Deploying Node.js with systemd
You can have the magic of the cloud too!

Ruben Vermeersch
Ruben Vermeersch

- Hacker: Web (~12yr), GNOME/Mono (~8yr), 2x GSOC, Mobile (mostly iOS)
- One year of PhD on scalable cloud deployments, ran away bored
- Co-founder of Flow Pilots
FINALLY. A PROFESSIONAL PARTNER FOR YOUR CORPORATE MOBILE NEEDS.

“WE NEED AN APP” IS NOT A MOBILE STRATEGY

At Flow Pilots, we believe that the smart use of mobile technologies will be a key driver in the success of modern companies. That is why we don’t just develop apps. We carefully design, build, deploy and manage the digital tools that help our customers increase revenues, decrease costs or bring freedom and flexibility to their employees.
Today

• How we deploy Node.js
• Why your own server is as good as a PAAS
• How it will get better

Linux! (your own server, EC2, ...)

Monday 21 January 13
1. Write code
2. Upload to server
3. Reload browser
What if it crashes?
What if it crashes?

```bash
runway2:~ # node myapp.js
```

```bash
runway2:~ # forever start myapp.js
```
What if it crashes?

Who starts forever?
Systemd

• Modern service manager for Linux

• Controversial (but awesome!)

• Fedora / Suse (by default)
  RHEL / CentOS / ... (soon)
  Any other distro (with hacks)

http://www.freedesktop.org/wiki/Software/systemd
[Service]
ExecStart=/opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js
Restart=always
StandardOutput=syslog
SyslogIdentifier=myapp
User=nobody
Group=nobody
Environment=PATH=/opt/nodejs/v0.8.16/bin:/usr/bin:/usr/local/bin
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
runway3:~ # systemctl enable myservice.service
ln -s '/etc/systemd/system/myservice.service' '/etc/systemd/system/multi-user.target.wants/myservice.service'
runway3:~ # systemctl start myservice.service
runway3:~ #
runway3:~ # systemctl enable myservice.service
ln -s '/etc/systemd/system/myservice.service' '/etc/systemd/system/multi-user.target.wants/myservice.service'
runway3:~ # systemctl start myservice.service
runway3:~ #
runway3:~ # systemctl status myservice.service
myservice.service
Loaded: loaded (/etc/systemd/system/myservice.service; enabled)
Active: active (running) since Wed, 16 Jan 2013 14:19:18 +0100; 2min 3s ago
Main PID: 14075 (node)
CGroup: name=systemd:/system/myservice.service
        14075 /opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js...
runway3:~ # 
# systemctl status myservice.service

myservice.service

  Loaded: loaded (/etc/systemd/system/myservice.service; enabled)
  Active: active (running) since Wed, 16 Jan 2013 14:19:18 +0100; 4min 50s ago

  Main PID: 14075 (node)
  CGroup: name=systemd:/system/myservice.service
         /opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js...

# kill 14075

# systemctl status myservice.service

myservice.service

  Loaded: loaded (/etc/systemd/system/myservice.service; enabled)
  Active: active (running) since Wed, 16 Jan 2013 14:24:15 +0100; 1s ago

  Main PID: 14233 (node)
  CGroup: name=systemd:/system/myservice.service
         /opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js...
Socket Activation
Not all workloads are equal

- Products: big volumes, often
- Client backends: infrequent peaks, randomly distributed
Not all workloads are equal

- Products: big volumes, often
- Client backends: infrequent peaks, randomly distributed

We can increase server density if we don’t have to run everything all the time!
How this works: without activation

TCP port 80

systemd

monitors

Node.js
How this works: with activation

TCP port 80

systemd → monitors → Node.js
How this works: with activation

TCP port 80

systemd → monitors → Node.js
How this works: with activation

TCP port 80

listens

systemd
How this works: with activation

TCP port 80

systemd

starts
socket hand-over

Node.js

Monday 21 January 13
How this works: with activation

TCP port 80

listens

systemd
Socket Activation with Node.js

• node-systemd
  https://github.com/rubenv/node-systemd

• node-autoquit
  https://github.com/rubenv/node-autoquit
runway3:myapp # npm install --save systemd
npm http GET https://registry.npmjs.org/systemd
npm http 304 https://registry.npmjs.org/systemd
systemd@0.2.2 node_modules/systemd
runway3:myapp #
runway3:myapp # npm install --save systemd

npm http GET https://registry.npmjs.org/systemd
npm http 304 https://registry.npmjs.org/systemd

systemd@0.2.2 node_modules/systemd
runway3:myapp #

var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(3000);
runway3:myapp # npm install --save systemd
npm http GET https://registry.npmjs.org/systemd
npm http 304 https://registry.npmjs.org/systemd
systemd@0.2.2 node_modules/systemd
runway3:myapp #

```
require('systemd');

var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(3000);
```
runway3:myapp # npm install --save systemd

npm http GET https://registry.npmjs.org/systemd
npm http 304 https://registry.npmjs.org/systemd

systemd@0.2.2 node_modules/systemd

runway3:myapp #

require('systemd');

var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen('systemd');
runway3:myapp # npm install --save systemd

npm http GET https://registry.npmjs.org/systemd
npm http 304 https://registry.npmjs.org/systemd

systemd@0.2.2 node_modules/systemd
runway3:myapp #

require('systemd');

var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(process.env.NODE_ENV == 'production' ? 'systemd' : 3000);
`runway3:myapp # npm install --save autoquit`

```
npm http GET https://registry.npmjs.org/autoquit
npm http 200 https://registry.npmjs.org/autoquit
npm http GET https://registry.npmjs.org/autoquit/-/autoquit-0.1.2.tgz
npm http 200 https://registry.npmjs.org/autoquit/-/autoquit-0.1.2.tgz
autoquit@0.1.2 node_modules/autoquit
runway3:myapp #
```
runway3:myapp # npm install --save autoquit
npm http GET https://registry.npmjs.org/autoquit
npm http 200 https://registry.npmjs.org/autoquit
npm http GET https://registry.npmjs.org/autoquit/-/autoquit-0.1.2.tgz
npm http 200 https://registry.npmjs.org/autoquit/-/autoquit-0.1.2.tgz
autoquit@0.1.2 node_modules/autoquit
runway3:myapp #

require('systemd');

var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(process.env.NODE_ENV == 'production' ? 'systemd' : 3000);
runway3:myapp # npm install --save autoquit

```
runway3:myapp #
```

```
root@runway3.flowpilots.com: /srv/www/myapp — ssh — 80x16
```

```
require('systemd');

var http = require('http');
var server = http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
});
server.listen(process.env.NODE_ENV == 'production' ? 'systemd' : 3000);
```
runway3:myapp # npm install --save autoquit

runway3:myapp #

```javascript
require('systemd');
require('autoquit');

var http = require('http');
var server = http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
});
server.listen(process.env.NODE_ENV === 'production' ? 'systemd' : 3000);
```
runway3:myapp # npm install --save autoquit

```
autoquit@0.1.2 node_modules/autoquit
```

```
require('systemd');
require('autoquit');

var http = require('http');
var server = http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
});
server.autoQuit();
server.listen(process.env.NODE_ENV === 'production' ? 'systemd' : 3000);
```
**Terminal Commands:**

```
runway3:myapp # npm install --save autoquit
```

**NPM Output:**

```
runway3:myapp #
```

**JavaScript Code:**

```javascript
require('systemd');
require('autoquit');

var http = require('http');
var server = http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
});
server.autoQuit({ timeout: 1800 });
server.listen(process.env.NODE_ENV == 'production' ? 'systemd' : 3000);
```
Dude...

Wait, what?
Kill your app!

- Forces your app tier to be stateless
- Step number one on the path to scaling horizontally
[Service]
ExecStart=/opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js
Restart=always
StandardOutput=syslog
SyslogIdentifier=myapp
User=nobody
Group=nobod
Environment=PATH=/opt/nodejs/v0.8.16/bin:/usr/bin:/usr/local/bin
Environment=NODE_ENV=production

[Install]
WantedBy=multi-user.target
[Service]
ExecStart=/opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js
Restart=always
StandardOutput=syslog
SyslogIdentifier=myapp
User=nobody
Group=nobody
Environment=PATH=/opt/nodejs/v0.8.16/bin:/usr/bin:/usr/local/bin
Environment=NODE_ENV=production
[Service]
ExecStart=/opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js
StandardOutput=syslog
SyslogIdentifier=myapp
User=nobody
Group=nobody
Environment=PATH=/opt/nodejs/v0.8.16/bin:/usr/bin:/usr/local/bin
Environment=NODE_ENV=production
[Socket]
ListenStream=/run/node-myapp.sk

[Install]
WantedBy=sockets.target
[Socket]
ListenStream=/run/node-myapp.sk

[Install]
WantedBy=sockets.target

/etc/systemd/system/myservice.socket

location / {
  proxy_set_header Host $host;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_pass http://unix:/run/node-myapp.sk/;
}

/etc/nginx/conf.d/vhost.myapp.conf
runway3:myapp # systemctl disable myservice.service
runway3:myapp # rm '/etc/systemd/system/multi-user.target.wants/myservice.service'
runway3:myapp # systemctl stop myservice.service
runway3:myapp # systemctl enable myservice.socket
runway3:myapp # ln -s '/etc/systemd/system/myservice.socket' '/etc/systemd/system/sockets.target.wants/myservice.socket'
runway3:myapp # systemctl start myservice.socket
runway3:myapp # systemctl disable myservice.service
rm '/etc/systemd/system/multi-user.target.wants/myservice.service'
runway3:myapp # systemctl stop myservice.service
runway3:myapp # systemctl enable myservice.socket
ln -s '/etc/systemd/system/myservice.socket' '/etc/systemd/system/sockets.target.wants/myservice.socket'
runway3:myapp # systemctl start myservice.socket
runway3:myapp # systemctl status myservice.socket
myservice.socket
    Loaded: loaded (/etc/systemd/system/myservice.socket; enabled)
    Active: active (listening) since Wed, 16 Jan 2013 16:04:47 +0100; 12s ago
    CGroup: name=systemd:/system/myservice.socket

runway3:myapp #
runway3:myapp # systemctl disable myservice.service
rm '/etc/systemd/system/multi-user.target.wants/myservice.service.service'
runway3:myapp # systemctl stop myservice.service
runway3:myapp # systemctl enable myservice.socket
ln -s '/etc/systemd/system/myservice.socket' '/etc/systemd/system/sockets.target.wants/myservice.socket'
runway3:myapp # systemctl start myservice.socket
runway3:myapp # systemctl status myservice.socket
my service.socket
Loaded: loaded (/etc/systemd/system/myservice.socket; enabled)
Active: active (listening) since Wed, 16 Jan 2013 16:04:47 +0100; 12s ago
    CGroup: name=syslm:///system/myservice.socket

runway3:myapp # systemctl status myservice.service
my service.service
Loaded: loaded (/etc/systemd/system/myservice.service; static)
Active: failed (Result: exit-code) since Wed, 16 Jan 2013 16:04:36 +0100; 32s ago
    Main PID: 14233 (code=exited, status=1/FAILURE)
    CGroup: name=syslm:///system/myservice.service

runway3:myapp #
runway3:myapp # curl http://localhost/
Hello World
runway3:myapp # systemctl status myservice.service
myservice.service
  Loaded: loaded (/etc/systemd/system/myservice.service; static)
  Active: active (running) since Wed, 16 Jan 2013 16:09:05 +0100; 3s ago
  Main PID: 14662 (node)
  CGroup: name=systemd:/system/myservice.service
         14662 /opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js...
runway3:myapp #
Monitoring
runway2:~ # systemctl status node-crash.flowpilots.com.service
node-crash.flowpilots.com.service
   Loaded: loaded (/etc/systemd/system/node-crash.flowpilots.com.service; static)
   Active: active (running) since Wed, 16 Jan 2013 16:14:31 +0100; 50s ago
   Main PID: 27721 (node)
   CGroup: name=systemd:/system/node-crash.flowpilots.com.service
         ^ 27721 node /opt/nodejs/v0.8.4/bin/coffee /srv/www/crash....

Jan 16 16:14:32 runway2.flowpilots.com node-crash.flowpilots.com[27721]: List...
Jan 16 16:14:32 runway2.flowpilots.com node-crash.flowpilots.com[27721]: - - ...
Jan 16 16:14:32 runway2.flowpilots.com node-crash.flowpilots.com[27721]: - - ...
Jan 16 16:14:37 runway2.flowpilots.com node-crash.flowpilots.com[27721]: - - ...
Jan 16 16:14:39 runway2.flowpilots.com node-crash.flowpilots.com[27721]: - - ...
Journal

• Log aggregation for the 21st century

• Allows passing extra metadata

→ A full audit trail, right in your system logging infrastructure
# yum install systemd-devel; npm install --save journald
runway3:myapp # yum install systemd-devel; npm install --save journald

```
require('systemd');
require('autoquit');

var journald = require('journald').Log;

var http = require('http');
var server = http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World
');
    journald.log({
        ACTION: 'sayHello',
        USER_ID: currentUserId
    });
});
server.autoQuit();
server.listen(process.env.NODE_ENV == 'production' ? 'systemd' : 3000);
```
More fun stuff
Resource Limiting

• All processes (and children) are grouped by CGroup

• Systemd can put limits on them!
Resource Limiting

- All processes (and children) are grouped by CGroup
- Systemd can put limits on them!
Access restrictions

- Separate filesystem namespaces per process

```
[Service]
ExecStart=/opt/nodejs/v0.8.16/bin/node /srv/www/myapp/app.js
StandardOutput=syslog
SyslogIdentifier=myapp
User=nobody
Group=nobody
Environment=PATH=/opt/nodejs/v0.8.16/bin:/usr/bin:/usr/local/bin
Environment=NODE_ENV=production
InaccessibleDirectories=/home
ReadOnlyDirectories=/srv/www
PrivateTmp=true
```
The future
Currently
Currently

- Automatically activated node.js services
Currently

- Automatically activated node.js services
- Centralized full-system logging
Currently

- Automatically activated node.js services
- Centralized full-system logging
- Resource control policies
Soon: Containers
Soon: Containers

• VMs are a popular way of segmenting users
Soon: Containers

- VMs are a popular way of segmenting users
- Extremely expensive
Soon: Containers

- VMs are a popular way of segmenting users
- Extremely expensive
- nspawn or LXC containers simulate a minimal Linux environment
Soon: Containers

- VMs are a popular way of segmenting users
- Extremely expensive
- `nspawn` or `LXC` containers simulate a minimal Linux environment
- Can be activated as well!
Take-away points

- Clouds seem like magically complex infrastructures
- They’re not, you can easily have it as well
- If nothing else, adopt socket activation
Resources

  (read the 20 parts of systemd for system administrators, they’re worth it, this talk is the tip of the iceberg)

- [https://github.com/rubenv/node-systemd](https://github.com/rubenv/node-systemd)

- [https://github.com/rubenv/node-autoquit/](https://github.com/rubenv/node-autoquit/)

• Will put this online at some point

• ruben@flowpilots.com

• @rubenv
Questions?
Extra slides
runway3:myapp # systemctl status myservice.service | grep Active
  Active: failed (Result: exit-code) since Wed, 16 Jan 2013 16:11:04 +0100; 3s ago
runway3:myapp # time curl http://localhost/
Hello World

real    0m0.048s
user    0m0.000s
sys     0m0.004s
runway3:myapp # systemctl status myservice.service | grep Active
  Active: active (running) since Wed, 16 Jan 2013 16:11:10 +0100; 1s ago
runway3:myapp # time curl http://localhost/
Hello World

real    0m0.008s
user    0m0.003s
sys     0m0.002s
runway3:myapp #