Chapter-14

Deployment of Our Application in the Live Environment

Deployment:

There are several deployment options are available to deploy our django web application. These options will be based on

1) **Scalability**
2) **Performance**
3) **Price**
4) **Security**
5) **Easy to use**

etc

The following are various deployment options

1) **PythonAnywhere.com**
   It is very simple and easy to host
2) **Digital Ocean → VPS(Virtual Privater Server)**
3) **Heroku**
4) **Amazon Cloud**

etc

Note: For every platform clear documentation steps are available.

Need of Version Control Systems:

1. To maintain multiple versions of the same product
2. At any point of time we can have a backup of previous version
3. We can see the difference between 2 or more versions of our code base
4. We can run mutliple versions of the same product simultaneously
5. It helps us to track project history over time and to collaborate easily with others. etc

The following is the list most popular version control systems

1) **GIT**
2) **Apache Subversion**
3) **Mercurial**
4) **Concurrent Version System(CVS)**
5) **GNU Bazaar**

etc

Git vs GitHub:

- **Git is a version control system that helps to track changes in our code**
- **GitHub is a company/website that helps manage git and and host our files on their site. i.e GitHub is remove hosting service to host our code repository.**
- **Similar to GitHub there are several hosting platforms like Gitlab,BitBucket etc**

Note: **If our application at remote hosting platform then deployment on various paltforms will become very easy.**

Version Control vs Hosting Platform vs Deployment Platform

How to install git:

https://git-scm.com/downloads

How to create account in github.com

Just login to github.com and create FREE account

Git Respository:

Git is a set of layers.
Each layer has a function. We can use git to move files between these layers.

Activtities related to Git Repository:

- **Create a folder named with my_repo which acts as Git Repository.**
- **Copy the required files to this folder for tracking purposes.**
- **Initialize Git by using the following command → git init**
- **By default git won't track any files. We have to add files to the Staging area, such files only can be tracked by GIT.**
- **We can add files to the staging area by using the following command → git add filename1**
- **To add all files present in current working directory we have to use → git add.**
- **Whenever we perform commit,for all files present in staging area,snapshots will be created by git. We can perform commit as follows → git commit -m 'comment'**
- **We can check the status by using the command → git status**

```
LENOVO@LENOVO-PC MINGW64 /e
$ cd my_repo1

LENOVO@LENOVO-PC MINGW64 /e/my_repo1
$ git init
Initialized empty Git repository in E:/my_repo1/.git/

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git add test.py

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git status
On branch master
No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.py

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git commit -m 'firstcommit'
[master (root-commit) b53bd68] firstcommit
 1 file changed, 3 insertions(+)
 create mode 100644 test.py

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git status
On branch master
nothing to commit, working tree clean

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git status
On branch 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:   test.py
no changes added to commit (use "git add" and/or "git commit -a")

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git add test.py

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git commit -m 'second'
[master 1677eb2] second
 1 file changed, 2 insertions(+), 1 deletion(-)

LENOVO@LENOVO-PC MINGW64 /e/my_repo1 (master)
$ git status
On branch master
nothing to commit, working tree clean
----------------------------
…OR create a new repository on the command line
echo "# django-deployment-first-application" >> README.md
git init
git add README.md
git commit -m "first commit"
```

```
git remote add origin https://github.com/djangodurga/django-deployment-first-application.git
git push -u origin master
--------------------------------------------------
LENOVO@LENOVO-PC MINGW64 ~
$ cd e:

LENOVO@LENOVO-PC MINGW64 /e
$ cd my_codebase/

LENOVO@LENOVO-PC MINGW64 /e/my_codebase
$ git init
Initialized empty Git repository in E:/my_codebase/.git/

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git add .

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   firstproject/db.sqlite3
        new file:   firstproject/firstproject/__init__.py
        new file:   firstproject/firstproject/__pycache__/__init__.cpython-36.pyc
        new file:   firstproject/firstproject/__pycache__/settings.cpython-36.pyc
        new file:   firstproject/firstproject/__pycache__/urls.cpython-36.pyc
        new file:   firstproject/firstproject/__pycache__/wsgi.cpython-36.pyc
        new file:   firstproject/firstproject/settings.py
        new file:   firstproject/firstproject/urls.py
        new file:   firstproject/firstproject/wsgi.py
        new file:   firstproject/manage.py
        new file:   firstproject/testapp/__init__.py
        new file:   firstproject/testapp/__pycache__/__init__.cpython-36.pyc
        new file:   firstproject/testapp/__pycache__/admin.cpython-36.pyc
        new file:   firstproject/testapp/__pycache__/models.cpython-36.pyc
        new file:   firstproject/testapp/__pycache__/views.cpython-36.pyc
        new file:   firstproject/testapp/admin.py
        new file:   firstproject/testapp/apps.py
        new file:   firstproject/testapp/migrations/__init__.py
        new file:   firstproject/testapp/migrations/__pycache__/__init__.cpython-36.pyc
```

```
        new file:   firstproject/testapp/models.py
        new file:   firstproject/testapp/tests.py
        new file:   firstproject/testapp/views.py

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git commit -m 'firstcommit'
[master (root-commit) c95b9d9] firstcommit
 22 files changed, 213 insertions(+)
 create mode 100644 firstproject/db.sqlite3
 create mode 100644 firstproject/firstproject/__init__.py
 create mode 100644 firstproject/firstproject/__pycache__/__init__.cpython-36.pyc
 create mode 100644 firstproject/firstproject/__pycache__/settings.cpython-36.pyc
 create mode 100644 firstproject/firstproject/__pycache__/urls.cpython-36.pyc
 create mode 100644 firstproject/firstproject/__pycache__/wsgi.cpython-36.pyc
 create mode 100644 firstproject/firstproject/settings.py
 create mode 100644 firstproject/firstproject/urls.py
 create mode 100644 firstproject/firstproject/wsgi.py
 create mode 100644 firstproject/manage.py
 create mode 100644 firstproject/testapp/__init__.py
 create mode 100644 firstproject/testapp/__pycache__/__init__.cpython-36.pyc
 create mode 100644 firstproject/testapp/__pycache__/admin.cpython-36.pyc
 create mode 100644 firstproject/testapp/__pycache__/models.cpython-36.pyc
 create mode 100644 firstproject/testapp/__pycache__/views.cpython-36.pyc
 create mode 100644 firstproject/testapp/admin.py
 create mode 100644 firstproject/testapp/apps.py
 create mode 100644 firstproject/testapp/migrations/__init__.py
 create mode 100644 firstproject/testapp/migrations/__pycache__/__init__.cpython-36.pyc
 create mode 100644 firstproject/testapp/models.py
 create mode 100644 firstproject/testapp/tests.py
 create mode 100644 firstproject/testapp/views.py

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git status
On branch master
nothing to commit, working tree clean

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git remote add origin https://github.com/djangodurga/django-deployment-first-
application.git

LENOVO@LENOVO-PC MINGW64 /e/my_codebase (master)
$ git push -u origin master
Enumerating objects: 29, done.
Counting objects: 100% (29/29), done.
Delta compression using up to 4 threads
```

Compressing objects: 100% (27/27), done.
Writing objects: 100% (29/29), 6.98 KiB | 376.00 KiB/s, done.
Total 29 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote: https://github.com/djangodurga/django-deployment-first-application/pull/new/master
remote:
To https://github.com/djangodurga/django-deployment-first-application.git
* [new branch] master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Deployment on pythonanywhere.com:

pricing and signup-->beginner account(free)
username:durgasoftdurga
pwd:github12345

Open the console and Create Virtual Environment:

Console → Bash →
$mkvirtualevn --python=python3.6 myproj
$mkvirtualenv --python=python3.6 myproj
here myproj is the name of the virtual environment
(myproj) 09:40 ~ $
If we are seeing this means our virtual enviroment created and active.

Which package already installed in this virtual env:

(myproj) 09:40 ~ $ pip list

```
Package    Version
---------- -------
pip        18.1
setuptools 40.5.0
wheel      0.32.2
```

Install django:

It is highly recommended to install the same version which is available on our local machine.
How to check our local machine django version:

C:\Users\LENOVO>python
Python 3.6.5 (v3.6.5:f59c093
Type "help", "copyright", "c
>>> import django

```
>>> django.__version__
'1.11'
```

We can install django in virtualenv as follows.
pip install -U django==1.11

How to check whether django installed properly:

(myproj) 09:48 ~ $ pip list

```
Package     Version
--------    -------
Django      1.11
pip         18.1
pytz        2018.7
setuptools  40.5.0
wheel       0.32.2
```

(myproj) 09:49 ~ $ which django-admin
/home/durgasoftdjango/.virtualenvs/myproj/bin/djan
go-admin

Copy Our Application from github to Our Virtual Environment (pythonanywhere):

https://github.com/djangodurga/django-second-deployment
clone or download
copy url: https://github.com/djangodurga/django-second-deployment.git
(myproj) 09:54 ~ $ git clone https://github.com/djangodurga/django-second-deployme
nt.git
Cloning into 'django-second-deployment'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 29 (delta 2), reused 29 (delta 2), pack-reused 0
Unpacking objects: 100% (29/29), done.
Checking connectivity... done.

(myproj) 09:55 ~ $ ls
README.txt django-second-deployment
(myproj) 09:56 ~ $ cd django-second-deployment/
(myproj) 09:56 ~/django-second-deployment (master)$ ls
secondproject
(myproj) 09:56 ~/django-second-deployment (master)$ cd secondproject/
(myproj) 09:56 ~/django-second-deployment/secondproject (master)$ python manage.py
makemigrations
No changes detected

(myproj) 09:57 ~/django-second-deployment/secondproject (master)$ python manage.py migrate

(myproj) 09:58 ~/django-second-deployment/secondproject (master)$ pwd
/home/durgasoftdjango/django-second-deployment/secondproject

This is the sourcecode path

Configuration on the web tab:

Add a new web app → Next →

If we want to develop a fresh application we have to select Django.
But if we have application already then we have to select

Manual configuration (including virtualenvs)

Next → Select Python Version → Python 3.6 → Next

Source code:
/home/durgasoftdjango/django-second-deployment/secondproject

Virtualenv:
/home/durgasoftdjango/.virtualenvs/myproj

WSGI Configuration:

WSGI configuration file:/var/www/durgasoftdjango_pythonanywhere_com_wsgi.py

In the configuration file just remove Hello World related things

```
# ++++++++++ DJANGO ++++++++++
# To use your own django app use code like this:
import os
import sys
#
## assuming your django settings file is at
'/home/durgasoftdjango/mysite/mysite/settings.py'
## and your manage.py is is at '/home/durgasoftdjango/mysite/manage.py'
path = '/home/durgasoftdjango/django-second-deployment/secondproject'
if path not in sys.path:
    sys.path.append(path)
os.chdir(path)
os.environ.setdefault('DJANGO_SETTINGS_MODULE','secondproject.settings')
```

```
import django
django.setup()
#
#os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'
#
## then:
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Add 'durgasoftdjango.pythonanywhere.com' to ALLOWED_HOSTS:

from bash shell

nano settings.py
move to the required position and add :
ALLOWED_HOSTS = ['durgasoftdjango.pythonanywhere.com']

ctrl+o to save
ctrl+x to close

Static Files:

By default in pythonanywhere deployment static files won't be considered. Hence while accessing our web application and django admin site look and feel will be changed. For that we have perform some configurations in the web tab of dashboard.

Static files related to admin site:

Static files:
URL: /static/admin
Path: /home/durgasoftdjango/.virtualenvs/myproj/lib/python3.6/site-packages/django/contrib/admin/static/admin

Note: After performing any configuration changes,compulsory we should reload our application

Static files related to our application:

To reflect css files,js files and images used in our application we have to perform the following configuration under Static files:

URL: /static
Path: /home/durgasoftdjango/django-second-deployment/secondproject/static

Note: It is highly recommended to disable DEBUG mode in production, because we should not display application level sensitive information to the end user. For this,in settings.py, we have to set DEBUG=False