



# ASSIGNMENT 1

Kevin Pérez Román, Jose Miguel López Riani

# Introduction

In this document we realized the activity 1 from the first assignment by the courses Operative systems.

## 1 Enumerate the milestones in the evolution of computer systems.

- **First Generation (1940s-1950s):** The computers in this era were characterized by vacuum tubes and plugboards, with no operating system. All tasks were programmed manually, and they were executed one at a time without any automation or task management.
- **Second Generation (1950s-1960s):** In this era, computers upgraded to transistors and batch systems, where jobs were grouped together to be processed in sequence. Simple operating systems began to emerge, introducing concepts that are still relevant today, such as executing tasks in a specific order and with defined priorities.
- **Third Generation (1960s-1970s):** Integrated circuits appeared, enabling the development of multiprogramming and time-sharing systems, which allowed multiple users to interact with the computer simultaneously. These advances significantly improved efficiency and resource access.
- **Fourth Generation (1970s-present):** The advent of microprocessors led to the creation of personal computers and graphical user interfaces (GUIs). This generation also saw the rise of distributed systems and networks, transforming how computing resources were used and shared.
- **Fifth Generation (Present and Future):** Characterized by parallel processing, real-time systems, cloud computing, virtualization, and advancements in mobile and embedded operating systems. This generation also explores emerging technologies such as artificial intelligence and machine learning, which are redefining the capabilities and functions of operating systems.

## 2 What are the four components of a computer system? Describe each one.

- **Hardware:** The physical components of a computer system, including the CPU, memory, I/O devices, and storage devices. It provides the basic computing resources for the system.
- **Operating System:** The software layer that manages the hardware resources, provides a user interface, and serves as an intermediary between applications and hardware. It ensures efficient and fair resource allocation.
- **Application Programs:** Software that performs specific tasks for users, such as word processors, web browsers, and database systems. These programs rely on the OS to access hardware resources.
- **Users:** Individuals or entities that interact with the computer system. Users can be humans, other computers, or automated processes.

## 3 What is the difference between a monolithic kernel and a microkernel?

- **Monolithic Kernel:** In a monolithic kernel, all OS services (such as file system management, process management, and device drivers) are integrated into a single large block of code that runs in kernel mode. This approach can be efficient but can lead to stability and security issues since a bug in one component can crash the entire system.
- **Microkernel:** A microkernel has a minimalistic design, where only essential services (like basic inter-process communication and scheduling) run in kernel mode. Other services, like file systems and device drivers, run in user mode as separate processes. This approach enhances modularity, security, and stability but can introduce performance overhead due to the need for communication between user-mode services.

## 4 Define an Operating System from two different perspectives.

- **User Perspective:** From the user's point of view, an operating system is an interface that provides a convenient way to interact with the computer hardware. It abstracts the complexity of hardware management and provides services such as file management, program execution, and I/O operations.
- **System Perspective:** From the system's point of view, an operating system is a resource manager that allocates and controls the computer's resources, such as CPU time, memory, and I/O devices, ensuring that different applications and users can operate efficiently and securely.

## 5 What is the purpose of system calls?

System calls are the interface through which user-level applications request services from the operating system. They provide a controlled entry point into the kernel, allowing programs to perform operations such as process control, file manipulation, and communication with hardware, without requiring direct access to the hardware.

## 6 What is a multiprogrammed operating system?

A multiprogrammed operating system is one that can execute multiple programs simultaneously by managing the CPU's time among them. This is achieved by keeping several jobs in memory at once and switching between them, allowing the CPU to be utilized more efficiently by reducing idle time.

## 8 What is a process?

A process is an instance of a program in execution. It consists of the program code (text), data, stack, and a Process Control Block (PCB) that contains information about the process's current state and resources. A process is the fundamental unit of work in an operating system.

## 9 What are the states of a process?

- **New:** The process is being created.

- **Running:** Instructions are being executed by the CPU.
- **Waiting (Blocked):** The process is waiting for some event (such as I/O completion) to occur.
- **Ready:** The process is ready to run but is waiting for CPU time.
- **Terminated:** The process has finished execution and is being removed from memory.

## 10 What information is stored in the Process Control Block (PCB) associated with a process?

- **Process State:** The current state of the process (e.g., running, waiting).
- **Program Counter:** The address of the next instruction to be executed.
- **CPU Registers:** The contents of all processor registers.
- **Memory Management Information:** Information such as page tables, segment tables, and base and limit registers.
- **Accounting Information:** CPU usage, process priority, and other accounting data.
- **I/O Status Information:** List of I/O devices allocated to the process, list of open files, etc.

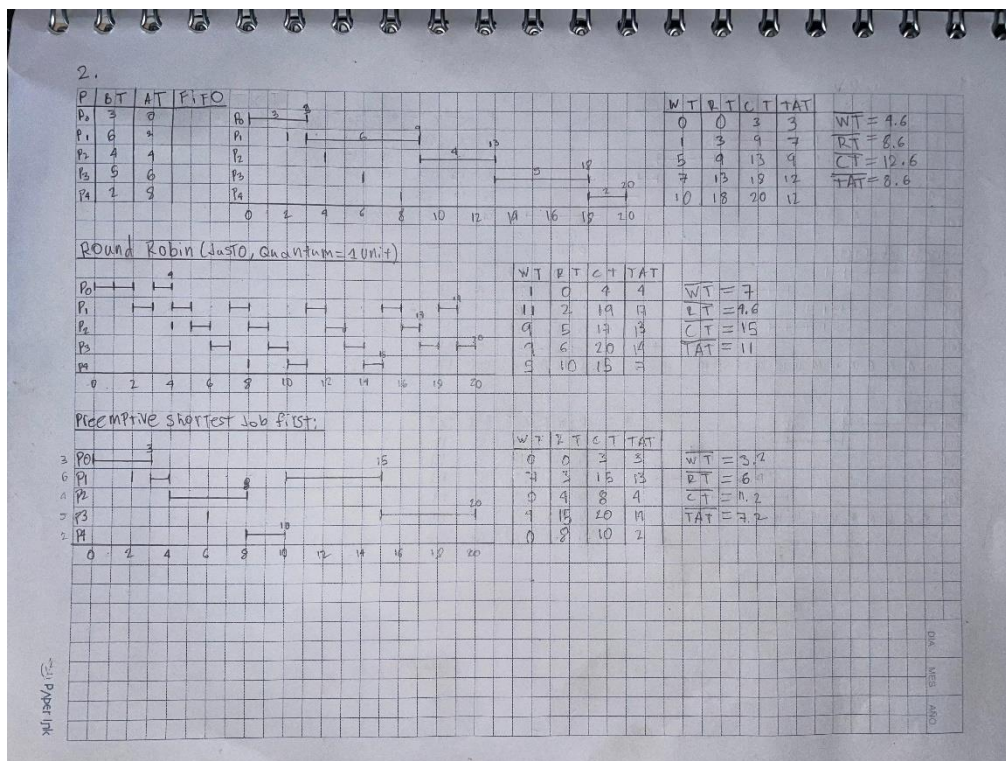
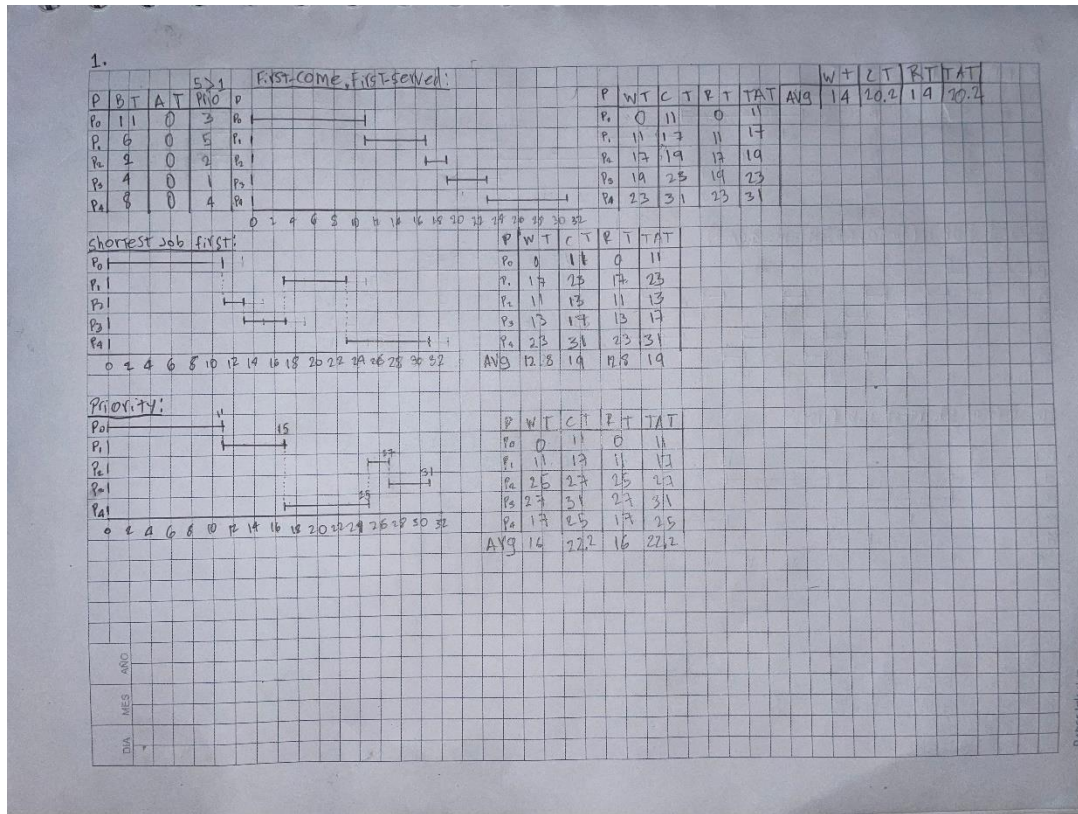
## 11 What are the main activities of an operating system in relation to process management?

- **Process Creation and Deletion:** Managing the creation and termination of processes.
- **Process Scheduling:** Deciding which process should be executed by the CPU next.
- **Process Synchronization:** Ensuring that processes do not interfere with each other while sharing resources.
- **Process Communication:** Facilitating communication between processes, whether they are on the same machine or across a network.
- **Deadlock Handling:** Ensuring that processes do not get stuck in a state where they cannot proceed because each is waiting for a resource held by another.

Typing this command will give the Client and Server versions available on your computer.

Paste a screenshot of result

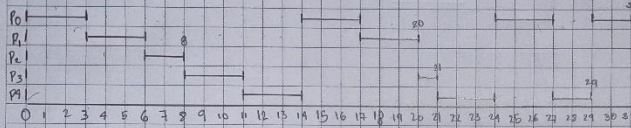
# Scheduling





3. P B T Round Robin, QUANTUM=3, Justo

se asume que el orden de la cola es el mismo de las etiquetas.



P	WT	RT	CT	TAT	
P0	20	0	31	51	WT = 15.6
P1	14	3	20	20	RT = 5.6
P2	6	0	9	9	CT = 21.8
P3	17	8	21	21	TAT = 21.8
P4	21	11	29	29	

DIA  
MES  
AÑO

Paper link

4.

se usa shortest job first, ya que con este orden de scheduling se adquiere el mejor response time.

si  $x \leq 3$ , el orden es:  $x, 3, 6, 9 = P_4, P_2, P_3, P_1, P_0$   
 si  $3 < x \leq 5$ , el orden es:  $3, x, 5, 6, 9 = P_2, P_4, P_3, P_1, P_0$   
 si  $5 < x \leq 6$ , el orden es:  $3, 5, x, 6, 9 = P_1, P_3, P_4, P_2, P_0$   
 si  $6 < x \leq 9$ , el orden es:  $3, 5, 6, x, 9 = P_2, P_3, P_1, P_4, P_0$   
 si  $x > 9$ , el orden es:  $3, 5, 6, 9, x = P_2, P_3, P_1, P_0, P_4$

en este caso  $x=6$ , entonces puede estar antes o después de P sin afectar el response time promedio

Paper link

DIA  
MES  
AÑO

# Docker

```
PS C:\Users\jmlp> docker version
Client:
 Version:           27.1.1
 API version:       1.46
 Go version:        go1.21.12
 Git commit:        6312585
 Built:             Tue Jul 23 19:57:57 2024
 OS/Arch:           windows/amd64
 Context:           desktop-linux

Server: Docker Desktop 4.33.1 (161083)
Engine:
 Version:           27.1.1
 API version:       1.46 (minimum version 1.24)
 Go version:        go1.21.12
 Git commit:        cc13f95
 Built:             Tue Jul 23 19:57:19 2024
 OS/Arch:           linux/amd64
 Experimental:      false
containerd:
 Version:           1.7.19
 GitCommit:         2bf793ef6dc9a18e00cb12efb64355c2c9d5eb41
runc:
 Version:           1.7.19
 GitCommit:         v1.1.13-0-g58aa920
docker-init:
 Version:           0.19.0
 GitCommit:         de40ad0
```

## Questions

1. How many arguments are absolutely required by the command 'docker pull' ?
2. Do you remember what a registry is?
  1. Docker pull requires at least one argument: the name or tag of the image to be downloaded from a registry.

```
Usage:  docker pull [OPTIONS] NAME[:TAG|@DIGEST]

Download an image from a registry

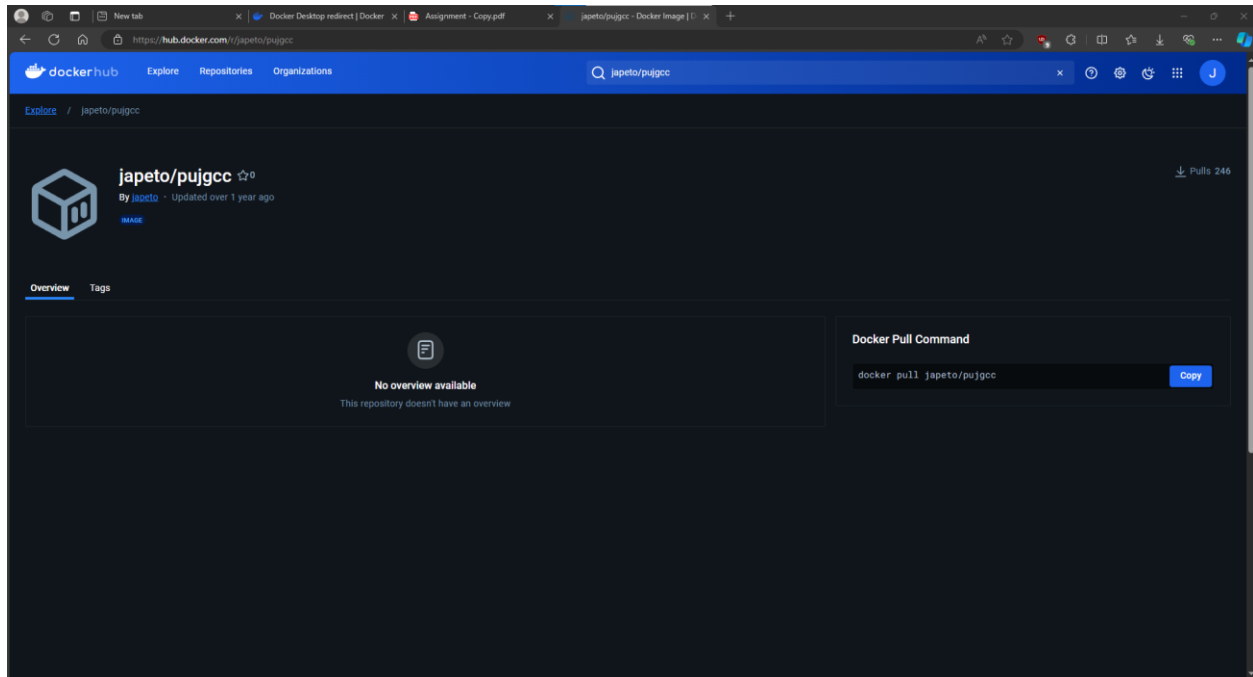
Aliases:
  docker image pull, docker pull

Options:
  -a, --all-tags           Download all tagged images in the repository
  --disable-content-trust  Skip image verification (default true)
  --platform string        Set platform if server is multi-platform
                           capable
  -q, --quiet              Suppress verbose output
```

2. A registry in docker is a system for versioning, storing and distributing Docker images.

Download a predefined image available on the DockerHub In a web browser, navigate to the DockerHub : <https://hub.docker.com/>

In the top search bar, type : `japeto/pujgcc` and paste a screenshot



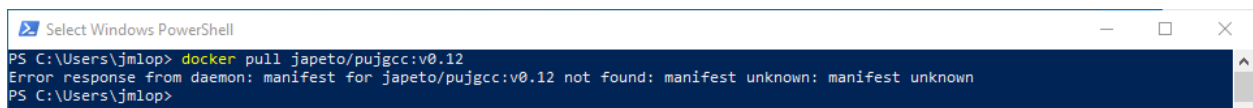
## Questions

1. How many times was the japeto image downloaded ?

1. The japeto image has been downloaded 246 times at the time it was consulted.

Execute the command inside a terminal.

Paste a screenshot of result



You will get an error as this image has no default tag ("latest"). So we need to specify one in the command line.

Go to the "Tags" tab and copy the pull command of version latest Paste a screenshot of result.



```

PS C:\Users\jmllop> docker pull japeto/pujgcc:latest
latest: Pulling from japeto/pujgcc
aa18ad1a0d33: Pull complete
15a33158a136: Pull complete
f67323742a64: Pull complete
c4b45e832c38: Pull complete
e5d4afe2cf59: Pull complete
1efbd2d5674a: Pull complete
022a58c161f7: Pull complete
6461d3be7619: Pull complete
8287aab02fcd: Pull complete
edd3d1d6fad6: Pull complete
c6dbbe32db7e: Pull complete
Digest: sha256:9b3d7bbf410396d0a26e6bcffbb54e4239736d5a23ad694b03d93c26f9fc6868
Status: Downloaded newer image for japeto/pujgcc:latest
docker.io/japeto/pujgcc:latest
PS C:\Users\jmllop>

```

**Question:**

**1. How many times do you see 'Pull complete' displayed ? Why ?**

1. There are 11 instances of "Pull complete displayed". Docker images are built in layers, which are incremental snapshots that make up the final image. When making a pull request, a pull is made for each layer of the image being downloaded. In our case, the image had 17 layers, of which 6 were empty, hence the 11 "Pull complete" displays.

Now, to be sure that the image was correctly pulled, let's see the list of all available downloaded images inside our workspace.

**Paste a screenshot of result**

```

PS C:\Users\jmllop> docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
japeto/pujgcc latest 05141310a94c 19 months ago 1.39GB
PS C:\Users\jmllop>

```

**Question:**

**1. What is the size of the japeto/pujgcc image ?**

1. The image is 1.39GB.

Among the Docker commands, we will now use the 'run' command.

**Question:**

**1. What are the options and parameters of the 'run' command ?**

1. The run command takes in options, the name of the image, a command and arguments:

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

As displayed in the terminal, the description of the command is 'Run a command in a new container'.

**Question :**

**1. What is the difference between an image and a container ?**

1. An image is a read only template that has the instructions needed for creating a container. On the other hand, a container is a running instance of a docker image; the execution environment in which the application runs.

**Now, to run the application, execute the following command:**

**Paste a screenshot of result**

```
PS C:\Users\jmlol> docker run japeto/pujgcc bash --help
GNU bash, version 4.3.30(1)-release-(x86_64-pc-linux-gnu)
Usage: bash [GNU long option] [option] ...
        bash [GNU long option] [option] script-file ...
GNU long options:
    --debug
    --debugger
    --dump-po-strings
    --dump-strings
    --help
    --init-file
    --login
    --noediting
    --noprofile
    --norc
    --posix
    --rcfile
    --restricted
    --verbose
    --version
Shell options:
    -ilrsD or -c command or -O shopt_option          (invocation only)
    -abefhkmnptuvxBCHP or -o option
Type `bash -c "help set"' for more information about shell options.
Type `bash -c help' for more information about shell builtin commands.
Use the `bashbug' command to report bugs.
PS C:\Users\jmlol>
```

**Find the paths to bind To bind our current folder to the /data/ folder located inside a container, we first need the absolute path of the current folder, obtained through the unix pwd command.**

**Paste a screenshot of result**

```
PS C:\Users\jmlol> pwd

Path
----
C:\Users\jmlol

PS C:\Users\jmlol>
```

I made a directory specifically for this, as the current path was my user directory:

```

PS C:\Users\jmllop> pwd

Path
----
C:\Users\jmllop

PS C:\Users\jmllop> ls

    Directory: C:\Users\jmllop

Mode                LastWriteTime         Length Name
----                -
d-----          9/1/2024   1:15 PM             .docker
d-----         2/12/2024   4:02 PM             .idlerc
d-----         5/15/2024   5:56 PM             .matplotlib
d-----         2/12/2024   2:51 PM             .vscode
d-r-----       1/7/2024   1:05 PM          3D Objects
d-----       1/7/2024   1:21 PM             ansel
d-r-----       1/7/2024   1:05 PM          Contacts
d-----       1/7/2024   1:07 PM          Documents
d-----       1/7/2024   1:43 PM          Downloads
d-r-----       1/7/2024   1:05 PM          Favorites
d-r-----       1/7/2024   1:05 PM             Links
d-r-----       5/28/2024   9:53 AM             Music
dar--l         2/18/2024   5:38 PM          OneDrive
d-r-----       1/7/2024   1:05 PM          Saved Games
d-r-----       1/28/2024  11:09 AM          Searches
d-r-----       9/1/2024   1:15 PM          Videos
-a-----       2/12/2024   7:37 PM          302 .gitconfig
-a-----       3/11/2024  12:44 PM           3 .node_repl_history

PS C:\Users\jmllop> mkdir dockerData

    Directory: C:\Users\jmllop

Mode                LastWriteTime         Length Name
----                -
d-----          9/1/2024   4:53 PM          dockerData

PS C:\Users\jmllop> cd dockerData
PS C:\Users\jmllop\dockerData> pwd

Path
----
C:\Users\jmllop\dockerData

PS C:\Users\jmllop\dockerData>

```

## Bind a local folder into a container

To perform the folder mapping between the current folder and /data inside the image, the syntax is simple.

## Paste a screenshot of result

```

PS C:\Users\jmllop\dockerData> docker run japeto/pujgcc:latest ls /data
ls: cannot access /data: No such file or directory
PS C:\Users\jmllop\dockerData> docker run -v ${PWD}:/data japeto/pujgcc:latest ls /data
PS C:\Users\jmllop\dockerData>

```

## Question:

### 1. Is the displayed list the same as what is in your current folder?

1. Yes, as the folder was empty, so there was nothing to list.

Finally, we can run C on a C or C++ file located in the Data folder. Change the name of the file to any of the provided files.

Paste a screenshot of result

```
PS C:\Users\jmllop\dockerData> docker run -v ${PWD}:/data japeto/pujgcc:latest gcc /data/helloworld.c
gcc: error: /data/helloworld.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
PS C:\Users\jmllop\dockerData>
```

The folder was empty, so I had to create the helloworld.c file:

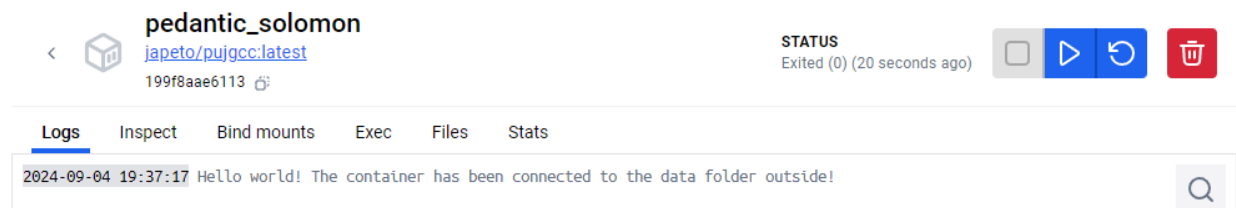
```
PS C:\Users\jmllop\dockerData> docker run -v ${PWD}:/data japeto/pujgcc:latest gcc /data/helloworld.c -o data/helloworld
PS C:\Users\jmllop\dockerData> docker run -v ${PWD}:/data japeto/pujgcc:latest ls /data/
helloworld
helloworld.c
PS C:\Users\jmllop\dockerData>
```

After running:

```
PS C:\Users\jmllop\dockerData> docker run -v ${PWD}:/data japeto/pujgcc:latest ./data/helloworld
Hello world! The container has been connected to the data folder outside!
PS C:\Users\jmllop\dockerData>
```

Use the start command to restart the container created in the last exercise Paste a screenshot of result

After running the last command, I checked which was the container last opened to use the name in start:



The screenshot shows the Docker Desktop interface for a container named 'pedantic\_solomon'. The container is based on the 'japeto/pujgcc:latest' image. The status is 'Exited (0) (20 seconds ago)'. The 'Logs' tab is selected, displaying a log entry from 2024-09-04 at 19:37:17: 'Hello world! The container has been connected to the data folder outside!'.

When running the command, the first problem was the fact that the -ti option didn't exist.

```
Options:
-a, --attach                Attach STDOUT/STDERR and forward signals
--detach-keys string        Override the key sequence for detaching a
                             container
-i, --interactive            Attach container's STDIN
```

The only two options available are -a and -i or -ai. So I ran the command that way.

```
PS C:\Users\jmllop\dockerData> docker start -ai pedantic_solomon /bin/bash
you cannot start and attach multiple containers at once
PS C:\Users\jmllop\dockerData> docker start -ai pedantic_solomon
Hello world! The container has been connected to the data folder outside!
PS C:\Users\jmllop\dockerData>
```

The other problem was that it was taking /bin/bash as a container, when it was an instruction. This is because each container is built when an instruction is run inside of them, so by starting this container, it already had a set goal. When removing the extra instruction, it restarted the container effectively.

Go back to the container using the exec command instead of the run command. Paste a screenshot of result

When running this command I ran into this error:

```
PS C:\Users\jmlop\dockerData> docker exec -ti pedantic_solomon /bin/bash

What's next:
  Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug pedantic_solomon
  Learn more at https://docs.docker.com/go/debug-cli/
Error response from daemon: container 199f8aae611356fefae642731e148688abfc9ee318b02d1008422554c6845ab7 is not running
PS C:\Users\jmlop\dockerData>
```

This is because exec runs an instruction in an already running container. As the container I referenced closed as soon as it completed the instruction it had, I couldn't run exec on it.

To solve this, I created a container that kept open:

```
PS C:\Users\jmlop\dockerData> docker run -d japeto/pujgcc:latest tail -f /dev/null
22ff4874f66dc5cd2f865003e1dfa805cd17271939b7b8106a0c54a9380500d2
PS C:\Users\jmlop\dockerData> docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
22ff4874f66d   japeto/pujgcc:latest   "tail -f /dev/null"     6 seconds ago Up 5 seconds             busy_herschel
PS C:\Users\jmlop\dockerData>
```

-d kept it running in the background and the instruction keeps it there indefinitely.

Now I can start the container, without using -d, as start already makes it a detached process. Not I can run the exec command, and pass an instruction to the already running container:

```
PS C:\Users\jmlop\dockerData> docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
9daa26841fd5   japeto/pujgcc:latest   "tail -f /dev/null"     22 seconds ago Up 17 seconds             distracted_mclean
PS C:\Users\jmlop\dockerData> docker exec -ti distracted_mclean /bin/bash
root@9daa26841fd5:/# exit
exit

What's next:
  Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug distracted_mclean
  Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\jmlop\dockerData> docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
9daa26841fd5   japeto/pujgcc:latest   "tail -f /dev/null"     About a minute ago Up About a minute             distracted_mclean
PS C:\Users\jmlop\dockerData>
```

When I exit the container, it doesn't stop, it keeps going.

Finally, I ran the stop command:

```
PS C:\Users\jmlop\dockerData> docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
9daa26841fd5   japeto/pujgcc:latest   "tail -f /dev/null"     2 minutes ago Exited (137) 5 seconds ago             distracted_mclean
PS C:\Users\jmlop\dockerData>
```