# Artificial Intelligence and Cloud Computing Final Report

Date: 2023.08.17

Author: Guodong Chen

School: Nanjing Normal University

Email: [isguodong.chen@outlook.com](mailto:isguodong.chen@outlook.com)

[Github repository for this project](https://github.com/be-Frozen/Artificial-Intelligence-and-Cloud-Computing.git)

## Assignment #1:Create Cluster Manager

### 1. Introduction

The Cluster Manager (CM) is a Python program designed to create and manage clusters of containers. It simulates a simple cloud environment consisting of multiple containers, functioning as virtual machines (VMs).

#### 1.1 Purpose of the Cluster Manager

The primary purpose of the Cluster Manager is to provide a simplified way of creating and managing container clusters. It allows users to create a cluster of containers, execute commands within the cluster, monitor cluster status, and perform actions such as stopping the cluster and deleting all containers.

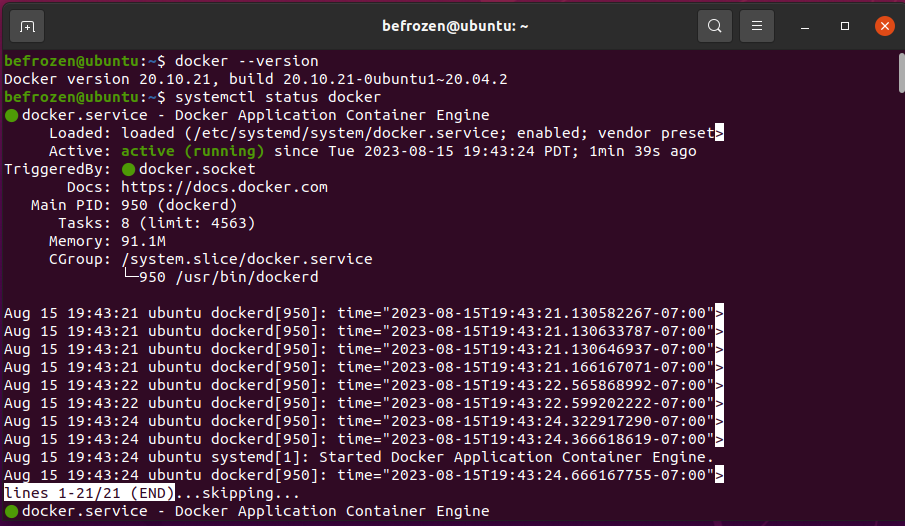
#### 1.2 Scope of Functionality

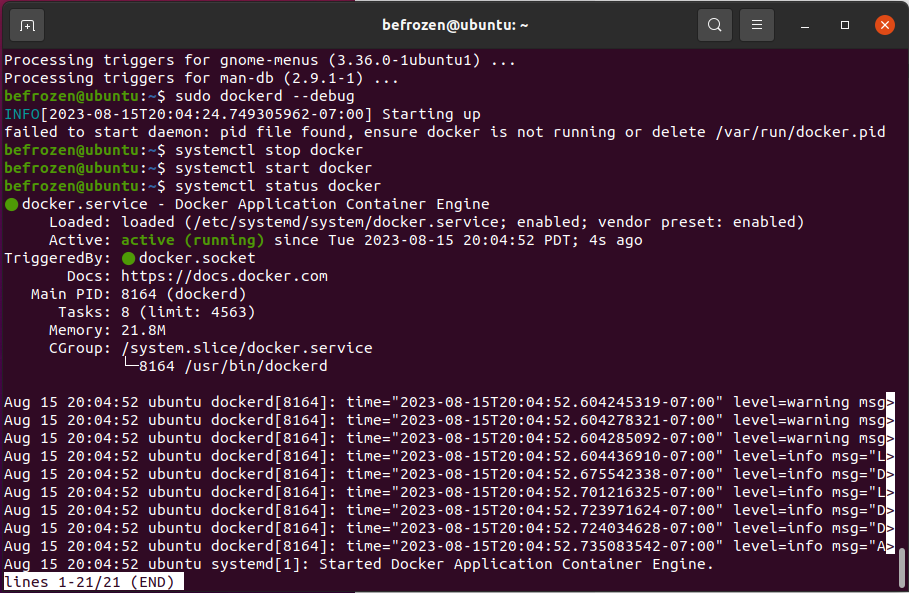
The CM supports the following key functionalities:

* **Create Cluster:** The CM can create a cluster of a specified number of containers. Each container represents a fake VM within the cluster.
* **List Cluster:** Users can view information about the containers within the cluster, including their IDs, status, and other relevant details.
* **Run Command:** Users can execute simple commands within the cluster. This enables interaction with the containers, simulating activities within a virtualized environment.
* **Stop Cluster:** The CM can halt the operation of the entire cluster, effectively stopping all containers.
* **Delete All Containers:** Users can delete all containers within the cluster, removing the entire simulated environment.

### 2. Usage Examples

To import the Docker SDK for Python and utilize its functionalities, users are required to ensure that the docker.service is up and running. If the docker.service is not active, you can follow these steps to rectify the issue and proceed successfully:

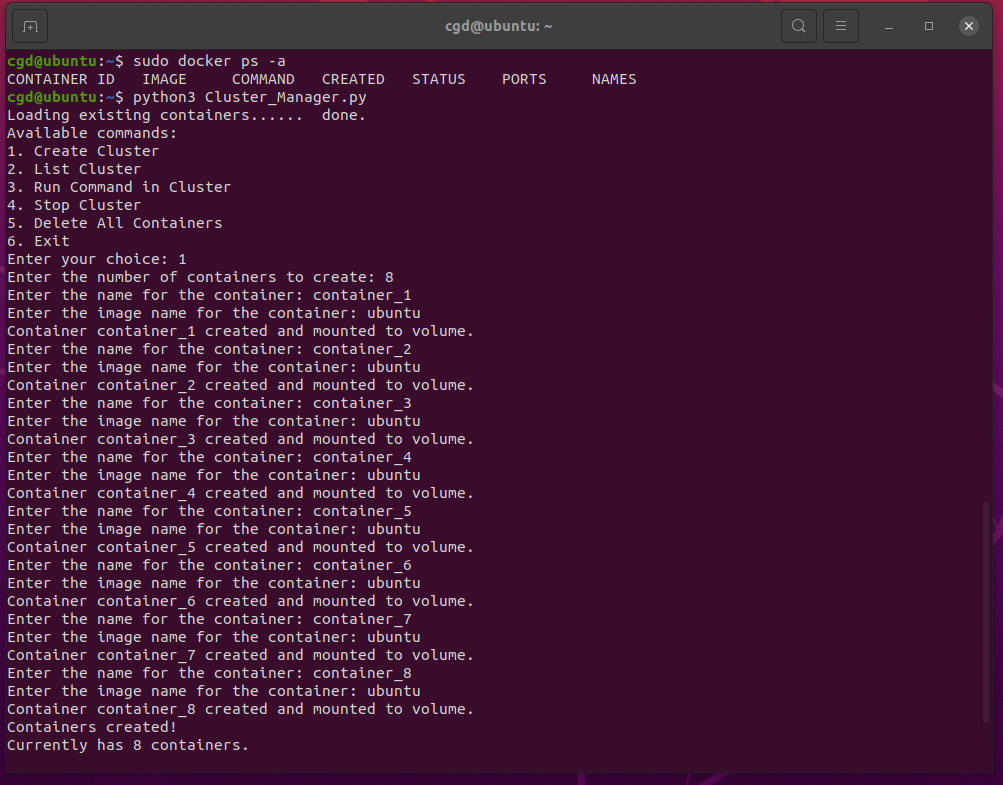




#### 2.1 Creating a Cluster

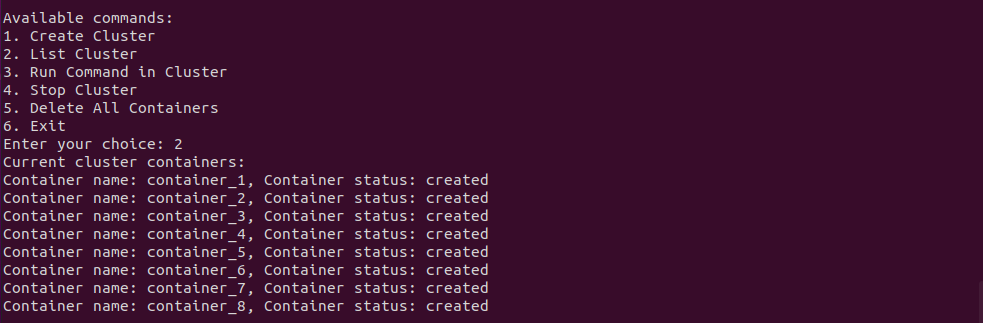
In this code snippet, users have the flexibility to specify the desired number of containers they wish to create. Additionally, they can individually set the names and images for each of these containers. Once these parameters are provided, the code automates the process of container creation according to the user's specifications.

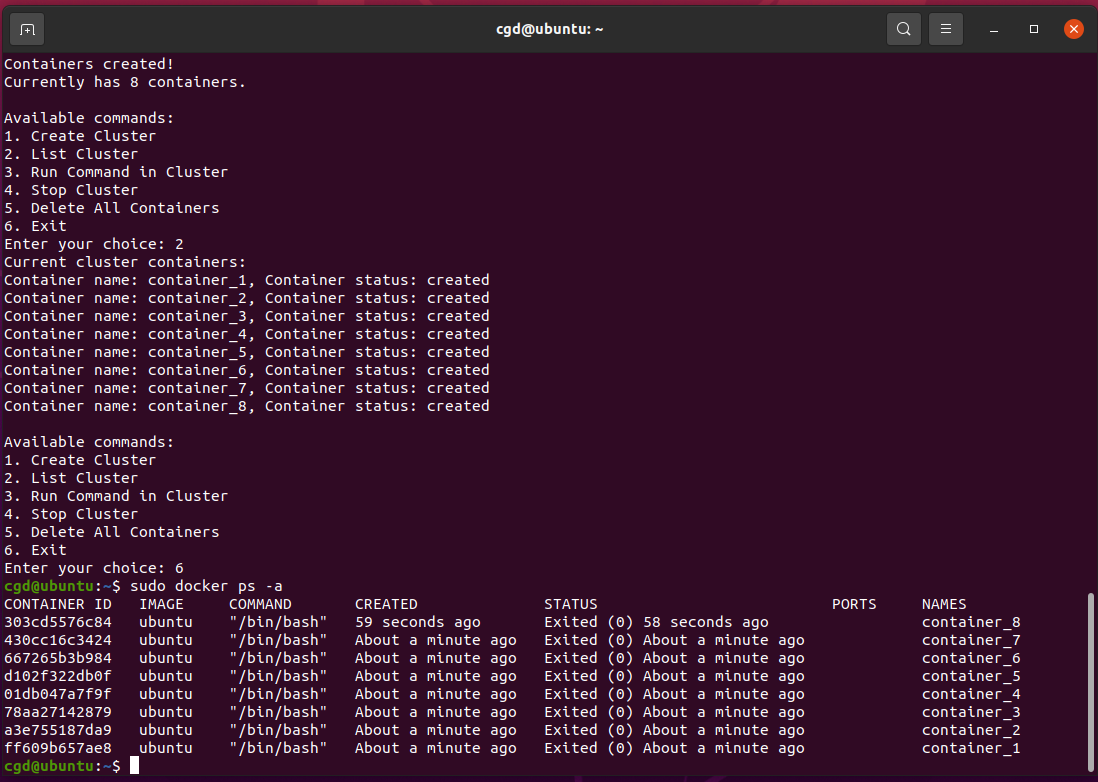
This level of customization ensures that users can tailor their container environment to their specific needs.



#### 2.2 Listing Cluster Information

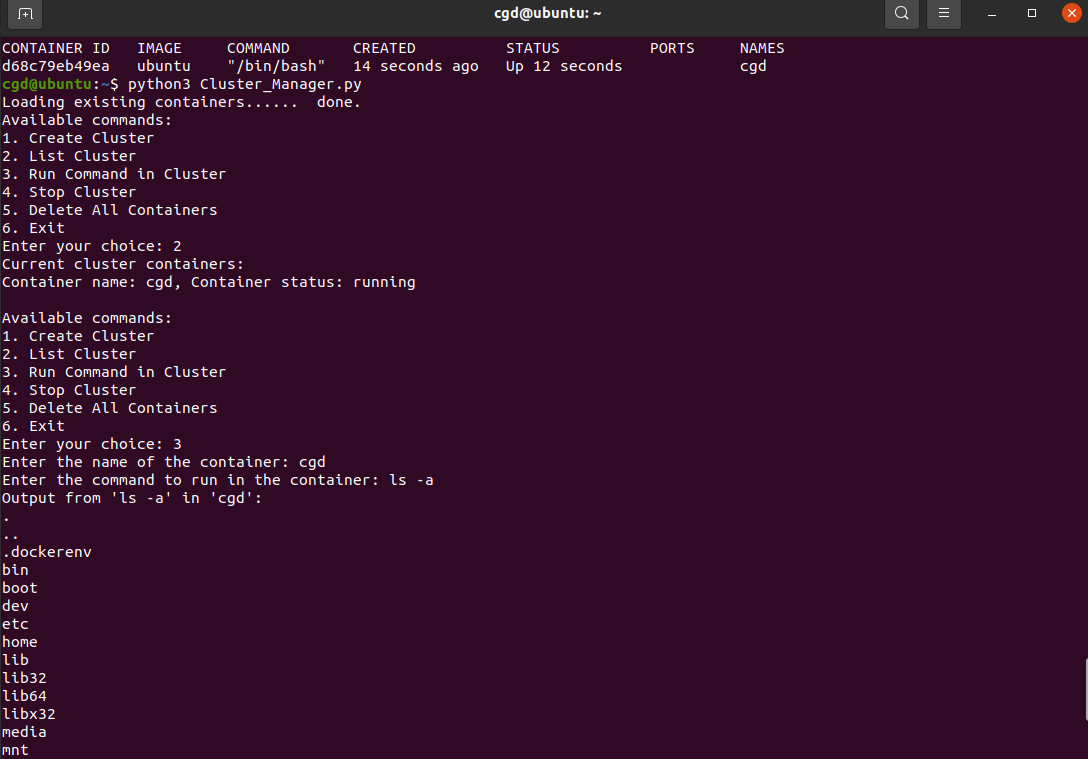
Once the containers have been successfully created, users have the option to check information about these containers. By selecting the "List Cluster" option, the names and statuses of each container will be displayed on the screen. This provides users with a clear overview of the current state of the containerized cluster.



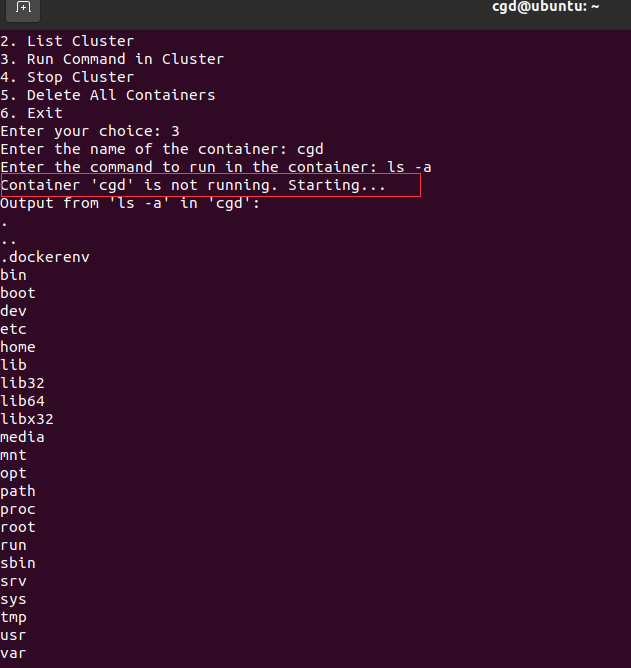
Users also have the option to utilize the command docker ps -a to view information about the created containers. This command provides a comprehensive overview of all containers, including their names, statuses, and additional details. 

#### 2.3 Running Commands in the Cluster

Users can actively interact with specific containers by entering the container's name along with the desired command they want the container to execute. The resulting output of the executed command will be displayed on the screen.

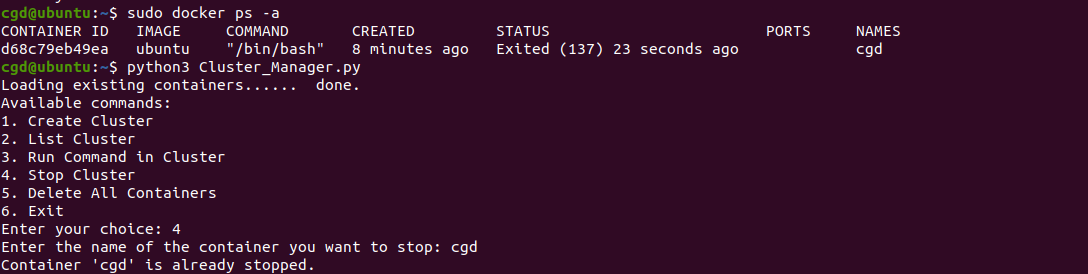


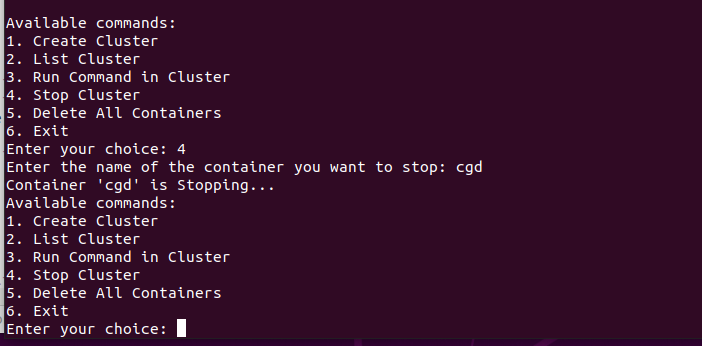
If the specified container is not currently running, the code will take the initiative to start the container automatically before executing the provided command.



#### 2.4 Stopping the Cluster

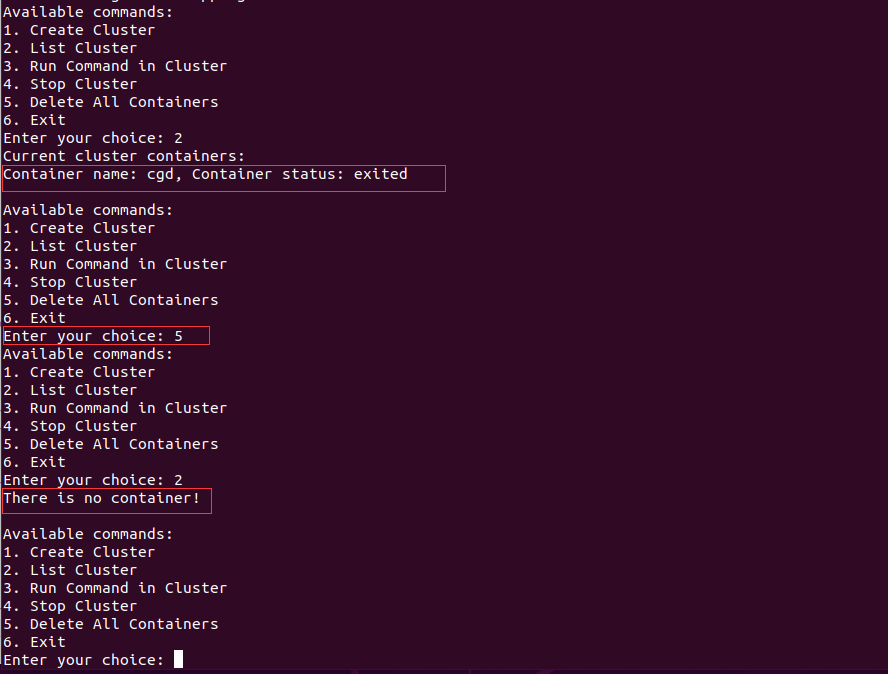
Users can choose a specific container they wish to stop within the cluster. If the chosen container is already in a stopped state, the code will provide a reminder to that effect. However, if the container is currently running, the code will proceed to stop it.

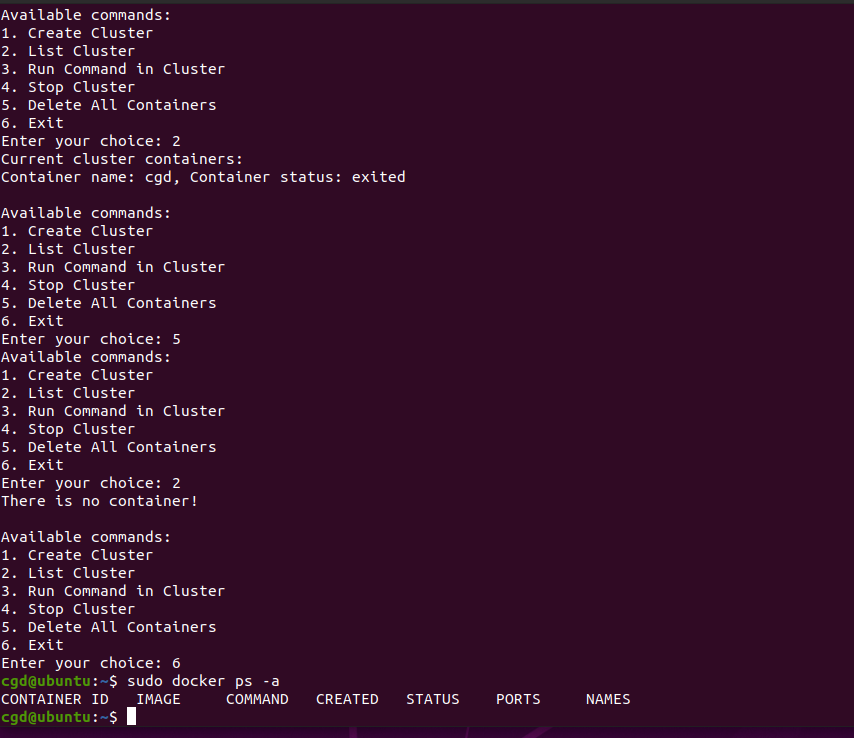




#### 2.5 Deleting All Containers

By selecting this option, all containers within the cluster will be deleted.





## Assignment#2: Data Processing in CM

This assignment presents a study on utilizing container managers to support data volumes for distributed data processing. We explore a scenario wherein a cluster of containers processes a large array of data collaboratively, with each container focusing on a distinct subset of the data. By employing shared data volumes, the processed results are made accessible to all containers. The primary objective is to compute and compare various statistical parameters, including sum, average, maximum, minimum, and standard deviation, for the subsets of data processed by each container.

### 1. Introduction

Containerization technology has gained prominence due to its efficiency and ease of deployment. Containers provide isolated and lightweight environments, making them suitable for various applications, including distributed data processing. In this report, we investigate the utilization of container managers in conjunction with shared data volumes for efficient and collaborative data processing.

A dataset comprising 100,000 numbers ranging from 1 to 100,000 is generated. This dataset is to be distributed among containers for processing.

### 2. Results

After creating four containers named Container\_Process\_1, Container\_Process\_2, Container\_Process\_3, and Container\_Process\_4, a new function called "Data Process" was introduced. In this phase, we observed each container printing out their respective results after processing the data divided from 1 to 100,000. The statistical parameters computed by each container for its allocated dataset subset are presented as follows:

* Container\_Process\_1:

Sum: 12,502,500

Average: 50,010

Maximum: 25,000

Minimum: 1

Standard Deviation: 7,219.86

* Container\_Process\_2:

Sum: 37,507,500

Average: 100,010

Maximum: 50,000

Minimum: 25,001

Standard Deviation: 7,219.86

* Container\_Process\_3:

Sum: 62,512,500

Average: 150,010

Maximum: 75,000

Minimum: 50,001

Standard Deviation: 7,219.86

* Container\_Process\_4:

Sum: 87,517,500

Average: 200,010

Maximum: 100,000

Minimum: 75,001

Standard Deviation: 7,219.86

