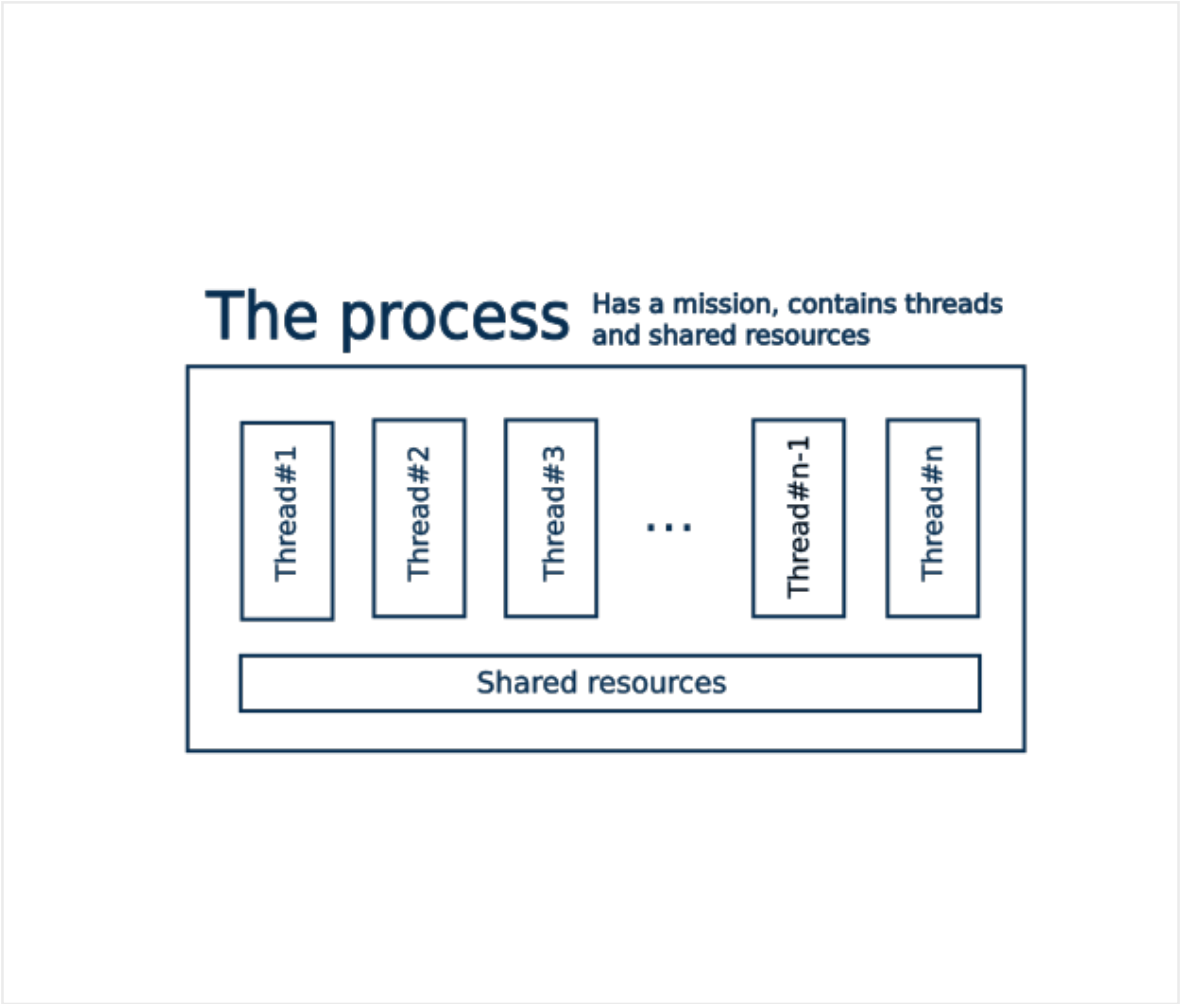
Processes And Threads

Process

The **process** is the self-contained unit of execution that has all the things necessary to accomplish the mission. In short, the process is the container for its threads, all necessities for their work, and their shared resources.

It’s cheaper to arrange access to shared resources once than to do it every time you spawn a new thread. The process has to have at least one thread as they do all the work.
There is no such thing as a thread without its process or a process without at least one thread.



If we look at the pizza business, a single pizza shop would serve as an analogy for the process. It has all the environment and equipment required for a worker to do the job. Equipment is expensive, so it's cheaper and more efficient when workers share it. There is no need for each worker to acquire personal equipment. On the other hand, the shop can't do anything without the workers. It is essential to have at least one worker because without them all the equipment would be useless.

Thread

In computer science, the **thread** of execution is a stream of instructions inside

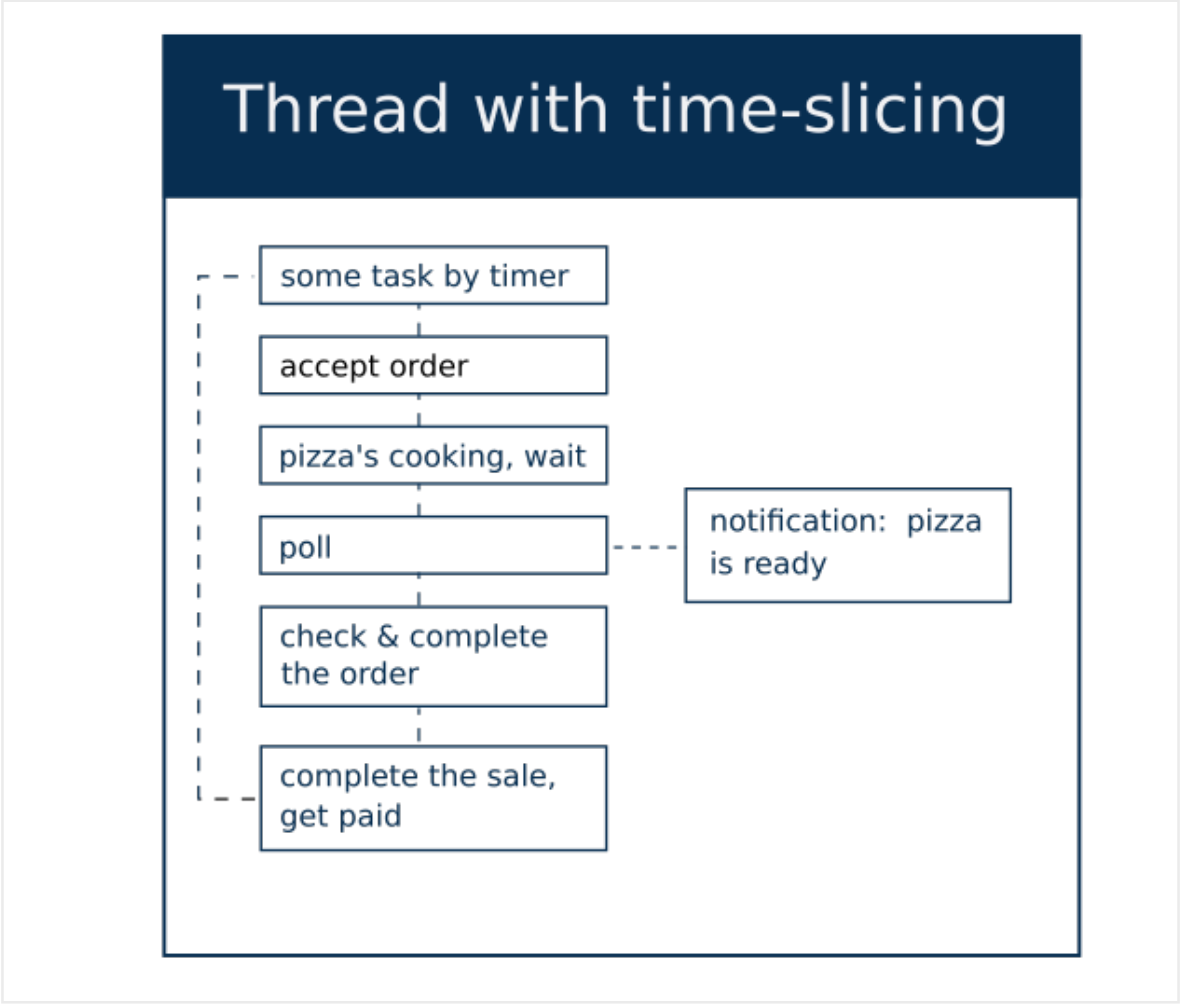
a process that can be scheduled and run independently. Each thread has its executor, and this executor can perform only one thread at a time. Several threads inside the same process can run concurrently or in parallel.

Workers in a pizzeria play the role of thread executors. Tasks that workers accomplish are the threads in the pizza shop "process".

Internal or lightweight concurrency
Workers can play different roles during the process of selling pizza. This concurrency is not among the workers but among the roles each worker plays. An important thing about these roles is that their tasks are typically fast enough and don't require a considerable amount of time and shared resources. They are **lightweight**.

If tasks are lightweight and don't require access to any shared resources except the executor's time and attention, there's no need to run them in different threads. It's cheaper to arrange their concurrent execution with time-slicing *inside one thread*.

The concurrency of this sort is called internal for obvious reasons. Often it is also called lightweight because the tasks inside such a thread are typically small and quick.



Concurrent tasks that compete only for executor's time and don't require a lot of resources to be completed can run concurrently inside the same thread.

The execution inside threads can be synchronous or asynchronous but never parallel.

https://www.geeksforgeeks.org/run-time-storage-organization/