

How to Manage Node Applications with PM2

Learn how to effectively manage Node.js applications using PM2, including installation, basic commands, monitoring, and best practices for production deployment.

Contents

01	Introduction	3
02	Prerequisites	3
03	Install PM2 Using NPM	3
04	Create an Example Node.js Application	3
05	Start the Application Using PM2	5
06	Manage the Application	6
07	List All Applications	6
08	Display Logs	7
09	Monitor the Resource Usage	7
010	Set up a Web-Based Dashboard	7
011	Configure PM2 to Start on Server Boot	9
012	Configure Applications to Start on Server Boot	10
013	Check for Updates	11
014	Update PM2	11
015	Best Practices and Security Considerations	11
016	Conclusion	13

Introduction

Node.js is an open-source JavaScript runtime that allows you to run JavaScript code outside the browser. Built on Chrome's V8 engine, it is known for its speed and scalability, making it ideal for real-time and high-performance applications. Running Node applications in production faces unique challenges — you need to handle unexpected crashes, manage processes, and ensure continuous uptime. PM2 is a Production Process Manager for Node.js applications that simplifies application management, allowing you to monitor, restart, and optimize your applications.

This article explains how to manage Node Applications with PM2, explore its essential features, and follow best practices for managing your Node applications in a production environment.

Prerequisites

- Have access to your workstation as the non-root sudo user running [macOS](#), [Linux](#), or [Windows](#) with Node and NPM installed.

Install PM2 Using NPM

- Install PM2 via the NPM package manager.

CONSOLE

```
$ sudo npm install pm2 -g
```

Create an Example Node.js Application

Follow the steps below to create an example Node.js application.

1. Navigate to the user's home directory.

```
CONSOLE
```

```
$ cd ~
```

2. Create a file named `app.js` using a text editor such as `vim`.

```
CONSOLE
```

```
$ vim app.js
```

3. Add the following code to the file.

```
JAVASCRIPT
```

```
const http = require('http');

const hostname = '0.0.0.0';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Save and exit the file.

The above `app.js` file is an example Node.js web application that listens on port 3000 and when it receives an HTTP request, it sends back a response with message "Hello World".

Start the Application Using PM2

- Start your Node.js application with PM2. Replace `app.js` with your application's entry file if needed.

CONSOLE

```
$ pm2 start app.js
```

Your output should be similar to the one below:

```
[PM2] Starting /home/linuxuser/app.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↻	status	cpu	memory
0	app	fork	0	online	0%	4.4mb

Below are some additional parameters to manage your application:

- `--name <app_name>` - specify an application name
- `--watch` - restart your application automatically when a file changes
- `--max-memory-restart <memory>` - a memory limit that triggers an automatic restart when reached
- `--log <log_file>` - specify a log file
- `--restart-delay <delay>` - delay between automatic restarts in milliseconds
- `--time` - prefix logs with time
- `--no-autorestart` - turn off automatic application restarts
- `--cron` - a cron job pattern for automatic application restarts
- `--no-daemon` - do not run the application in a daemon process

Visit the [official documentation](#) to view all available parameters.

Manage the Application

Manage your Node.js application using PM2 and perform actions like restart, reload, stop and delete PM2 processes.

- Restart your PM2-managed application.

CONSOLE

```
$ pm2 restart app
```

- Reload your PM2-managed application.

CONSOLE

```
$ pm2 reload app
```

- Stop your PM2-managed application.

CONSOLE

```
$ pm2 stop app
```

- Delete your PM2-managed application.

CONSOLE

```
$ pm2 delete app
```

List All Applications

- List all the applications that are managed by PM2.

CONSOLE

```
$ pm2 list
```

Display Logs

- View the logs produced by your application in real-time.

```
CONSOLE
```

```
$ pm2 logs
```

- View older logs, by specifying the number of lines using `--lines` flag.

```
CONSOLE
```

```
$ pm2 logs --lines 200
```

Monitor the Resource Usage

- Monitor the amount of resources consumed by your applications in near real time.

```
CONSOLE
```

```
$ pm2 monit
```

Set up a Web-Based Dashboard

- Configure a web-based PM2 dashboard.

```
CONSOLE
```

```
$ pm2 plus
```

Follow the prompts to create or log in to your PM2 account. If you don't have an account, complete the required information to create one and accept the license agreement.

You'll see the following output during the setup process:

```
[PM2 I/0] Do you have a pm2.io account? (y/n) n
[PM2 I/0] No problem! We just need a few details to create your account.
[PM2 I/0] Please choose a username: vultrexampleuser
[PM2 I/0] Please choose an email: admin@example.com
[PM2 I/0] Please choose a password: *****
...
[PM2 I/0] Do you accept the terms and privacy policy (https://pm2.io/legals/terms_conditions.pdf)? (y/n) y
[PM2 I/0] Successfully authenticated
[PM2 I/0] Successfully validated
[+] PM2+ activated!
[PM2 I/0] Successfully connected to bucket PM2 Plus Monitoring
[PM2 I/0] You can use the web interface here: https://app.pm2.io/#/bucket/67cb27ccd623d57f6d4c63c0
```

Replace `vultrexampleuser` with your preferred username, `admin@example.com` with your log in email address and provide a strong password to secure your PM2 account.

Next Steps:

1. Open a web browser and visit the provided web address.
2. Log in using your PM2 account credentials.
3. Start monitoring and managing your applications through the PM2 Plus dashboard.

Note

PM2 Plus is a subscription-based offering that incurs a cost, depending on the number of processes you want to monitor. You can review the pricing details on the PM2 Plus page.

Configure PM2 to Start on Server Boot

1. Generate a startup script and configure PM2 to start on server boot.

CONSOLE

```
$ pm2 startup
```

Your output should be similar to the one below:

```
[PM2] To setup the Startup Script, copy/paste the following command:  
sudo env PATH=$PATH:/home/linuxuser/.npm/versions/node/v22.14.0/bin /home/  
linuxuser/.npm/versions/node/v22.14.0/lib/node_modules/pm2/bin/pm2 startup  
systemd -u linuxuser --hp /home/linuxuser
```

2. Copy the command from previous command output and execute it to make the PM2 process persistent.

CONSOLE

```
$ sudo env PATH=$PATH:/home/linuxuser/.npm/versions/node/  
v22.14.0/bin /home/linuxuser/.npm/versions/node/v22.14.0/lib/  
node_modules/pm2/bin/pm2 startup systemd -u linuxuser --hp /  
home/linuxuser
```

Your output should be similar to the one below:

```
...  
[PM2] Writing init configuration in /etc/systemd/system/pm2-linuxuser.service  
Created symlink /etc/systemd/system/multi-user.target.wants/pm2-linuxuser.service  
→ /etc/systemd/system/pm2-linuxuser.service.  
...
```

Configure Applications to Start on Server Boot

Boot

Configure your PM2 applications to start automatically on server boot by saving them into an application list.

- Save the running applications to the list.

CONSOLE

```
$ pm2 save
```

Your output should be similar to the one below:

```
[PM2] Saving current process list...  
[PM2] Successfully saved in /home/linuxuser/.pm2/dump.pm2
```

- If you need to restore the saved applications manually from the list, use the below command.

CONSOLE

```
$ pm2 resurrect
```

Your output should be similar to the one below:

```
[PM2] Resurrecting  
[PM2] Restoring processes located in /home/linuxuser/.pm2/dump.pm2  
[PM2] Process /home/linuxuser/app.js restored
```

id	name	mode	♾	status	cpu	memory
0	app	fork	0	online	0%	21.3mb

Check for Updates

- Print a list of outdated NPM packages on your system.

CONSOLE

```
$ npm outdated -g
```

This command displays a list of globally installed NPM packages that have newer versions available.

Update PM2

1. Update the PM2 package via NPM. The update stops all your running PM2 processes until upgrade is completed and your applications will face downtime during this period.

CONSOLE

```
$ npm install pm2@latest -g
```

2. Perform an in-memory update of PM2.

CONSOLE

```
$ pm2 update
```

Best Practices and Security Considerations

Prevent Downtime With Cluster Mode

Your PM2 processes may restart multiple times during their lifecycle, causing temporary downtime. Use PM2's cluster mode to reduce downtime. Cluster

mode runs multiple instances of your application. When a restart is required, only one instance stops and restarts at a time, while other instances continue to serve your application.

1. Delete the application before switching to cluster mode.

CONSOLE

```
$ pm2 delete app
```

2. Start the application in cluster mode, specifying the number of instances with the `-i` parameter.

CONSOLE

```
$ pm2 start app.js -i 3
```

This command starts three instances of your application in cluster mode using PM2.

3. Scale the application by specifying the new number of instances.

CONSOLE

```
$ pm2 scale app 4
```

This command will scale up your application to have **4** instances of `app` application.

Eliminate Failed Requests With Graceful Shutdown

PM2 ensures graceful shutdowns by sending signals to your application so it can terminate cleanly. This process helps prevent data loss or corruption by allowing your app to finish active tasks before exiting.

- PM2 handles graceful shutdowns as follows:
 1. Before shutdown, PM2 sends a `SIGINT` signal to your application.
 2. Your application intercepts the signal.

3. It finishes active requests and closes open resources like database connections.
4. The application exits.

Add the following code to enable graceful shutdown in your application:

JAVASCRIPT

```
process.on('SIGINT', function() {
  // sigint signal received, log a message to the console
  console.log('Shutdown signal intercepted');
  // close the http server
  server.close();
  // exit the process
  process.exit();
});
```

The above code enables graceful shutdown in your application. It intercepts the `SIGINT` signal, logs a message to the console, closes the HTTP server, and exits the process.

Conclusion

In this article, you learned how to install and use PM2 to manage Node.js applications in a production environment. You configured PM2 to start applications automatically on system boot, monitored resource usage, viewed logs in real-time, and explored web-based dashboard capabilities with PM2 Plus. By following these steps, you can keep your Node.js applications running reliably, recover from crashes automatically, and simplify operational tasks using PM2's robust feature set. For more information please visit the [official documentation](#).



VULTR

