GNU/Linux Administration - Support #393

How To Use PM2 to Setup a Node.js Production Environment

05/20/2014 08:33 PM - Daniel Curtis

Status:	Closed	**Start date:**	05/20/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	100%
Category:	Node.js	**Estimated time:**	0.20 hour
Target version:		**Spent time:**	0.50 hour

Description

This tutorial aims to help you setup up aDebian server to run Node.js applications, including apps based on Express, Geddy, or Sails. These instructions will help you avoid some security mistakes, as well as provide some surprising benefits such as:

- You will not run your app as root; therefore, your app will be more secure.
- You will be using port 80 to run your app, which typically can only be accessed by the root user. (You will able to run your app using a custom URL such as http://mysite.com - but you will not have to specify a port.)
- Your application will restart if it crashes, and it will keep a log of unhandled exceptions.
- Your application will restart when the server starts - i.e. it will run as a service.

Create a Safe Account to Run Your Code

Start by logging into the server:

```
ssh root@192.168.20.9
```

As most of us understand, if you run your code using the root account, and if a hostile party compromises the code, that party could get total control of your VPS. To avoid this, let's setup a safe account that can still perform root operations if we supply the appropriate password. For the purposes of this tutorial, let's call our safe user "safeuser" – you can name it whatever you like. For now, log on as the root user and follow these steps:

1. Create the user with a folder in /home/safeuser/:

   ```
   useradd -s /bin/bash -m -d /home/safeuser -c "safe user" safeuser
   ```

2. Create a password for safeuser - you will be asked to type it twice after you enter the following command:

   ```
   passwd safeuser
   ```

3. Give the safe user permission to use root level commands:

   ```
   usermod -aG sudo safeuser
   ```

Login as the Safe User

Log out of your root session by typing:

```
exit
```

Please note that the command to log on as the safe user is the same command you used before, but the user name has changed. Once you have logged on as the safe user, every time you want to run a command that has root privileges, you are going to have to proceed the command with the word sudo. From the command line on your own machine, log on using the command that appears below.

```
ssh safeuser@192.241.xxx.xxx
```

Install GIT

One you have logged on, install GIT (we are going to use GIT to install Node.js.). Installing it on Debian is easy:

```
sudo apt-get install git
```

The command sudo indicates that you want to run this command as root. You will be prompted for your password - i.e. the safeuser password. When you provide your password, the command will run.

Install Latest Node.JS

Please note that v0.10.24 is the most recent version of Node as of this writing. If there is a newer version, please use that version number instead:

```
sudo apt-get install build-essential
sudo apt-get install curl openssl libssl-dev
git clone https://github.com/joyent/node.git
cd node
git checkout v0.10.24
./configure
make
sudo make install
```

When you type sudo make, a lot of things are going to happen. Be patient.

When the make install command finishes, make sure all went well by typing:

```
node -v
```

If all went well, you should see:

    v0.10.24.

Give Safe User Permission To Use Port 80

Remember, we do NOT want to run your applications as the root user, but there is a hitch: your safe user does not have permission to use the default HTTP port (80). You goal is to be able to publish a website that visitors can use by navigating to an easy to use URL like http://mysite.com.

Unfortunately, unless you sign on as root, you'll normally have to use a URL like http://mysite.com:3000 - notice the port number.

A lot of people get stuck here, but the solution is easy. There a few options but this is the one I like. Type the following commands:

```
sudo apt-get install libcap2-bin
sudo setcap cap_net_bind_service=+ep /usr/local/bin/node
```

Now, when you tell a Node application that you want it to run on port 80, it will not complain.
Use NPM To Install A Package Called PM2.

NPM is a package manager that you will use to install frameworks and libraries to use with your Node.js applications. NPM was installed with Node.js. PM2 is a sweet little tool that is going to solve two problems for you:

- It is going to keep your site up by restarting the application if it crashes. These crashes should NOT happen, but it is good know that PM2 has your back. (Some people may be aware of Forever.js, another tool that is used to keep node based sites running - I think you will find that PM2 has a lot to offer.)
- It is going to help you by restarting your node application as a service every time you restart the server. Some of use know of other ways to do this, but pm2 makes it easier, and it has some added flexibility.

Install PM2

Install PM2 by typing the following at the command line:

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

Create a Simple Node App

This is where you can test your environment to be sure everything is working as it should. In this example, I will use the IP address, but your goal should be to use a domain name. Look at these instructions later: How To Set Up a Host Name with DigitalOcean

First, create a simple node app just for testing. At the command line type:

```
nano app.js
```

Then enter the following lines of code into the nano editor:

```
var http = require('http');
var server = http.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.end("Hello World\n");
});

server.listen(80);
console.log("Server running at http://127.0.0.1:80/");
```

Press ctrl-X to exit - when nano asks if you want to save, answer yes.

Now you have a node based application called app.js that you can use to test your environment.

You can run app.js at the command line by typing:

```
node app.js
```

Do that, and you should be able to see your hello world text by using a browser and typing your IP address as the URL. You can interrupt execution by pressing crtl-C. This is NOT how we want to run our application. We will use PM2 to run it instead of using Node directly.

Run your app using PM2, and ensure that your node.js application starts automatically when your server restarts

There are some huge benefits for you if you run your application using pm2. Instead of running your app as above, run it using the following command:

```
pm2 start app.js
```

What are the advantages of running your application this way?

- PM2 will automatically restart your application if it crashes.
- PM2 will keep a log of your unhandled exceptions - in this case, in a file at /home/safeuser/.pm2/logs/app-err.log.
- With one command, PM2 can ensure that any applications it manages restart when the server reboots. Basically, your node application will start as a service.

Run this command to run your application as a service by typing the following:
sudo env PATH=$PATH:/usr/local/bin pm2 startup -u safeuser debian

Please note, you may not be using safeuser as the user name - use the name that corresponds to your setup. You should see the following report:

> Adding system startup for /etc/init.d/pm2-init.sh ...
> /etc/rc0.d/K20pm2-init.sh -> ../init.d/pm2-init.sh
> /etc/rc1.d/K20pm2-init.sh -> ../init.d/pm2-init.sh
> /etc/rc6.d/K20pm2-init.sh -> ../init.d/pm2-init.sh

/etc/rc2.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc3.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc4.d/S20pm2-init.sh -> ../init.d/pm2-init.sh
/etc/rc5.d/S20pm2-init.sh -> ../init.d/pm2-init.sh

After thought: You may notice a file folder called node in the safeuser directory. It was used during installation, but you no longer need it. You can delete it by typing the following:

```
rm -rf /home/safuser/node
```

Important Clarification: There is a startup script that starts your Node applications, but you will avoid a lot of confusion if you understand how it works. The script is called 'pm2-init.sh.' It lives in the 'etc/init.d/' directory, but it does NOT start app.js. Instead, it starts the programs that were running under PM2 the last time the server shutdown.

This is important. If your node application does not show up in the list when you type pm2 list, then your app will not restart when the server restarts. Follow the proper instructions for starting your apps using pm2 to ensure that they will restart:

```
pm2 start app.js
```

Resources

https://www.digitalocean.com/community/articles/how-to-use-pm2-to-setup-a-node-js-production-environment-on-an-ubuntu-vps

History

#1 - 05/25/2014 11:33 AM - Daniel Curtis

- Status changed from Resolved to Closed

- % Done changed from 90 to 100

#2 - 02/15/2015 09:10 PM - Daniel Curtis

- Project changed from 86 to GNU/Linux Administration

- Category set to Node.js