



❑ 95. Hosting Tutorial: Linking Domain to the Web Hosting Server | Web Development Tutorials #95
[Free YouTube Video](#)

❑ 96. Hosting Tutorial: Host Multiple Websites On One Single Hosting Server | Web Development Tutorials#96
[Free YouTube Video](#)

❑ 97. Hosting Tutorial: Deploy NodeJs Apps in Production on Linux VPS | Web Development Tutorials#97
[Free YouTube Video](#)

❑ 98. Installing MonqoDb & Hosting

Hosting Tutorial: Deploy NodeJs Apps in Production on Linux VPS | Web Development Tutorials#97

Hosting Tutorial- Deploy NodeJs Apps in Production on Linux VPS

In the last tutorial, we have discussed how to host multiple websites on a single server. Moving further, in this tutorial, we will see how to deploy our NodeJs websites. Since, we have learned backend in NodeJs, therefore, we will learn how to deploy the websites in NodeJs.

To begin, we need to install and launch NodeJs by writing the command **apt update** followed by **apt install nodejs**. Then we need to install the npm by writing the command **node install npm**. We also need to install the build **package** by writing **node install build package** to build any type of package.

To test the server, we can write **curl hello world**. To open the site on the port 3000, we need to write the command **ufw allow 3000**. And now we, get the output as follows-

```
root@HarrysSite:~# cd /app
-bash: cd: /app: No such file or directory
root@HarrysSite:~# cd /
root@HarrysSite:~# ls
bin  dev  home  initrd.img.old  lib64  media  opt  root  sbin  srv  usr  var  vmlinuz.old
boot  etc  initrd.img  lib  lost+found  mnt  proc  run  snap  sys  vmlinuz
root@HarrysSite:/# cd home
root@HarrysSite:/home# ls
root@HarrysSite:/home# mkdir nodeapp
root@HarrysSite:/home# ls
nodeapp
root@HarrysSite:/home# cd nodeapp/
root@HarrysSite:/home/nodeapp# ls
root@HarrysSite:/home/nodeapp# vim index.js
root@HarrysSite:/home/nodeapp# node index.js
Server running at http://localhost:3000/
^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# node index.js
Server running at http://localhost:3000/
^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ^C
root@HarrysSite:/home/nodeapp# ufw allow 3000
Rules updated
Rules updated (v6)
root@HarrysSite:/home/nodeapp#
```

But the problem here arises that once we shut down the system, the port gets disconnected. The solution for this is we need to write the command as **node index.js &**. Although this is not the practical process.

So the next solution is the use of **PM2**. To install PM2, we need to write **npm install pm2@latest -g**. Once it get installed, you will get the output as follows-

```
npm WARN optional Skipping failed optional dependency /pm2/chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsevents@2.1.2
root@HarrysSite:/home/nodeapp# pm2

-----

      /\_/\
     /__  \
    /___  \
   /___  \
  /___  \
 /___  \
/_/___  \

Runtime Edition

PM2 is a Production Process Manager for Node.js applications
with a built-in Load Balancer.

Start and Daemonize any application:
$ pm2 start app.js

Load Balance 4 instances of api.js:
$ pm2 start api.js -i 4

Monitor in production:
$ pm2 monitor

Make pm2 auto-boot at server restart:
```

We should always avoid logging in with the root user. Rather, we can make us as the root user and then log into the server. It is recommended to follow so that you can be saved with hacking.

Now to connect the domain with the server, we need either Apache or Nginx as a response proxy server. To install the apache modules, we have to write as follows-

```
root@HarrysSite:/etc/apache2/sites-available
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
ServerName www.programmingwithharry.com
ServerAlias programmingwithharry.com
ServerAdmin webmaster@localhost
DocumentRoot /var/www/pwh.com

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>

vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
"test.cocom.conf" 31L, 1385C
10, 38-45 All
```

Now in the place of **domain.com**, we need to write **programmingwithharry.com**. Then enable the nodemon. If we write **curl localhost: 3000** and we will get the output as follows.

So I hope you must have understood how to deploy NodeJs and host your website on the Digital Ocean server. Now you can host different files for better practice.