Deploying Node.js Applications with PM2

POSTED ON APR 15, 2015

Guest post: *Today's post will feature a tutorial for deploying Node.js apps on Digial Ocean by Raghuvir Kasturi, a multi-talented full stack engineer.*

Deploying Node.js applications is hard

Ask anyone who's jumped on the self-taught hacker bandwagon, gone through an online tutorial or two and then attempted to get their basic ToDo app to actually run on anything but localhost.

This tutorial walks through setting up a very basic push-to-deploy workflow using PM2, nginx & GitHub. This workflow has been tested on a Digital Ocean 14.04 box, but could be adapted for other systems with some tweaking.

What is PM2?

From the official repo:

> *PM2 is a production process manager for Node.js applications with a built-in load balancer.*

PM2 gives you an accessible and comprehensive toolkit to deploy and manage your Node.js applications in a production environment. While this tutorial focuses on the deployment arm of PM2, I would encourage you to spend some time looking through the other features; the CLI is extremely user-friendly.

What you'll need

You'll need the following (or some variants thereof):

- A VPS that you can SSH into (I use an Ubuntu 14.04 box provided by Digital Ocean).
- A GitHub account

I use a Mac as my personal computer, but everything going forward should work on any *NIX OS. If you use Windows you might have to Google around for hacks.

Initial setup

Node.js/npm

The first thing to do is make sure both your computer and your server have Node.js (and npm) installed. There are numerous ways to do this, depending on your OS and requirements. The easiest way to get it is via their website. If you need to install it via CLI – on your server – here's a good guide.

Once that's done, you'll need PM2 installed on both your personal computer and your server.

```

$ npm install -g pm2@latest

```

If you need to update PM2 on either your computer or the server, you can install the latest at any point using

```

$ npm install -g pm2@latest

```

and then run

```

$ pm2 updatePM2

```

This will update the in-memory PM2 to the latest installed version.

GitHub

For this tutorial, we'll be using GitHub as the DVCS of choice. This means your workflow will be something similar to this

- Work on your local copy of your code and test/run locally until you're satisfied.
- Once satisfied, push up to GitHub.

- Use PM2 to deploy your latest version from GitHub.

This means your server needs to have SSH access to GitHub in order to pull your code and restart the server with the changes. You'll need to create a new SSH key on your server and add the public key to GitHub. I've posted a guide on working with SSH keys that may be of help if you're unfamiliar with them.

nginx

To serve our new Node app, we're going to use nginx as the front-end server which will then proxy requests to our Node server. Nginx provides a lot of tools out of the box that we can use to avoid unnecessary stress on our Node server, such as GZIP encoding, static file serving, and HTTP caching.

Your server will need to have nginx installed, and the easiest way to do this is by using the OS's package manager. For example, on Ubuntu 14.04

```

$ sudo apt-get update
$ sudo apt-get install nginx

```

Make sure it's working by checking your IP or hostname in a browser. If all went well, you should see the default nginx landing page.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Once you have it installed, you'll need to set up the proxy configuration for your Node.js server.

A sample config file for your app looks something like this (and would site inside /etc/nginx/sites-available)

```

server {
    server_name your.domain.com;
    return 301 $scheme://domain.com$request_uri;
}

server {
  listen 80;

server_name domain.com;

client_max_body_size 10G;

location / {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_pass http://127.0.0.1:3000;
    proxy_redirect off;
    proxy_buffering off;
  }
}

```

You should change your proxy_pass to whatever host/port your Node.js server is running on.

Deploy

Now that you've got everything set up, you're ready to deploy! We're going to deploy a simple Express app.

app.js

```

var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hello, World!');
});

var server = app.listen(3000, function() {

console.log('Server listening at http://%s:%s', host, port);

});

```

PM2 uses a special file called **ecosystem.json** as its configuration file. Similar to npm, PM2 gives you a way to create one via the CLI.

```

$ pm2 ecosystem

```

This will generate a new ecosystem.json in your current folder. We'll modify it a bit to suit our (very basic) needs.

```

{

"apps" : [
    {
      "name"       : "basicApp",
      "script"     : "app.js",
      "env": {
        "COMMON_VARIABLE": "true"
      },
      “env_production” : {
        “NODE_ENV”: “production”
      }
    }

],

“deploy” : {
    “production” : {
      “user” : “node”,
      “host” : “238.141.2.56”,
      “port” : “5674”,
      “key” : “/Users/user/.ssh/production_server”,
      “ref”  : “origin/master”,
      “repo” : “git@github.com:user/repo.git”,
      “path” : “/var/www/production”,
      “post-deploy” : “npm install –production && pm2 startOrRestart ecosystem.json –env production”
    },
    “dev” : {
      “user” : “node”,
      “host” : “238.141.2.56”,
      “port” : “5674”,
      “key” : “/Users/user/.ssh/production_server”,
      “ref”  : “origin/master”,
      “repo” : “git@github.com:user/repo.git”,
      “path” : “/var/www/development”,
      “post-deploy” : “npm install –production && pm2 startOrRestart ecosystem.json –env dev”
    }
  }
}

```

This allows you to have multiple deployment options based on your needs (dev, staging, production), all of which are managed with a single ecosystem file.

Once this is done, initialise the folders on your server

```

$ pm2 deploy ecosystem.json <environment> setup

```

where environment is the deployment environment. It must be defined in your ecosystem.json for the setup to work. For example

```

$ pm2 deploy ecosystem.json production setup

```

```

$ pm2 deploy ecosystem.json production
```

```

```

This will deploy your code, install all dependencies, and run your Node.js server. You may have to restart nginx for the proxy to take effect, but if all has gone well, you should see your shiny new app if you navigate to your host or IP in a browser!

Closing thoughts

pm2 deploy

PM2's deploy allows you to revert changes, update, list all the deploy commits, and much more – giving you a lot of control if things break.

```

$ pm2 deploy <configuration_file> <environment> <command>

Commands:
    setup                    run remote setup commands
    update                   update deploy to the latest release
    revert [n]               revert to [n]th last deployment or 1
    curr[ent]                output current release commit
    prev[ious]               output previous release commit
    exec|run <cmd>           execute the given <cmd>
    list                     list previous deploy commits
    [ref]                    deploy to [ref], the "ref" setting, or latest
tag

```

ecosystem.json

If you get errors with the "post-deploy" saying it couldn't find the npm, node, or pm2 commands, you may have to supply absolute paths for the executables, or make sure your $PATH variable for the user running the Node app is set up correctly.

Happy deploying!

About Raghu

Raghuvir Kasturi is a software engineer with a background in physics, based out of Bangalore, India. Raghu is interested in designing and building scalable and modular systems that can be leveraged on the modern multi-device web via meaningful user experiences. His current focus is on high impact data visualisations and build & deployment strategies for web & mobile applications. You can find Raghu's writings here

Submit a Comment

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Submit Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Recent Posts

- A Tale of Hosting Horror at GoDaddy Managed WordPress
- Intro to Destructuring Objects (ES6)
- Installing Kaltura 10.x (Jupiter) on Microsoft Azure
- Deploying Node.js Applications with PM2
- Quickly Use Street Address with Google Maps in Angular