**KINIK, Koray**

Sep 9, 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 Bonus: 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 wrote it as a simple cli app.  \_\_main\_\_.py:  *""" Entry point module, in case you use `python -mcsci\_e88\_hw0`.  setup.py points to here. The function 'main' below runs the app.  The cli app does what question 1 wants."""* import argparse  from csci\_e88\_hw0.cli import main  if \_\_name\_\_ == "\_\_main\_\_":  """  Enter the inputs: numFiles, numLines from the command line (int)."""   # parse the cli arguments:  parser = argparse.ArgumentParser(description='Flip a switch to pull full dataset.')  parser.add\_argument('--numFiles', required=True, type=int)  parser.add\_argument('--numLines', required=True, type=int)  args = parser.parse\_args()   # run the app:  main(args)  cli.py:  *""" Module that contains the command line app.  Koray Kinik  CSCI E-88  Homework 1  Sep 6, 2019 """* from luigi import build import subprocess  from .homework\_1 import GenerateFiles   def main(args=None):  *"""  Trigger the luigi task that generates multiple files.* ***:param*** *args: cli arguments: numFiles, numLines* ***:return****: None  """* # First, start the luigid as a daemon run, so that  # the central-scheduler can run parallel tasks:  with subprocess.Popen('pipenv run luigid --port 8082', shell=True,  stdout=subprocess.PIPE, stderr=subprocess.STDOUT):   # Next, trigger the luigi task that does the job.  # Note that number\_of\_files = number\_of\_workers.  build([  GenerateFiles(numFiles=args.numFiles, numLines=args.numLines)  ], workers=args.numFiles)  homework\_1/\_\_init\_\_.py:  *""" Homework 1, Question 1.  This is where things happen:*  *Create numLines txt files, each with numLines, each on different core.  For parallel processing, I used luigi's ‘central scheduler’.  It uses ‘multiprocessing’ under the hood.*  *"""* import numpy as np from luigi import WrapperTask, ExternalTask, IntParameter, LocalTarget   class SingleTxtTarget(ExternalTask):  *"""(luigi) Task that creates one single txt file."""* fileNo = IntParameter()  numLines = IntParameter()   def output(self):  *"""Create a local txt target (non-atomic)."""* return LocalTarget(f'data/koray\_kinik\_{self.fileNo}.txt')   def run(self):  *"""Create rand data and save as txt to the LocalTarget."""* with self.output().open('w') as f:  data = np.random.randint(10, size=(self.numLines, 3))  np.savetxt(f.name, data, fmt='%i')   class GenerateFiles(WrapperTask):  *"""Luigi task to generate files, given params."""* numFiles = IntParameter()  numLines = IntParameter()   def requires(self):  *"""Factory. Trigger one task for each integer in the list."""* return [  SingleTxtTarget(  fileNo=i, numLines=self.numLines  ) for i in range(self.numFiles)  ]  (  About luigi:  <https://luigi.readthedocs.io/en/stable/central_scheduler.html?highlight=scheduler#using-the-central-scheduler>  )  setup.py:  setup(  name='csci-e88-hw0',  …  …  entry\_points={  'console\_scripts': [  'csci-e88-hw0 = csci\_e88\_hw0.cli:main',  ]  },  … |
|  |

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]

|  |
| --- |
| Run the app:    The output:    The daemon: (monitoring the process) |

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]

|  |
| --- |
| Owner: 755240292069 |

Describe how you connected to your machine:

|  |
| --- |
| On the terminal:    On the IDE:  I also configured my PyCharm Pro to connect to the machine.  I followed these instructions to configure my PyCharm project interpreter.  <https://www.codementor.io/abhishake/pycharm-setup-for-aws-automatic-deployment-m7n8uu2n4>  First, I saved my \*.pem file to my local project folder.  Then, copied the access info to my instance on AWS page:    I created security group (for pycharm access):    Next, on PyCharm: Preferences/Project Interpreter:        Next:      Comments so far:  I see that python 2.7 is installed on root. That is ok, (and needed).  I will install python 3.7 in a container anyways.  Next, installed Python 3.7.4  Step 2.a: Installed HomeBrew (<https://docs.brew.sh/Homebrew-on-Linux>) and added to PATH.    Step 2.b: Installed Pipenv.  Pipenv is a package manager and installs Python3 in a venv.  <https://docs.pipenv.org/en/latest/install/#installing-pipenv>    Next, I point my python executable to python 3.7 in the project:    Now I am inside a pipenv (which uses venv under the hood).  And I have python 3.7 installed in it.  Next, ran my App on my EC2 instance:    …  Program ran successfully.    And here are the list of files created on remote: |

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

|  |
| --- |
| (Note that I run inside the pipenv shell. Python 3.7.4 is installed inside the venv.) |

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

|  |
| --- |
| From the terminal:    Also, on the IDE:  How PyCharm does it automatically:  I click the ‘sync’ button (top right on the Remote Host menu) to refresh the files from local to my instance. |

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]

|  |
| --- |
| Pull the image and start redis server:    Confirmed it is up and running:    Next, I run redis-cli inside the container I just built (“redis1”). I can ping from local to the container through port 6379.    I create another container and confirm redis1 container is linked to “redis”.    Next, I go into the shell and launch the redis **client1**. It connects to my redis server, “redis1”.    Next, I launch a second container with **client2**: |

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

|  |
| --- |
| (Note: I did not use “--rm” when I created my client containers, so I have to remove them manually once I exit from the shell(s).) |

**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]

|  |
| --- |
| Pulled the image:    Started my new container using the postgres image. I name it “postgres1”. I used -v to store the database I am about to create.    Next, running inside the “postgres1” container, I created a DB named “mydb”. |

Show how you connect to the DB [5 points]

|  |
| --- |
| Inside this container, there is a built in CLI tool (*psql*).  I can use it to use to connect to “mydb”:    Next, I create a table and then enter some mock records. |

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

|  |
| --- |
| (I want to also verify the table from another linked container.)    Finally, deleted all records from my table.    Also, on DBeaver: |

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

Show your Docker Compose configuration [7 points]

|  |
| --- |
| **…** |

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

|  |
| --- |
| Next, on a new terminal window,    Next, I connect to the redis server container (already running):      Postgres        Thanks! |