

לעבור מ DB לוקאלי לענן:

פותחים חשבון ב mlab כדי לשמור בענן את DB של מונגו.

בקובץ הראשי של הפייתון:

- Add secret_key

- Install dnspython package

```
pip install dnspython
```

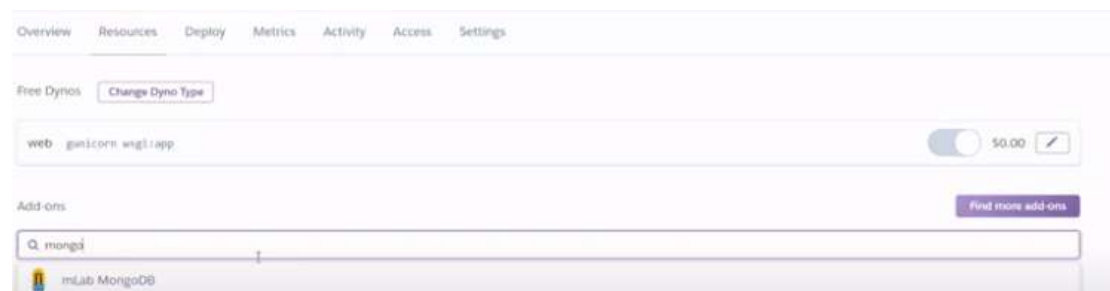
```
7 # MongoDB Connection
8 app.config["MONGO_URI"] = "mongoexample"
9 app.config['MONGO_DBNAME'] = 'mongodb://localhost:27817/mongoexample'
10 |
11 mongo = PyMongo(app)
```



```
# MongoDB Connection
app.config["MONGO_URI"] = "mongodb+srv://flaskmongo:27817@adwwebtech-88v1b.mongodb.net/usersDatabase?retryWrites=true&majority"
app.config['MONGO_DBNAME'] = 'usersDatabase'
app.config['SECRET_KEY'] = 'secret_key'
```

שילוב mlab עם Heroku:

באתר של Heroku להירשם ולפתוח חשבון, לאחר מכן על מנת לשלב את DB נבצע את הצעדים כמו בתמונה.



הוספת משתנים מבלי לחשוף סיסמאות ושם משתמש:

באתר של Heroku נכנס ללשונית settings ושם נבחר ב config vars את המשתנים והערכים שניתן להם, כך נוכל להשתמש בהם בקוד.

במקום הכתובת של ה local host כתבנו את הדבר הבא:

```
app.config["MONGO_URI"] = os.environ.get("MONGODB_URI")
```

```
'app.config['MONGO_DBNAME'] = 'multiTrade
```

```
'app.config['SECRET_KEY'] = 'mySecret
```

The screenshot shows the Heroku Settings page for an application named 'multitrade'. The 'App Information' section displays details such as Region (United States), Stack (heroku-18), Framework (Python), Slug size (60.1 MIB of 500 MIB), and Heroku git URL (https://git.heroku.com/multitrade.git). The 'Config Vars' section is visible, showing a table with columns for KEY and VALUE. A 'MONGODB_URI' key is partially visible with a masked value. There is an 'Add' button to add new configuration variables.

כאשר `os.environ.get("MONGODB_URI")` זה הכתובת של `mlab` כולל שם משתמש וסיסמה שמוזנת באתר Heroku. (צריך לעשות `Heroku (import os # for Env`).

התקנת Heroku ועליית הפרויקט לרשת.

בשלב הבא נתקין את התוכנה של Heroku CLI בקישור <https://devcenter.heroku.com/articles/heroku-cli>

* חשוב בפרויקט לעשות `pip install gunicorn`

לאחר מכן נפתח תיקייה חדשה עבור הפרויקט לפתוח `cmd` ולרשום `heroku login` - לוודא שזה האימייל הנכון.

אחרי שההתחברות הצליחה - בתוך ה `cmd` נלך לתיקייה שיצרנו. ונרשום את הפקודה `virtualenv env` אם זה לא מצליח נבצע את 2 הפקודות הבאות:

Run `pip uninstall virtualenv` and then `pip install virtualenv`

לאחר מכן נלך לתיקיה `env` ומשם ל `scripts` ונפעיל את הקובץ `activate` ע"י זה שנרשום פשוט את שם הקובץ ונלחץ `enter`.

```
C:\Users\CaiteEdna\Documents\my-project>env\Scripts\activate
(env) C:\Users\CaiteEdna\Documents\my-project>
```

נוודא שאנחנו בתיקיה הראשית של הפרויקט ונבצע את הפקודה `git init`.

אחרי זה נכנס לאתר של Heroku ללשונית `deploy` ונבצע את הפקודה:

```
git:clone -a multitrade
```

ניצור בתיקיית הפרויקט את הקובץ `Procfile` (ללא סיומת) אשר נראה כך:

```
web: gunicorn app:app
clock: python app.py
```

הערה : ה `clock` עבור ה `scheduler`.

חשוב! נשים לב שלקובץ הראשי קוראים `app.py`.

נבצע את הפקודות הבאות ברצף:

```
git add .
git commit -am "make it better"
git push heroku master
```

לאחר מכן נבצע את הפקודה :

```
pip freeze > requirements.txt
```

על מנת שנעלה את כל הדרישות לפרויקט וכך הם יותקנו לנו באופן אוטומטי.

לאחר מכן נבצע שוב את הפקודות :

```
git add .  
git commit -am "make it better"  
git push heroku master
```

כדי להפעיל את הscheduler נבצע את הפקודה הבאה:

```
heroku ps:scale clock=1
```

כאשר clock זה ה label אשר מופיע בקובץ ה Procfile .

קישורים בהם נעזרנו:

<https://medium.com/@summerxialinqiao/connect-flask-app-to-mongodb-atlas-using-pymongo-328e119a7bd8>

<https://www.youtube.com/watch?v=8sqLY28FS5Y>

<https://www.youtube.com/watch?v=sqJSdJbOOU0>

<https://www.youtube.com/watch?v=n1P8B53CCxs>