Carlebach, Mark
September 4, 2019
CSCIE-88, 2019 Fall

Homework 1: AWS and Docker Setup

This document is a template for your solutions submission. You are free to add additional information in this submission if you would like. Extra screenshots and extra documentation is perfectly fine. Screenshots must always be viewable. If a screenshot is too blurry to be viewed or is chopped off in a key area you will not receive full credit for it.

Please identify which problems were completed. If any were incomplete, please identify where you encountered problems.

for example:
Problem 1: 100% complete
Problem 2: 100% complete
Problem 3: 100% complete
Problem 4: 100% complete
Problem 5: 100% complete

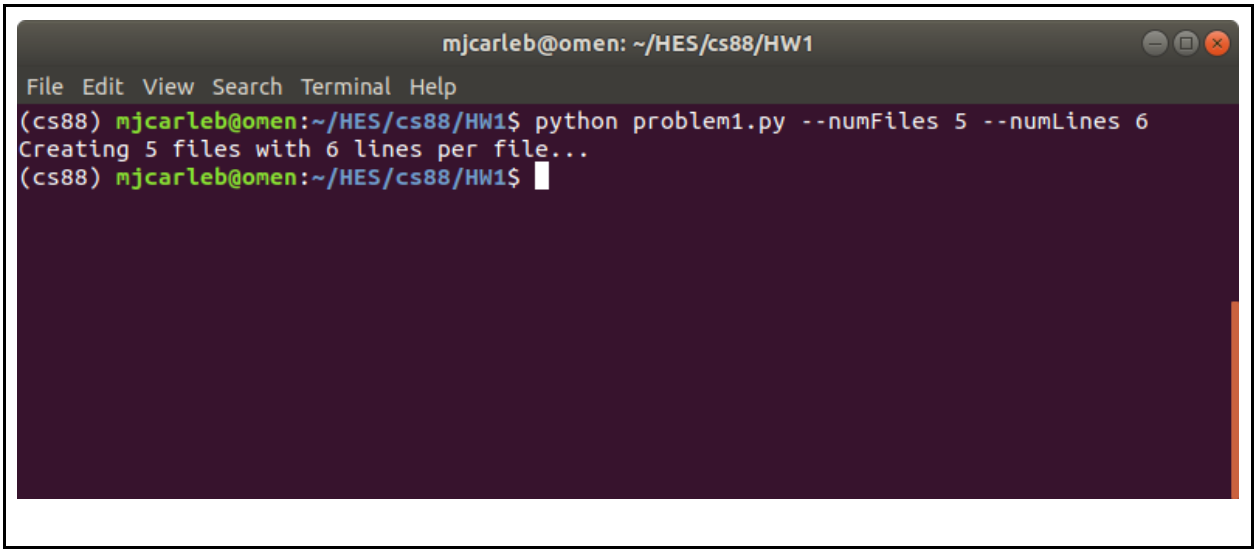
Problem 1: [25 points] File generator program

Paste your source code into the following area. All code should be heavily commented, and easily readable. [15 points]

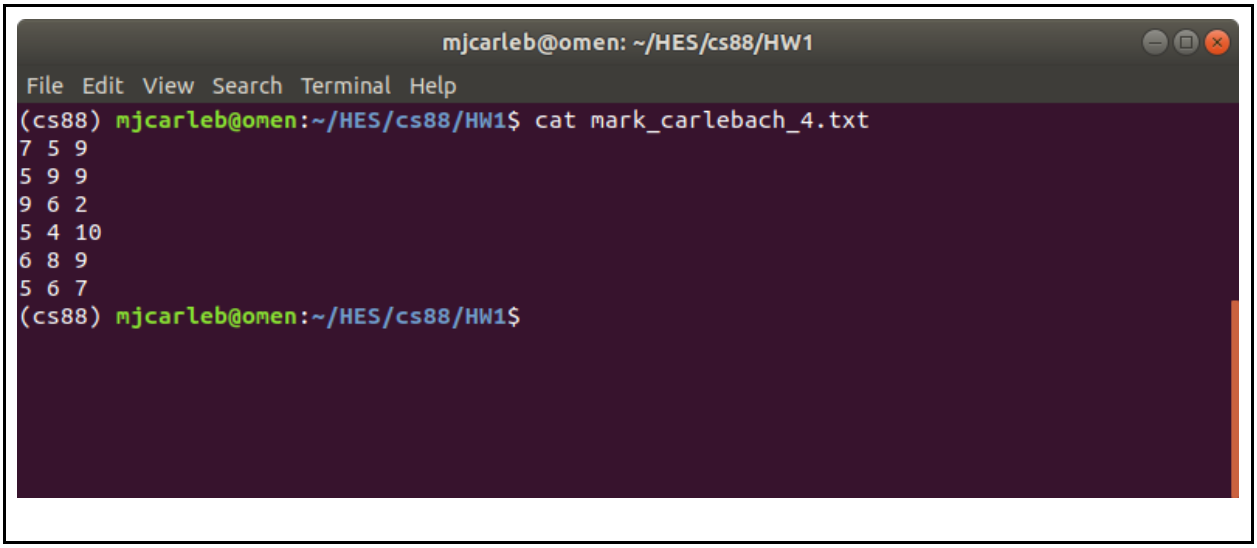
```
"""
I watched Youtube video by Derek Banas "Learn to Program 14:  Python Threads"
to learn how to use threads in Python for this assignment.
"""

# Import needed libraries
import random
import threading
import argparse

# Set random seed for repeatability
random.seed(88)

# input arguments
# Example of running:  python problem1.py --numLines 3 --numFiles 4
parser = argparse.ArgumentParser()
parser.add_argument("--numFiles", help="Number of files to create")
parser.add_argument("--numLines", help="Number of lines in each file")
args = parser.parse_args()
numFiles = int(args.numFiles)
numLines = int(args.numLines)
```

```
def exec_thread(thread_num, numLines):
    """On a thread, create file with 3 random ints on each of numLines
    :thread_num (int):  the thread number
    :numLines (int):  the number of lines to write in file
    :returns None, side effects only
    """

    # Create list numLines long with each item being a 3 random ints
    lines = [f"{random.randint(0, 10)} "+\
            f"{random.randint(0, 10)} "+\
            f"{random.randint(0, 10)}\n" for i in range(numLines)]

    # Open a file with name that references thread_num
    with open(f"mark_carlebach_{thread_num}.txt", mode="w") as f:
            # Write lines to file
            f.writelines(lines)

# Echo system arguments
print(f"Creating {numFiles} files with {numLines} lines per file...")

# For each file/thread...
for i in range(numFiles):

    # Create thread to execute function that creates file
    thread = threading.Thread(target=exec_thread,
                              args=(i, numLines))

    # Start the thread
    thread.start()
```

Paste an example of your code output into the following area. This can be a screenshot (ideally), or a copy/paste of console text. [5 points]

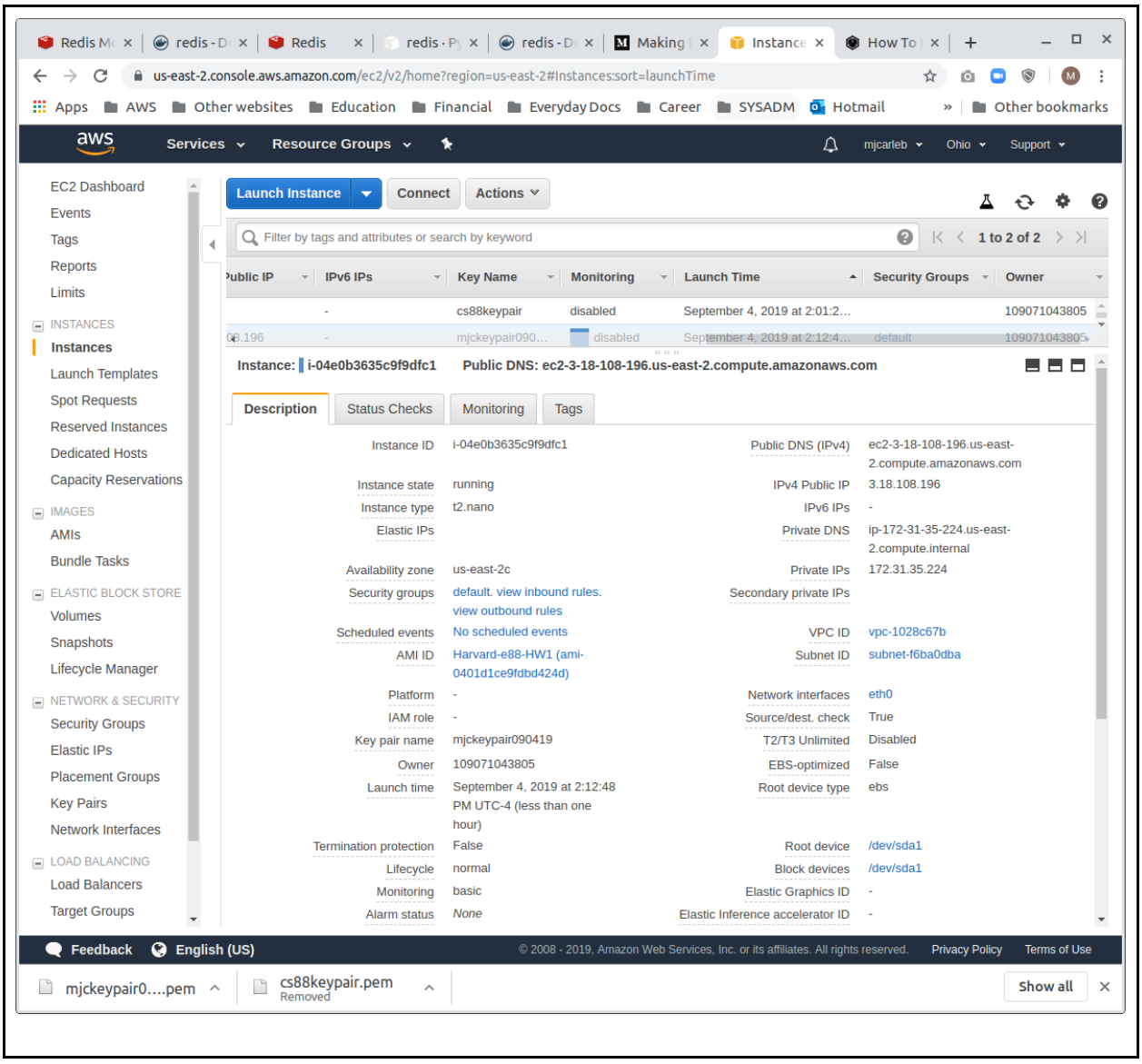


Paste an example of the contents of one of your generated files in the following area. [5 points]



Problem 2: [25 points] Set up a machine and demonstrate that it works

Paste a screenshot of your machine, include your owner information and creation date in your screenshot. [15 points]



Describe how you connected to your machine:

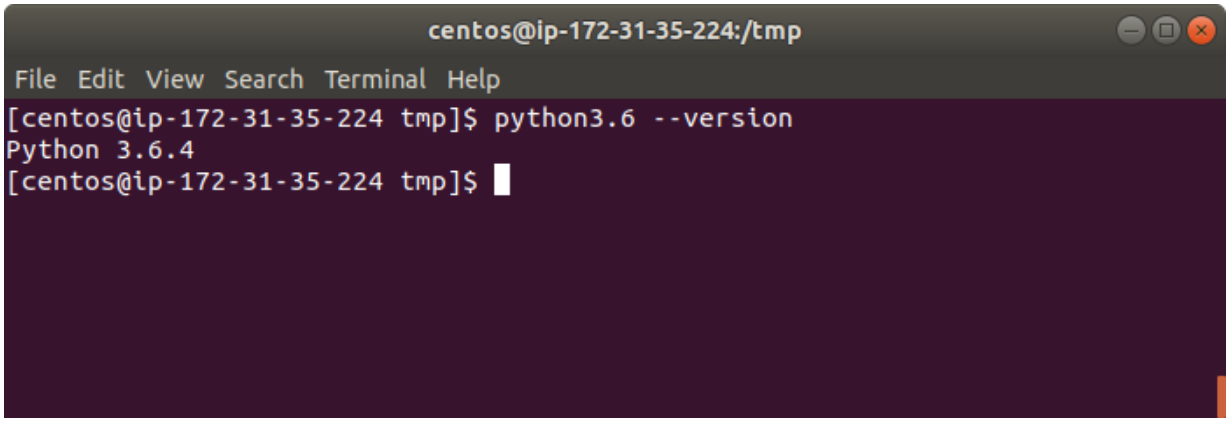
$ ssh -i "~/mjckeypair090419.pem" centos@ec2-3-18-108-196.us-east-2.compute.amazonaws.com

Show which Java and/or Python version is installed on your machine:

I saw the machine had Python 2.7 installed. So, I upgraded to 3.6 using the following commands (from this reference: https://www.rosehosting.com/blog/how-to-install-python-3-6-4-on-centos-7/):

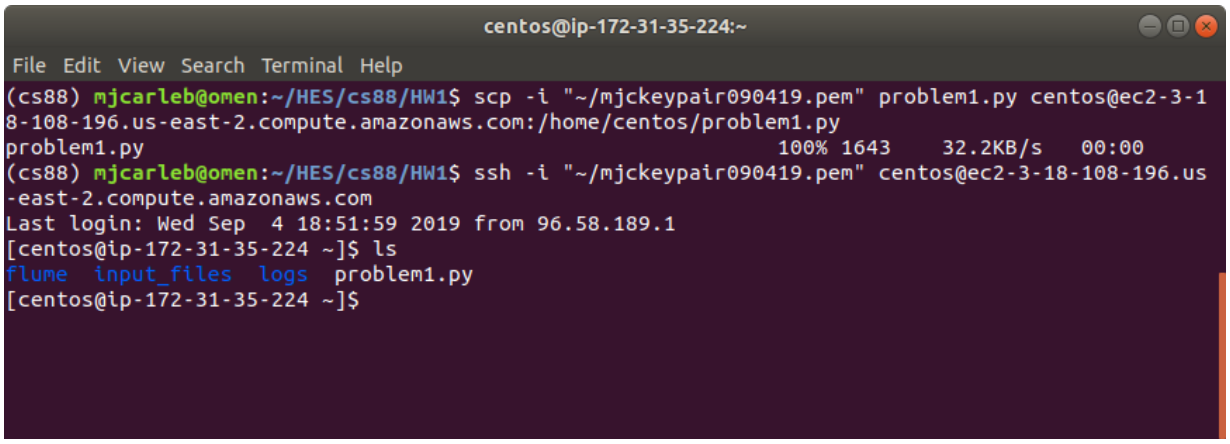
sudo yum install -y https://centos7.iuscommunity.org/ius-release.rpm
sudo yum update
sudo yum install -y python36u python36u-libs python36u-devel python36u-pip

I now, I have Python 3.6 as shown in the screenshot below:

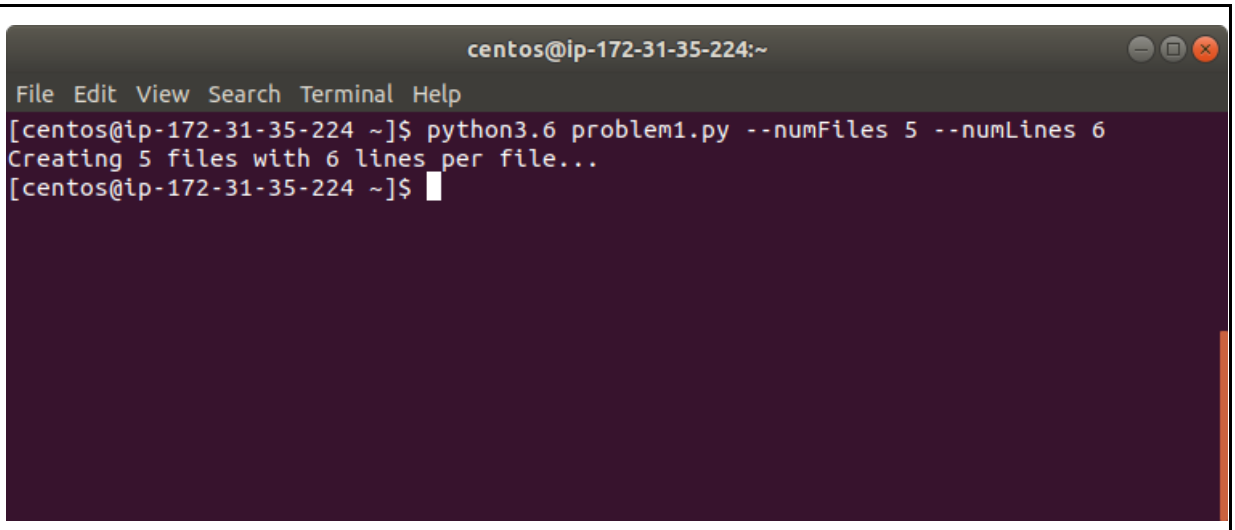


Paste a screenshot of the command you used to transfer your program to your machine [5 points]

scp -i "~/mjckeypair090419.pem" problem1.py centos@ec2-3-18-108-196.us-east-2.compute.amazonaws.com:/home/centos/problem1.py



Paste a screenshot of your program execution from within your machine. [5 points]



Problem 3: [25 points] Run Redis server and clients as Docker containers and demonstrate that they work

Show all the commands you used, in sequence, to start your Redis server and clients [15 points]

docker network create my_network

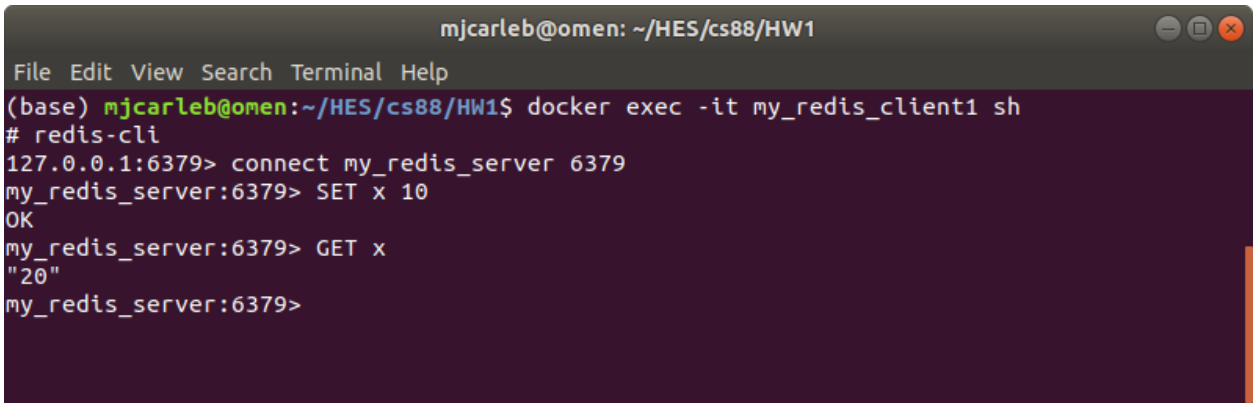
docker run --name my_redis_server -d --network my_network redis redis-server --appendonly yes

docker run --name my_redis_client1 -itd --network my_network redis

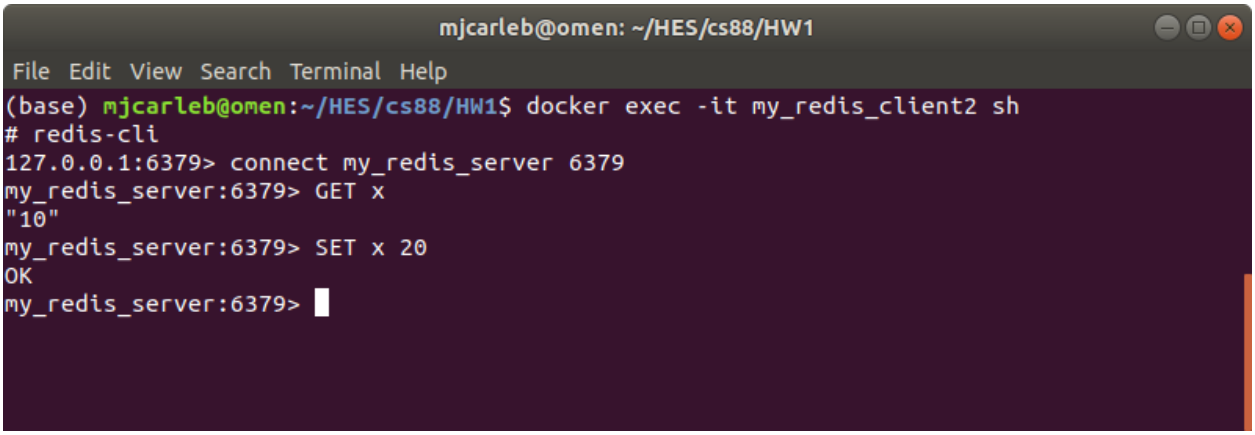
docker run --name my_redis_client2 -itd --network my_network redis

Show the value of 'x' in the clients, as described in problem 3 [10 points]

Working inside my_redis_client1:



Working inside my_redis_client2:



Problem 4: [25 points] Run Postgres DB as Docker container and demonstrate that it works

Show all the commands you used, in sequence, to start your Postgres server. [10 points]

```
docker network create my_network

docker run --network my_network --name my_postgres_server -e
POSTGRES_PASSWORD=first1 -d postgres
```

Show how you connect to the DB [5 points]

```
docker run -it --rm --network my_network postgres psql -h my_postgres_server -U postgres
```

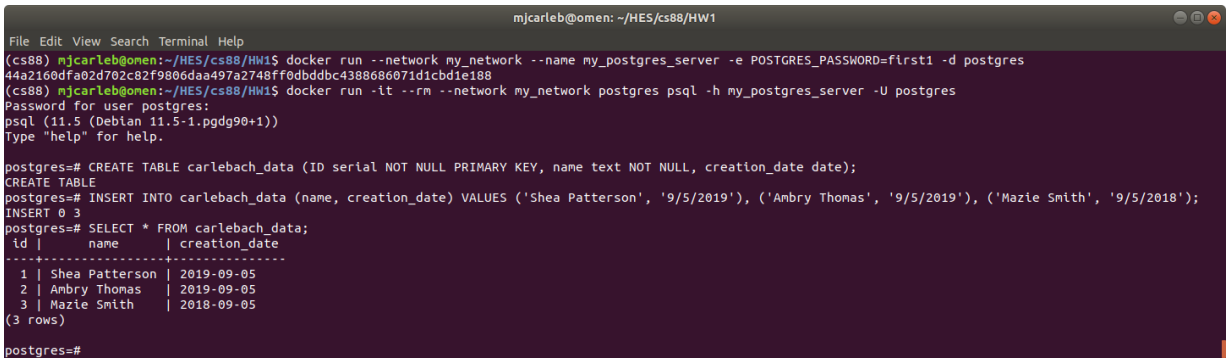
Show results of querying your database for all records. [10 points]

You did not ask, but here are psql commands to create datbase:

- CREATE TABLE carlebach_data (ID serial NOT NULL PRIMARY KEY, name text NOT NULL, creation_date date);
- INSERT INTO carlebach_data (name, creation_date) VALUES ('Shea Patterson', '9/5/2019'), ('Ambry Thomas', '9/5/2019'), ('Mazie Smith', '9/5/2018');
- SELECT * FROM carlebach_data;
- delete from carlebach_data *;

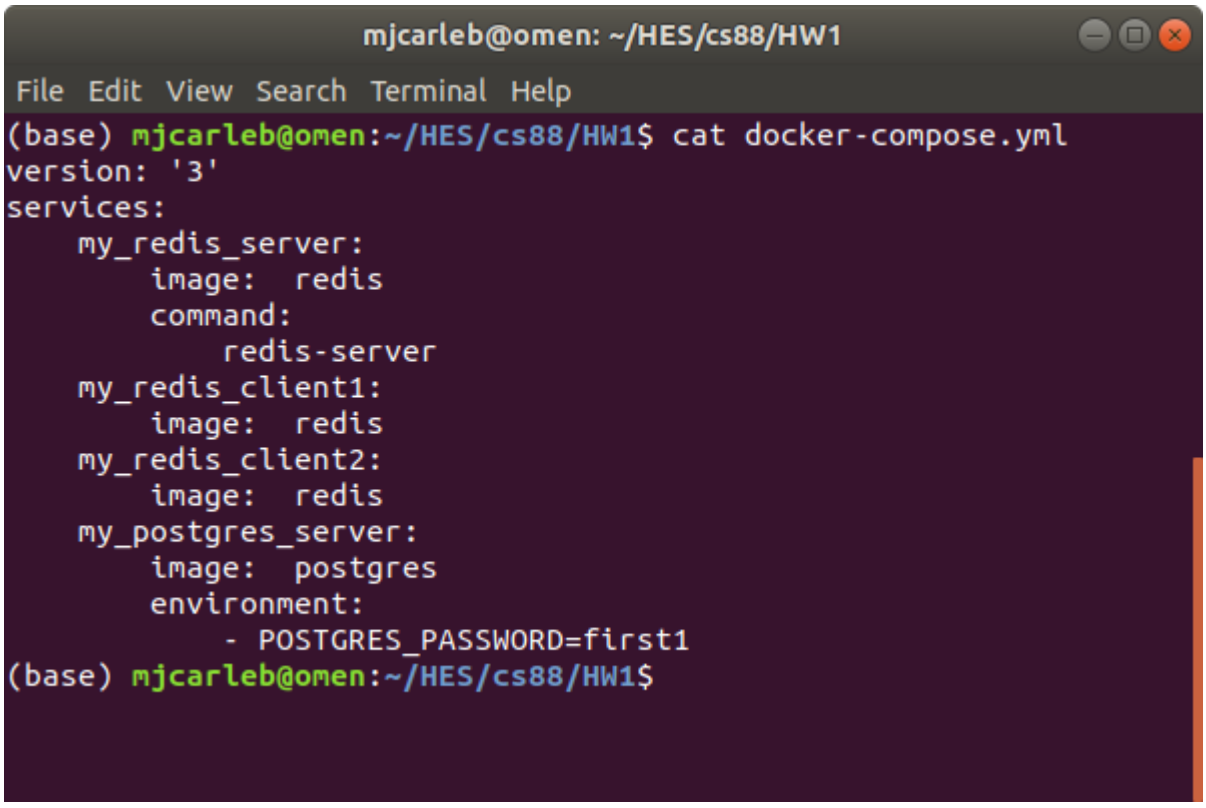
Here are query results:

```
 id |      name      | creation_date
----+----------------+---------------
  1 | Shea Patterson | 2019-09-05
  2 | Ambry Thomas   | 2019-09-05
  3 | Mazie Smith    | 2018-09-05
```



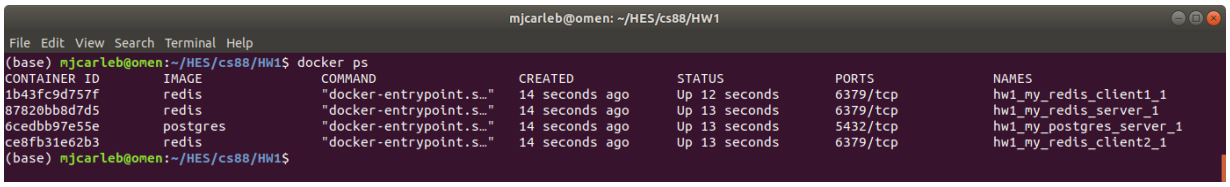
Problem 5: [Bonus, 15 points]: Start multiple Docker container via Compose

Show your Docker Compose configuration [7 points]

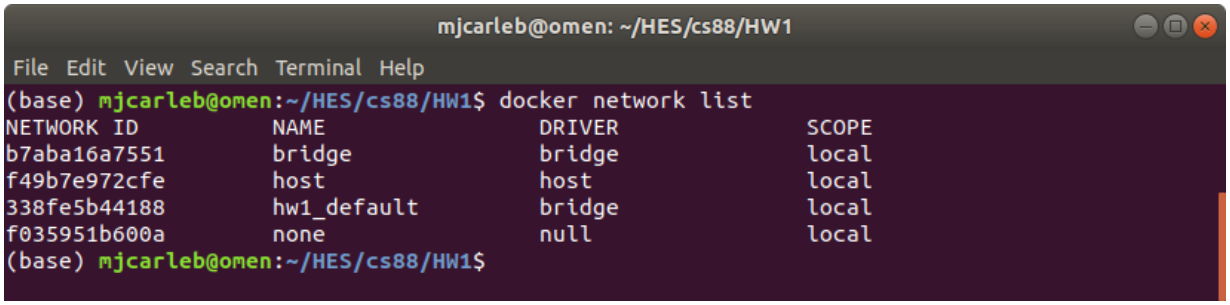


Show that the Redis server, 2 Redis clients, Postgres server are all functional [8 points]

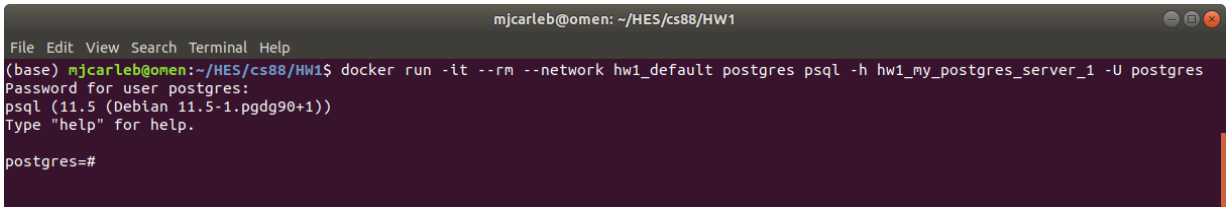
Show all servers running with $docker ps:



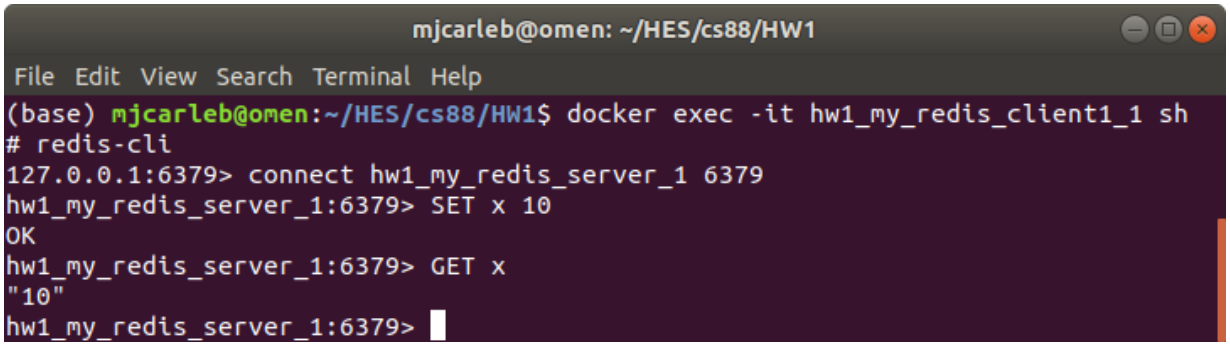
Get network name created by docker-compose with $docker network list:



Connect to postgres server with $docker run -it --rm --network hw1_default postgres psql -h hw1_my_postgres_server_1 -U postgres:



From redis_client1, update redis_server with $docker exec -it...:



From redis_client2, read value from redis_server (that was created by redis_client1):