Assignment 4

180919

* Understand and report your opinion about restful/restless, stateless/stateful

A RESTful Web Service is Stateless, meaning that it doesn’t keep a client state on the server. The client is responsible for passing its context to the server; then the server can store this context to process the client's further request. For example, session maintained by server is identified by session identifier passed by the client.

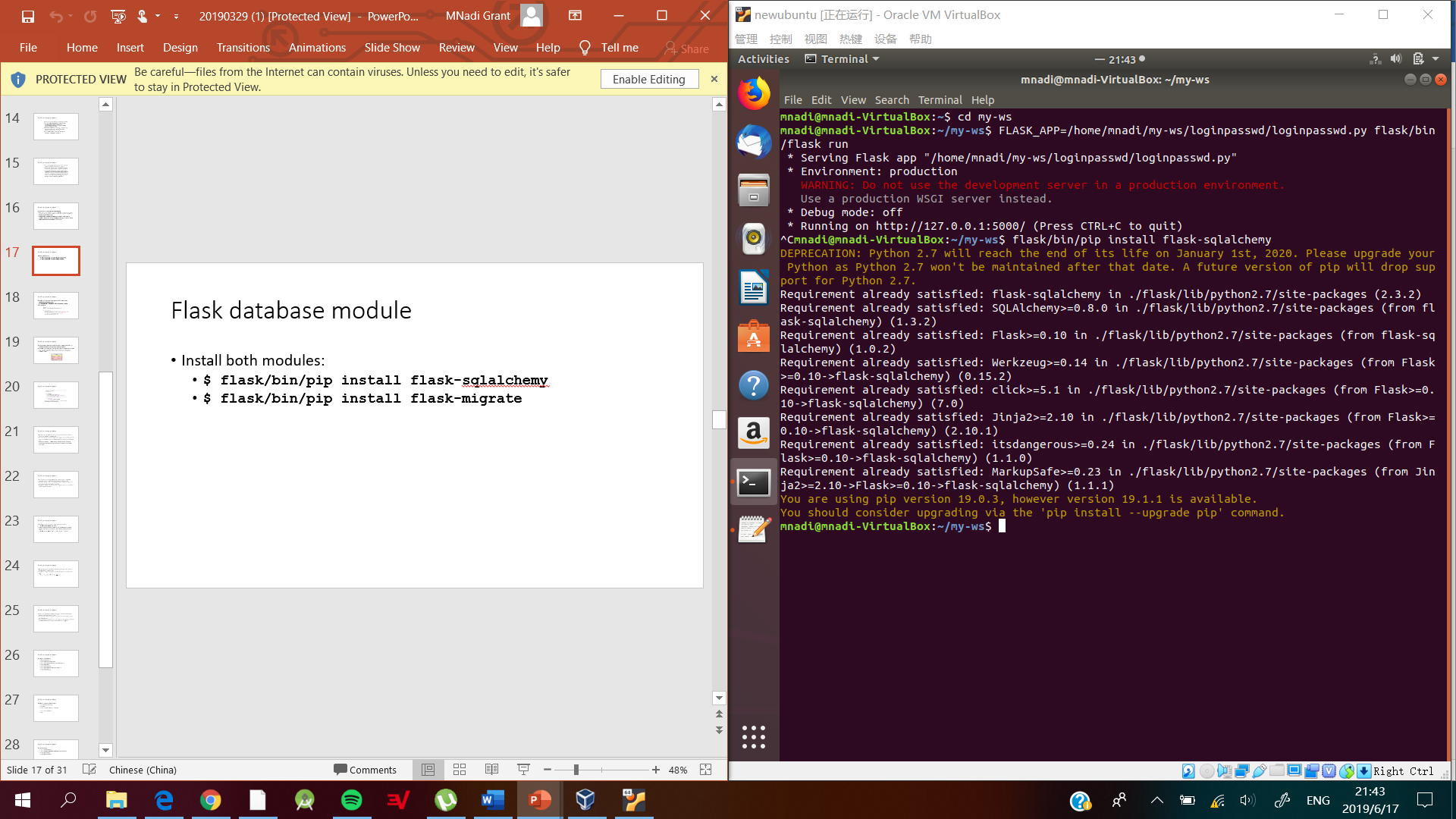
A RESTless Web Service is StateFul, meaning that it doesn’t adhere to the RESTful principle. For example, a web server stores the username and password of a user in a backend manner and uses it to identify the user. As you use the web service, everything you do is referenced back to this stored state.

* Understand SQLAlchemy module as a middle-layer for Flask-DB

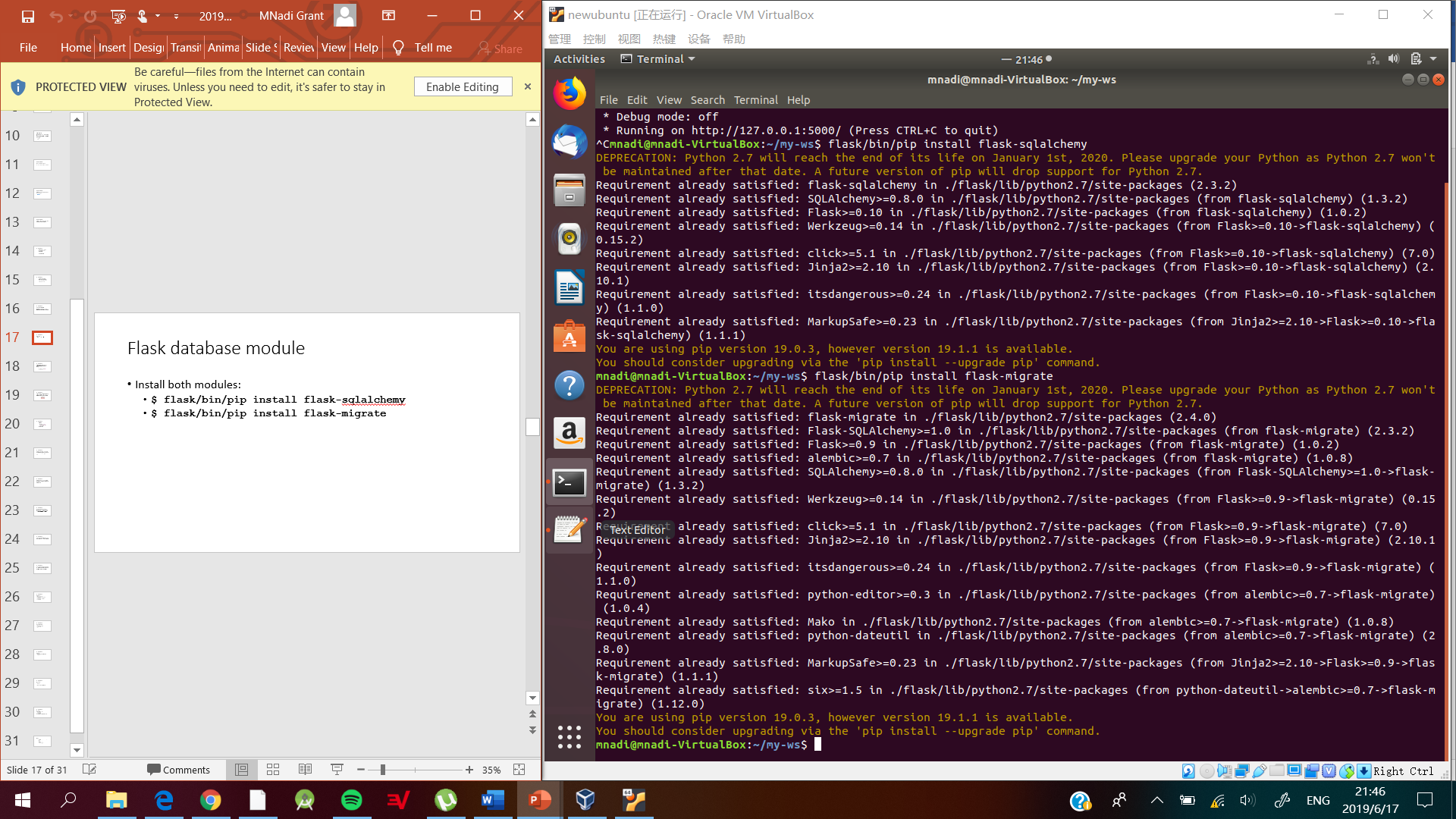
Flask uses the extension Flask-SQLAlchemy which adds support for SQLAlchemy to the application. The is done to simplify, using SQLAlchemy with flak by providing useful default and extra assistance in order to complete common task. The ORM layer within SQLAlchemy translations map objects created from these classes into rows in the proper database tables. As a middle layer, the SQLAlchemy module uses the migration framework which makes doing changes in the database schema easy.

* Install SQLAlchemy & migrate for Flask, and try simple examples

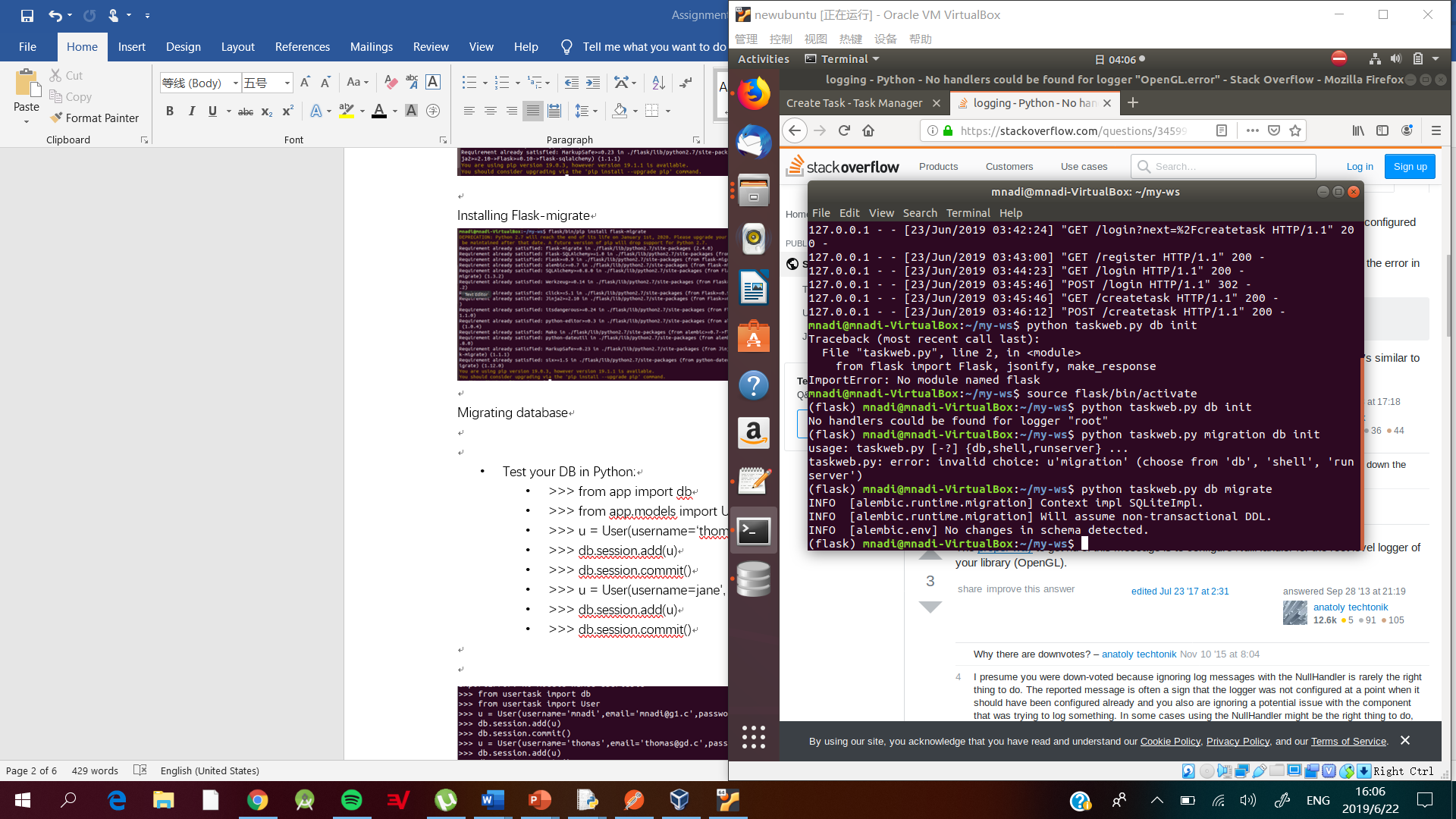
Installing SQLAlchemy



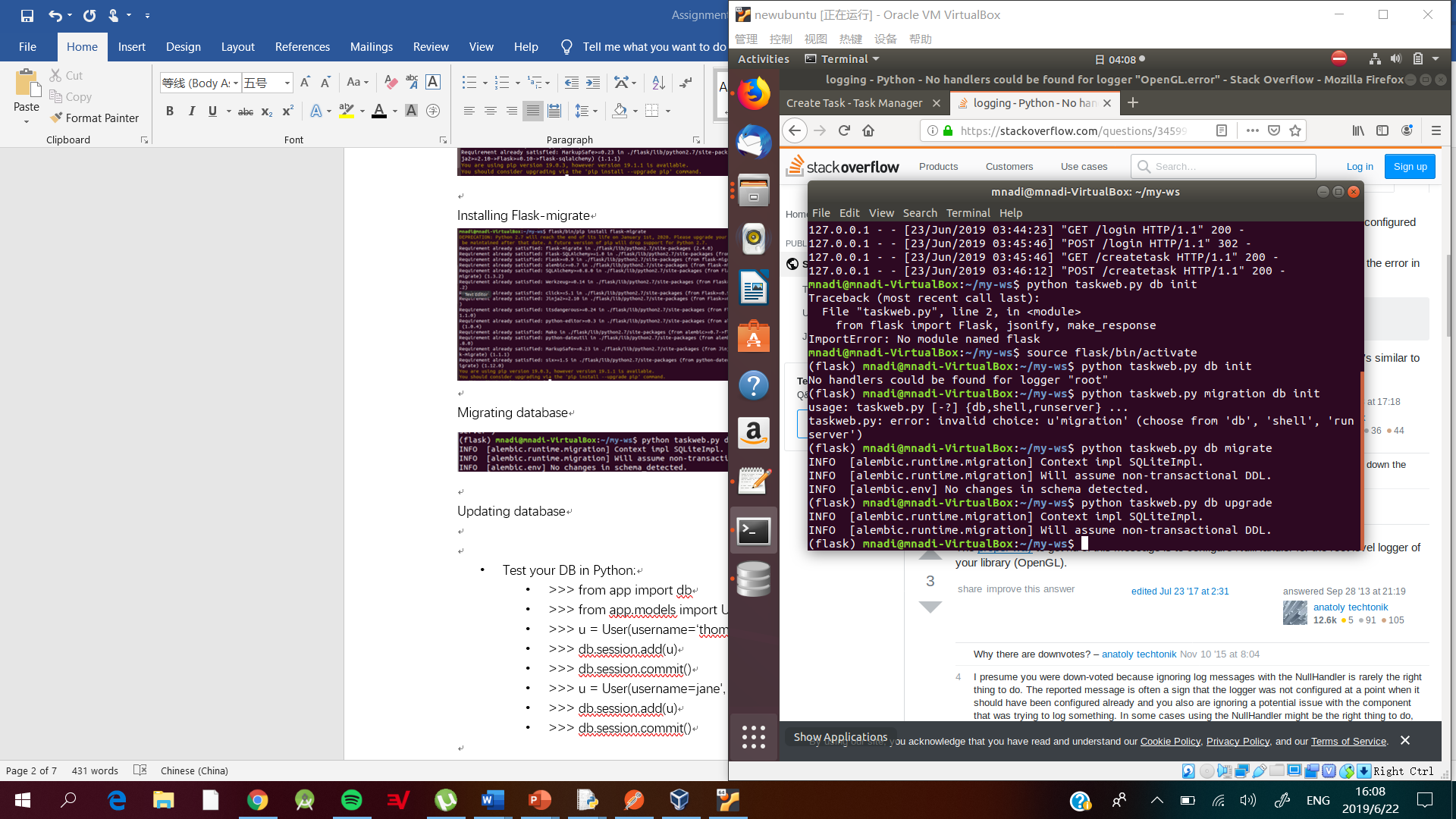
Installing Flask-migrate



Migrating database



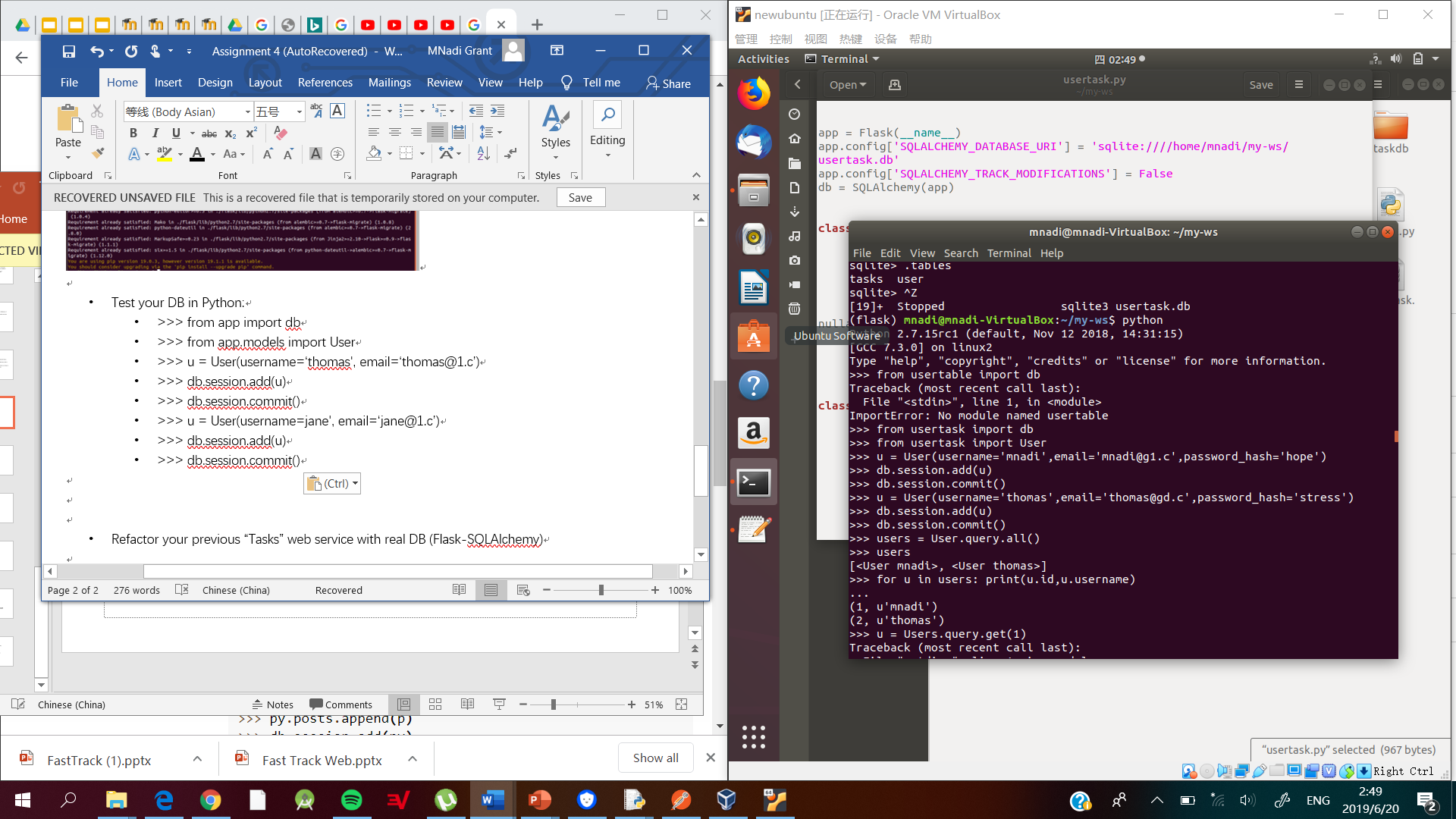
Updating database



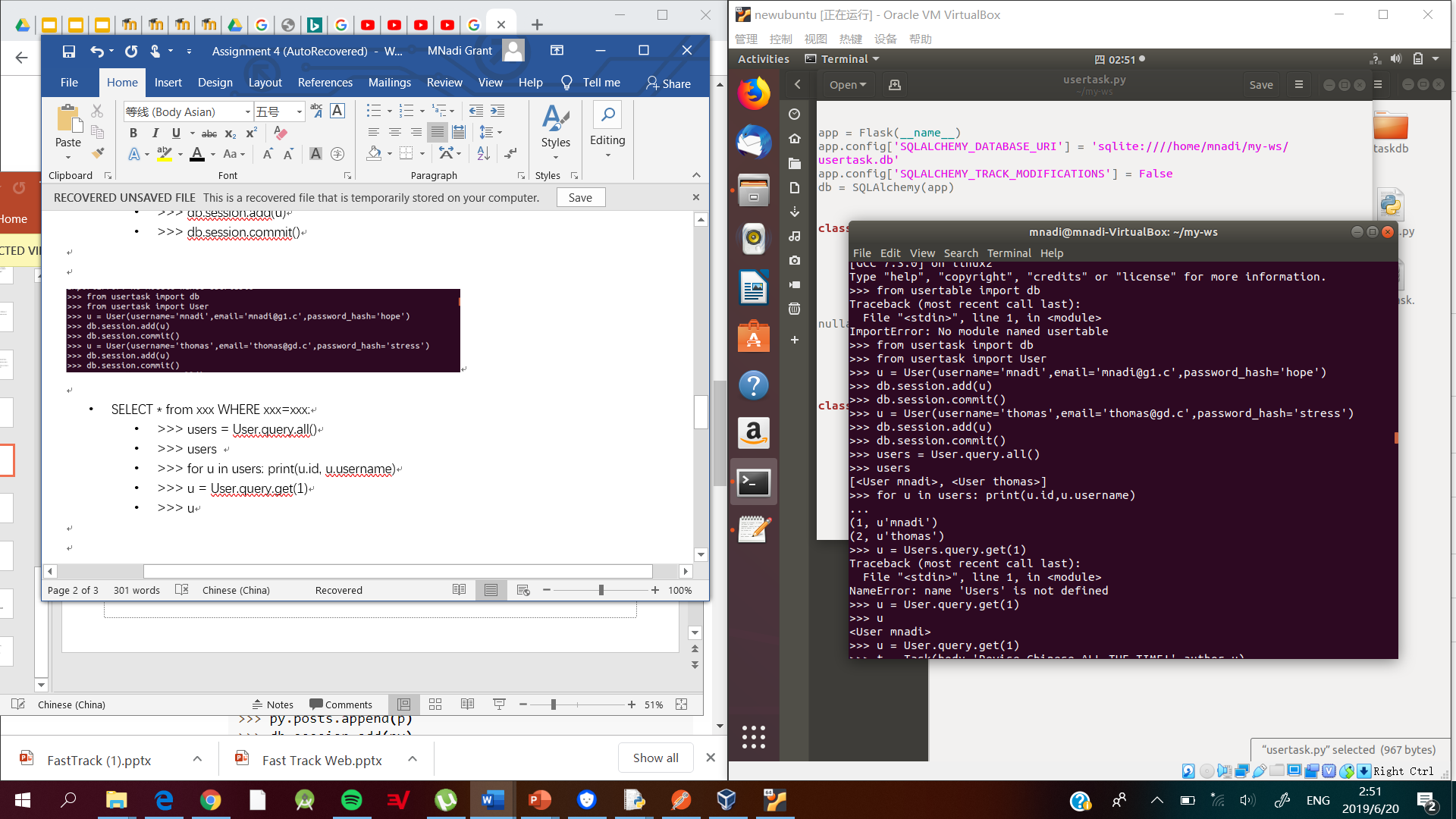
**NOTE: Due to no changes being made to the database the following was executed.**

TESTING EXAMPLES SHOWN IN SLIDE

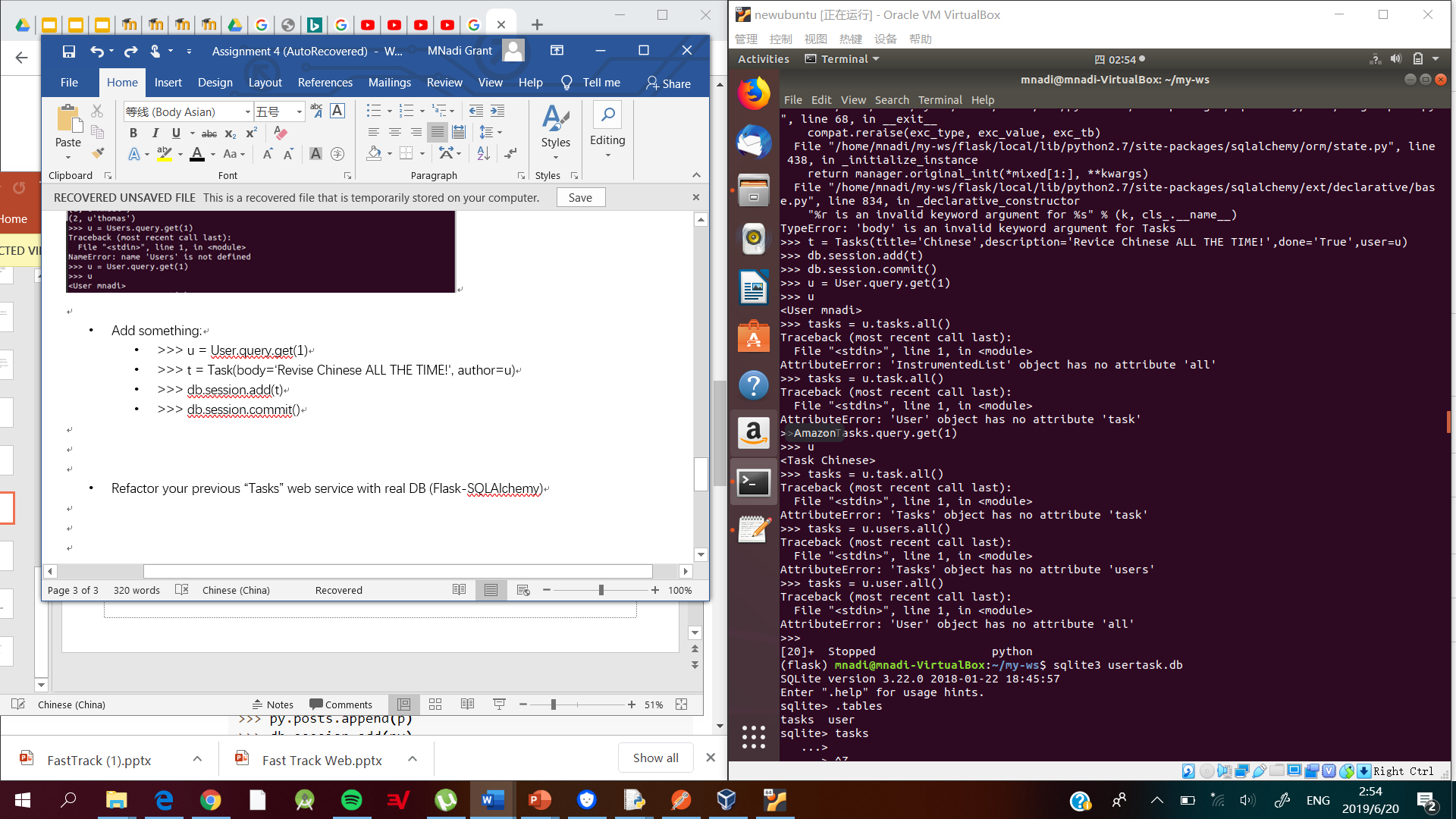
* Test your DB in Python:
  + >>> from app import db
  + >>> from app.models import User
  + >>> u = User(username=‘thomas', email=‘thomas@1.c’)
  + >>> db.session.add(u)
  + >>> db.session.commit()
  + >>> u = User(username=jane', email=‘jane@1.c’)
  + >>> db.session.add(u)
  + >>> db.session.commit()



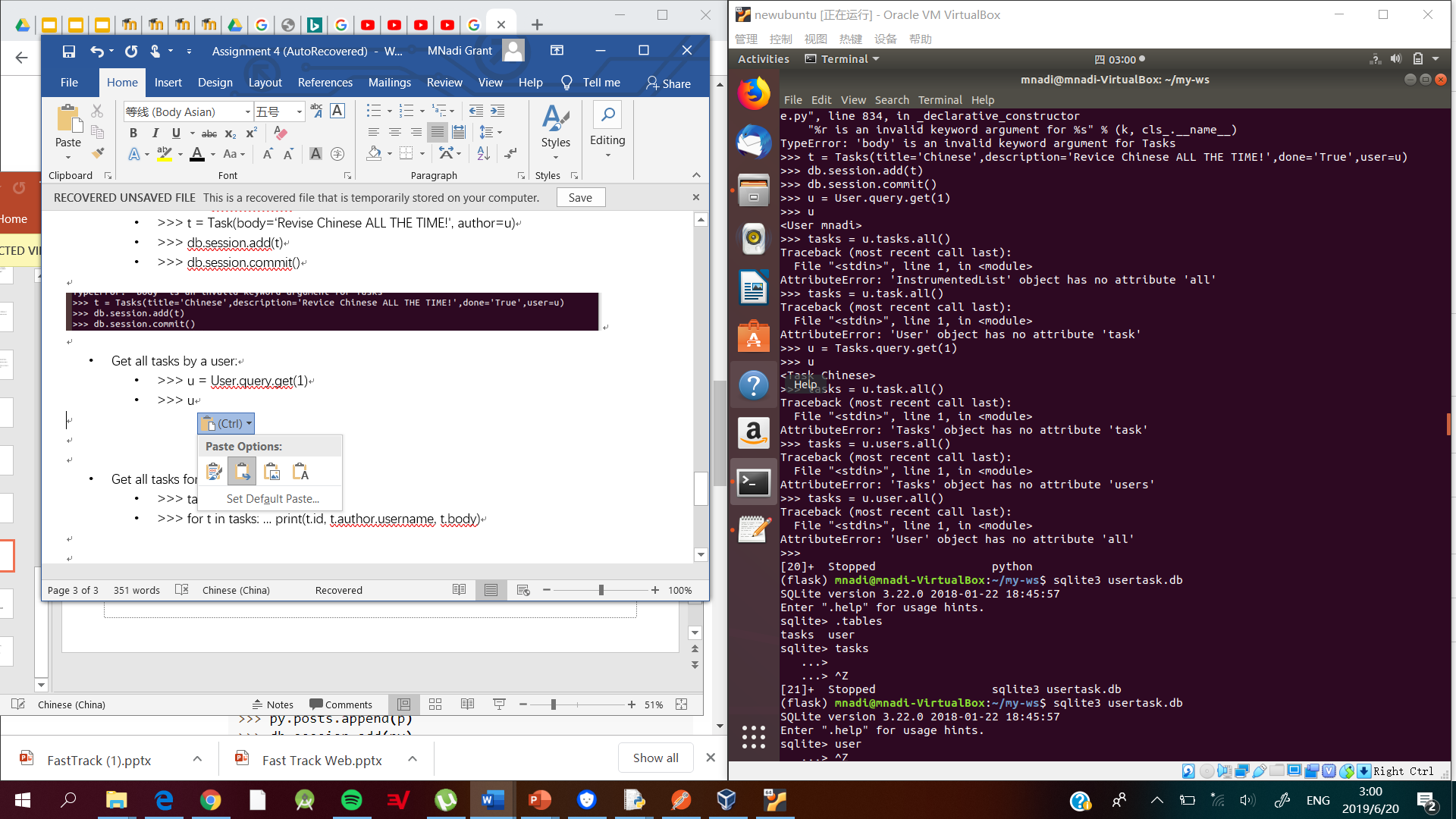
* SELECT \* from xxx WHERE xxx=xxx:
  + >>> users = User.query.all()
  + >>> users
  + >>> for u in users: print(u.id, u.username)
  + >>> u = User.query.get(1)
  + >>> u



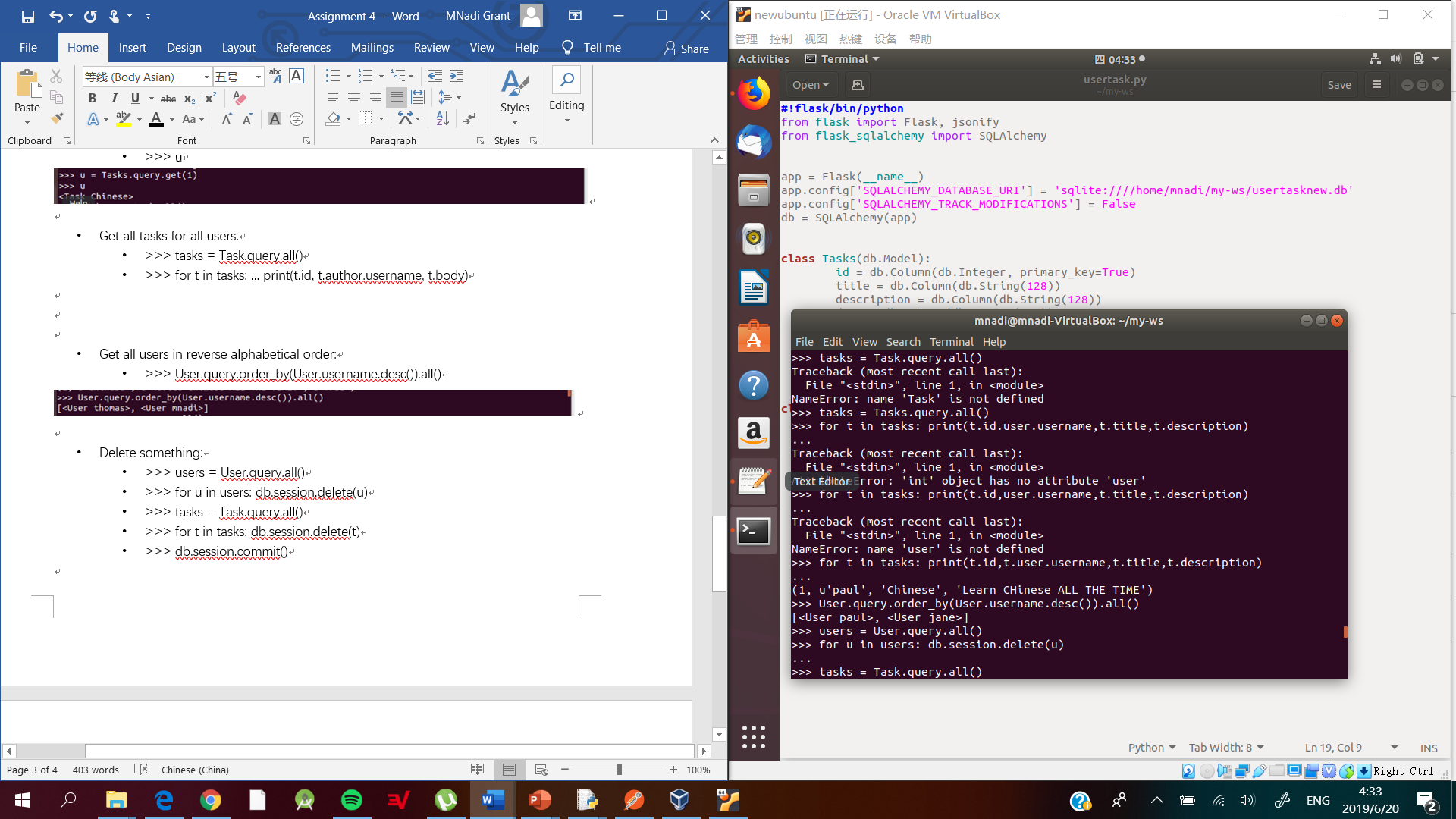
* Add something:
  + >>> u = User.query.get(1)
  + >>> t = Task(body=‘Revise Chinese ALL THE TIME!', author=u)
  + >>> db.session.add(t)
  + >>> db.session.commit()



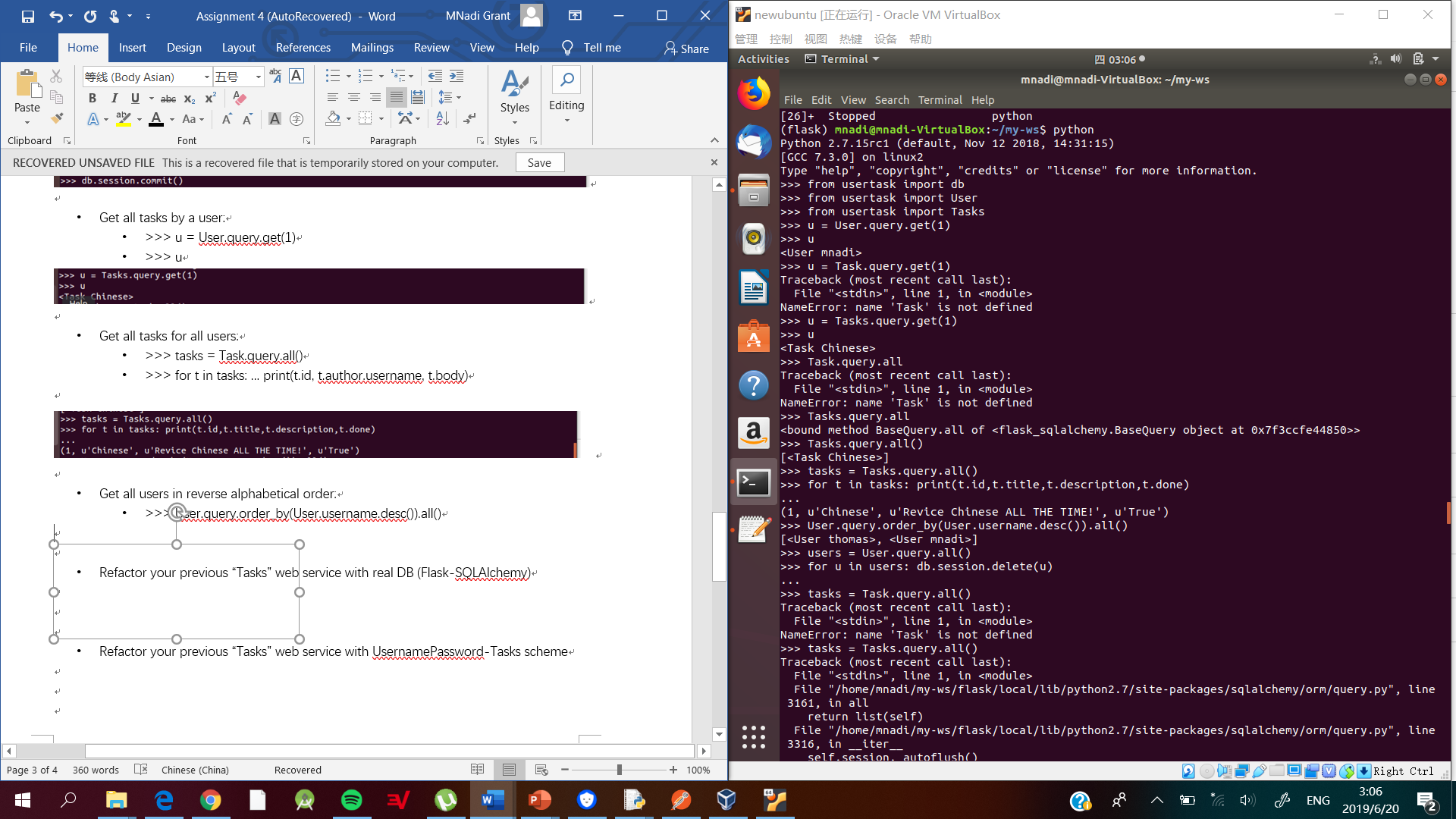
* Get all tasks by a user:
  + >>> u = User.query.get(1)
  + >>> u



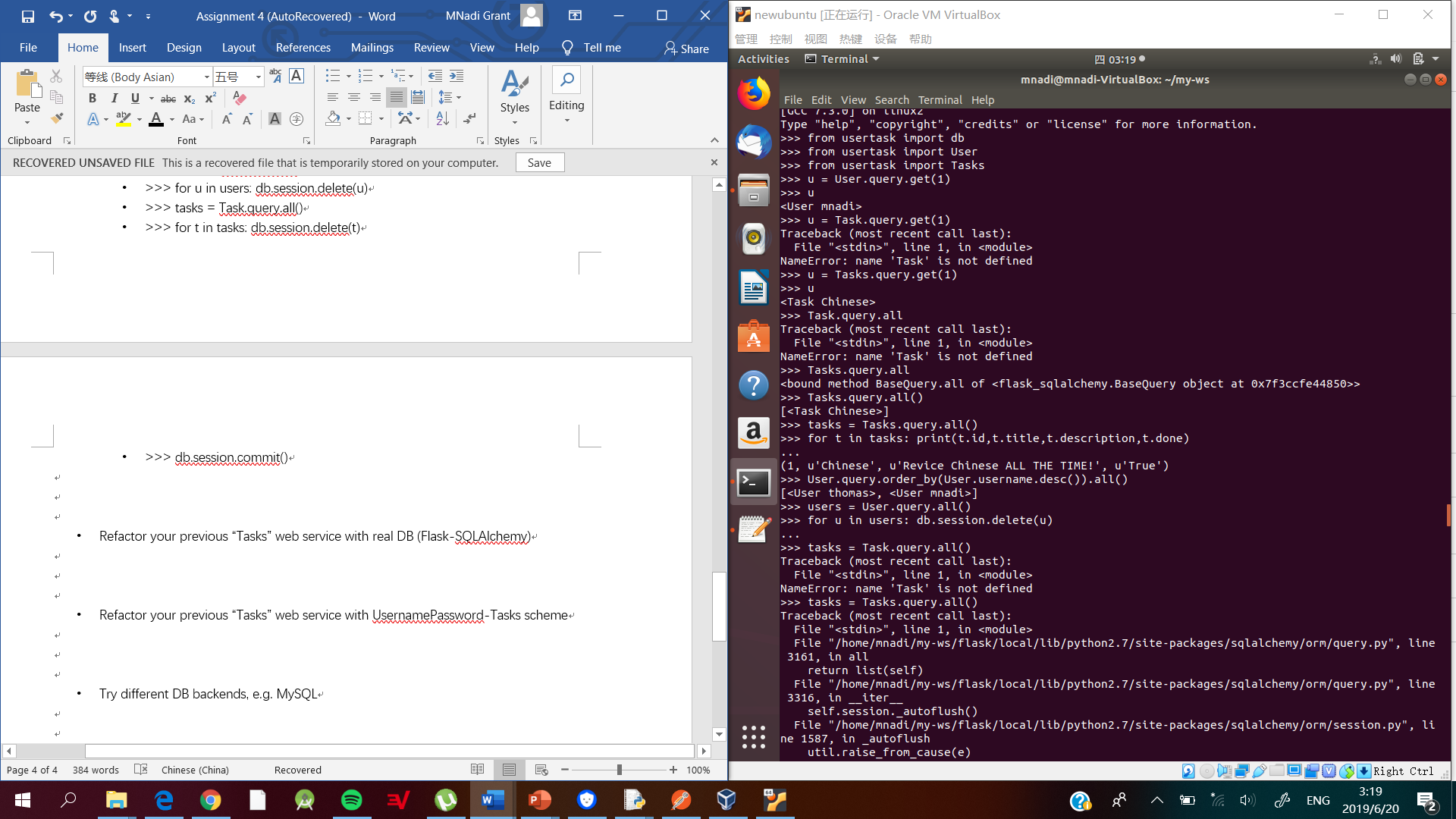
* Get all tasks for all users:
  + >>> tasks = Task.query.all()
  + >>> for t in tasks: ... print(t.id, t.author.username, t.body)



* Get all users in reverse alphabetical order:
  + >>> User.query.order\_by(User.username.desc()).all()



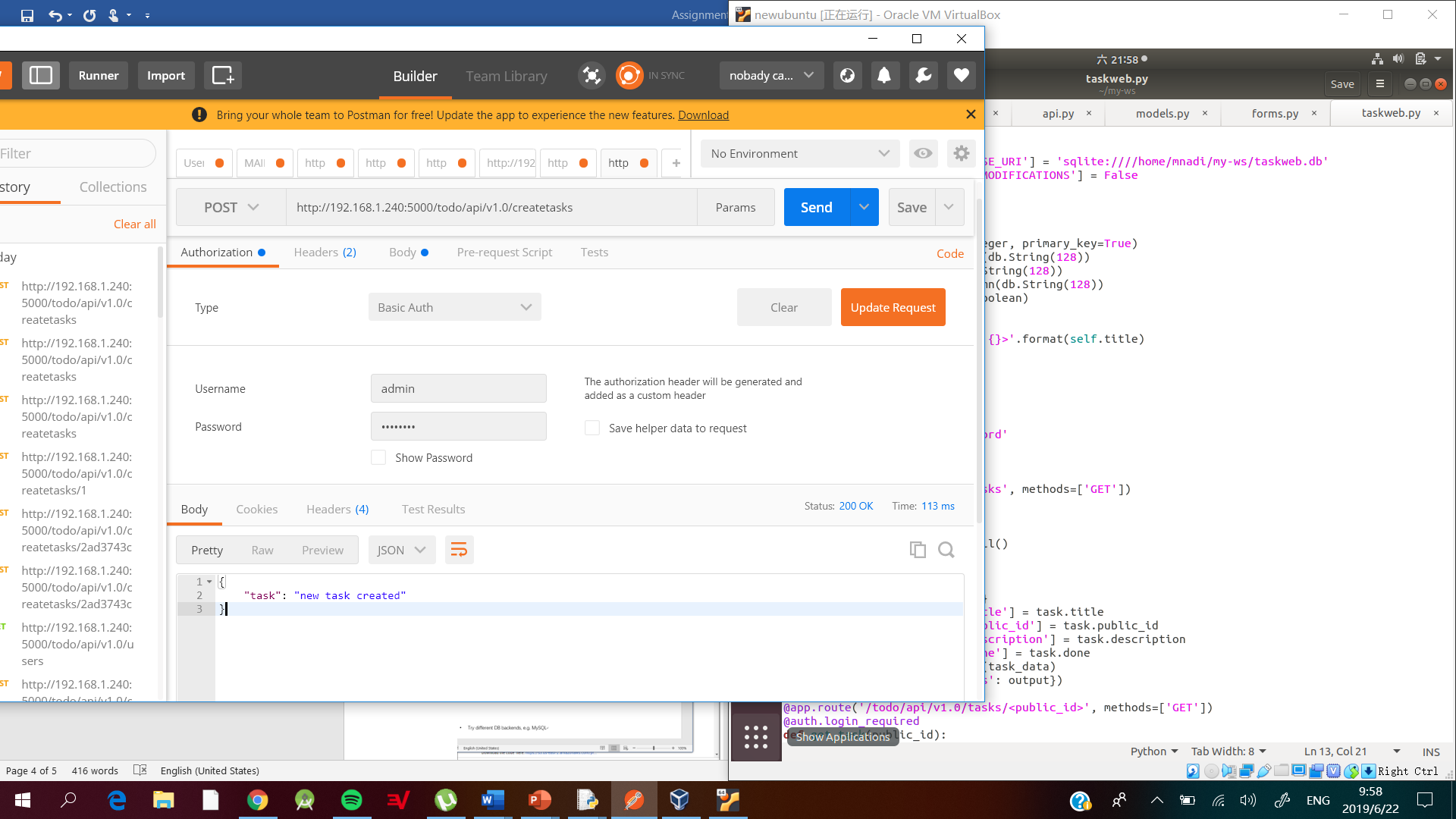
* Delete something:
  + >>> users = User.query.all()
  + >>> for u in users: db.session.delete(u)
  + >>> tasks = Task.query.all()
  + >>> for t in tasks: db.session.delete(t)
  + >>> db.session.commit()



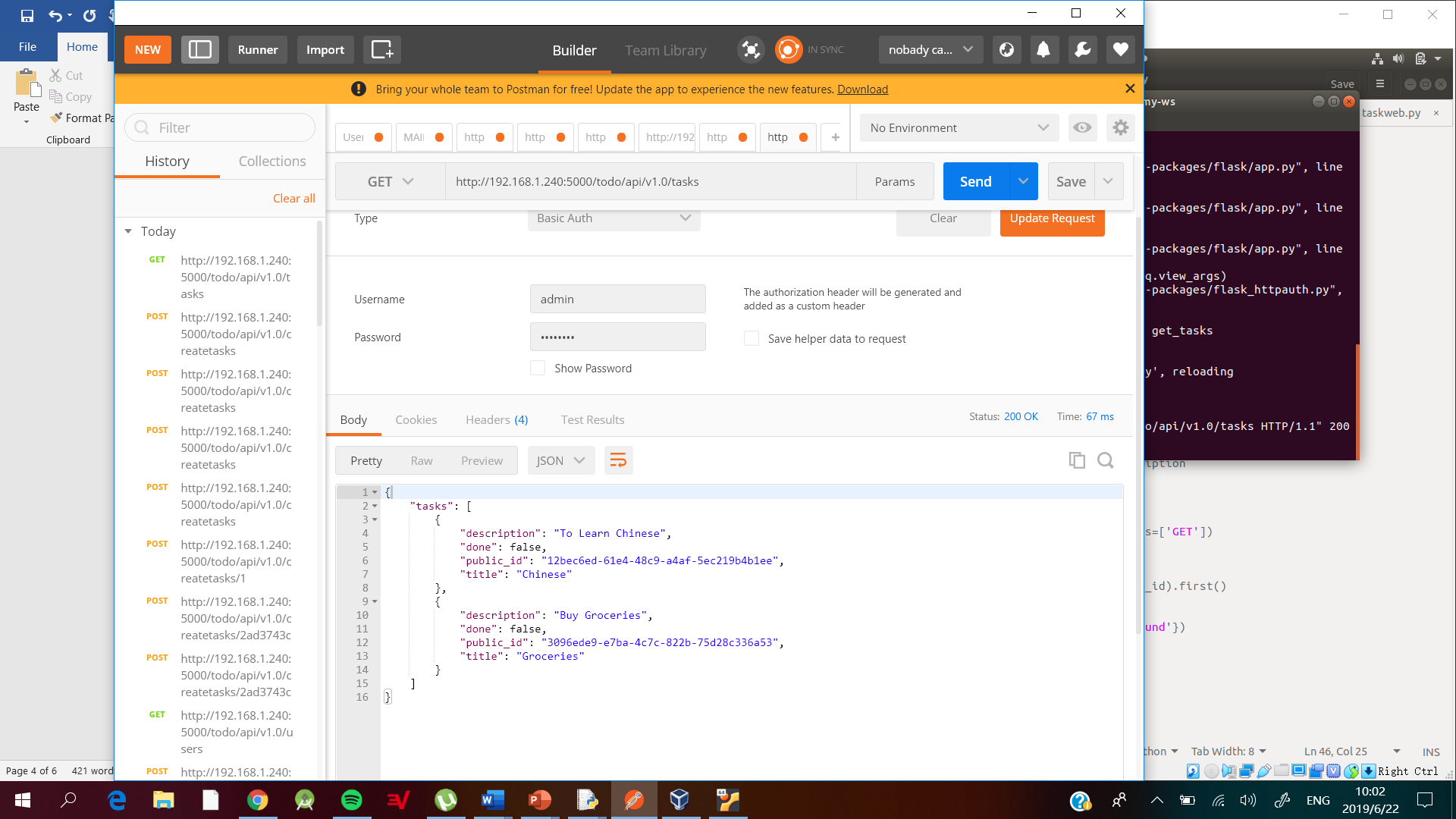
An error occurred when trying to view all tasks because of the relationship between the Task and User table.

* Refactor your previous “Tasks” web service with real DB (Flask-SQLAlchemy)

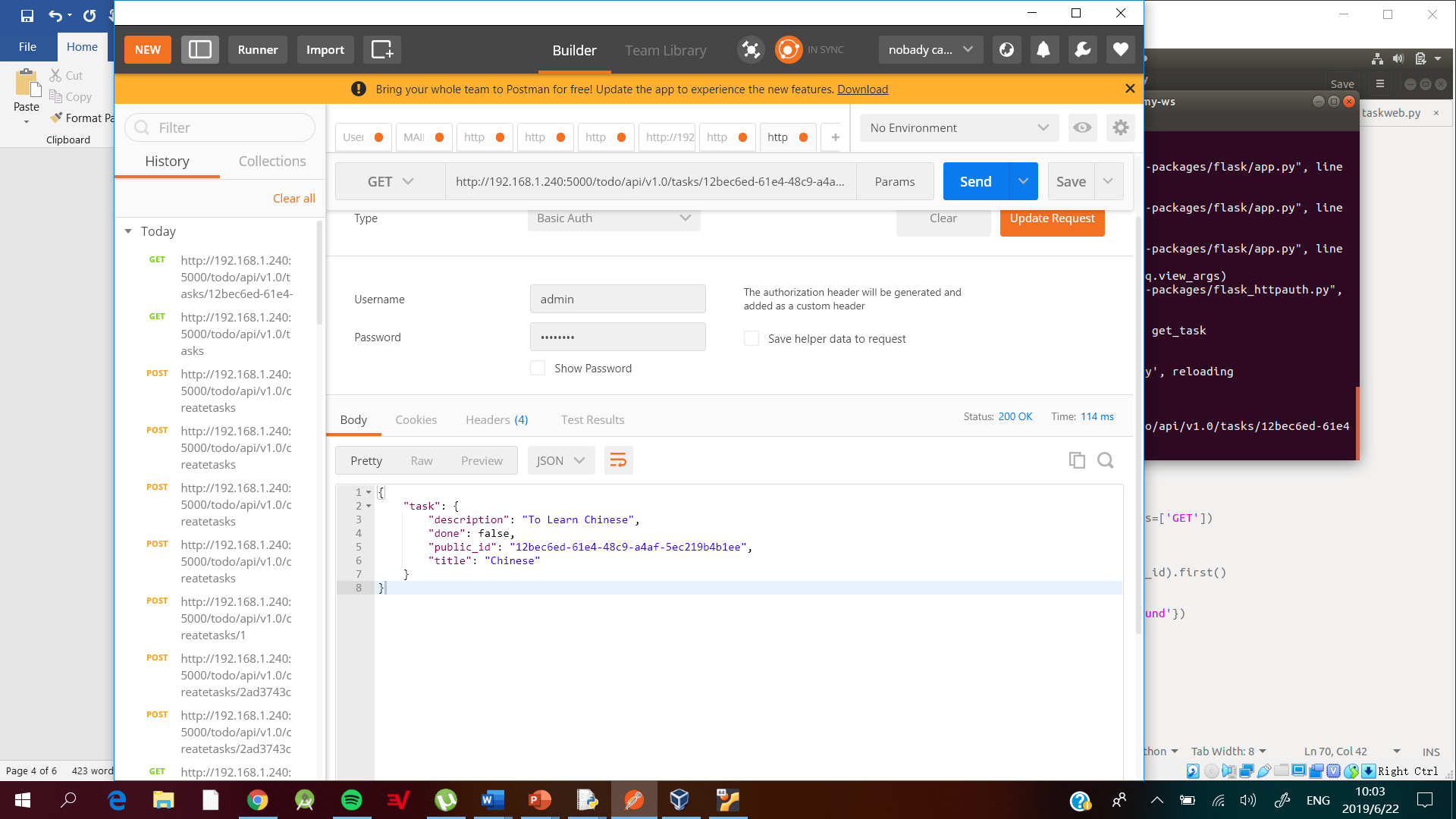
Create tasks



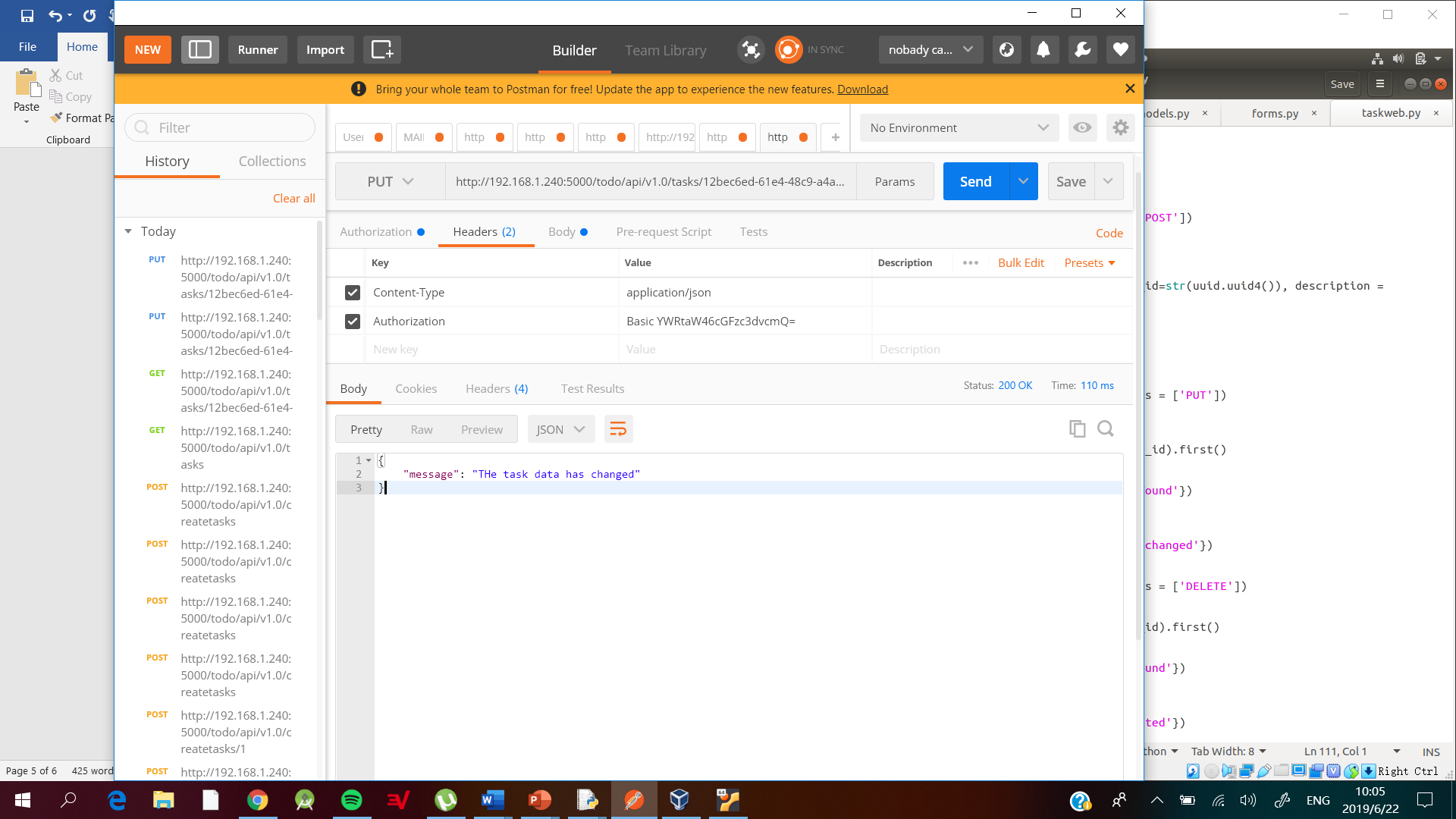
Get all Tasks



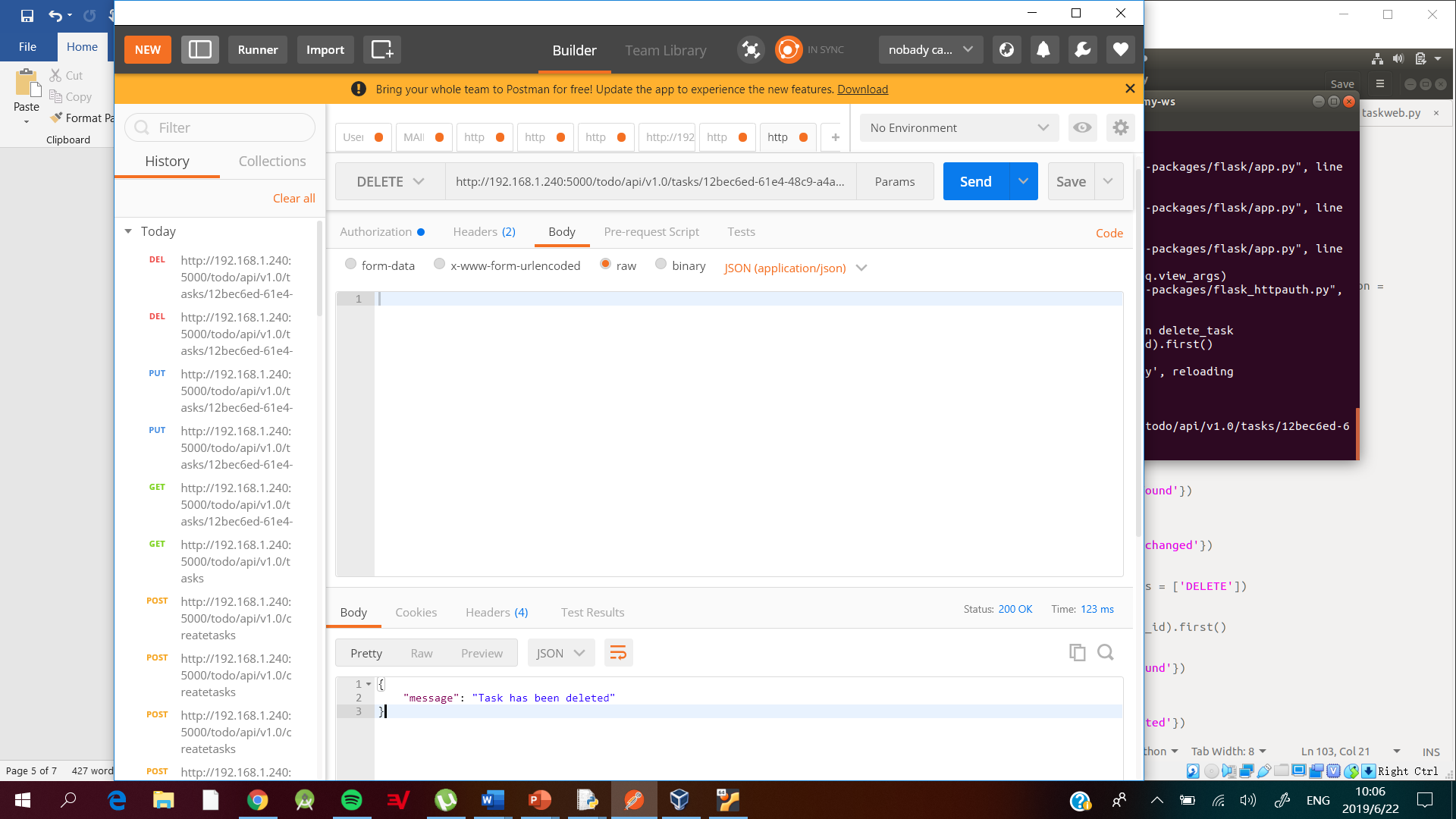
Get task



Update Tasks

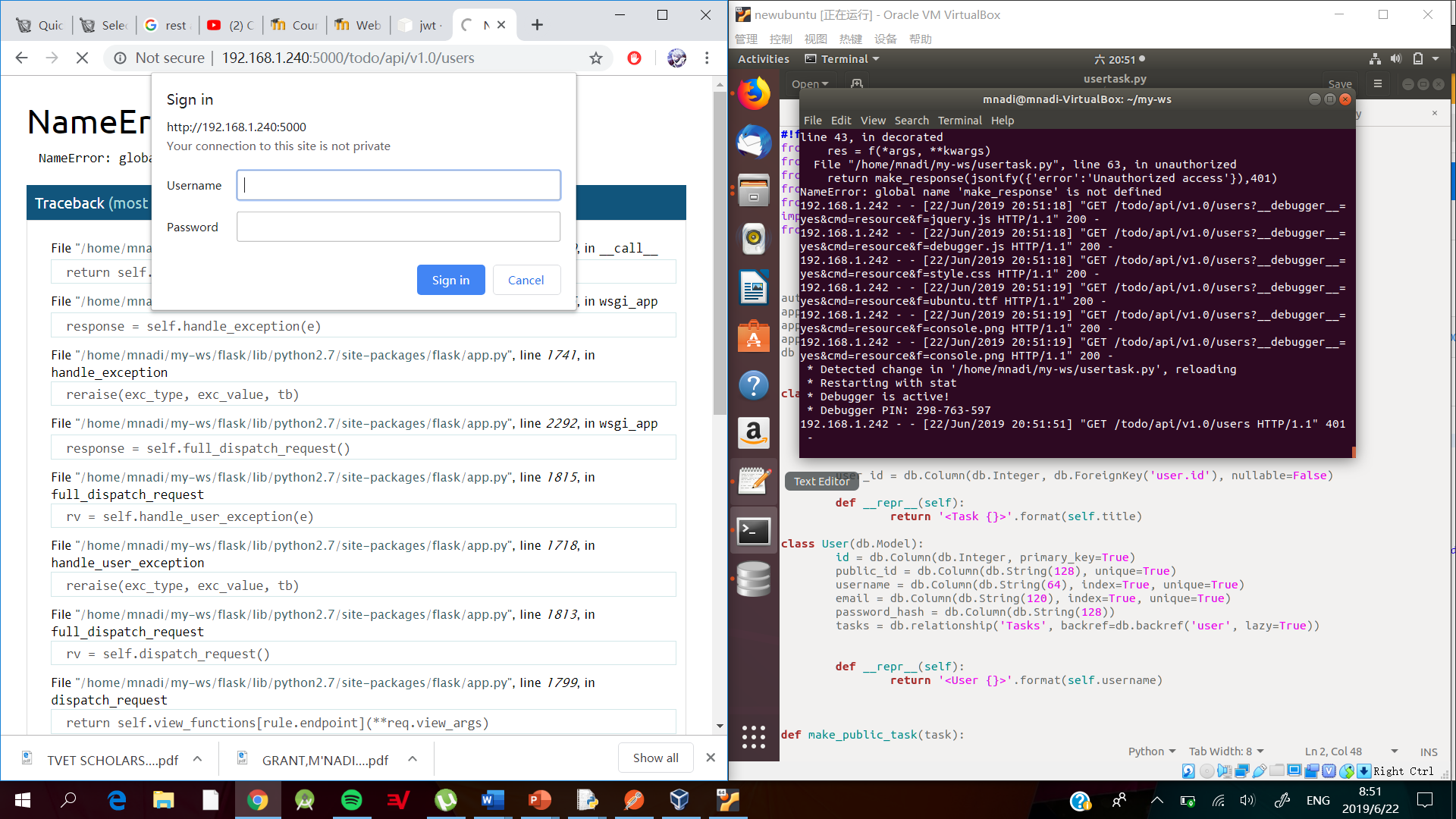


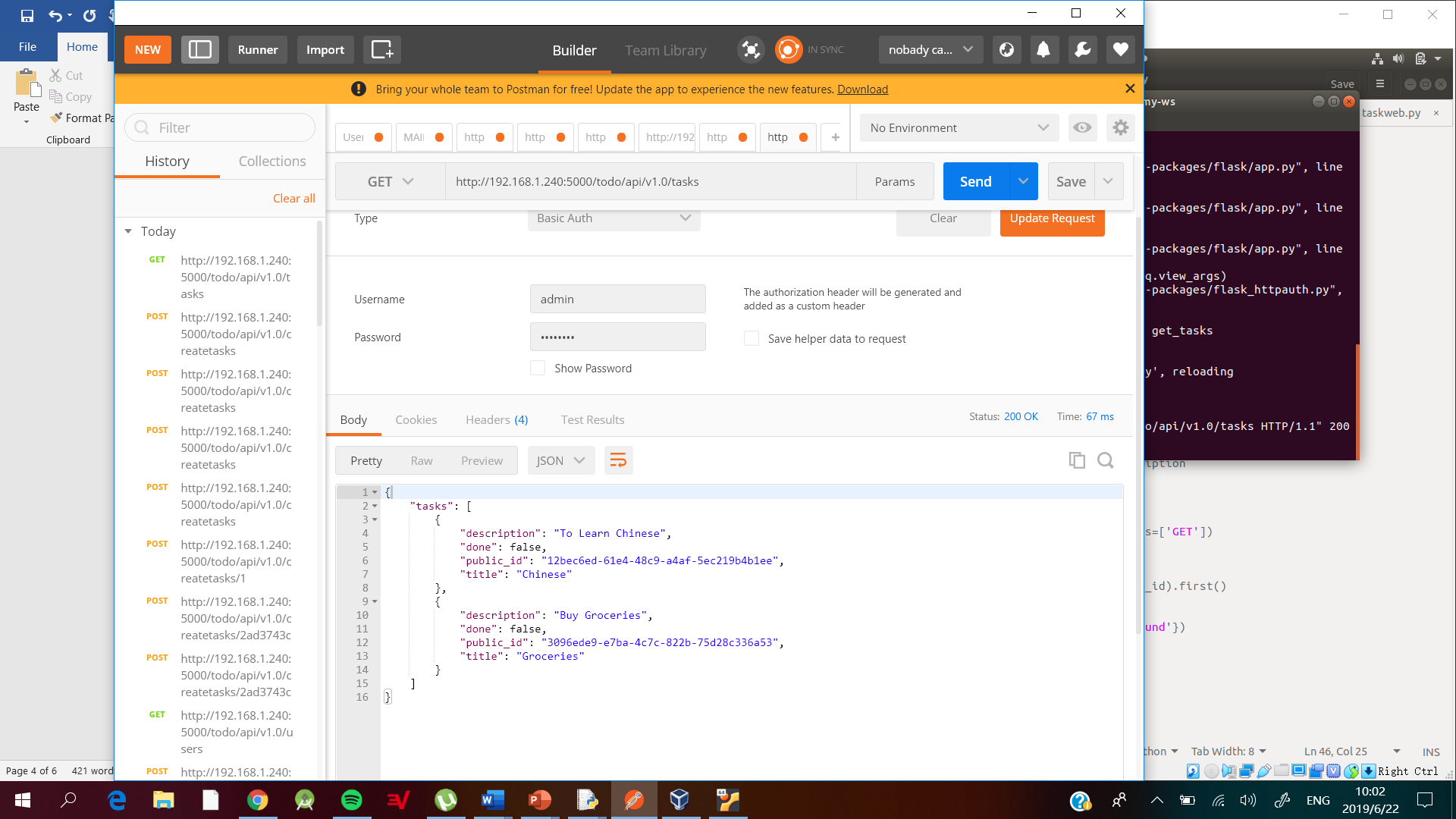
Delete Task



* Refactor your previous “Tasks” web service with UsernamePassword-Tasks scheme

The “Tasks” web service has been updated to done work for specific users.





* Try different DB backends, e.g. MySQL