Using NodeSource on Cloud Foundry

Patrick Mueller
About Me

Patrick Mueller

• Senior Node Engineer at NodeSource

• Fooling around on the internet since the late 1980’s

• Developing with Node.js development on Cloud Foundry since 2013

• Worked at IBM for 30 years, developing a variety of software platforms including IDEs, mobile runtimes and libraries, and server platforms

• He can bore you to death talking about how great development in Smalltalk was.
• **Understanding Cloud Foundry** and how to run Node.js applications on it.

• Get started running **Hello World on Cloud Foundry**.

• Learn some **Node.js tips n’ tricks for Cloud Foundry** to optimizing your development cycle.

• **Use N|Solid to diagnose your Cloud Foundry Applications** to find performance and memory issues.
Understanding Cloud Foundry
Understanding Cloud Foundry

- **Platform-as-a-Service** product
  - very similar to Heroku, shares some technology

- **Open Source**, part of Cloud Foundry Foundation

- **Commercial** hosted, and on-prem deployments from multiple vendors
  - Pivotal, IBM, etc

- **Supports multiple languages** out-of-the-box
  - Node.js, Ruby, Python, Java, Go
Understanding Cloud Foundry - Deploying a Node.js application

- Create a **manifest.yml** file in your project root directory
- Run **cf push** to deploy app
  - sends your application code to a staging machine
  - staging machine obtains Node.js runtime, runs **npm install**
  - packages result in a droplet
  - runs the droplet in a container
Running Hello World on Cloud Foundry
'use strict'

const http = require('http')

const PORT = process.env.PORT || '3000'  // PORT env var provided by CF

const server = http.createServer(onRequest)
server.listen(PORT, onListening)

function onRequest (req, res) {
  res.end('<h1>Hello, World</h1>')
}

function onListening () {
  console.log(`Server listening on http://localhost:${PORT}`)
}
Running Hello World - package.json and manifest.yml

package.json

```json
{
    "name": "hello-world",
    "version": "1.0.0",
    "dependencies": {}
}
```

manifest.yml

```yaml
---
applications:
- name: hello-world-nodejs
  memory: 128M
  command: node server
```
$ cf push
Using manifest file /path/to/hello-world/manifest.yml

Updating app hello-world-nodejs in org pcfdev-org / space pcfdev-space as user...
OK...

~ 30 seconds later...

requested state: started
instances: 1/1
usage: 128M x 1 instances
urls: hello-world-nodejs.local.pcfdev.io
buildpack: node.js 1.5.15

<table>
<thead>
<tr>
<th>state</th>
<th>since</th>
<th>cpu</th>
<th>memory</th>
<th>disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>#0</td>
<td>running</td>
<td>0.0%</td>
<td>10.8M of 128M</td>
<td>39M of 512M</td>
</tr>
</tbody>
</table>
Node.js tips n’ tricks for Cloud Foundry
Node.js CF Tips n’ Tricks - skip uploading files not used at runtime

• Use .cfignore to have files not uploaded to staging machine to
  - Quicker turn around time on cf push
  - Add node_modules when starting a new app, tests whether your dependencies in package.json are correct, saves time.
  - Add other directories not used at runtime to .cfignore, like tests, docs, etc
Node.js CF Tips n’ Tricks - use cfenv to parse VCAP_SERVICES

- Cloud Foundry provides two environment variables - **VCAP_SERVICES** and **VCAP_APPLICATION** which provides environmental information for your application.
  - JavaScript object in JSON format
  - a bit unwieldy to traverse

- Use **cfenv** to get simple access to the data, eg:
  
  ```javascript
  appEnv.getServiceCreds('redis-counters')
  ```

- [https://www.npmjs.com/package/cfenv](https://www.npmjs.com/package/cfenv)

- `npm install cfenv --save`
Using N|Solid to Diagnose your Cloud Foundry Applications
Using N|Solid on Cloud Foundry

- Run your apps with the N|Solid Runtime by using the N|Solid buildpack instead of the Node.js buildpack
- Bind your app to a `nsolid-storage` service
- Run N|Solid Storage and Console servers
Using N|Solid on Cloud Foundry

• N|Solid buildpack
  - Supports the same functionality as the Node.js buildpack
  - Installs N|Solid Runtime instead of open source Node.js
  - Sets N|Solid env vars based on `nsolid-storage` service and VCAP environment variables
  - Use from GitHub, or download bundled (offline) buildpack to install in Cloud Foundry
  - Open Source - contribute new features!
Running Hello World - changes to manifest.yml

changes to manifest.yml

```yaml
---
aplications:
- name: hello-world-nodejs
  memory: 128M
  command: node server
  buildpack: https://github.com/nodesource/nsolid-buildpack-cf.git
  services:
    - nsolid-storage
```
Using N|Solid on Cloud Foundry

**Bind your app to nsolid-storage service**

- Create a User-Provided Service from JSON

```bash
cf cups nsolid-storage -p example.json
```

```
{
    "sockets": {
        "command": "nsolid-storage.example.com:9001",
        "data": "nsolid-storage.example.com:9002",
        "bulk": "nsolid-storage.example.com:9003"
    }
}
```

- Bind the service to all apps using the N|Solid buildpack
- Provides coordinates to the N|Solid Storage server
Using N|Solid on Cloud Foundry

• Run N|Solid Storage and Console servers

  - For production, should run in a network visible to the Cloud Foundry applications

  - For “trying it out”, the N|Solid Storage and Console servers can be run as Cloud Foundry applications w/caveats:
    - Historical data and settings cannot be persisted
    - Requires `cf ssh` to be enabled

  - Coming “Some Day” - provision and run N|Solid servers when creating an nsolid-storage service
Demo
References
References during webinar, post questions to Twitter with hashtag: #needtonode

- These Slides and Samples
  http://pmuellr.github.io/slides/

- N|Solid Buildpack for Cloud Foundry
  https://github.com/nodesource/nsolid-buildpack-cf

- N|Solid Samples for Cloud Foundry
  https://github.com/nodesource/nsolid-cf

- Node Package cfenv
  https://www.npmjs.com/package/cfenv

- N|Solid Console demo (available during demo section)
  https://pjm-nsolid-console.cfapps.io/
Thank you.

Patrick Mueller
pmuellr@nodesource.com
@pmuellr