**Choi, Mason**

September 1, 2022

CSCIE-88, 2022 Fall

**Assignment 1: 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.

NOTE: I primarily work using a monitor, meaning that screenshots may appear zoomed out with small contents. Please feel free to zoom in, the quality should hold and make every part of the image clear. Thank you!!

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

|  |
| --- |
| Problem 1: 100% complete  Problem 2: 100% complete  Problem 3: 100% complete  Problem 3B: Bonus: 100% complete  Problem 4: 100% complete |

**Problem 1: [25 points] Local dev setup**

Paste a screenshot of your dev environment (IDE of choice). Specify which IDE you are using, as well as whether you are using it on your local machine or on a AWS EC2 instance. [6 points]

|  |
| --- |
| Visual Studio Code on local machine but may use Vim in some cases for practice/comfort and Pycharm for more Python specific control. |

|  |
| --- |
| Graphical user interface  Description automatically generated |

Paste your source code into the following area. If you used the provided programs without changes, just mention that (No need to paste the code). If not, paste the new source code and indicate the changes you made. Remember that all code should be heavily commented, and easily readable. [5 points]

|  |
| --- |
| I used the provided source code, only changing arguments num\_file and num\_lines to 3 and 10 respectively.  # Copyright (c) 2020 CSCIE88 Marina Popova  # this is a very simple multi-processing application and it can be improved in many ways ...  # it takes two input arguments: num\_files and num\_lines , and creates the specified number of files  # each files has num\_lines lines with 3 random numbers per line  import argparse  import multiprocessing  import random  # default number of files and lines  NUM\_OF\_FILES = 3 # CHANGE  NUM\_OF\_LINES = 10 # CHANGE  # Function to generate a file with the specified number of lines  def generate\_file(num\_lines, file\_number):  '''  Function to generate a file with the specified number of lines, each with 3 random numbers  '''  filename = "cscie88\_fall2022\_" + str(file\_number) + ".txt"  # Open writer and output lines  file = open(filename, "w")  for i in range(num\_lines):  line = str(random.randint(0, 10)) + " " + str(random.randint(0, 10)) + " " + str(random.randint(0, 10))  file.write(line + "\n")  file.close()  print(filename + " written!")  def parse\_arguments():  '''  Argument parser  num\_files = Number of files  num\_lines = Number of lines  '''  parser = argparse.ArgumentParser(description='Set the number of files and number of lines')  parser.add\_argument("num\_files", nargs='?', type=int, help="Number of files to create", default=NUM\_OF\_FILES)  parser.add\_argument("num\_lines", nargs='?', type=int, help="Number of lines per file", default=NUM\_OF\_LINES)  args = parser.parse\_args()  return (args)  def main():  '''  Get arguments, setup multiprocessing, create files  '''  arguments = parse\_arguments()  num\_files = arguments.num\_files  num\_lines = arguments.num\_lines  print("Program arguments: num\_files = " + str(num\_files) + "; num\_lines = " + str(num\_lines))  jobs = []  for file\_number in range(num\_files):  t = multiprocessing.Process(target=generate\_file, args=(num\_lines, file\_number))  jobs.append(t)  t.start() # new child process is started at this point, it has its own execution flow  for curr\_job in jobs: # wait for all processes to finish  curr\_job.join()  print("Program completed OK")  if \_\_name\_\_ == "\_\_main\_\_":  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. [7 points]

|  |
| --- |
| Text  Description automatically generated |

Paste an example of the contents of one of your generated files in the following area. [7 points]

|  |
| --- |
| cscie88\_fall2022\_2.txt CONTENTS:  6 3 1  4 4 3  1 9 4  5 6 6  2 9 8  7 7 1  4 2 7  5 1 9  10 0 2  5 6 1 |

**Problem 2: [30 points] Set up a machine and demonstrate that it works**

Paste a screenshot of your AWS machine, include your owner information and creation date in your screenshot. [10 points]

|  |
| --- |
| Graphical user interface  Description automatically generated |

Describe how you connected to your machine. [5 points]

|  |
| --- |
| I used ssh to connect in a new local terminal. |

Show which Java and/or Python version is installed on your machine. [5 points]

|  |
| --- |
| Text  Description automatically generated |

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

|  |
| --- |
| Graphical user interface  Description automatically generated with medium confidence |

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

|  |
| --- |
| Text  Description automatically generated |

**Problem 3: [30 points] Apache Web server in a local Docker container**

Paste a screenshot of your simple html page with an image[10 points]

|  |
| --- |
| Graphical user interface, application  Description automatically generated |

Show all command(s) used, in sequence, to launch the Apache Web server image on your Docker and corresponding console output [10 points]

|  |
| --- |
| PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> nvim index.html  PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> docker build -t assignment1\_container .  [+] Building 1.1s (7/7) FINISHED  => [internal] load build definition from Dockerfile 0.0s  => => transferring dockerfile: 31B 0.0s  => [internal] load .dockerignore 0.0s  => => transferring context: 2B 0.0s  => [internal] load metadata for docker.io/library/httpd:2.4 0.9s  => [internal] load build context 0.0s  => => transferring context: 247B 0.0s  => CACHED [1/2] FROM docker.io/library/httpd:2.4@sha256:70999c4a17c796dd28f86f9c847b30f28abaed6ef1fd72a44282b1c9 0.0s  => [2/2] COPY index.html /usr/local/apache2/htdocs/ 0.0s  => exporting to image 0.1s  => => exporting layers 0.0s  => => writing image sha256:7f4c704d3a85a98398b473a11445ad25bae6f14bb2de06cdad39c03b67c2f579 0.0s  => => naming to docker.io/library/assignment1\_container 0.0s  Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them  PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> docker run -dp 8080:80 assignment1\_container  4c105039785597dcdfb9050168716be8a44f811c8aaa0f97769831c8ac0017a1  PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> |

show URL and a screenshot of accessing your new page [10 points]

|  |
| --- |
|  |

**Problem 3B [Bonus, 10 points]: Docker on AWS**

Show all command(s) used, in sequence, to create and running the Docker container on your EC2 instance[5 points]

|  |
| --- |
| PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> scp -i ../CSCIE88.pem index.html ubuntu@3.16.157.18:/home/ubuntu  index.html 100% 208 4.7KB/s 00:00  PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> scp -i ../CSCIE88.pem Dockerfile ubuntu@3.16.157.18:/home/ubuntu  Dockerfile  PS C:\Users\Mason Choi\Projects\CSCIE88\assignment1> ssh -i ../CSCIE88.pem ubuntu@3.16.157.18  ubuntu@ip-172-31-46-0:~$ docker build -t assignment1 .  Sending build context to Docker daemon 129.1MB  Step 1/2 : FROM httpd:2.4  2.4: Pulling from library/httpd  7a6db449b51b: Pull complete  b4effd428409: Pull complete  6b29c2b62286: Pull complete  c2123effa3fc: Pull complete  152876b0d24a: Pull complete  Digest: sha256:70999c4a17c796dd28f86f9c847b30f28abaed6ef1fd72a44282b1c941238804  Status: Downloaded newer image for httpd:2.4  ---> a981c8992512  Step 2/2 : COPY index.html /usr/local/apache2/htdocs/  ---> 68ea6103365b  Successfully built 68ea6103365b  Successfully tagged assignment1:latest  ubuntu@ip-172-31-46-0:~$ docker run -dp 8080:80 assignment1  3048167feca4fc9ea0e913dab5e837aa51c2950d05dd8617807753bc0538ee47  ubuntu@ip-172-31-46-0:~$ curl http://localhost:8080  <html><body><h1>It works!</h1></body></html>  ubuntu@ip-172-31-46-0:~$ curl http://localhost:8080/index.html  <!DOCTYPE html>  <html>  <body>  <img  src="https://codingart-book.com/images/malevich01.png"  style="display: block; margin-left: auto; margin-right: auto; padding: 40px 0"  />  </body>  </html>  ubuntu@ip-172-31-46-0:~$ |

show URL and a screenshot of accessing your new page [5 points]

|  |
| --- |
| Text  Description automatically generated with low confidence |

**Problem 4: [15 points]: Research**

Determine the BDP system type. Explain your decision [15 points]

|  |
| --- |
| Type 2  Mux is a clear example of a Type 2: Historical Pre-aggregated + Limited Real-Time + Pre-defined queries systems Big Data Process.  Historical Pre-aggregated/Pre-defined queries: At the heart of the process, Mux aggregates records by a pre-defined criteria every hour or so. This pre-aggregation limits the querying possibilities, eliminating any answers to ad-hoc questions, but increases efficiency when answering vital predetermined questions such as customer ID and watch time.  Limited Real-Time: Mux states “Our CDN partners use different mechanisms to deliver access logs. Some CDNs use [Syslog](https://en.wikipedia.org/wiki/Syslog) to continually push records, typically achieving less than one minute of delay from the time of the original request. However, other CDNs use HTTP push mechanisms where they’ll issue an HTTP POST and can take several hours to fully deliver all access logs,” in its explanatory webpage. It explains that they value real-time data, usually achieving extremely low latency. Of course, this real-time access is limited, as noted by Mux’s use of “HTTP push mechanisms” that can take several hours. Mux makes use of as much real-time data as they can, but due to different process requirements and “late records”, it is limited. |