Building and Deploying 12 Factor Apps in Scala, Java, Ruby, and Node.js

Neil Shannon

February 24, 2017

About Me

Polyglot Developer/Architect

Founder at NTS Development LLC

Staff Software Engineer @ The Home Depot

@ntshannon | neil@nts-dev.com

https://linkedin.com/in/neilshannon

https://github.com/neilshannon

Agenda

What is a 12-factor app and why do I care?

How do I build a 12 factor app in Java? Scala? Ruby? Node.js?

How do I deploy my app to Pivotal Cloud Foundry?

12 factors for the Impatient

I. Codebase - use version control (e.g. git)

II. **Dependencies** - use a dependency manager (e.g. gradle/maven/sbt)

III. **Config** - separate configuration from code (use the OS environment)

IV. **Backing Services** - reference resources such as DBs by URLs in the config

V. Build release run - separate build from run. Use versions.

VI. **Processes** - run the app as one or more *stateless* processes.

VII. **Port binding** - app should be self-contained. No app server.

VIII. Concurrency - scale horizontally

IX. Disposability - fast startup, graceful shutdown

X. **Dev/Prod parity** - keep environments similar

XI. Logs - treat logs as event streams (no FileAppenders!)

XII. Admin Processes - treat admin processes as one-off events

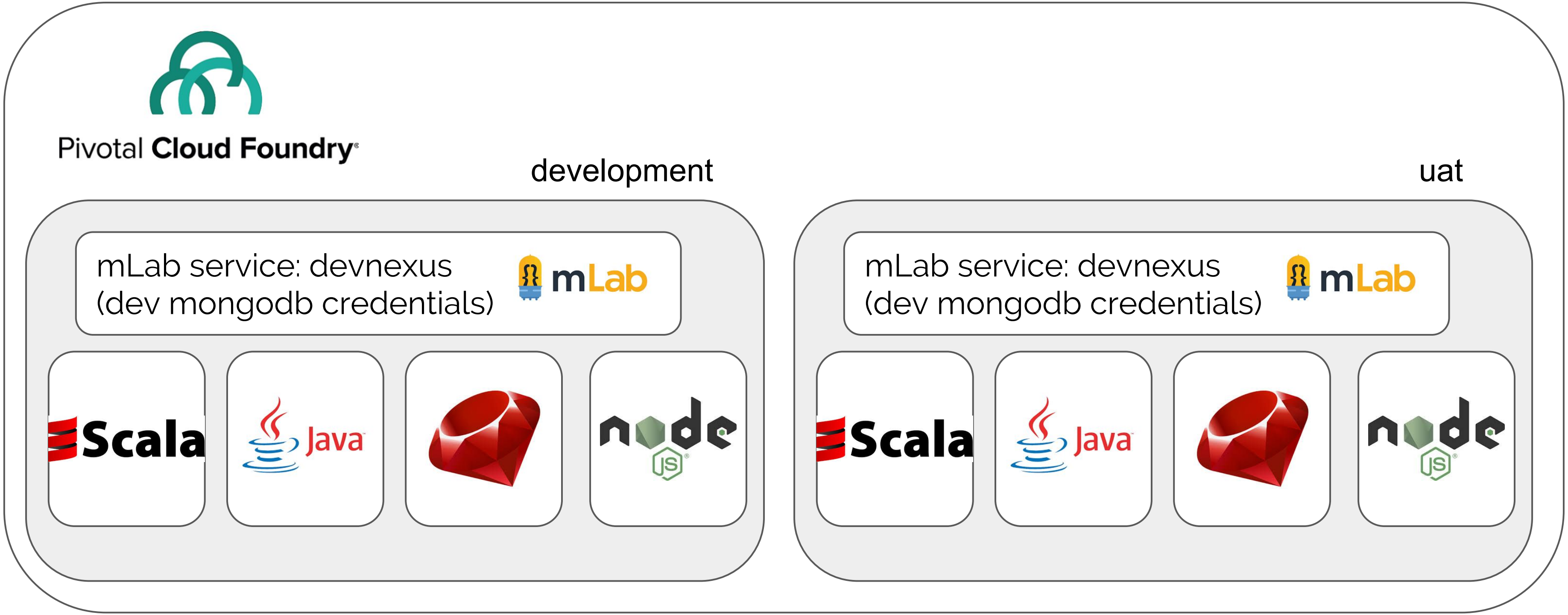
A 12-factor app is an application designed to be deployed in the cloud.

Deployment Environment

CLOUDFOUNDRY

Pivotal Cloud Foundry®

The Environment



development

mLab service: devnexus
(dev mongodb credentials)

uat

mLab service: devnexus
(dev mongodb credentials)

Same

- Application Code
- Access to config
- Build model
- Deployment model
- Environment

Different

- Config values
- Endpoint URIs
- Number of instances

What are we going to do next?

We’re going to build an executable JAR file containing a Java web service and its dependencies.

We will tell Pivotal Cloud Foundry how to execute our application using a cloud manifest (manifest.yml).

We’re going to push our JAR to Pivotal Cloud Foundry and boot up our application.

Let’s Get Started

```
$ git clone https://github.com/neilshannon/devnexus-microservices
```

Prepare Environment

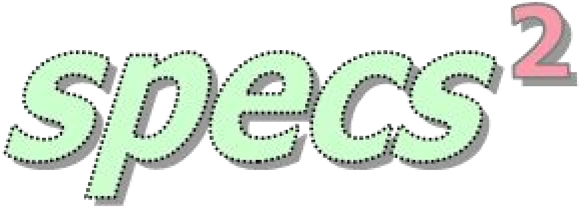
1. Create org in Pivotal Web Services
2. Create space in Pivotal Web Services
3. Create MongoLab service

Prepare cloud manifest

manifest.yml

```
---
applications:
- name: devnexus-microservices-scala
 buildpack: java_buildpack
 path:
target/scala-2.11/devnexus-microservices-scala-assembly-0.1.0-SNAPSHOT.jar
 services:
   - devnexus
```

Scala Stack



Scala Stack

Build - sbt

Test - specs2

Platform - Scalatra hosted in embedded Jetty container

Persistence - ReactiveMongo

REST - Scalatra

Deploy - Cloud Foundry

Scala pitfalls and caveats!

Scala generates a lot of code with long class names.

CloudFoundry aufs only supports file lengths of a maximum of 243 characters.

Guess how many characters this generated class has...

```
Copying into the container failed: stream-in: nstar: error streaming in: exit status 2. Output: tar:
shapeless/TupleTypeableInstances$$anon$17$$anonfun$cast$17$$anonfun$app$$$$9e2cdc6fa02ebaeef8cffbef37753
221$$$$$128$$anonfun$apply$129$$anonfun$apply$130$$anonfun$apply$131$$anonfun$apply$132$$anonfun$apply$1
33$$anonfun$apply$134$$anonfun$apply$135.class: Cannot open: File name too long
```

Java Stack

spring
by Pivotal

Java Stack

Build - gradle

Test - JUnit

Platform - Spring Boot

Persistence - Spring Data MongoDB

REST (with hypermedia!)- Spring Data REST

Deploy - Cloud Foundry (Pivotal Web Services)

Ruby Stack

rack-test

Ruby Stack

Build - bundler

Test - Minitest, rack-test

Platform - Sinatra

Persistence - mongodb (Ruby official driver)

REST - Sinatra

Deploy - Cloud Foundry (Pivotal Web Services)

Node Stack

Node Stack

Build - npm

Test - mocha, chai

Platform - restify

Persistence - mongodb (official Node.js MongoDB driver)

REST - restify

Deploy - Cloud Foundry (Pivotal Web Services)

Resources

The 12 Factor App - http://12factor.net

Cloud Foundry - https://www.cloudfoundry.org/

Pivotal Web Services - https://run.pivotal.io/