Node.js

In Production

20.01.2011

About

Contributor

Felix Geisendörfer

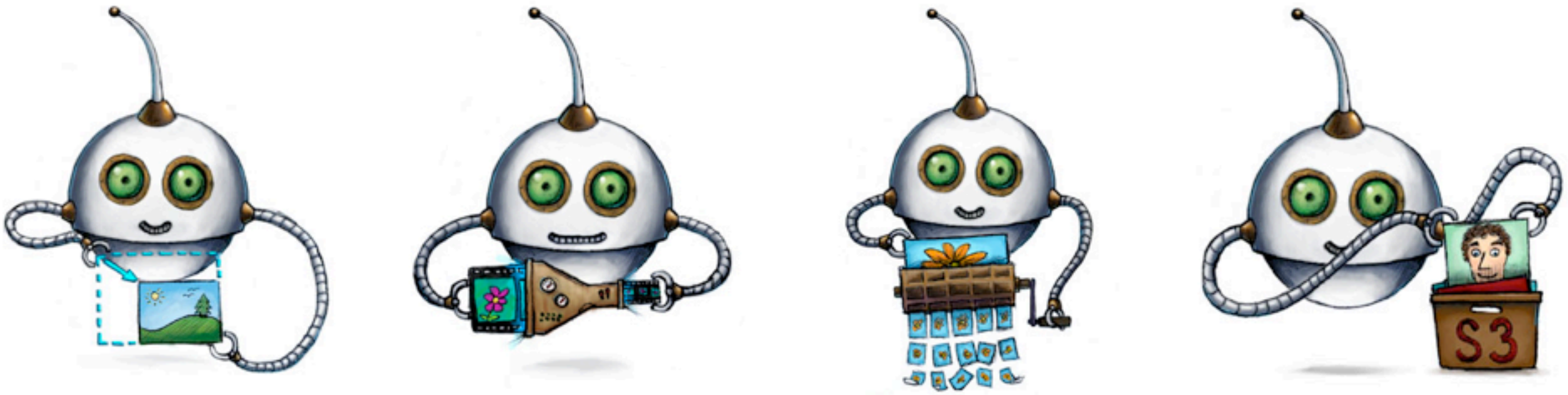
Co-founder

node.js driver

node-formidable

- Joined the mailing list in June 26, 2009
- Co-Transloadit
- When I joined there where #24 people
- First patch in September 2009
- Now mailing list has > 3400 members

Transloadit



File uploading & processing as a service for other web applications.

- The app we run in production
- Not a “typical” app, no html rendering, only JSON APIs

Are you running
node in production?

Our imagination

Our first attempt

~1 Year ago

Transloadit v1

- Complete failure

- Node randomly crashed with:

```
(evcom) recv() Success
```

- A mess of code, hard to maintain

* http://transloadit.com/blog/2010/04/to-launch-or-not

Today

Well, almost :)

Transloadit v2

Now

- Processed > 2 TB of data
- Not seen a node bug in production yet
- Clean code base, 100% test coverage

How to get there?

Hosting

Managed / Heroku-style

- Joyent (no.de)
- Nodejitsu (nodejitsu.com)
- Duostack (duostack.com)
- Nodefu (nodefu.com)
- Heroku (heroku.com)

Self-managed

- **Amazon Ec2**
- Linode
- Rackspace Cloud Servers
- etc.

Deployment

Deployment

- Heroku service
- Custom deploy script

Custom “deploy script”

```
$ git clone <my-project> /var/www/<my-project>
$ nohup bin/server.js &
```

Enough to get started

Staying alive

The problem

```
SyntaxError: Unexpected token ILLEGAL
    at Object.parse (native)
    at Server.<anonymous> (/path/to/my-script.js:4:8)
    at Server.emit (events.js:45:17)
    at HTTPParser.onIncoming (http.js:862:12)
    at HTTPParser.onHeadersComplete (http.js:85:31)
    at Socket.ondata (http.js:787:22)
    at Socket._onReadable (net.js:623:27)
    at IOWatcher.onReadable [as callback] (net.js:156:10)
```

Your node process is killed by any runtime error

The wrong solution

```
process.on('uncaughtException', function(err) {
  console.log('Something bad happened: '+err);
  // continue on ...
});
```

Continuing after an exception will leave your system in unpredictable state

The right solution

```
process.on('uncaughtException', function(err) {
  console.log('Something bad happened: '+err);
  // Every process has to die one day, it's life
  process.exit(0);
});
```

Exiting the program will prevent corrupted state

Even better

```
var Hoptoad = require('hoptoad-notifier').Hoptoad;
Hoptoad.key = 'YOUR_API_KEY';

process.on('uncaughtException', function(err) {
  console.log('Something bad happened: '+err);
  Hoptoad.notify(err, function() {
    process.exit(0);
  });
});
```

Send your exception to a logging service.

Your process is dead.

Now what?

Use a process monitor

- **Monit ***
- Upstart
- God (ruby)
- forever (node.js)

* http://kevin.vanzonneveld.net/techblog/article/run_nodejs_as_a_service_on_ubuntu_karmic/

Debugging

Debugging tools

- console.log
- node-inspector (use webkit inspector.)
- node debug <script.js> (new in 0.3.x)

Unit tests

(Real ones, not integration tests)

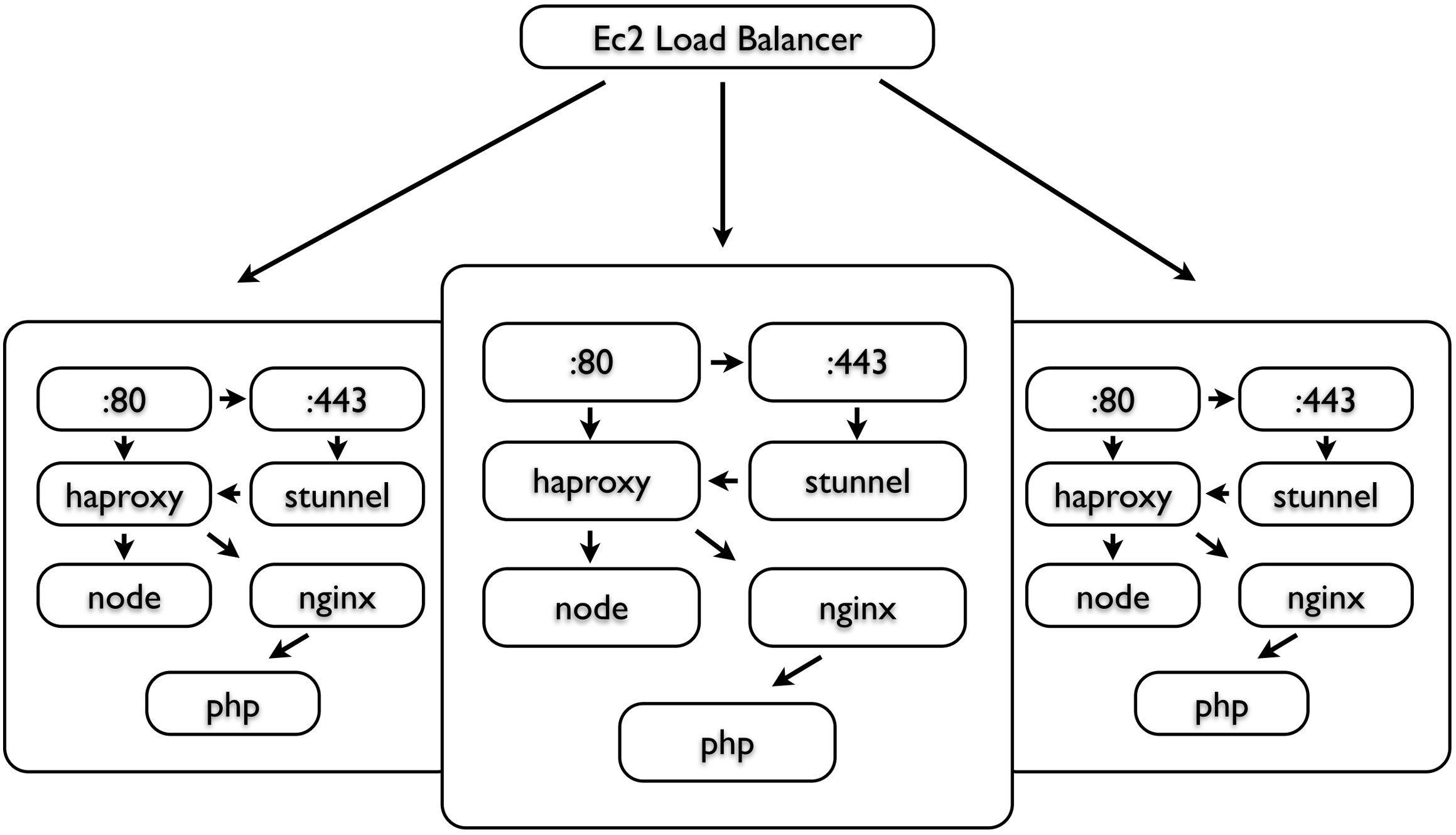
Load balancing

Load Balancing

- Haproxy
- **Amazon Load Balancer**
- Nginx (supports only http 1.0)

Our stack

Our stack


Ec2 Load Balancer
:80
:443
haproxy
stunnel
node
nginx
php
:80
:443
haproxy
stunnel
node
nginx
php
:80
:443
haproxy
stunnel
node
nginx
php

Workers

- Use workers a la Unicorn
- All workers listen on a shared socket
- Master process starts & kills workers

Workers

- spark
- fugue
- **custom**

master.js

```
var netBinding = process.binding('net');
var spawn = require('child_process').spawn;
var net = require('net');

var serverSocket = netBinding.socket('tcp4');
netBinding.bind(serverSocket, 8080);
netBinding.listen(serverSocket, 128);
```

```
for (var i = 0; i < 10; i++) {
  var fds = netBinding.socketpair();
  var child = spawn(process.argv[0], [__dirname+'/worker.js'], {
    customFds: [fds[0], 1, 2]
  });
  var socket = new net.Stream(fds[1], 'unix');
  socket.write('hi', 'utf8', serverSocket);
}
```

worker.js

```
var net = require('net');
var http = require('http');
var socket = new net.Socket(0, 'unix');

socket
  .on('fd', function(fd) {
    startServer(fd);
  })
  .resume();

function startServer(fd) {
  http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('Hello from: '+process.pid+'\n');
  }).listenFD(fd);
  console.log('Worker ready: '+process.pid);
}
```

Questions?

@felixge / felix@debuggable.com