On Window

```
$ env.bat
```

4. Migrate ข้อมูล ลงฐานข้อมูล และ สร้าง superuser (หากยังไม่ได้ทำการรันเซิร์ฟเวอร์ก็ให้ทำการเริ่มคอนเทนเนอร์ด้วยคำสั่ง `docker start some-mariadb` ในอีกหน้าต่างของ Command Prompt ก่อน สามารถเช็คสถานะได้โดยคำสั่ง `>docker ps` ได้ในหน้า terminal ที่จะทำการ Migrate)

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying polls.0001_initial... OK
  Applying sessions.0001_initial... OK
$ python manage.py createsuperuser
Username (leave blank to use 'test'): admin
Email address: admin@example.com
Password:
Password (again):
Superuser created successfully.
$ python manage.py runserver
```

## บัญญัติข้อที่ 5. Build, release, run

Hand on: Docker build

## บัญญัติข้อที่ 6. Processes

พิจารณาไม่ให้แอปจะต้องมีการเก็บ state ไว้ใน process ของตัวเอง และถ้า App จะต้องใช้งานร่วมกันระหว่างหลาย process ให้ใช้ IV. Backing services เข้ามาช่วยเช่น Database เช่น Session สำหรับ Django สามารถเลือกที่เก็บ session ไว้ใน Database ได้ด้วยการเปิด 'django.contrib.sessions' ใน INSTALLED\_APPS ไฟล์ `mysite/settings.py`

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
```

```
]
```

ระบบไฟล์ เช่น Application ของเรา มีการให้ User upload file มาหรือไม่ ถ้าเป็นเช่นนั้น ให้ พิจารณา แยก เก็บไฟล์เหล่านั้น ไว้ใน storage ต่างหาก ซึ่ง Docker ก็สามารถ Mount Path จาก Host เข้าไปใน Container ได้ เพิ่มแยกที่เก็บไฟล์ไว้ต่างหาก เช่น การสร้าง MySQL server เราได้ทำการ mount path mysql-datadir ไปยัง folder ที่เก็บข้อมูลของ MySQL /var/lib/mysql ผ่าน parameter -v

```
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -e MYSQL_DATABASE=mysite -e MYSQL_USER=user -e MYSQL_PASSWORD=password -v /home/test/mysql-datadir:/var/lib/mysql -p 3306:3306 mariadb
```

## ปัญญุดีข้อที่ 7. Port binding

การเข้าถึง service โดยการให้ Port binding ซึ่งเป็นรูปแบบทั่วไป ในการเข้าถึง บริการ โดยการสร้าง Application หากมีการให้เข้าถึง ปกติ ควรจะต้องเข้าถึงได้ผ่าน Port และเป็นลักษณะที่จับในตัวเอง เช่น PHP ที่จะต้องอาศัย Web server ในการให้บริการ จำเป็นจะต้อง ออกให้บริการในรูปแบบผ่าน Port ได้เลย จึงควรรวม Web server เข้ามาใน Application ด้วยเลย แต่ในตัวอย่างนี้ Django มี Built-in Web server (python manage.py runserver) ซึ่งสามารถให้บริการผ่านเว็บได้ทันที ด้วยตัวเอง และสำหรับการทำงานในระดับ production python ก็มีส่วนเสริม ที่ทำหน้าที่เป็น web server แยกได้อีก เช่นกัน

```
$ python manage.py runserver
Performing system checks...

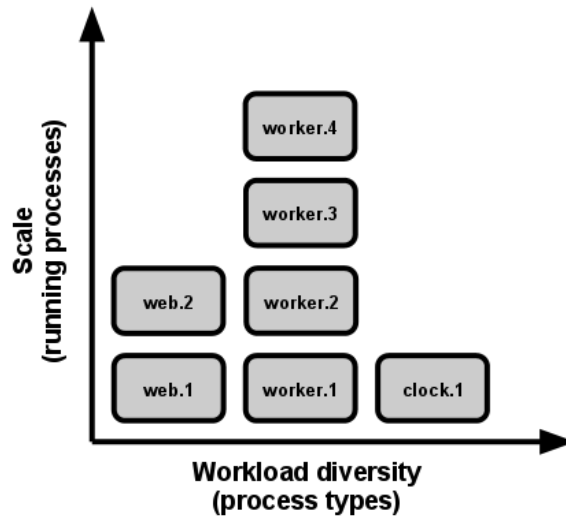
System check identified no issues (0 silenced).
June 28, 2018 - 13:00:54
Django version 2.0.6, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

นอกจากนี้แล้ว Docker ยังช่วยเรื่องการ Bind port ขณะ run ให้มีความยืดหยุ่น จากการสร้าง MySQL จะเห็นว่า มีการเพิ่ม parameter -p 3306:3306 หมายถึง การ Binding port ของเครื่อง host (3306) กับ ภายใน container (3306) ช่วยให้สามารถกำหนด Port ใช้งานได้ ขณะ run เช่น run หลายๆ เว็บ server ได้พร้อมกัน แม้ใน container จะระบุ port 80 แต่ ก็สามารถ binding port ขณะ run ได้แยกจากได้หลาย Port

```
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -e MYSQL_DATABASE=mysite -e MYSQL_USER=user -e MYSQL_PASSWORD=password -v /home/test/mysql-datadir:/var/lib/mysql -p 3306:3306 mariadb
```

## ปัญญุดีข้อที่ 8. Concurrency

จากที่เราได้มีการพิจารณาให้ Application ของเรา ตามข้อ VI. Processes จุดมุ่งหมาย คือ การพยายามทำให้ App สามารถ scaling ได้โดยการเพิ่มจำนวน ตามจำนวน process ซึ่งหาก App ของเรา มีความเป็น stateless และ shared nothing ก็จะทำให้การขยายตัวเป็นไปได้ง่าย ข้อพิจารณาคือ App เราสามารถ run หลายๆ process เพื่อรองรับ Load ได้หรือไม่



รูปที่ 4 เปรียบขนาดของ running process และ process types

ซึ่งมีข้อดี คือ เป็น Horizontal scale หากทรัพยากรของเครื่องใดเครื่องหนึ่งไม่พอ สามารถกระจายงาน ไปสู่เครื่องอื่นๆ ได้ ลักษณะของ run แบบ process ที่แนะนำหรือ run เป็น foreground สอดคล้องกับ ลักษณะการทำงาน ของ Container และการเปิดปิด ควบคุม จากระบบดูแลภายนอก

### บัญญัติข้อที่ 9. Disposability

Application จะต้องสามารถ เริ่มทำงานอย่างรวดเร็ว และ ปิดการทำงานได้อย่างเรียบร้อย ทั้งแบบปกติ และผิดปกติ เพราะการทำงานแบบ Container จะมีการเปิดปิด ตลอดเวลา เนื่องการจัดสรรทรัพยากร โดยเฉพาะ การปิดการทำงาน จะต้องเป็นไปอย่างเรียบร้อย เช่น หาก Application ถูกปิด ระหว่าง กำลังทำงานกับ Database Application จะต้องมีการ Rollback transaction โดยนักพัฒนาจะต้องตระหนักไว้เสมอว่า มีความเป็นไปได้ ตลอดที่ Application จะถูกปิดกลางคัน

ตัวอย่าง SQL transaction Statement

```
START TRANSACTION;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summary=@A WHERE type=1;
COMMIT;
```

สำหรับ Django มีการจัดการ Transaction ของ Query ที่มีความเกี่ยวเนื่องกันอัตโนมัติ แต่เราสามารถกำหนดให้ แต่ละ HTTP Request เป็น Transaction ถ้า Response สำเร็จ จึง COMMIT TRANSACTION ด้วยการ ตั้งค่า ATOMIC\_REQUESTS ใน mysite/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.environ['MYSQL_DATABASE'],
        'USER': os.environ['MYSQL_USER'],
        'PASSWORD': os.environ['MYSQL_PASSWORD'],
        'HOST': os.environ['MYSQL_HOST'],
        'PORT': os.environ['MYSQL_PORT'],
        'ATOMIC_REQUESTS': True,
    }
}
```

อ่านเพิ่ม <https://docs.djangoproject.com/en/2.0/topics/db/transactions/>

### ปัญหาข้อที่ 10. Dev/prod parity

การทำให้ Dev/Prod environment เหมือนกัน ให้มากที่สุด แบ่งออกเป็น สอง ประเด็นคือ

- คนที่ deploy app ควรจะเป็นคนที่ dev ด้วย โดยมีเครื่องมือ ที่เอื้อ ให้เค้าสามารถ deploy app ได้ด้วยตัวเอง เพื่อให้เค้าสามารถแก้ไขปัญหาได้เลย
- Software/Tools ควรจะเป็น ตัวเดียวกัน เช่น Database ควรจะเป็น ยี่ห้อ และรุ่นเดียวกัน Docker สามารถเข้ามาช่วยตรงนี้ได้ ด้วยการใช้งานที่ง่ายขึ้น กำหนด version ได้ตามต้องการ

### ปัญหาข้อที่ 11. Logs

Application ควรจะส่ง Log ออกทาง stdout ซึ่ง จะถูกจัดการผ่าน ระบบที่เข้ามาดูแล Process อื่นๆ เช่น journalctl หรือ docker เองก็มีการจัดเก็บของ ที่ผ่านออกมาทาง stdout ซึ่งเราสามารถไป โปรแกรมจัดเก็บ Log ไป tap ออกมาได้

โดย default Django จะแสดง Log ออกมาทาง stdout อยู่แล้ว หาก กำหนด DEBUG = True ซึ่งหาก run production ควรจะตั้งค่า DEBUG = False จึงควรกำหนดให้มีการแสดง Log ออกมาทาง stdout ใน mysite/settings.py

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['console'],
            'level': os.getenv('DJANGO_LOG_LEVEL', 'INFO'),
            'propagate': True,
        },
    },
}
```

### ปัญหาข้อที่ 12. Admin processes

งานประเภท admin ให้ run ในลักษณะเดียวกันกับ process หลักของ App แต่เป็นลักษณะ one-off process และให้ code ในส่วนนี้ อยู่ในโฟลเดอร์เดียวกับ project

งานลักษณะนี้ เช่น การปรับโครงสร้าง ฐานข้อมูล, การลบข้อมูลเก่า, การแก้ไขข้อมูลที่ผิดพลาด ใน Django สนับสนุนประเภทนี้ ด้วยการให้ management command เช่น

```
$ python manage.py migrate
```

นอกจากนี้ สามารถ สร้าง management command เองได้

<https://docs.djangoproject.com/en/2.0/howto/custom-management-commands/>

สร้างไฟล์ polls/management/\_\_init\_\_.py

สร้างไฟล์ polls/management/commands/\_\_init\_\_.py

สร้างไฟล์ polls/management/commands/closepoll.py

```

from django.core.management.base import BaseCommand, CommandError
from polls.models import Question as Poll

class Command(BaseCommand):
    help = 'Closes the specified poll for voting'

    def add_arguments(self, parser):
        parser.add_argument('poll_id', nargs='+', type=int)

    def handle(self, *args, **options):
        for poll_id in options['poll_id']:
            try:
                poll = Poll.objects.get(pk=poll_id)
            except Poll.DoesNotExist:
                raise CommandError('Poll "%s" does not exist' % poll_id)

            poll.opened = False
            poll.save()

            self.stdout.write(self.style.SUCCESS('Successfully closed
poll "%s"' % poll_id))

```

ทดสอบการเรียก management command

```
$ python manage.py closepoll 1
```

บันทึกการเปลี่ยนแปลง ใน Git และ Push เก็บที่ Codebase

```

$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

        modified:   mysite/settings.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        polls/management/
        requirements.txt

no changes added to commit (use "git add" and/or "git commit -a")

$ git add mysite/settings.py polls/management requirements.txt

```

```
$ git commit -m "12 Factor App"
[master eba517e] 12 Factor App
7 files changed, 57 insertions(+), 3 deletions(-)
create mode 100644 polls/management/__init__.py
create mode 100644 polls/management/commands/__init__.py
create mode 100644 polls/management/commands/closepoll.py
create mode 100644 requirements.txt

$ git pull origin master
From https://gitlab.com/ridnarong/wunca37
 * branch          master      -> FETCH_HEAD
Already up to date.

$ git push origin master
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 1.87 KiB | 1.87 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0)
To https://gitlab.com/ridnarong/wunca37.git
 11096d7..eba517e  master -> master
```