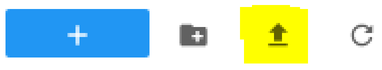
Guided Programming: Creating a Producer

44-671: Streaming Data

Note: In my experience the Docker application remembers my files between starts and stops, but I cannot guarantee that behavior. If you do not intend on finishing this assignment in a single sitting I recommend downloading your producer notebook (by right clicking on it) just in case it does not persist when you stop your Docker application.

1. Start your Docker application you set up in the Software Installation assignment
2. Open a web browser and open JupyterLab (`http://localhost:8888/lab`)
3. Using the "upload" button (highlighted in yellow in the image below), upload the following files:



   - `d3.v4.min.js`
   - `d3consumer.ipynb`

4. Open `d3consumer.ipynb` and change the name of the RabbitMQ instance to what your instance is called.
5. Create a new Python Notebook named `yourname-producer.ipynb` (replace `yourname` with your name; you can right click on the file in the left pane to rename the file)
6. Create two new cells in your notebook, and write the following code (replace the RabbitMQ instance with your instance name). Use the new cell button to create the cells; don't run the cells just yet:

```python
import pika
from random import choice
import pickle
from time import sleep

# change 'streaming-data_rabbitmq_1' to the name of your rabbitMQ Docker instance
connection = pika.BlockingConnection(pika.ConnectionParameters('streaming-data_rabbitmq_1'))
channel = connection.channel()
channel.queue_declare(queue='gasket')

points = [[0.,0.], [1.,0.], [.5, .866]]

def nextpt(pt):
    vr = choice(points)
    return (pt[0] + vr[0])/2, (pt[1]+vr[1])/2

pt = [.1,.1]

while(True):
    channel.basic_publish(exchange='', routing_key='gasket', body=pickle.dumps((pt[0], pt[1])))
    pt = nextpt(pt)
    sleep(.05)
```

```python
connection.close()
```

7. Start the consumer; in `d3consumer.ipynb` select the second code cell (that starts with the comment `#Run this cell`) and run it (either with the run button or Ctrl + Enter). Make sure you see a `[*]` next to the cell; no output should appear yet. If you encounter errors you probably need to fix the RabbitMQ instance name; if there are other errors contact your instructor for help debugging.

8. Once your consumer is running, go back to your other notebook and start your first code cell; there should be no errors; if there are, fix them as above.
9. Go back to the `d3consumer.ipynb` notebook; You should notice the plot above the consumer code changing.
10. Let your code run for a while; you should see a shape start to appear. Once the shape starts solidify, take a screenshot of the shape.
11. Stop the code by doing the following steps in order:
    (a) Open your producer notebook and hit the "stop" button (the square) that is above the notebook (you should see an "error" that says `Keyboard Interrupt`).
    (b) Run the second code cell (that contains `connection.close()`)
    (c) Open the `d3consumer.ipynb` notebook and hit the stop button.
    (d) Run the code cell that contains `connection.close()`
12. Add a new cell to the bottom of your producer notebook; in the dropdown box above the notebook that says "Code", select "Markdown", and answer the following questions (if you want to format the questions nicely, you can find a good Markdown tutorial at `https://www.markdowntutorial.com/` and a reference at `https://guides.github.com/features/mastering-markdown/`):
    - Your producer was generating (x, y) coordinates that the consumer was plotting; what shape was being create? Hint: you may want to search for the term "chaos game"
    - The first code cell in the producer notebook is what generates the points and adds them to the RabbitMQ; what lines of code would you need to modify to access a live source of data (such as access logs or social media) instead of modifying and generating points?
    - Based on your reading and any understanding of the code you may have, if you wrote another producer that wrote to a different queue, would our consumer take that data?
    - What do you think would happen if we started another producer writing to the same queue?
13. "Run" your Markdown cell (so it formats nicely)

When you are finished, save your notebook and export it as a PDF. Submit both your PDF and screen shot of the shape that was created to the assignment on the course website. You can stop your Docker application while you are not using it to save resources on your computer.