How To Separate Local, Development and Production Settings

1. Separate local, development and production settings.
   - Make a settings directory

```
mysite/
 |-- mysite/
 |    |-- __init__.py
 |    |-- settings.py
 |    |-- urls.py
 |    +-- wsgi.py
 +-- manage.py

To

mysite/
 |-- mysite/
 |    |-- __init__.py
 |    |-- settings/         <--
 |    |    |-- __init__.py  <--
 |    |    +-- base.py      <--
 |    |-- urls.py
 |    +-- wsgi.py
 +-- manage.py
```

   - Put all common settings in base.py

<base.py>

```
"""
Django settings for dustandsepia project.

Generated by 'django-admin startproject' using Django 2.2.5.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'dustandsepia.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [TEMPLATES_DIR],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'dustandsepia.wsgi.application'

# Password validation
#
https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-valida
tors

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidato
r',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True
```

```
USE_TZ = True
```

- Put any settings that might change in both local.py and development.py

```
                     <local.py and development.py>

from .base import *

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'qf@*avph%!$*-$j+=6_js3k6lapm+b@$h8fh^hhv32&(7m$nrs'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

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


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'
```

- Later you will change these settings on your development server, but for now leave them the same. Test if this works by typing

  ```
  > python3 manage.py runserver --settings=myproject.settings.local
  ```

  And visit the appropriate url in your browser. If you get the message that there ‘X unapplied migrations’, that means that your django project is unable to find your sqlite db file. In base.py, tweak the following line to fix that.

```
BASE_DIR =
os.path.dirname(os.path.dirname(os.path.dirname(os.path.abspath(__file__
))))
TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')
```

  - You can add production.py or someotherenv.py later
  - Now change wsgi.py to indicate changes settings module

    ```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'myproject.settings')
    ```

    will change to

    ```
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
    'myproject.settings.development')
    ```

  - You can run project locally by typing `python3 manage.py runserver --settings=myproject.settings.local`

2. Push to git. Before you do this, make sure you have a gitignore file that ignores your sqlite db, your virtual env, and any .pyc and .env files.
3. Clone git repo on remote server and create a remote branch to make changes for deployment on remote server.\

   ```
   > git checkout -b test_dev_deployment
   ```

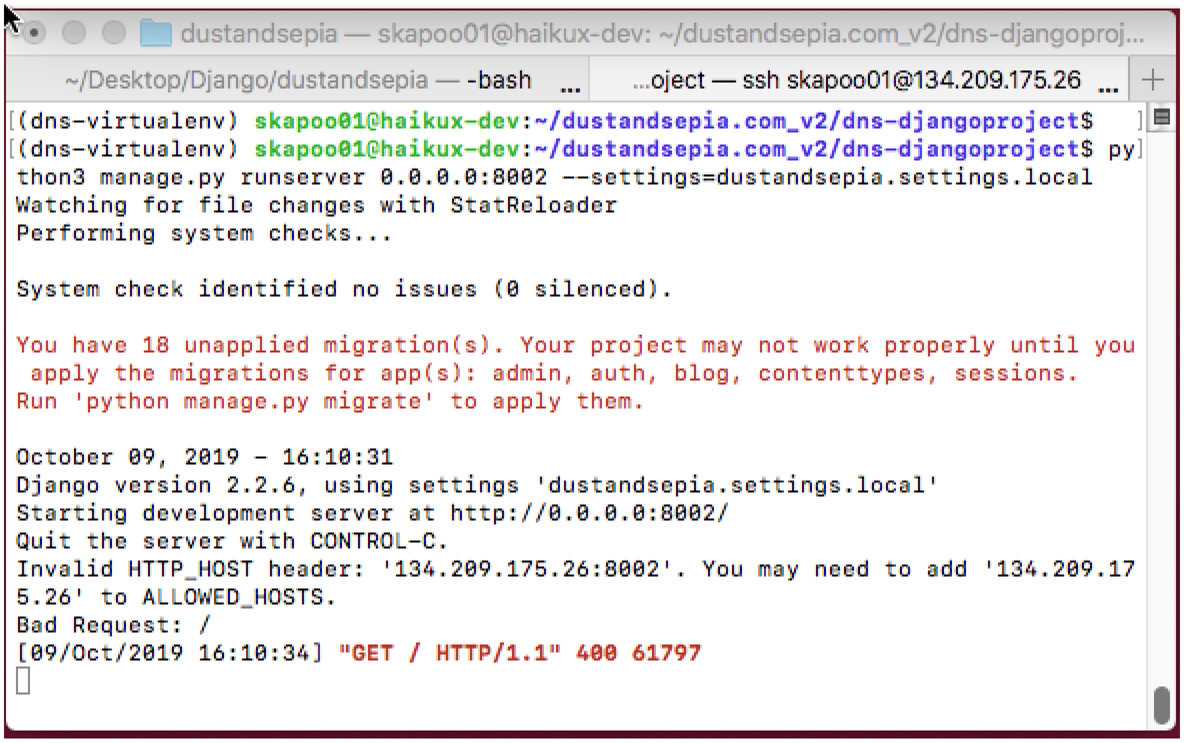
   Now before doing anything else, create a virtual env and install any modules needed from requirements.txt.

   ```
   > virtualenv project-virtualenv
   > source project-virtualenv/bin/activate
   > python3 -m pip install --upgrade pip
   > pip install -r requirements.txt
   ```

   Now go to your project directory where manage.py is and type

   ```
   (myproject-venv) > python3 manage.py runserver
   --settings=myproject.settings.local
   ```

   Your terminal will tell you at this point that you have unapplied migrations:


dustandsepia — skapoo01@haikux-dev: ~/dustandsepia.com_v2/dns-djangoproj...
~/Desktop/Django/dustandsepia — -bash
...oject — ssh skapoo01@134.209.175.26

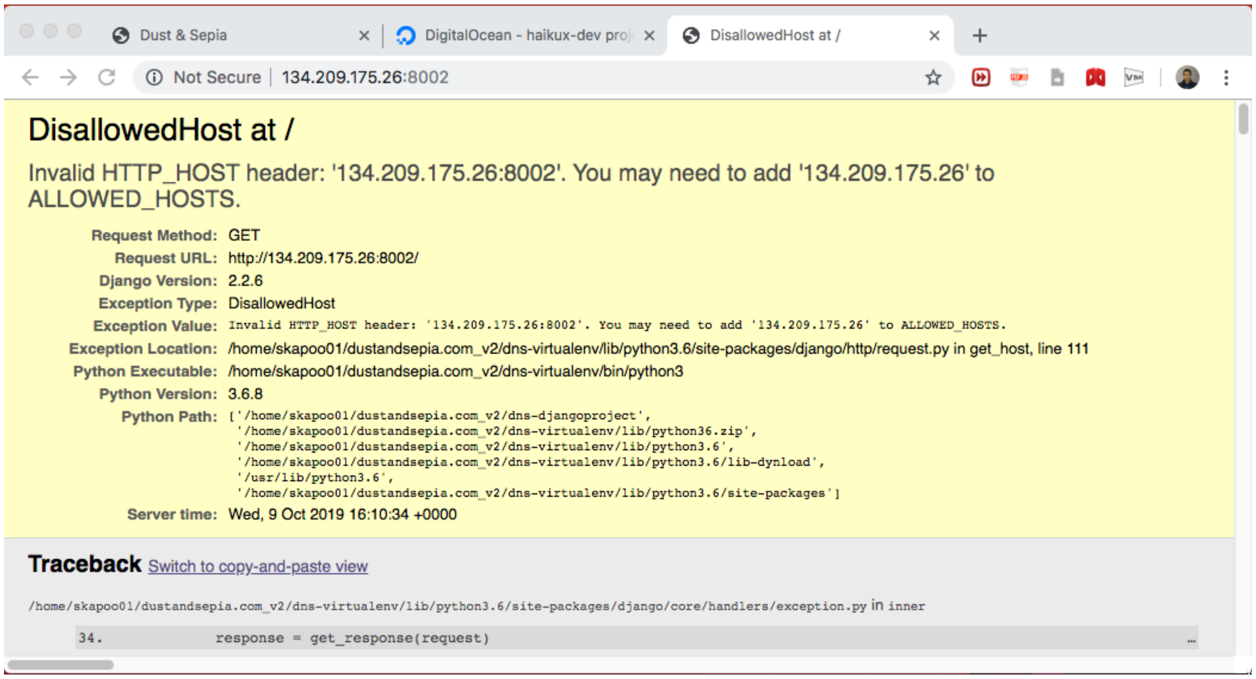
```
[(dns-virtualenv) skapoo01@haikux-dev:~/dustandsepia.com_v2/dns-djangoproject$   ]
[(dns-virtualenv) skapoo01@haikux-dev:~/dustandsepia.com_v2/dns-djangoproject$ py]
thon3 manage.py runserver 0.0.0.0:8002 --settings=dustandsepia.settings.local
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you
 apply the migrations for app(s): admin, auth, blog, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

October 09, 2019 - 16:10:31
Django version 2.2.6, using settings 'dustandsepia.settings.local'
Starting development server at http://0.0.0.0:8002/
Quit the server with CONTROL-C.
Invalid HTTP_HOST header: '134.209.175.26:8002'. You may need to add '134.209.17
5.26' to ALLOWED_HOSTS.
Bad Request: /
[09/Oct/2019 16:10:34] "GET / HTTP/1.1" 400 61797
```

And if you visit the appropriate address in your browser:



DisallowedHost at /

Invalid HTTP_HOST header: '134.209.175.26:8002'. You may need to add '134.209.175.26' to ALLOWED_HOSTS.

```
     Request Method: GET
        Request URL: http://134.209.175.26:8002/
     Django Version: 2.2.6
     Exception Type: DisallowedHost
    Exception Value: Invalid HTTP_HOST header: '134.209.175.26:8002'. You may need to add '134.209.175.26' to ALLOWED_HOSTS.
 Exception Location: /home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python3.6/site-packages/django/http/request.py in get_host, line 111
  Python Executable: /home/skapoo01/dustandsepia.com_v2/dns-virtualenv/bin/python3
     Python Version: 3.6.8
        Python Path: ['/home/skapoo01/dustandsepia.com_v2/dns-djangoproject',
                      '/home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python36.zip',
                      '/home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python3.6',
                      '/home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python3.6/lib-dynload',
                      '/usr/lib/python3.6',
                      '/home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python3.6/site-packages']
        Server time: Wed, 9 Oct 2019 16:10:34 +0000
```

Traceback Switch to copy-and-paste view

```
/home/skapoo01/dustandsepia.com_v2/dns-virtualenv/lib/python3.6/site-packages/django/core/handlers/exception.py in inner
     34.            response = get_response(request)                                   …
```

This is a good sign. Naturally you have not created a database yet and have not added your server_ip to your ALLOWED_HOSTS in settings, so this is the expected result. It tells you everything is in order?

4. In development and production settings, make all changes required to run your project on a remote server
    - Make a database and add an admin user to it
    - Make a virtual environment in your project base directory. Change requirements.txt to include any needed modules for deployment like pscopg2 and pip install -r requirements.txt
    - Change development.py to use your database and add your domain/server_ip to allowed hosts, as well as add a static root
    - Attempt runserver locally
    - Add env variables to activate, and appropriately modify production.py
    - See if you can run project with gunicorn
    - If yes, create a .env file with your environment variables
    - Now create a systemd service file for your project that will run a gunicorn daemon, and in this file indicate the path to the file containing env variables. First run the gunicorn on an IP port to see if it works, then change it to a unix socket.
    - Now add a nginx file that can reverse proxy your server_ip or domain to the unix socket for your project
    - If your project works, then you're all set and you can remove env variables from your activate script
5. Store any sensitive info in env variables, and make sure your gitignore will ignore files containing them. You can add the environment variables to your virtualenv activate script if you wish to run gunicorn from the console, or create a .env file and indicate this in a gunicorn service file if you are running gunicorn via systemd service.
6. Test whether deployment on server
7. Push remote branch to git, and then try to see if the remote branch can run locally with your local settings on your local machine without any changes
8. If yes, then merge the remote branch with master, and you should be able to run concurrent development from master. If not, troubleshoot. Finally be sure to update master and development branches both locally and on your remote server to avoid a git headache.
9. To add features or make changes, first test them locally on master. Then push to git and try to run the project on the remote server. You will need to download any required dependencies in your virtual environment, and run any needed migrations. However, if there are any changes in your local settings other than that (such as database changes, server changes, changes in static asset delivery, etc.) then you might need to make changes in your production settings. In this case it is recommended making or using a development branch and merging only after testing that the branch can run locally still.

https://medium.com/@stschindler/one-django-settings-py-to-rule-them-all-dac2f6be8159

https://djangostars.com/blog/configuring-django-settings-best-practices/

https://stackoverflow.com/questions/25076295/gunicorn-environment-variable-setting

https://stackoverflow.com/questions/1783405/how-do-i-check-out-a-remote-git-branch

https://simpleisbetterthancomplex.com/tips/2017/07/03/django-tip-20-working-with-multiple-settings-modules.html