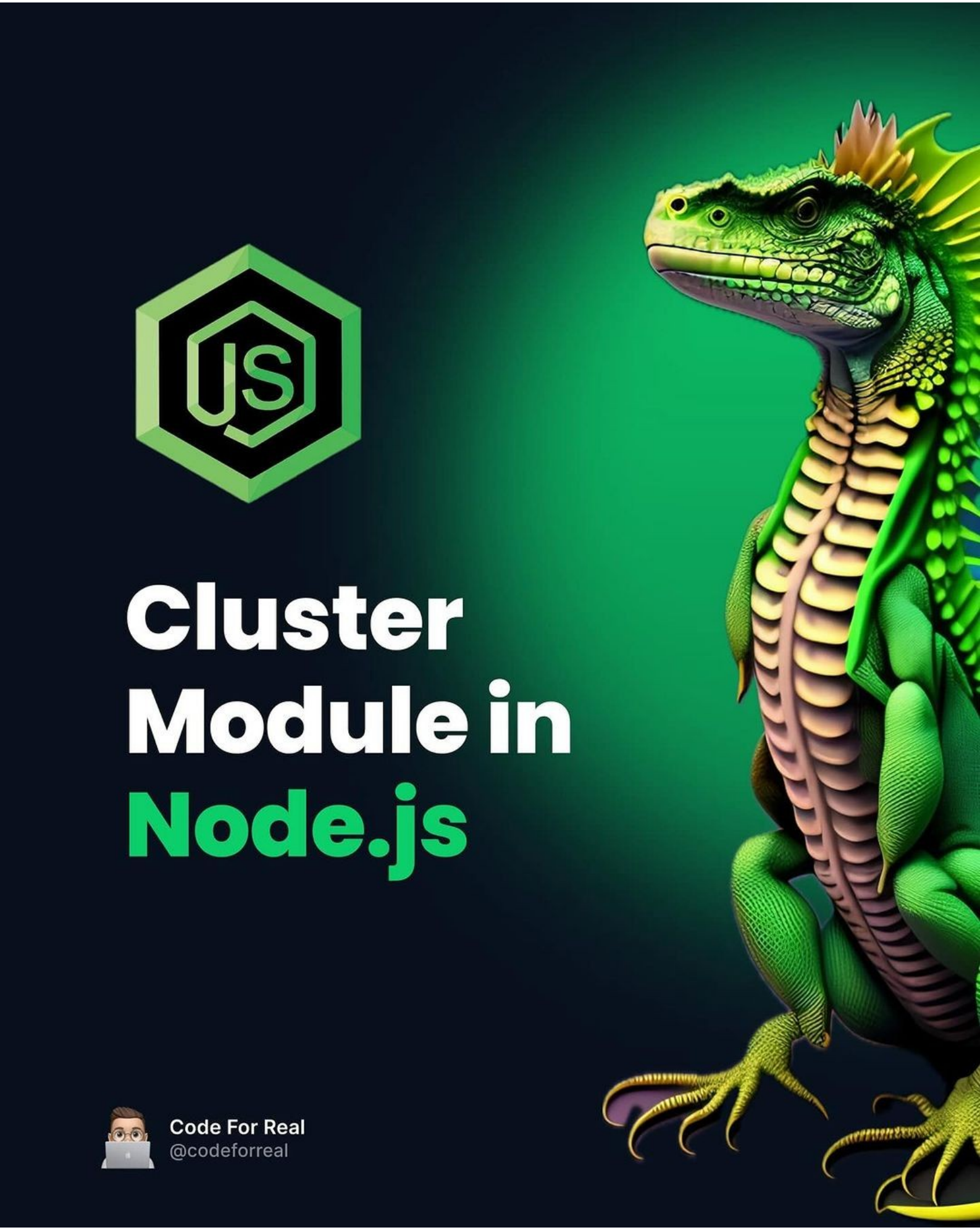
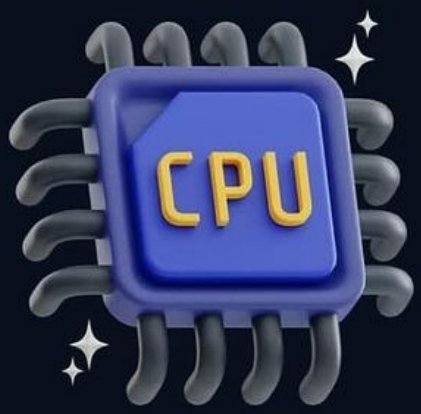

Cluster
Module in
Node.js
Code For Real
@codeforreal


Code For Real
@codeforreal
Give a follow
What is cluster module & how do we use it in Node.js?
Dont' forget to save it
Save



In **Node.js**, the **cluster module** is used to create child processes (workers) that share the same server port. This allows you to take advantage of multi-core CPUs by distributing the workload among multiple processes. Each child process operates independently and can handle incoming requests, improving the overall performance and concurrency of your application.

The cluster module is especially useful for applications that need to handle a high number of concurrent connections, such as web servers. It helps in utilizing the full capacity of your server's CPU cores and provides better scalability and responsiveness.

Let's dive into a detailed explanation with code examples:

Code For Real
@codeforreal

Save

```
const cluster = require('cluster');
const http = require('http');
const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
  // Fork workers for each CPU core
  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }

  cluster.on('exit', (worker, code, signal) => {
    console.log(`Worker ${worker.process.pid} died`);
    // Restart the worker if it exits
    cluster.fork();
  });
} else {
  // Each worker process runs this code

  const server = http.createServer((req, res) => {
    res.writeHead(200);
    res.end('Hello, Node.js Cluster!');
  });

  server.listen(8000, () => {
    console.log(`Worker ${process.pid} listening on port 8000`);
  });
}
```

Code For Real
@codeforreal

Save

What's going on here?

Let's go over the code step by step:

1. The cluster.isMaster condition is used to determine whether the current process is the master process or a worker process.

2. If the process is the master, it forks workers equal to the number of CPU cores available on the system. Each worker will have its own process.

3. If a worker process exits (dies), the master process detects this event using the cluster.on('exit', ...) event listener. It then restarts a new worker to replace the one that exited.

4. If the process is a worker, it creates an HTTP server that listens on a port. Each worker can handle incoming requests independently.

5. The workers share the same port (8000 in this case) because they are spawned from the same master process. The operating system takes care of distributing incoming connections among the worker processes.

Code For Real
@codeforreal

bash

```
> node ./cluster-example.js
Worker 34949 listening on port 8000
Worker 34950 listening on port 8000
Worker 34952 listening on port 8000
Worker 34951 listening on port 8000
Worker 34954 listening on port 8000
Worker 34953 listening on port 8000
Worker 34955 listening on port 8000
Worker 34956 listening on port 8000
```

8 CPU cores are forked and are listening to the same port 8000.

Code For Real
@codeforreal

Save

Remember that while the cluster module can greatly improve the performance of your application, it might not be suitable for all use cases. Applications that are I/O-bound (such as those that perform a lot of file I/O or network requests) might not benefit as much from clustering as applications that are CPU-bound. Additionally, handling shared state and communication between worker processes can introduce complexities that need to be managed carefully.

Code For Real
@codeforreal