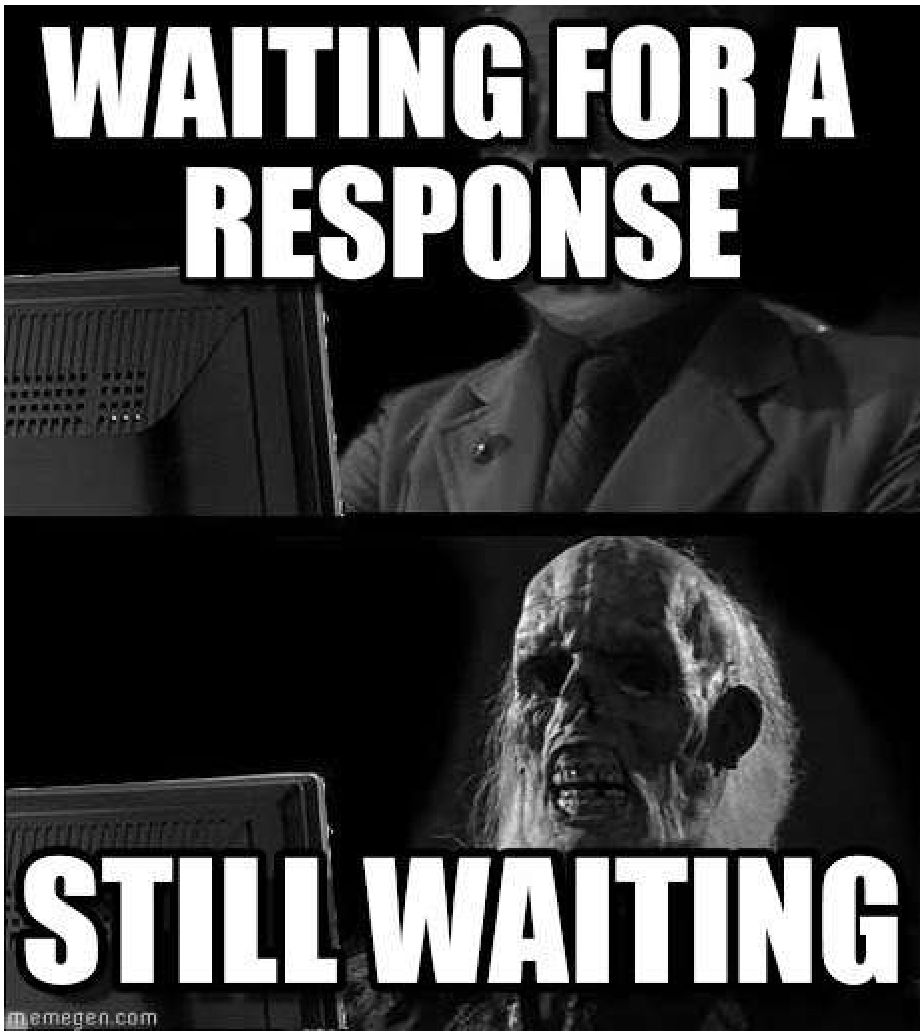
Handling Long Running API Requests in Nodejs

Dale Nguyen
Jul 5, 2018 · 4 min read



I have been playing around with designing and deploying RESTful API for a while. And it is interesting to handle and present data for a request from a client. Today, I got a question about how should I handle a request that takes a certain amount of time to process. We cannot let customer waiting for about 10' in order to get the response back. More than that, any other requests that come in during data processing will have to interaction patterns that we can use:

- **Request-Response (synchronous)**: An answer is returned with a response. For example, in e-commerce applications, a user is notified immediately if a submitted order has been processed or if there are any issues.
- **Fire-and-forget (asynchronous)**: A request has been received and an alternative means to get the response is provided. For example, when users import a large amount of data that needs to be processed, they receive acknowledgement that the data has been received and instructed to check an import queue to view the status. Or, a message is sent upon import completion.

This leads to the notion of **Message Queue** which can offload processing and free up resources so that the application can handle other requests.

In this demo, I will use RabbitMQ to handle API requests.

Step 1: Get RabbitMQ ready

You can either register an account from CloudAMQP or install the service from your local machine from RabbitMQ.

After creating an account and an instance from CloudAMQP, you can click on the instance and view the cloud hosted RabbitMQ information.

automatically run on your machine. And you can access it through **'amqp://localhost'** URL in your app.

Step 2: Implement Message Queue to RESTful API

In this example, I suppose that we will have a long processing POST request that may take few minutes to accomplish.

Before doing anything, we need to install amqp.node package

```
npm install amqplib
```

Then, inside your route, you need to import the package

```
// For TypeScript
import * as amqp from 'amqplib/callback_api';

// For JavaScript
var amqp = require('amqplib/callback_api');
```

Then edit the POST route functions. For example:

```
app.route('/messages).post(req: Request, res: Response) {

 amqp.connect('amqp://localhost', (err, conn) => {
  conn.createChannel((err, ch) => {
   const q = 'hello';
   ch.assertQueue(q, {durable: false});

   // I suppose the process will take about 5 seconds to finish
   setTimeout(() => {
    let msg = 'Get data from message queue!';
    ch.sendToQueue(q, new Buffer(msg));
    console.log(` [X] Send ${msg}`);
   }, 5000)
  });
```

Now, our API will listen to a POST request. For example from : https://localhost:3000/messages

For testing the configuration, I will use request package for making POST request from client side.

```
// receive.js
var amqp = require('amqplib/callback_api');
var request = require('request');

request.post('https://localhost:3000/messages', function (error,
response, body) {

    console.log(response.body);

    // Connect to the server and wait for the queue
    amqp.connect('amqp://localhost', (err, conn) => {
        conn.createChannel((err, ch) => {
            var q = 'hello';

ch.assertQueue(q, {
                durable: false
            });
            console.log(' [*] Waiting for messages in %s. To exit
press CTRL+C', q);
            ch.consume(q, msg => {
                console.log(' [x] Received %s', msg.content);
                conn.close();
            }, {
                noAck: true
            });
        });
    })

});
```

Then I run the test server and making sample POST request.

queue!", and the POST request also get the message and print out from console log.

Although this is just a test project, you now know a method to handle long running process when call the RESTful API application.

References:

- Messaging with RabbitMQ in Node.js
- RabbitMQ tutorial

Nodejs Message Queue API Restful Api Api Development