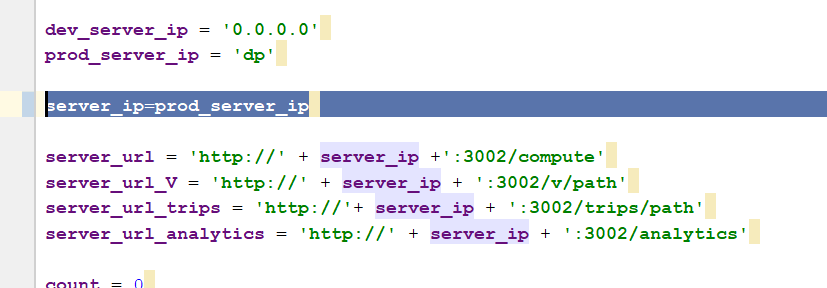
# ­Installation steps for Windows

## Deploying FastLanes on any operating system

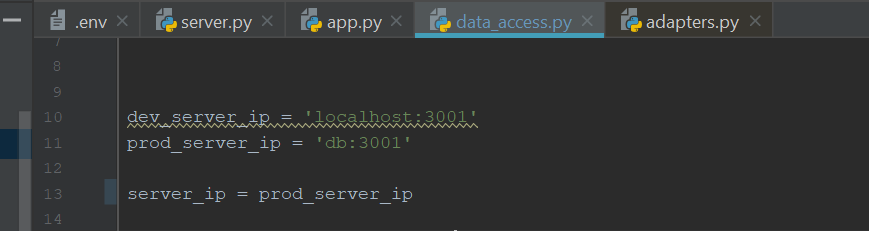
1. Download Docker Community Edition for your operating system
2. Download Git
3. Using Git Bash in the target folder, clone the FastLane code:

*Git clone* [*https://github.com/arnonkahani/FastLane.gitb*](https://github.com/arnonkahani/FastLane.gitb)*.*

1. When you pull from GitHub, the system is configured for Dev, so ports changes are required <<This needs to be changed>>>
2. Open app.js and change the server\_ip var to take the prod\_server\_ip, like this:



1. Open data\_access.py and change the server\_ip to take the prod\_server\_id like this:



1. Open command line in the FastLane root folder (you should have a docker-config.yml file there) and type *docker-compose up*
2. Proceed on Chrome to <http://localhost:8080> and play around with the tool

## Developing FastLanes on Windows

The following software tools are required to be able to debug and develop new functionalities for FastLanes:

1. Download Docker Community Edition for your operating system
2. Download Git
3. Using Git Bash in the target folder, clone the FastLane code:

*Git clone https://github.com/arnonkahani/FastLane.git*

1. Download NodeJS
2. Download PostgreSQL, post GIS amd pg4admin – **CHECK IF THIS CAN REMOVED AND ONLY HAVE P4GADMIN**
3. Download Python 3.6
4. Download PyCharm Pro (not community edition, in order to support flask) – students have 6-month trial license that can be extended when needed using BGU email
   1. Install dependencies and plugins (Ctrl+Alt+S -> Project -> Project Interpreter -> :
      1. Flask Integration
      2. GeoJson
      3. Shapely
      4. requests
5. Download Webstorm – license as above
6. Open git bash terminal (right-click in any Windows Explorer window) in the FastLanes root folder and pip-install any **needed module**

# Setting up Fastlanes for Dev & Debug on Windows

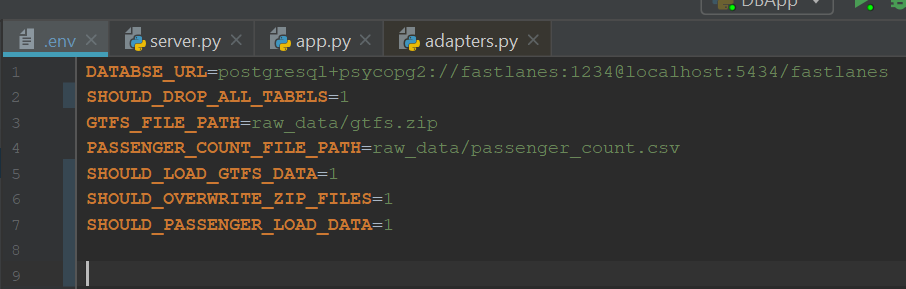
## General Note about Deployment vs. Developments environments

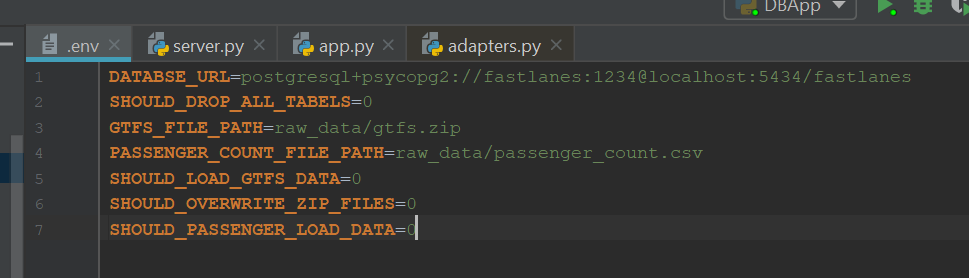
While the docker environment is great for deployment on different operating systems, it does mean that the containers are Linux-based. So, if one tries to, let’s say, try to start the PostgreSQL DB (or PostGIS in our case) using traditional postgres server and psql terminal on **Windows**, all sorts of errors are displayed because the DB was created on Linux.

Therefore, the easiest way to setup a development environment is using the docker only for the postGIS DB and run the other servers using PyCharm and WebStorm.

## Run FastLanes for Dev & Debug

### If you already setup the environment once

1. **Start the PostGIS DB:** 
   1. Open a new command line in the FastLanes root folder
   2. Type *Docker-compose up postgis*
   3. Wait until the terminal logs: “database system is ready to accept connections
2. **Start up the UI server:**
   1. Open Webstorm
   2. Run app.js
3. **Start up the Data process and Database servers**
   1. Open PyCharm
   2. If you wish to load data do the following, otherwise go to the next step (c):
      1. Replace the existing files with new GTFS/Passenger files under *C:\Dev\FastLane\DB\raw\_data*
      2. *Open the .env* file in PyCharm and make sure that the corrsesponding variables are set to 1:
      3. 
   3. Run **both** app.py files under DP folder and DB folder
   4. Change the *.env* file back to 0s in order to avoid data overwriting next time you restart the system:



1. **Access UI**
   1. Navigate in Chrome to <http://localhost:8080> and hack away

### First time configuring the environment

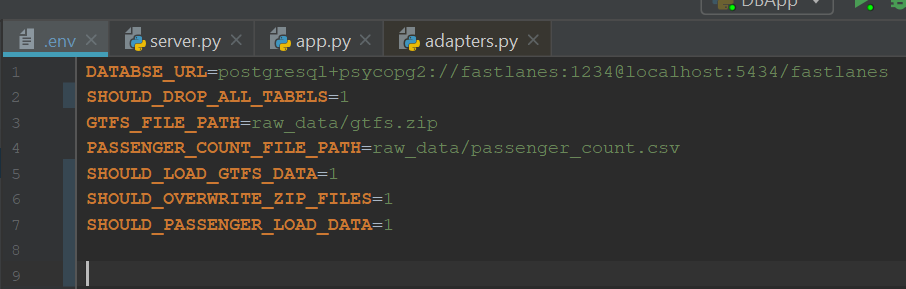
1. **Create a volume for PostGIS data**

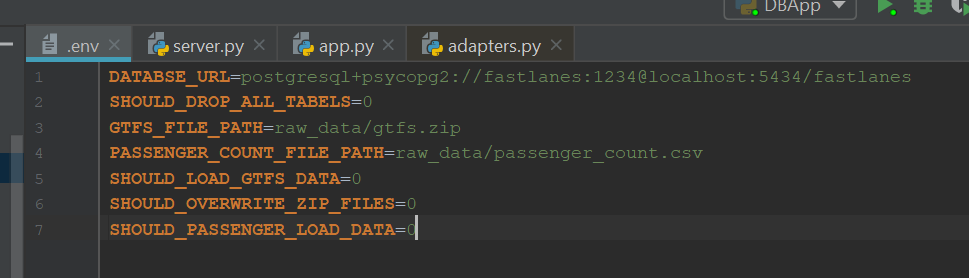
Because the initial docker configuration was created on Mac, a volume needs to be created (the original docker configuration file is docker-compose\_orig\_for\_mac\_linux.yml – the Windows configuration file is now the main one: docker-compoe.yml)

* 1. Open a new command line in the FastLanes root folder
  2. Type *dcoker volume create --name=postgis\_data*

1. **Start the PostGIS DB:** 
   1. Type *docker-compose up postgis*
   2. Wait until the terminal logs: “database system is ready to accept connections
2. Validate the DB is accessible using **PG4Admin**
   1. Run PG4Admin program
   2. Click *Add New Server*
   3. Fill in the following details:
      1. General -> Name: *FastLanes*
      2. Connection -> Host Name/Adress: *localhost*
      3. *Port -> 5434* (note: this is the port that the docker exposes for accessing the DB outside of the container. Inside the container the port is the default postgres port 5432)
      4. Maintenance database: *postgres*
      5. Username*: fastlanes*
      6. Click *Save*
      7. When prompted for a password, type *1234*
   4. Depending if the system was run previously and data was already populated into it, validate the schema tables and data exist:
      1. Go to *Servers -> FastLanes -> Databases -> fastlanes -> Schemas -> Public -> tables*
      2. Right click on any table to view the first 100 rows.
3. Configure **Webstorm**
   1. Open a terminal in the project root folder/UI and run *npm install* to install all required dependencies.
   2. Run Webstorm
   3. Open existing project by selecting the FastLanes root folder
   4. Run app.js
4. **Access UI**
   1. Navigate in Chrome to <http://localhost:8080>­
5. Configure **PyCharm**
   1. Run PyCharm
   2. Open existing project by selecting the FastLanes root folder
6. **Set proper DB server configuration**

When the DB server is launched, it uses the .env file in order to determine whether GTFS/passenger counts data will be populated and override existing data or not.

* 1. Open Pycharm -> DB -> .env
  2. If you wish to load data do the following, otherwise go to the next step (c):
     1. *Download israel-public-transportation from ftp://gtfs.mot.gov.il*
     2. Replace the existing files with new GTFS/Passenger files under *C:\Dev\FastLane\DB\raw\_data*
     3. *Open the .env* file in PyCharm and make sure that the corrsesponding variables are set to 1:
     4. 
  3. Run **both** app.py files under DP folder and DB folder
  4. Change the *.env* file back to 0s in order to avoid data overwriting next time you restart the system:



1. **Run the DB & DP Servers**
   1. In Pycharm, Run **both** app.py files under DP folder and DB folder
   2. If you encounter “can’t resolve ..” errors, install the missing package as mentioned in the beginning of this document