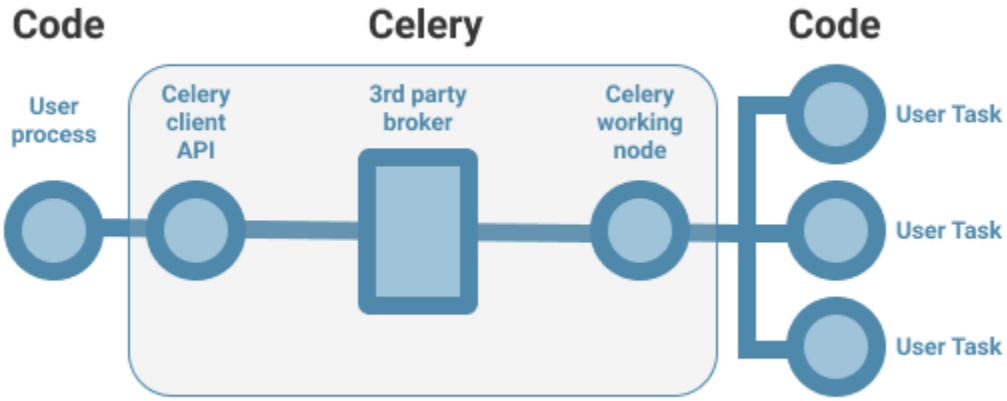
1. **What is Celery?**

   Ans. Celery is a distributed job queue that simplifies the management of task distribution. Developers break datasets into smaller batches for Celery to process in a unit of work known as a job.

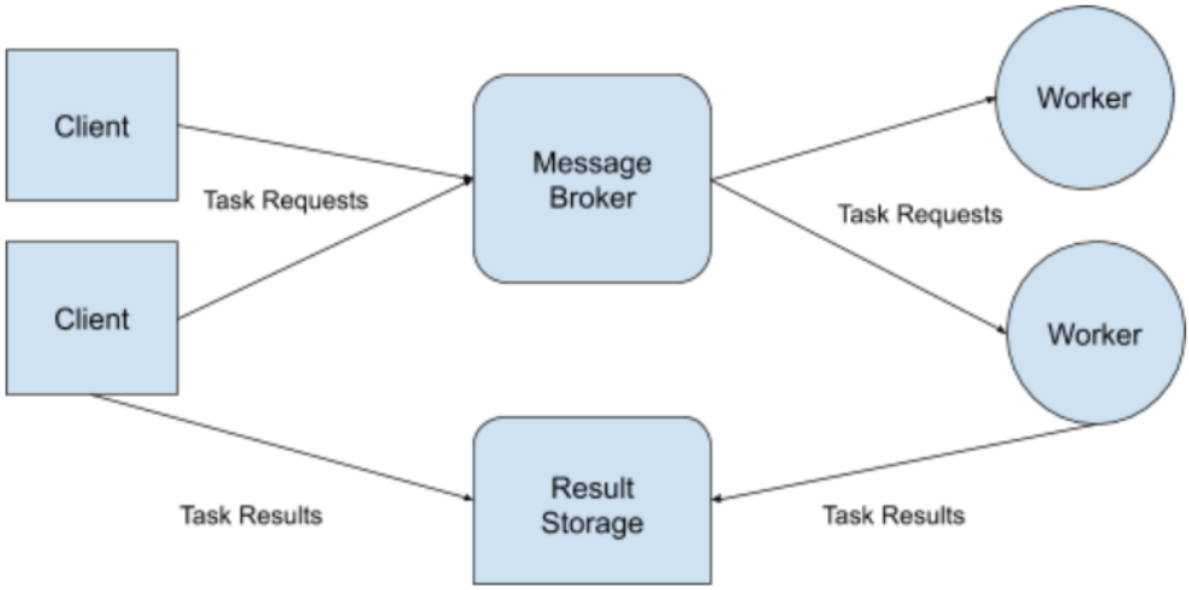
   It is an asynchronous task queue based on distributed message passing. Task queues are used as a strategy to distribute the workload between threads/machines.



2. **What are the Components of Celery?**

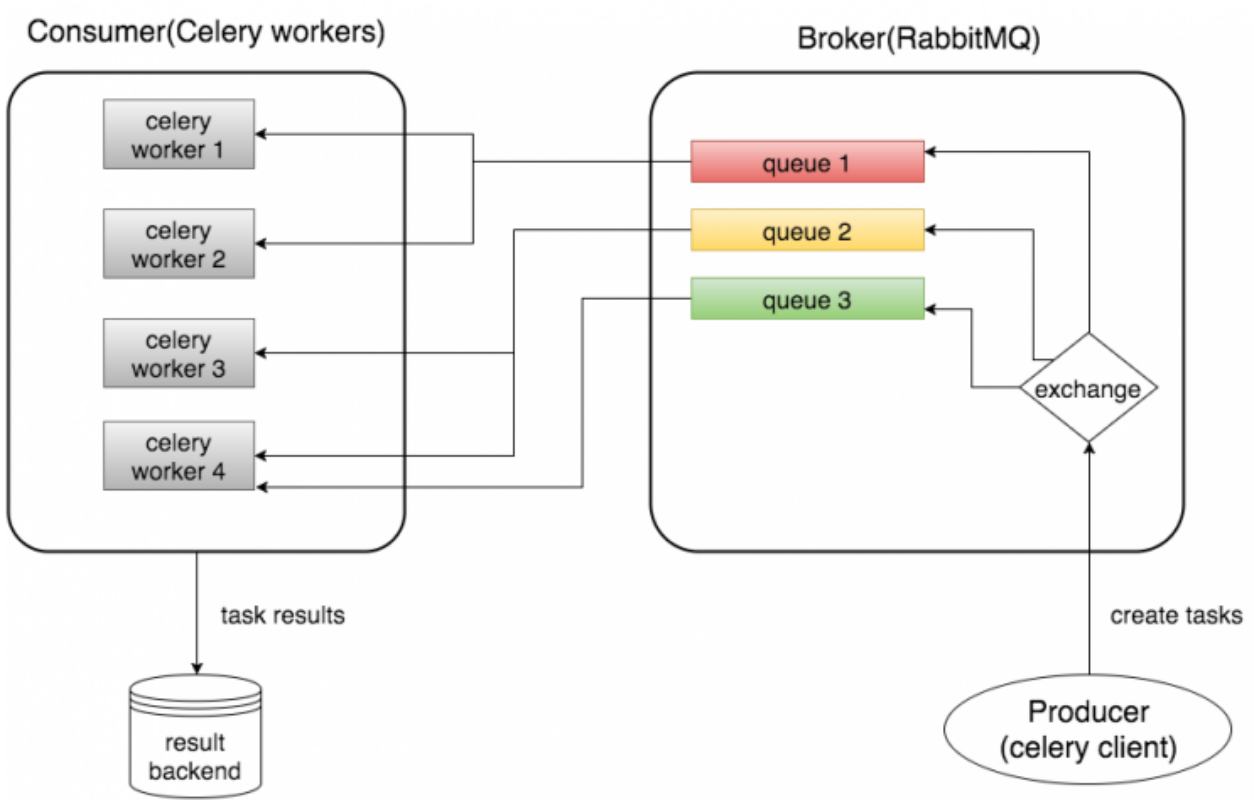
   Ans . Celery has three components

   - The celery client which will interact with our application and runs parallel with it
   - The celery workers which will run the long-running tasks given by our application asynchronously
   - The message broker which the celery client will use to keep track of the tasks and queue it in so that the celery workers can consume them
   - The components and its communication are depicted in the below diagram



3. **What is a Broker?**

   Ans. The Broker (RabbitMQ) is responsible for the creation of task queues, dispatching tasks to task queues according to some routing rules, and then delivering tasks from task queues to workers.



4. Why should we use celery instead of RQ?

Ans. These are the reasons why we use Celery instead of RQ. RQ is designed to be simpler all around. Celery is designed to be more robust. They are both excellent.

Documentation. RQ's documentation is comprehensive without being complex, and mirrors the project's overall simplicity - you never feel lost or confused. Celery's documentation is also comprehensive, but expect to be re-visiting it quite a lot when you're first setting things up as there are too many options to internalize

Monitoring. Celery's Flower and the RQ dashboard are both very simple to setup and give you at least 90% of all information you would ever want

Broker support. RQ only supports Redis. This means less documentation on "what is a broker", but also means you cannot switch brokers in the future if Redis no longer works for you. For example, Instagram considered both Redis and RabbitMQ with Celery. This is important because different brokers have different guarantees e.g. Redis *cannot* (as of writing) guarantee 100% that your messages are delivered.

Priority queues. RQs priority queue model is simple and effective - workers read from queues in order. Celery requires spinning up multiple workers to

5. What is Celery Worker?

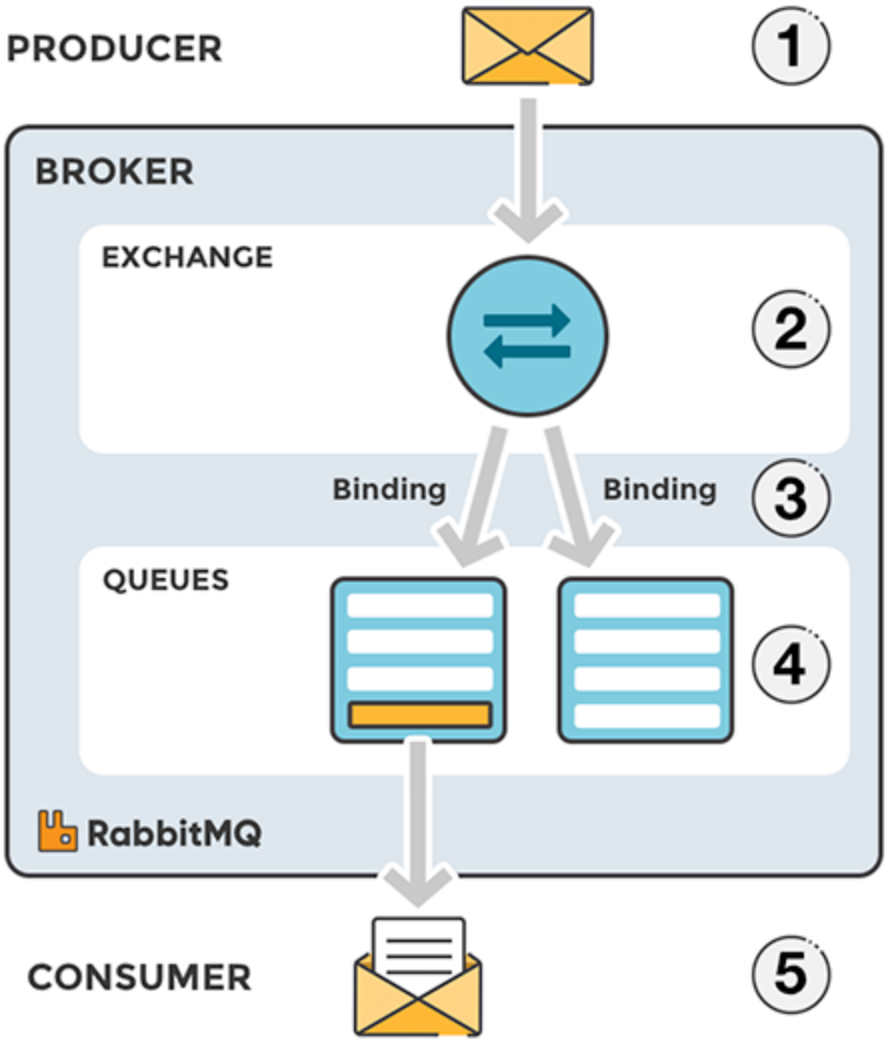
Ans: The Celery worker itself does not process any tasks. It spawns child processes (or threads) and deals with all the book keeping stuff. The child processes (or threads) execute the actual tasks. These child processes (or threads) are also known as the *execution pool*

6. What is RabbitMQ?

Ans. RabbitMQ is a message-queueing software also known as a *message broker* or *queue manager*. Simply said; it is software where queues are defined, to which applications connect in order to transfer a message or messages.

7. What is an exchange in RabbitMQ?

Ans. Exchanges are **message routing agents**, defined by the virtual host within RabbitMQ. An exchange is responsible for routing the messages to different queues with the help of header attributes, bindings, and routing keys. A binding is a "link" that you set up to bind a queue to an exchange



8. What is the routing key in RabbitMQ?

Ans. The routing key is a message attribute added to the message header by the producer. Think of the routing key as an "address" that the exchange is using to decide how to route the message. A message goes to the queue(s) with the binding key that exactly matches the routing key of the message.

9. What are the types of exchanges available in RabbitMQ?

Types of exchange are :

1. Direct exchange
2. Default exchange
3. Topic Exchange
4. Fanout Exchange
5. Headers Exchange
6. Dead Letter Exchange

10. Design a diagram of Producer and Consumer application with Rabbitmq as Message Broker