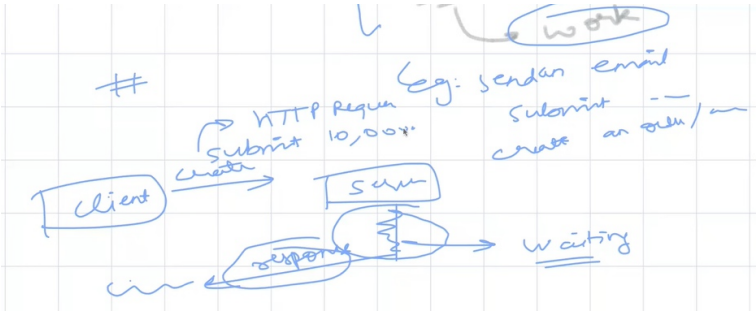
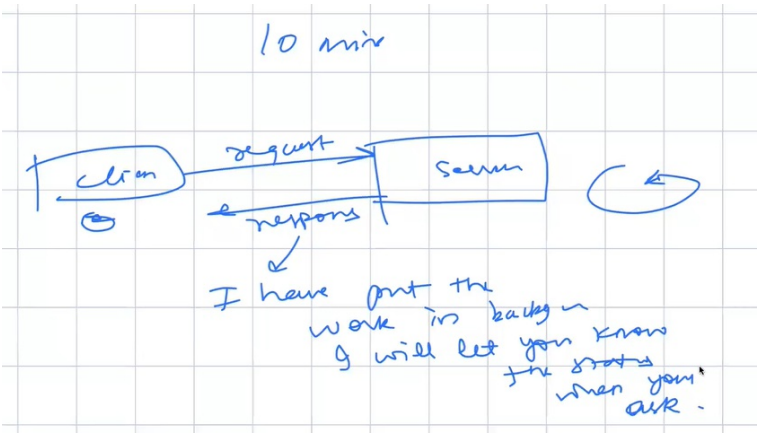
Q&A

Created	@January 21, 2024 9:36 PM
Status	Open
Updated	@September 17, 2024 5:42 PM



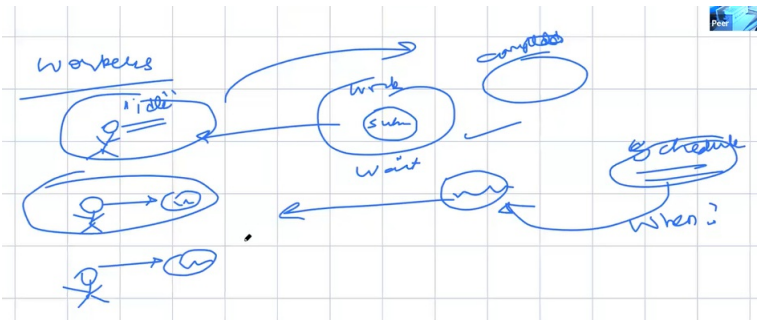
We know that the process is usually client send HTTP request, then the server will take it and do the job, and we will wait until the job is done so we can get the response back to the client. But what if the request is to **submit 10.000 sales invoice records?** Suppose it takes 10 minutes, in this case, we can't keep the user waiting for so long. So, we have to do the job in the background instead.



So, now the server acknowledges that it has received the request from the client, and immediately responds saying the work have been put in the background (queueing), and it will let the client know the status if asked.

This is useful not only for the UX, but also requests usually have **timeouts**. Suppose the timeout limit is 5 minutes, then this request will be cancelled after 5 minutes and every changes that the job/work made will be rolled back.

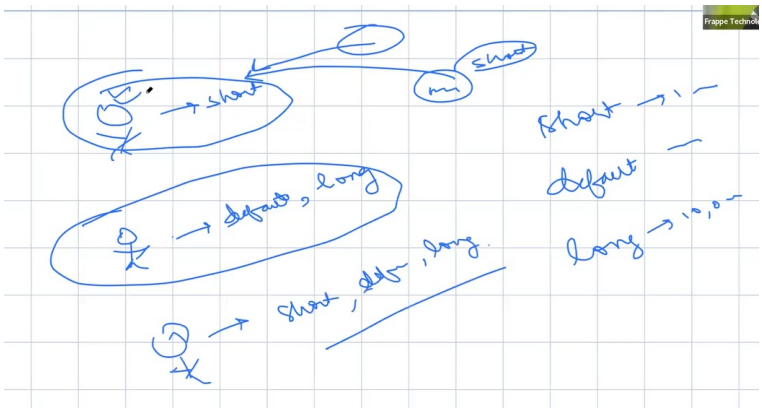
There are multiple ways to give notification for when the job is done. First, it's called pulling where client will keep asking every minute if the job is done or not. Second, we can use **realtime** (socket.io) event that will listens to it.



We can have n number of workers to do the background jobs. An `idle` worker will grab the work and do it, and then it will mark it as completed when it's done. If there is no idle worker, then that work has to wait until a worker is free. **We can configure the**

number of background workers on the common_site_config.json. Gunicorn workers are different (they handle the requests).

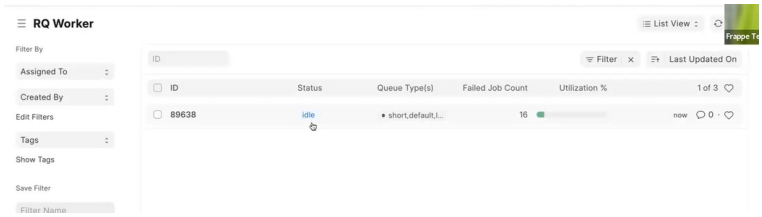
In development mode, usually the # of worker is set to 1. **Scheduler** (like scheduler events) **is only responsible for scheduling the work/job** (when the time comes, it will tell the worker to do the job). **The one that is responsible to do the actual work/job is the background workers.**



In Frappe, there are 3 types of task/queue: short, default, long

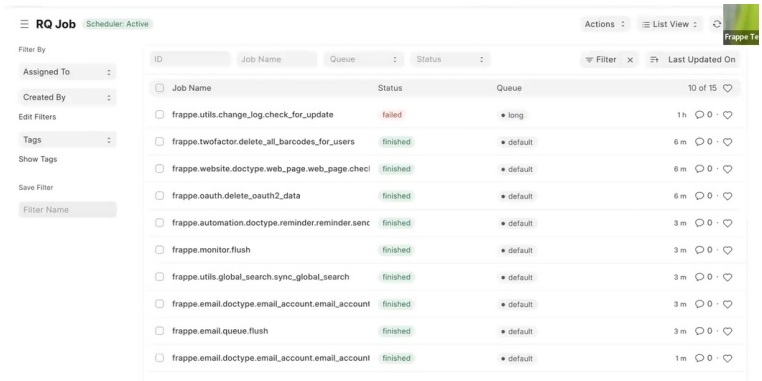
Now, we can define which types of task that a worker is supposed to take. For example, if we have 2 workers, we can set it in such a way so that `worker 1` will only take `short` tasks, while `worker 2` will only take `default` & `long` tasks.

To handle these queueing, frappe uses `redis queue`. We can see the worker on the `RQ Worker` virtual doctype that comes from Redis directly, it tracks the background worker that we have:

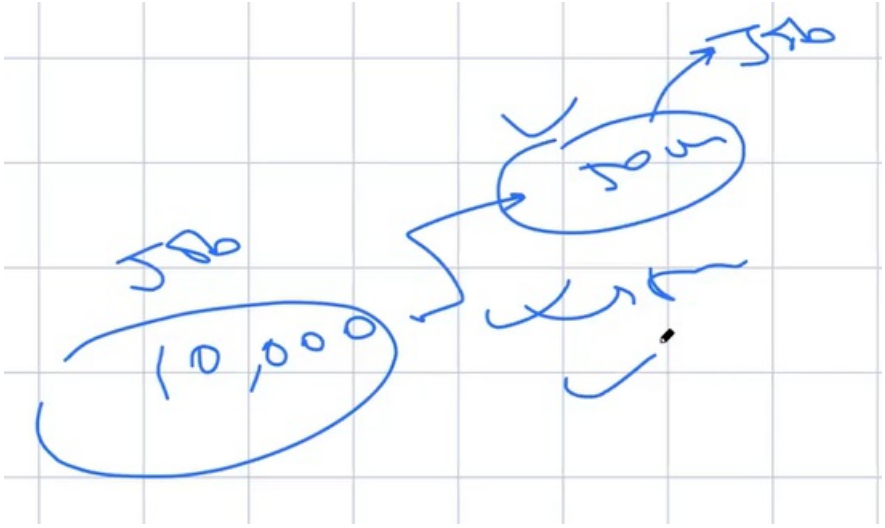


From here, we can see the worker we have, what status it's currently in, the queue type it handles, etc. Utilization % means that if it was running for 1 hour, how many of that 1 hour is it actually working.

Similarly, frappe also tracks the background jobs on `RQ Job` doctype:



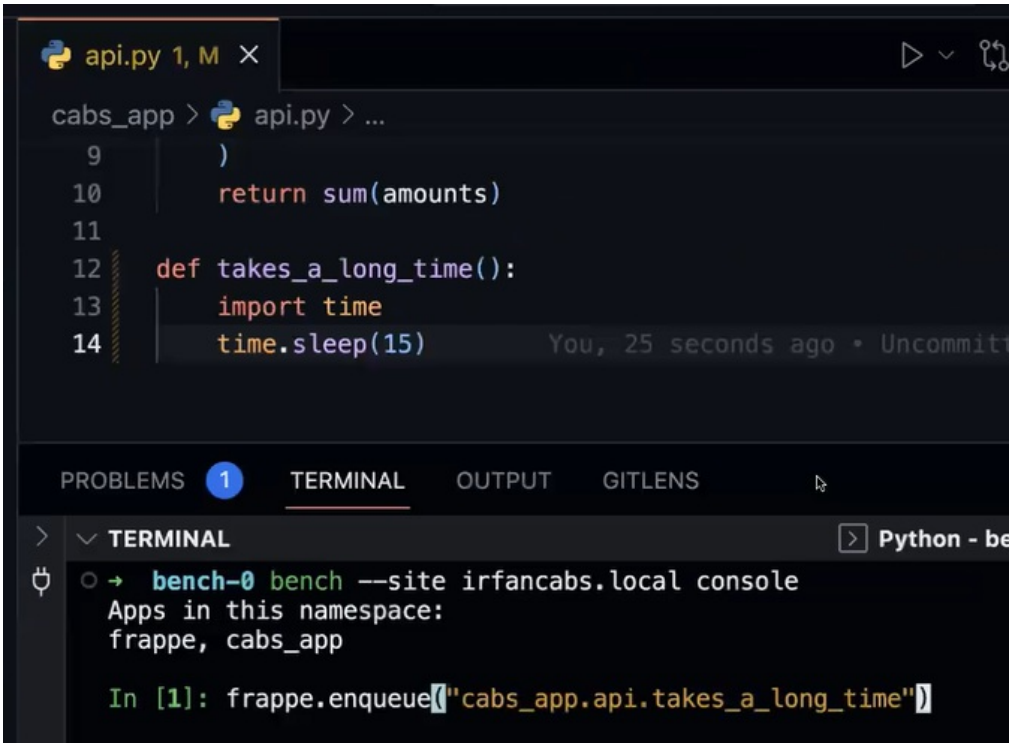
Default timeout for `default` is 5 minutes, while for `long` it's 25 minutes.



There is also this thing called **batching**, where we can break down a big job into multiple smaller jobs so that we can work on them in parallel.

How to queue our jobs:

We can use the method `frappe.enqueue()` and provide the dotted path to the method.



If we run the command above, we can see the job being queued and worked on the `RQ Job` virtual doctype. We can also tag the queue by adding the argument `queue="short" (or other type)`.

To add workers, we can just add new entries for the worker process on **Procfile** and give them different names: