Lecture 8

Testing CI/CD

Subdirectory

Airline0

Admin.py

class PassengerAdmin(admin.ModelAdmin):

filter\_horizontal = ("flights",)

Extend the existin model

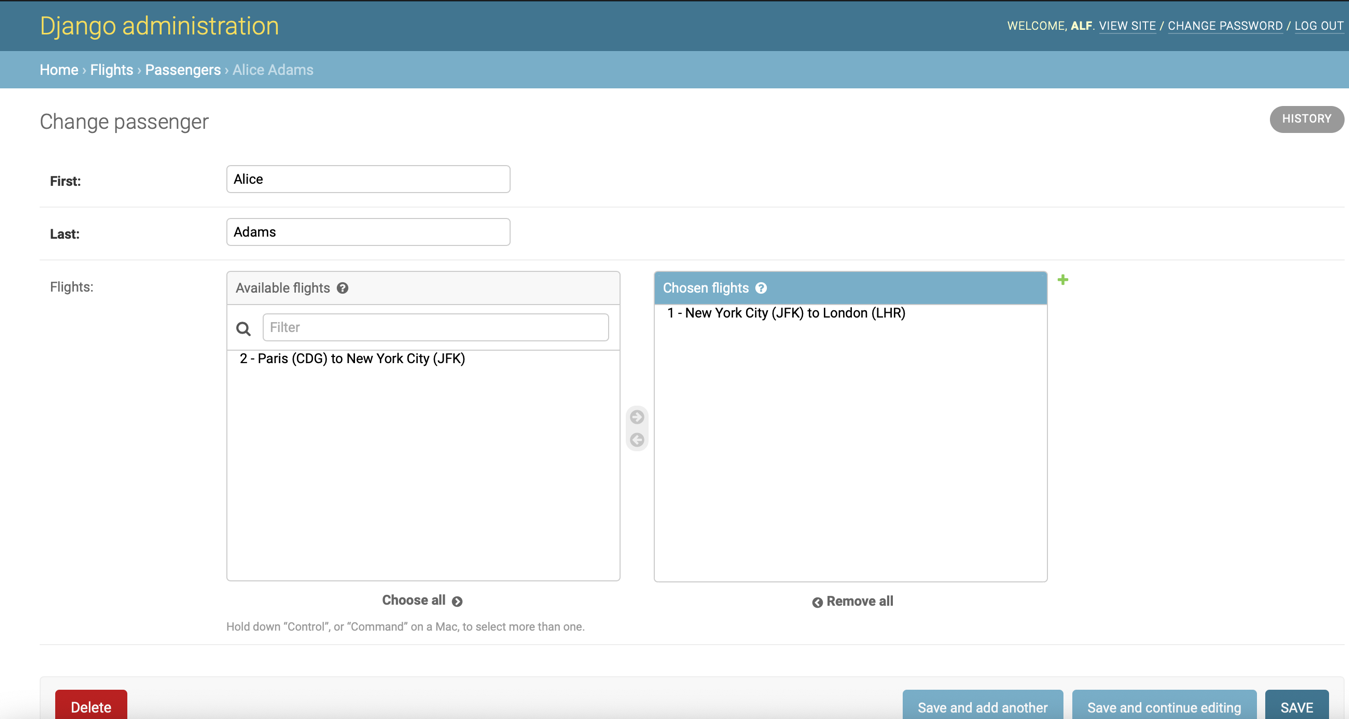
Determin how the users can use

Only wen adding passenger

Filter horizontal, taking passenger and manipulate it

Python manage.py runserver

Here when we want to add passengers appears an horizontal option to choose



In admin.py

Define a new class

class PassengerInline(admin.StackedInline):

model = Passenger.flights.through

extra = 1

StackedInline

It is to modify or add passengers

It has a model that relates individual passengers with flights

Extra=1

Means that we can add one passenger at time

Ç

class FlightAdmin(admin.ModelAdmin):

inlines = [PassengerInline]

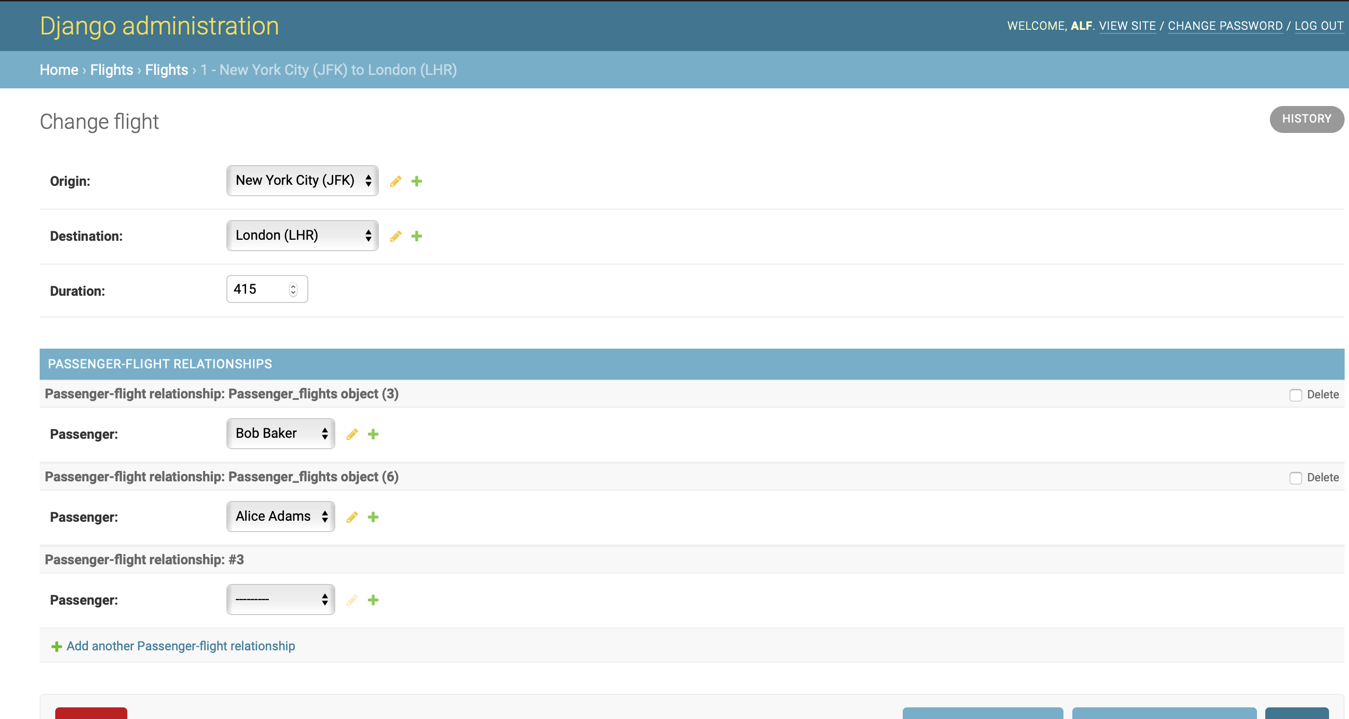
I want to add this section

admin.site.register(Flight, FlightAdmin)

We add to the FlightADmin to the administration

Therefore, I can see passenger-flight relationships

It makes possible to see

Changes in flight, origin, destinagtion, duration and also passengers, add or change information about passengers!

Encorage to see the manual of admin

How to use the javascript and ccs

Base.html

We use

{% load static %}

<link rel="stylesheet" href="{% static 'flights/styles.css' %}"/>

In flights/flights/styles.css

In flights there is static folder

Inside is flights directory

Inside is the styles.css

Note: there is a flights directory inside static and templates!

It feels redundant, it is because we have multiple routes

Testing

It is to know whether some changes in a function or in one part of the program is not gonna affect unnecessary the process in another part of the sytem

Also, when I have different inputs I want to know whether the functions are going to work as expected

Which tools we have?

First we are going to test it in python and from there we are going to test in Django and web applications.

Prime.py

First I enter to python3.8

Then type (note: prime.py is the python file and is\_prime is the function inside this file)

>>> from prime import is\_prime

>>>is\_prime(23)

>>>True

>>>is\_prime(28)

>>>False

>>>is\_prime(25)

>>>False

This process is very tedious and not very efficient

What can I do?

To write a program that does it for me

tests0.py

It compares the result of prime.py with the expected results and tells me whether is right or not

Tests0.sh

This file helps to run several times tests0 with particular values

It is run as:

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % ./tests0.sh

If an error occurs, it presents:

ERROR on is\_prime(8), expected False

ERROR on is\_prime(25), expected False

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

If no errors exist, running the file .sh just produces nothing

So it means that under the test I set up, the program passed, but it does not mean that is completely right. For this, we need to explore all possibilities of errors.

There are a couple of things in python that can be useful

One is the assert command

It evaluates something after this word and if it is true, it says nothing otherwise marks an error

>>> assert True

>>> assert False

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AssertionError

>>>

Assert0.py

I run it in python

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % python3.8 assert0.py

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

It produces nothing because the content is right

However, if it is not true

Assert1.py

I run it in python

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % python3.8 assert1.py

Traceback (most recent call last):

File "assert1.py", line 4, in <module>

assert square(10) == 101

AssertionError

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % echo $?

1

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

In yellow echo %? Tells the code the exit program. If no errors it produces 0, but 1 means there was an error

There is a python library called unittest which make easier to test the programs

Tests1.py

import unittest

class Tests(unittest.TestCase):

def test\_1(self):

"""Check that 1 is not prime."""

self.assertFalse(is\_prime(1))

def test\_2(self):

"""Check that 2 is prime."""

self.assertTrue(is\_prime(2))

def test\_8(self):

"""Check that 8 is not prime."""

self.assertFalse(is\_prime(8))

def test\_11(self):

"""Check that 11 is prime."""

self.assertTrue(is\_prime(11))

def test\_25(self):

"""Check that 25 is not prime."""

self.assertFalse(is\_prime(25))

def test\_28(self):

"""Check that 28 is not prime."""

self.assertFalse(is\_prime(28))

When I run this program:

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % python3.8 tests1.py

......

----------------------------------------------------------------------

Ran 6 tests in 0.000s

OK

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

I did not get any error

However if intentionally I make an error in prime.py

for i in range(2, int(math.sqrt(n))):

then running tests1.py gives the following results

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % python3.8 tests1.py

...F.F

======================================================================

FAIL: test\_25 (\_\_main\_\_.Tests)

Check that 25 is not prime.

----------------------------------------------------------------------

Traceback (most recent call last):

File "tests1.py", line 26, in test\_25

self.assertFalse(is\_prime(25))

AssertionError: True is not false

======================================================================

FAIL: test\_8 (\_\_main\_\_.Tests)

Check that 8 is not prime.

----------------------------------------------------------------------

Traceback (most recent call last):

File "tests1.py", line 18, in test\_8

self.assertFalse(is\_prime(8))

AssertionError: True is not false

----------------------------------------------------------------------

Ran 6 tests in 0.001s

FAILED (failures=2)

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 %

Note: F means failed, and dot means passed

Now we apply unittest in a web application

In airline1

Airline1/flights/tests.py

This is a program that helps test the code

Tests.py

from django.test import TestCase

TestCase imports unittest

1. Create airports

a1 = Airport.objects.create(code="AAA", city="City A")

1. Create flights

Flight.objects.create(origin=a1, destination=a2, duration=100)

We are going to look for a function that it will be interesting to test

In models.py

We are going to choose a function that checks whether we created a valid flight (for instance, we want to avoid to have a flight that goes from Paris to Paris)

class Flight(models.Model):

origin = models.ForeignKey(Airport, on\_delete=models.CASCADE, related\_name="departures")

destination = models.ForeignKey(Airport, on\_delete=models.CASCADE, related\_name="arrivals")

duration = models.IntegerField()

def is\_valid\_flight(self):

return (self.origin != self.destination) and (self.duration >= 0)

This function returns true or false options. It validates that the origin is different to the destination and the flight duration is larger than 0

Now we are going to test it

In flights/flights/flight.html

It was added additionally:

<li>Valid: {{ flight.is\_valid\_flight }}</li>

Just to produce if a flight is valid

I run Django

Src $ python manage.py runserver

It says the property:

* Flight Number: 2
* Origin: Paris (CDG)
* Destination: New York City (JFK)
* Duration: 435
* Valid: True
* Passengers:
  + No passengers

Now we are testing erroneous options

Tests.py

Flight.objects.create(origin=a1, destination=a2, duration=100)

Flight.objects.create(origin=a1, destination=a1, duration=200)

Flight.objects.create(origin=a1, destination=a2, duration=-100)

The first one is right

The second one is erroneous

The third one is erroneous

Also in the following code we are testing if in the airport is related with the departures from

def test\_departures\_count(self):

a = Airport.objects.get(code="AAA")

self.assertEqual(a.departures.count(), 3)

this code checks whether in the airport AAA there are three departures

another example:

def test\_arrivals\_count(self):

a = Airport.objects.get(code="AAA")

self.assertEqual(a.arrivals.count(), 1)

This code checks whether in the airport AAA there is only one arrival

In this other code:

def test\_valid\_flight(self):

a1 = Airport.objects.get(code="AAA")

a2 = Airport.objects.get(code="BBB")

f = Flight.objects.get(origin=a1, destination=a2, duration=100)

self.assertTrue(f.is\_valid\_flight())

here is tested whether it is a valid flight

in this other code

def test\_invalid\_flight\_destination(self):

a1 = Airport.objects.get(code="AAA")

f = Flight.objects.get(origin=a1, destination=a1)

self.assertFalse(f.is\_valid\_flight())

it is teste whether a flight is invalid

in this other code

def test\_invalid\_flight\_duration(self):

a1 = Airport.objects.get(code="AAA")

a2 = Airport.objects.get(code="BBB")

f = Flight.objects.get(origin=a1, destination=a2, duration=-100)

self.assertFalse(f.is\_valid\_flight())

it is tested whether the duration of flight is invalid

note: .get is a single query to get something unique

Note: when tests.py is running, it does not run the actual program, but rather a image of it in order to avoid mess up the original database

How to run the test?

Airline1 $ python manage.py test

It is the result:

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

.....

----------------------------------------------------------------------

Ran 5 tests in 0.015s

OK

Destroying test database for alias 'default'...

(djangodev) josealfredocarrillosalazar@iMac-de-Jose airline1 %

The five dots in yellow means that five tests where run in 0.05 s and they are OK and then it discarded the test database

By the way, each test in the tests.py file was run independly from the others so there are not interactions or interferences among them

It is not the only things we can test

How to test the template

Airlines2 directory

In this it is also included Client:

from django.test import Client, TestCase

This is gonna simulate a client request

And there will be a response from the server

In this case we use the index function and index.html

This function calls all flights and list them in index.html

Code is:

def test\_index(self):

c = Client()

response = c.get("/")

self.assertEqual(response.status\_code, 200)

self.assertEqual(response.context["flights"].count(), 2)

First we create a new client ( c).

Second it will get a response from adding / that downloads the default index.html

Third I have to make sure that the response is successful with a code equal to 200

Fourth we want to check the content of response, that there are two flights

It works because

self.assertEqual(response.context["flights"].count(), 2)

in flights/views.py we pass “flights” into index.html

def index(request):

context = {

"flights": Flight.objects.all()

}

return render(request, "flights/index.html", context)

Thus the program gets “flights” and count the number of flights and it compares to 2, in the example.

In this other code we test whether a particular flight number is a valid flight

def test\_valid\_flight\_page(self):

a1 = Airport.objects.get(code="AAA")

f = Flight.objects.get(origin=a1, destination=a1)

c = Client()

response = c.get(f"/{f.id}")

self.assertEqual(response.status\_code, 200)

in this other code we test whether a particular flight number is invalid or does not exist

def test\_invalid\_flight\_page(self):

max\_id = Flight.objects.all().aggregate(Max("id"))["id\_\_max"]

c = Client()

response = c.get(f"/{max\_id + 1}")

self.assertEqual(response.status\_code, 404)

This is more complex.

First we obtain the max\_id we have

max\_id = Flight.objects.all().aggregate(Max("id"))["id\_\_max"]

we use the special function called Max id value and then we test whether we have a higher value than the Max value

response = c.get(f"/{max\_id + 1}")

so I should get a 404 code that does it not exist

and it will check assert equal to true

In this other code

def test\_flight\_page\_passengers(self):

f = Flight.objects.get(pk=1)

p = Passenger.objects.create(first="Alice", last="Adams")

f.passengers.add(p)

c = Client()

response = c.get(f"/{f.id}")

self.assertEqual(response.status\_code, 200)

self.assertEqual(response.context["passengers"].count(), 1)

It tests whether the passenger page is working properly. It tests whether when we add one passanger in one flight number it does actually exists as one passanger added

In the following code

def test\_flight\_page\_non\_passengers(self):

f = Flight.objects.get(pk=1)

p = Passenger.objects.create(first="Alice", last="Adams")

c = Client()

response = c.get(f"/{f.id}")

self.assertEqual(response.status\_code, 200)

self.assertEqual(response.context["non\_passengers"].count(), 1)

here we don’t add a new passanger and we expect this is gonna be false

then we run:

cd airline2

(djangodev) josealfredocarrillosalazar@iMac-de-Jose airline2 % python3.8 manage.py test

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

..........

----------------------------------------------------------------------

Ran 10 tests in 0.064s

OK

Destroying test database for alias 'default'...

(djangodev) josealfredocarrillosalazar@iMac-de-Jose airline2 %

For instance, if I modified in views.py

def flight(request, flight\_id):

try:

flight = Flight.objects.get(pk=flight\_id)

except Flight.DoesNotExist:

raise Http404("Flight does not exist")

context = {

"flight": flight,

"non\_passengers": flight.passengers.all(),

"passengers": Passenger.objects.exclude(flights=flight).all()

}

return render(request, "flights/flight.html", context)

when I run the test file I get:

I get

(djangodev) josealfredocarrillosalazar@iMac-de-Jose airline2 % python3.8 manage.py test

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

..FF......

======================================================================

FAIL: test\_flight\_page\_non\_passengers (flights.tests.FlightsTestCase)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/josealfredocarrillosalazar/Documents/CourseProgpythonjava/Lecture8\_Programs/src8/airline2/flights/tests.py", line 83, in test\_flight\_page\_non\_passengers

self.assertEqual(response.context["non\_passengers"].count(), 1)

AssertionError: 0 != 1

======================================================================

FAIL: test\_flight\_page\_passengers (flights.tests.FlightsTestCase)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/Users/josealfredocarrillosalazar/Documents/CourseProgpythonjava/Lecture8\_Programs/src8/airline2/flights/tests.py", line 74, in test\_flight\_page\_passengers

self.assertEqual(response.context["passengers"].count(), 1)

AssertionError: 0 != 1

----------------------------------------------------------------------

Ran 10 tests in 0.063s

FAILED (failures=2)

Destroying test database for alias 'default'...

(djangodev) josealfredocarrillosalazar@iMac-de-Jose airline2 %

Which means that it found failures

If I correct then I get all fine

We can see other unittest methods

assertEqual

assertNotEqual

assertTrue

assertFalse

assertIn

assertNotIn

…

Another example of testing

(djangodev) josealfredocarrillosalazar@iMac-de-Jose src8 % cd selenium

(djangodev) josealfredocarrillosalazar@iMac-de-Jose selenium % open counter.html

(djangodev) josealfredocarrillosalazar@iMac-de-Jose selenium %

It is a simple application that increases or decreases numbers clicking in two buttons

How I test this?

What is happening inside a webpage?

In the directory of selenium

Run python3.8

Then

>>> from selenium import webdriver

>>>

Nota:

Par esto tuve que instalar selenium

a) Installed java 8 update 251.app

b) installed jdk

c) download selenium-server-standalone-3141.59.jar

Note: command java -version (shows the java version)

Note: Lo anterior creo no era necesario!!!

downloaded selenium for python

(ver https://pypi.org/project/selenium/)

Esto si fue suficiente:

(djangodev) josealfredocarrillosalazar@iMac-de-Jose lecture8\_programs % pip install -U selenium

Collecting selenium

Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)

|████████████████████████████████| 904 kB 383 kB/s

Collecting urllib3

Using cached urllib3-1.25.9-py2.py3-none-any.whl (126 kB)

Installing collected packages: urllib3, selenium

Successfully installed selenium-3.141.0 urllib3-1.25.9

después en el folder de selenium del lecture8, I run python3.8

y use el comando: from selenium import WebDriver

y listo!!!!

Luego

>>> from tests import file\_uri

Nota: para lograr esto tuve que:

) Fui al menu de safari, preferencias, avanzado y seleccioné “Monstar menu desarrollo en la barra de menús”

b) en el menú de desarrollo activé la opción permitir automatización remota

además, en el archivo tests.py en el directorio de selenum lecture 8

hice que

driver=webdriver.Safari()

then:

>>> file\_uri("counter.html")

'file:///Users/josealfredocarrillosalazar/Documents/CourseProgpythonjava/Lecture8\_Programs/src8/selenium/counter.html'

>>>

Note: if a copy

'file:///Users/josealfredocarrillosalazar/Documents/CourseProgpythonjava/Lecture8\_Programs/src8/selenium/counter.html'

And paste it in the webpage I can run the counter.html webpage from my computer

>>> uri = file\_uri("counter.html")

>>> uri

'file:///Users/josealfredocarrillosalazar/Documents/CourseProgpythonjava/Lecture8\_Programs/src8/selenium/counter.html'

>>>

When I type:

>>>Driver=webdriver.Safari()

>>>

Then I got a Safari window ready for automation

If I type:

>>> driver.get(uri)

>>>

The result is that in the opened Safari window appears the counter.html page

Then in counter.html I can see that

<button id="increase">+</button>

<button id="decrease">-</button>

Thus I can create a variable to find an element with the id “increase” which is a button

>>> plus=driver.find\_element\_by\_id("increase")

And then click on it

>>> plus.click()

Authomatically the value in counter.html page increases

Now we are going to test the aspects of the webpage

First we test the title of the page

def test\_title(self):

driver.get(file\_uri("counter.html"))

self.assertEqual(driver.title, "Counter")

Second we test the increase button

def test\_increase(self):

driver.get(file\_uri("counter.html"))

increase = driver.find\_element\_by\_id("increase")

increase.click()

self.assertEqual(driver.find\_element\_by\_tag\_name("h1").text, "1")

Third we test the decrease button

def test\_decrease(self):

driver.get(file\_uri("counter.html"))

decrease = driver.find\_element\_by\_id("decrease")

decrease.click()

self.assertEqual(driver.find\_element\_by\_tag\_name("h1").text, "-1")

Fourth we test multiple increases

def test\_multiple\_increase(self):

driver.get(file\_uri("counter.html"))

increase = driver.find\_element\_by\_id("increase")

for i in range(3):

increase.click()

self.assertEqual(driver.find\_element\_by\_tag\_name("h1").text, "3")

When I run the tests.py from selenium directory, the result was:

(djangodev) josealfredocarrillosalazar@iMac-de-Jose selenium % python3.8 tests.py

....

----------------------------------------------------------------------

Ran 4 tests in 0.335s

OK

(djangodev) josealfredocarrillosalazar@iMac-de-Jose selenium %

CI/CD

How to create convenient and robust software

Continuous integration (it is about different people working in the same proyect and then after developing their bits, they want to integrate in the main branch. Also, anytime the different branches of the project are integrated in the main branch, we want to test whether there are not errors or bugs that affect other parts of the software. Also I want the test is make automatically anytime the new branch is added to the main branch)

Frequent merges to main branch

Automated unit testing

Continuous delivery (When we maka modification to our branch, we want continuously apdate the main branch

Automated application deployment

We are going to explore some of the utilities for doing it. There are many application for this (CI Tools) like CicleCI, Codeship, Jenkins, Travis CI. One is Travis one of the most populars

Travis

When I push I will be notified of changes, then ttravis is going to pull code and run tests. Make sure that pass tests and it notify of changes to travis again

First we need a file for configutation for Travis

We use a file called YAML

YAML has a series of keys and values

For instance:

Key1: value1

Key2: value2

Key3:

* Itme1
* Item2
* Item3

Then the configuration is like this:

Language: python

Python:

* 3.8

Install:

-pip install -r requrements.txt

Script:

-python3.8 manage.py test

Directory airline3

There is a file called

.travis.yml

It has the following:

language: python

python:

- 3.6

install:

- pip install -r requirements.txt

script:

- python manage.py test

In travis I make any time I push Travis will know

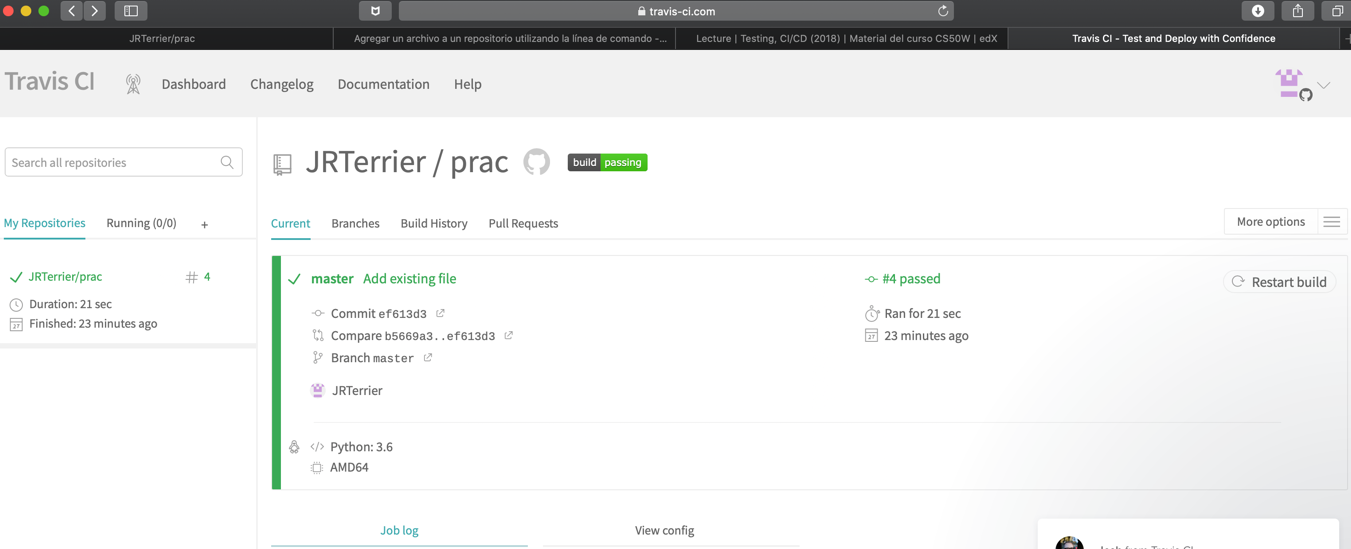
In terminal:

$ git add .travis.yml

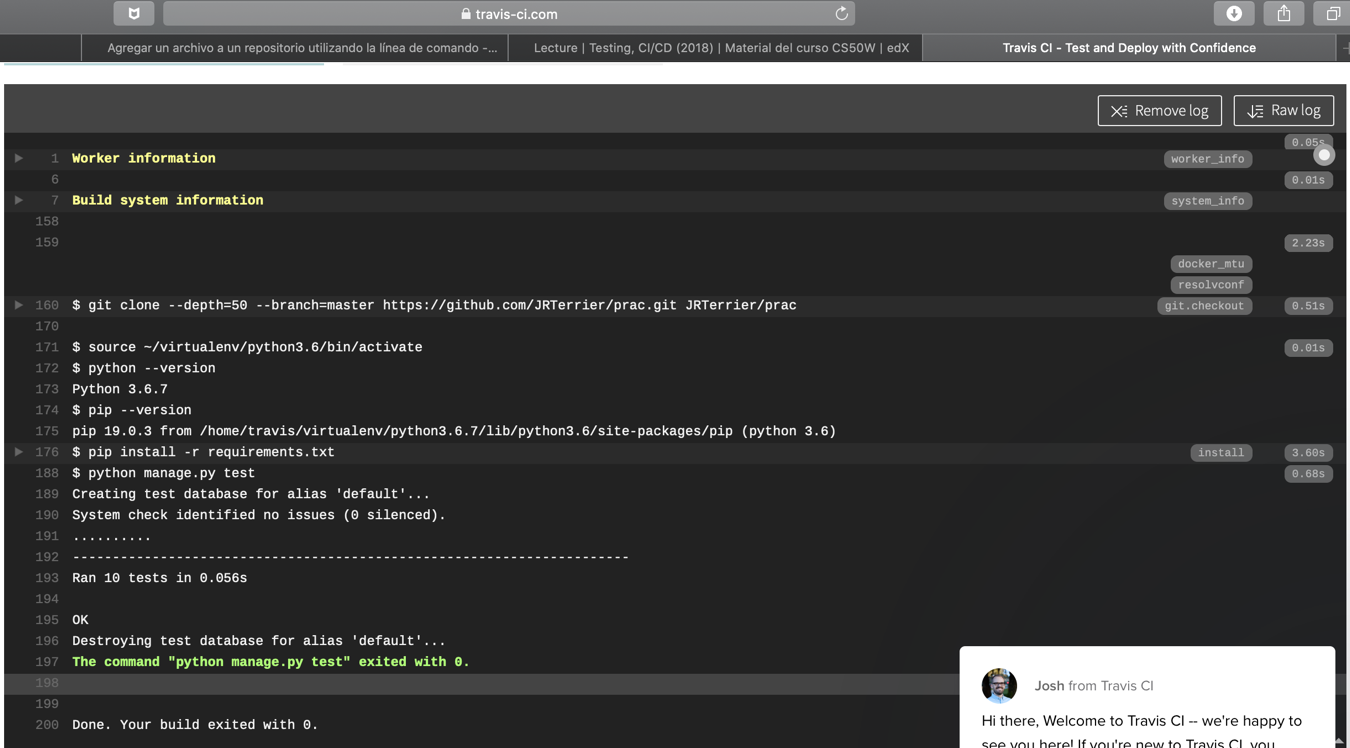
$ git commit -m “Add .travis.yml”

$ git push

If we go to Travis

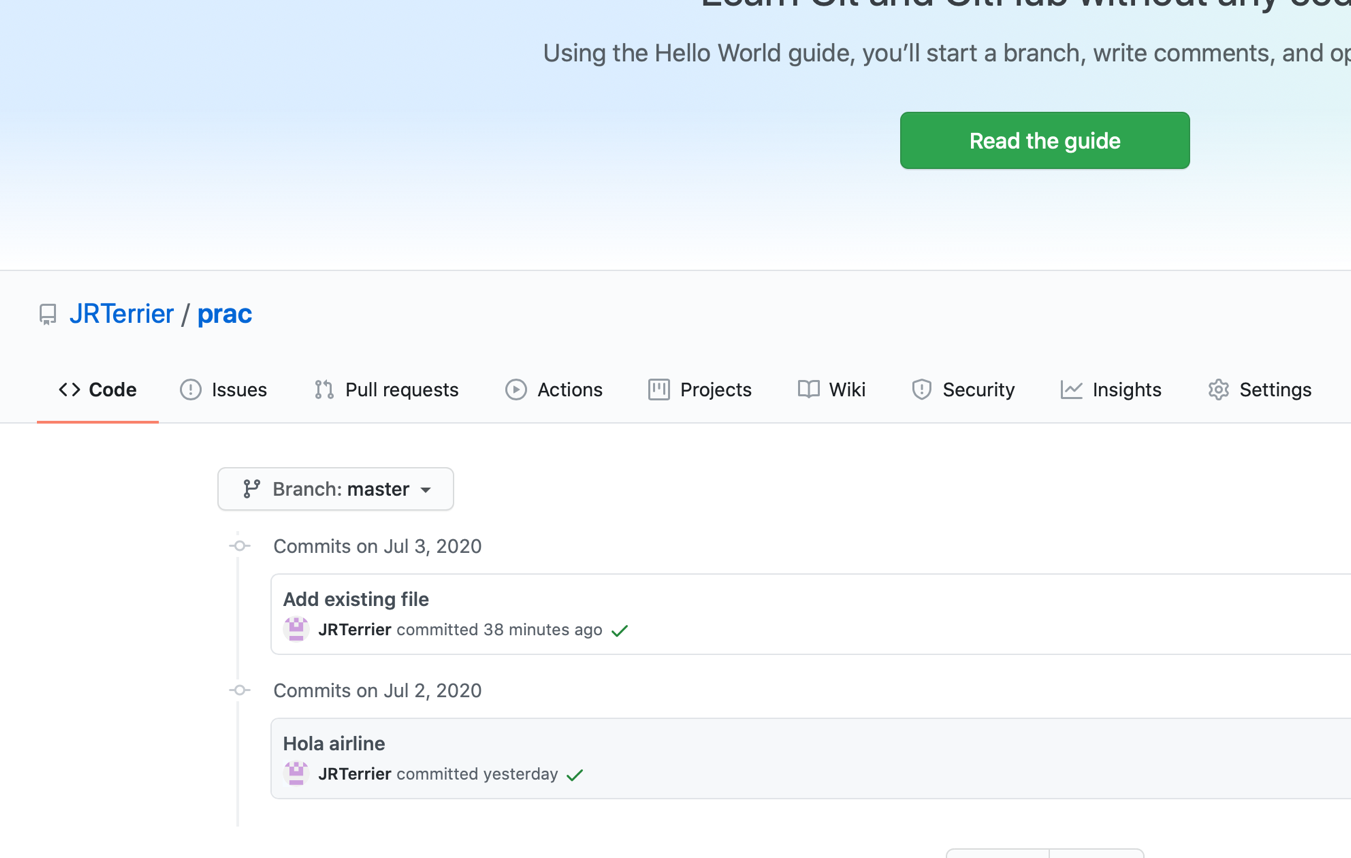


It produces automatic running



In this case, the program run 10 tests and produced a exit code 0, it means there were not errors

In this case it says that there were not errors in the commits. If any error, it will be appeared a red cross



Travis also:

Make sure whether we are sending notifications for instance a slat channel to say if something is succeded

Automatic deployment to a server somewhere

If all the test is fine, it can make an automatic deployment to internet, otherwise it does not

When we make an application, there is not guarantee that it is gonna work in another computer, probably in the computer we want to run the program does not have even Django, or they are using an older version of Django, so there are all types of compatibility errors between whether the app was created and where the app is run. So what I need to make sure is that the computer where the app is running, has the same environment needed for it to run, the libraries are installed. Which strategies I could follow?

The answer is a virtual machine!!

So I create a virtual operative system on top of the current one.

Which can be one drawback?

It is slower!

The answer is containerization when I need to install just some instances. It is creating containers; they are not completely virtual environments or machines but rather programs that I need to be installed, so if we can make sure the image I am using is the same than the image in the remote computer.

One of the most popular programs to make comtanerization is called Docker.

Docker is just going to build estright away from the host OS, and it is gonna construct containers to keep isolated the new programs, and it has the libraries, fpr the application to run. For instance, it can run flask and postgres in different containers.

The example is in airline4 directory

In there, there is a file called Dockerfile

FROM python:3

WORKDIR /usr/src/app

ADD requirements.txt /usr/src/app

RUN pip install -r requirements.txt

ADD . /usr/src/app

FROM python:3.8

This is to make sure we have python ver 3.98

WORKDIR /usr/src/app

This is the work dir for convention

ADD requirements.txt /usr/src/app

This is to add the file where the requirements are:

Django==2.0

psycopg2

The requirements are Django and psycopg2 (python library to work with postgres)

Then we install the requirements:

RUN pip install -r requirements.txt

Finally, add all my application files to the convention directory

ADD . /usr/src/app

Settings.py

In this case, postgres is gonna be used and in settings is defined:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'postgres',

'USER': 'postgres',

'HOST': 'db',

'PORT': 5432,

}

}

So the file is gonna make all the sincronization is the file docker-compose.yml

version: '3'

services:

db:

image: postgres

migration:

build: .

command: python3 manage.py migrate

volumes:

- .:/usr/src/app

depends\_on:

- db

web:

build: .

command: python3 manage.py runserver 0.0.0.0:8000

volumes:

- .:/usr/src/app

ports:

- "8000:8000"

depends\_on:

- db

- migration

It specified all the services will provide my web application.

The first service is a database from postgres

It will migrate everychanges

Built all

The command will be running python3.8 manage.py migrate

This is in order to run all the migrations

Volumes helps to link different files

Depends\_on makes sure whether the database is installed before the migration takes place.

Then, another service is gonna run is called web, which is related when the app will run, and is the actually one that will be running the application.

There is again a build an command an volumes. In the command there is ip address is running,

Ports run, my container has its self port, it says that my container may have its own port. So it says, take the ip address of the contaniner in the remote computer and send it to my own computer

Depends on db and that migration is over

Then, I run in my terminal:

Airline4 $ docker-compose up

And the first things we can see:

Starting airline4\_db\_1 … done

Starting ariline4\_migration\_1 … done

Starting airline4\_web\_1 … done

Then I can go to 0.0.0.0:8000 (webpage!)

I then I see what I have in my web application running docker

Note: I tried to, it was very difficult. But finally it worked. See file “salida guardada de la terminal.txt “ en airline4 directory

I made a few changes:

File: settings.py

Original:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'postgres',

'USER': 'postgres',

'HOST': 'db',

'PORT': 5432,

}

}

Modified:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'postgres',

'USER': 'postgres',

'PASSWORD': 'postgres',

'HOST': 'db',

'PORT': 5432,

}

}

Docker-compose.yml

Original:

version: '3'

services:

db:

image: postgres

migration:

build: .

command: python3 manage.py migrate

volumes:

- .:/usr/src/app

depends\_on:

- db

web:

build: .

command: python3 manage.py runserver 0.0.0.0:8000

volumes:

- .:/usr/src/app

ports:

- "8000:8000"

depends\_on:

- db

- migration

Modified:

version: '3'

services:

db:

image: postgres

environment:

- POSTGRES\_DB=postgres

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=postgres

migration:

build: .

command: python3.8 manage.py migrate

volumes:

- .:/usr/src/app

depends\_on:

- db

web:

build: .

command: python3.8 manage.py runserver 0.0.0.0:8000

volumes:

- .:/usr/src/app

ports:

- "8000:8000"

depends\_on:

- db

- migration

The result in a web page when we enter as 0.0.0.0:8000

Flights

Only that!!

All we have done is start up the application, it does not seem much but we have gain connect the docker file and and the docker compose file, that define exactly how to start up the web application. It uses postgres database , and it install all needed, docker is caring about it and it is running the database in the container. So I can provide the respository to anybody and as long as they have docker installed they can run this application and we expect to behave the same way. This is mainly when several people are working in the same project and they don’t need to install all the programs to make sure it will be working. So if we deploy an application to amazon for instance, we make sure that they will run the application and it will behave as we designed in their server. All the servers are running in the same docker container? No

If we type docker ps

It list all the containers we have and in this case are two, the web page and de database in separated container

And then we can go inside a container by

$ docker exec -it a4b223a3ee03 bash -l

Then we got inside the

root@ a4b223a3ee03:/usr/src/app# ls

Dockerfile airline db.squlite3 docker-compose.yml flights manage.py requirements.txt

Then I can run

root@ a4b223a3ee03:/usr/src/app# python manage.py shell

>>>

there we can make changes, additions, etc

if I need to make changes we don’t need to go inside the container unless I need for instance modify the database, for instance if I need to add a new airport

So we can interact between travis and docker for CI/CD and it makes sure that we are running similar environments since de design up to the deployment