10 things not to do when running Node.js application

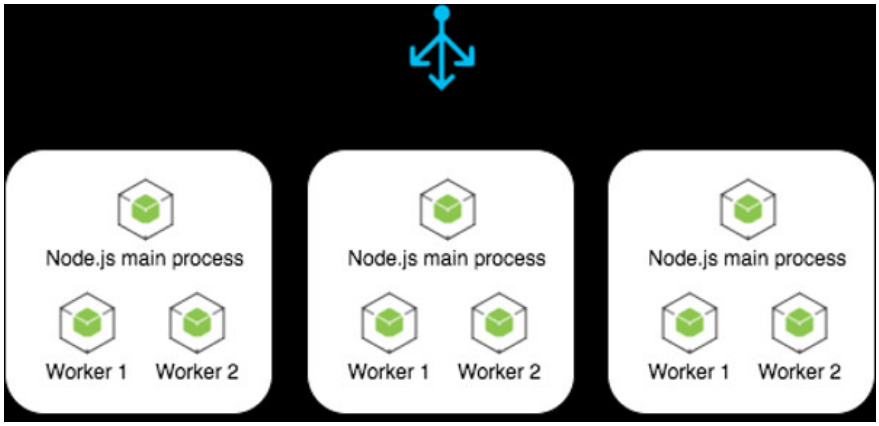
Let's TipsMake.com list 10 things you should not do when running the Node.js application in this article offline!

1. Arrays and objects in JavaScript are like stories and newspapers!
2. 10 useful tips for new programmers
3. Top 10 basic network troubleshooting tools that IT people need to know

In Hashnode, most of us often use Node.js. I personally am a big fan of Node.js and have learned a lot during the time of managing Hashnode. When I interacted with other developers, I realized that many of them didn't make the most of Node and made certain things in the wrong ways. Therefore, this article will tell you about what not to do when running a Node application in practice. Let's start with TipsMake.com to list **10 things you should not do when running Node.js application** !

1. Do not use Node.js Cluster

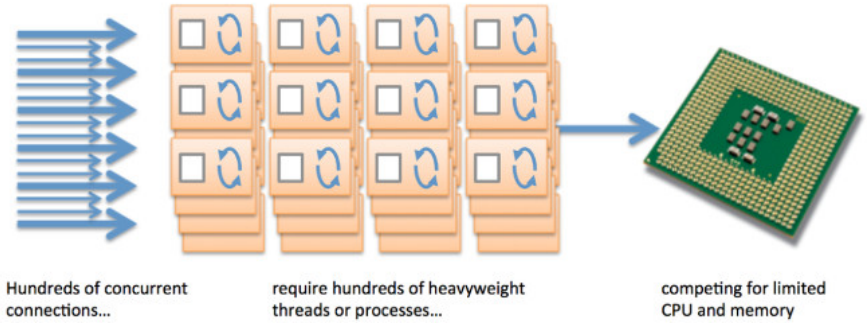
Node.js is a unique stream in nature and 1.5GB memory limit. Therefore, it cannot automatically take advantage of many CPU cores. The good news is that the Cluster module allows you to create multiple sub-processes using IPC to communicate with the parent process. The overall process of controlling employees (also known as child processes) and all connections are distributed in a round robin rotation (round-robin).



Clustering performance enhancements allow you to reduce downtime when deploying to 0 easily. Besides, the number of child processes can be created without being limited by the number of CPU cores of the machine.

Personally, the author of the article, found clustering to be a must for any Node.js application running in practice and there is no reason not to apply it.

2. Handle heavy tasks on Web Server



Node / Express server is not suitable to handle heavy tasks and requires a lot of computing. For example, a simple web application needs to send a series of emails to users. Although you can do this in Node.js web server, it will significantly affect performance. It's better to separate these heavy-duty tasks into **micro services** (architecture of many small services) and deploy them into separate Node applications. Furthermore, you can use the **message queue** (the immobile communications communication model, the exchange between the sender and the receiver does not happen simultaneously at the same time) like **RabbitMQ** to communicate between micro architectures. services.

When you post on Hashnode and attach it to keywords, we insert articles into the feeds of thousands of users. This is a heavy task. Just a few months ago, the job of inserting news articles into the news site was still done by the web server itself. As the traffic to the site increased, we found that growth inhibition appeared and congestion occurred. Then, if you post an article and add "General Programming" or "JavaScript" keywords, the whole site will be stalled for a few seconds. When looking carefully, the problem lies in the task of pushing the article into a user's news site. The main solution is to move the entire module to handle that task to another machine and start this task via the message queue.

Therefore, it is important to remember that Node.js is suitable for handling events and non-blocking I / O. Any task that can cause time to complete should be handled by a separate process.

3. Do not use process management tools

Obviously, the process management tool has many benefits, but most developers first deploy their applications in fact do not use it. In Hashnode, we use **pm2** , a powerful management tool for Node.js application.

Moreover, if using pm2 you can easily start running your application in `cluster` mode.

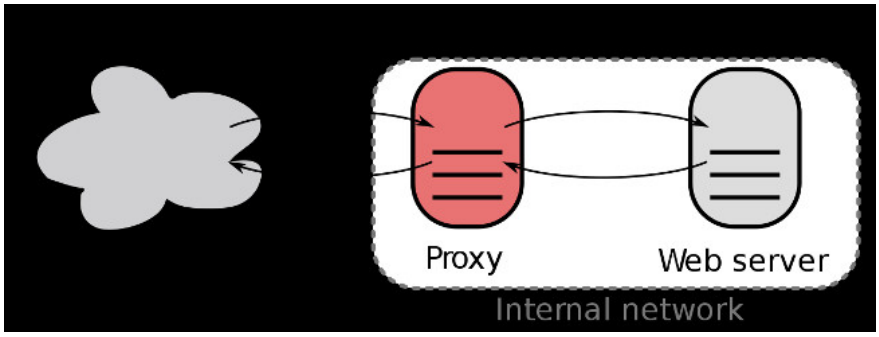
```
pm2 start app.js -i 2
```

In the command in the above example, `i` is the number of child processes that we want to create in cluster mode. The best thing is that you can run each child process in turn without affecting each other. Therefore, the application does not have to waste time on deployment. The following command will run the application again:

```
pm2 reload app
```

If you use pm2, refer to Keymetrics, a Node.js application monitoring service (based on pm2).

4. Do not use reverse proxy



I have seen programmers run the Node application on port 80 and provide static resources straight through this port. You should remember that running the Node application on port 80 is not a good idea and is actually dangerous in most cases. Instead, you should run on another port like port `3000` and use **nginx** (or something like HAProxy?) As a reverse proxy before the Node.js application.

The above configuration protects your application from direct contact with Internet traffic, server expansion and easier load balancing.

5. Lack of supervision

Bad things like unexpected errors and frequent exceptions. Do you know what's worse? It is not known what error occurred in the system. When using the process management tool, an unprocessed error will occur every time the process is restarted. So unless you check the logs, you will know which problems have occurred. The solution for this is to use monitoring services and notify via email or sms whenever your process is "stopped" and restarted.

6. Do not delete console.log

When developing applications, we use `console.log` to check everything. However, sometimes we forget to delete these commands which consume CPU time and resources. The best way to avoid this is to use a `debug` module. Therefore, unless you run the application in the `DEBUG` environment, console.log statements will be printed.

7. Maintain Global status within web Node processes

Sometimes, programmers store values ??of session ids, socket connections, . in memory. This is absolutely not recommended and should be avoided at any cost. If you save the session ids in memory, you will see the user logged out every time the server restarts. Moreover, this also leads to difficulties and problems when extending applications, increasing the servers. Your web server only focuses on processing requests, accessing the web and not storing any information in memory.

8. Do not use SSL

With a website for users, there is no reason not to use SSL by default. Sometimes, I see programmers who read SSL keys from a file and use it in the Node process. You should use reverse proxy before Node.js application and install SSL in the reverse proxy.

In addition, regularly check the latest weaknesses of SSL and apply the fastest processing possible.

9. Lack of basic security measures

Security is always important and users worry about security is a good thing. Besides basic security checks, you should use things like NSPs to detect weaknesses in your project.

Don't use old versions of Node and Express. Immediately remove versions that are no longer maintained and upgrade to security.

10. Do not use VPN

Always download your application on a private network so that only trusted customers can communicate with your server. People often forget this simple thing when implementing and encounter many problems later. Before downloading the application, you need to think about the architecture and infrastructure.

For example, if your Node server is running on port `8080` and has configured nginx as revers proxy, it is important to make sure that only nginx can communicate with the server at this port. This port should be

completely isolated and not accessed by other things on the network.

Above, I have listed a list of **10 things you should not do while running Node.js.** I will continue to update more information if any. What is the checklist you follow in the Node.js deployment? Let me know in the comment section below.

Author: Sandeep Panda

Refer to some more articles:

1. 12 extremely useful tricks for JavaScript programmers
2. Journey to change jobs from fashion models to software engineers within 1 year
3. Top 6 outsourcing trends will change IT industry by 2018

Having fun!