**ECC - Assignment 3**

by

([sdiware@iu.edu](mailto:sdiware@iu.edu))

**Docker Installation:**

1. I have used Jetstream because it has preinstalled docker in root user.

A black background with white text

Description automatically generated

Note: Since the docker was installed on root user to access the volumes I had to use sudo commands.

**Directory Structure:**

Below is the directory which I created to perform this assignment.

1. As seen in screenshot below, I have created a folder Assignment3, which consists of docker-compose.yml, serverFolder and clientFolder.
2. Inside of clientFolder I have created a dockerfile.client and client.py.
3. Inside of serverFolder I have created a dockerfile.server and server.py.

A computer screen shot of a computer program

Description automatically generated

**Files:**

1. Docker-compose.yml:

* Below is the screenshot of docker-compose.yml.
* It defines two services, "server" and "client," connected to a custom network named "mydockerNetwork."
* Each service utilizes named volumes ("servervol" and "clientvol").
* Volumes are used to persist data for the "server" and "client" services in the /serverdata and /clientdata
* The "client" service depends on the "server" service, ensuring proper service startup order.

A screen shot of a computer program

Description automatically generated

1. Dockerfile.server:

* This Dockerfile sets up a Python 3.9 environment, defines the working directory as /app, and specifies a volume at /serverdata.
* It copies a Flask-based Python script named server.py into the container's /app directory, installs the Flask library.
* It specifies the default command to run the Flask server on port 8080 when the container starts.

A screen shot of a computer

Description automatically generated

1. Server.py:

* It creates a Flask web application with a single route ('/') that generates a random file (randomFile.txt),
* Calculates its SHA-256 checksum and accepts the file with the checksum included in the response headers.
* The server listens on the specified port 8080.

A computer screen shot of a black screen

Description automatically generated

1. Dockerfile.client
   * This Dockerfile sets up a Python 3.9 environment for a client application, installs the requests library, copies a Python script (client.py) into the container's /app directory.
   * It defines a volume at /clientdata.

A screen shot of a computer

Description automatically generated

1. Client.py:

* It defines a client application that gets a file with checksum from a specified server URL (<http://server:8080/>).
* The file is saved at /clientdata/receivedFile.txt, and its checksum is verified using SHA-256.
* If the checksum verification fails, an error message is printed.

A computer screen shot of text

Description automatically generated

**Questions:**

**Build and Run your server and client container**.

* I have used the docker-compose.yml to define and configure multi-container.
* Using docker-compose build as below I have built and run my containers – assignment3-client\_1 and assignment3-server\_1.
* The docker-compose build command is used to build or rebuild the docker images defined in my docker compose file.
* When I run this command in the directory where my docker-compose.yml file is located, it reads the configuration and builds images for all the services defined in the file.

A screenshot of a computer program

Description automatically generated

* After that I executed the command **docker-compose up -d** to start the services mentioned in the docker compose file.
* I have used -d flag which is detached mode use to run services in the background.
* Below is the screenshot of docker-compose up -d and as we can see, network **mydockerNetwork** is created, with two volumes **assignment3\_servervol** and **assignment3\_clientvol** and two services **assignment3\_server\_1** and **assignment3\_client\_1**.

A screen shot of a computer screen

Description automatically generated

* Below is the screenshot of docker-compose ps command and we can see both containers are up and running.

A screen shot of a computer

Description automatically generated

**Communication between the two:**

* Using the client.py and server.py the communication is happening between the containers, and it can be seen using the file transfer that is happening within the network between the client and server containers.
* I have created a network named **mydockerNetwork** as seen in the below screenshot.

A computer screen with white text

Description automatically generated

* Both my containers **assignment3\_server\_1** and **assignment3\_client\_1** is running the same network mydockerNetwork which can be seen in the screenshot below.
* I have configured all of these in docker-compose.yml as mentioned in File section above.

A computer screen shot of white text

Description automatically generated

* I also created two volumes **assignment3\_clientvol** and **assignment3\_servervol** as shown below to store the container data.

A black screen with blue text

Description automatically generated

* Below is screenshot of volumes prior to creating the volumes for the assignment.

A screen shot of a computer

Description automatically generated

* Below is screenshot of volumes after creating the volumes for the assignment.

A screenshot of a computer

Description automatically generated

* After successfully running the containers, the randomFile.txt in the server.py is transferred from server to client and recievedFile.txt is received at the client side.
* I verified this by navigating to the volumes as shown below.

A screen shot of a computer

Description automatically generated

* Also, I verified the cksum too for both the files and it is the same as seen below. (I have done this for additional verification).

A screen shot of a computer

Description automatically generated

* Note: Since the docker was installed on root user to access the volumes I had to use sudo commands.
* Although I am sending the checksum and verifying it in client.py too as mentioned in files section.

**Client Container Shell:**

* Below is the screenshot of client container shell.
* I used docker-compose exec client sh to open the client container shell.
* I navigated to /clientdata to verify the received file.
* As seen below the file is received and checksum is also the correct one.

A black screen with a black background

Description automatically generated