Rock Solid Cloud Native Apps with Spring

Neil Shannon, Founder, NTS Development LLC
@ntshannon

About Me

- Polyglot Engineer/Architect
- President / Founder at NTS Development LLC
- Java, Scala, Ruby, Node.js, Groovy, JavaScript
- >10 years professional development/architecture experience

@ntshannon | neil@nts-dev.com

https://linkedin.com/in/neilshannon

https://github.com/neilshannon

spring

Agenda

- What is a “cloud native” or “12-factor” app and why do I care?
- How can I build one with Spring and maintain solid software quality? How can I bridge the gap between my local
and cloud environments?
- Testing/Code/Deployment Demo

The 12 Factors for the Impatient

1. Codebase - use version control (e.g. git)
2. Dependencies - use a dependency manager (e.g. gradle/maven/sbt)
3. Config - separate configuration from code (use the OS environment)
4. Backing Services - reference resources such as DBs by URLs in the config
5. Build release run - separate build from run. Use versions.
6. Processes - run the app as one or more stateless processes.
7. Port binding - app should be self-contained. No app server.
8. Concurrency - scale horizontally
9. Disposability - fast startup, graceful shutdown
10. Dev/Prod parity - keep environments similar
11. Logs - treat logs as event streams (no FileAppenders!)
12. Admin Processes - treat admin processes as one-off events

Cloud Native Apps

Stateless

- Stateless apps can scale horizontally
- No “sticky sessions” or server-side state

Use the environment

- Configuration is stored in the environment
- App access configuration in a uniform manner

Self-contained

Everything (dependencies, http server, code, static content) is included in the deployment artifact

Are you an engineer?

simpsonscrazy.com

spring

Quality?

> Testing ***is*** the engineering rigor of software development.
>
> *Neal Ford*

Tech Stack

Spring Libraries

- **Spring Boot** - context/runtime/web/embedded Tomcat
- **Spring Cloud Connectors** - cloud configuration locally/in the cloud
- **Spring Data Mongo DB** - mongo connectivity
- **Spring Data REST** - HATEOAS web services for Mongo collections
- **Spring Test / Spring Boot Test** - testing support

Local Environment

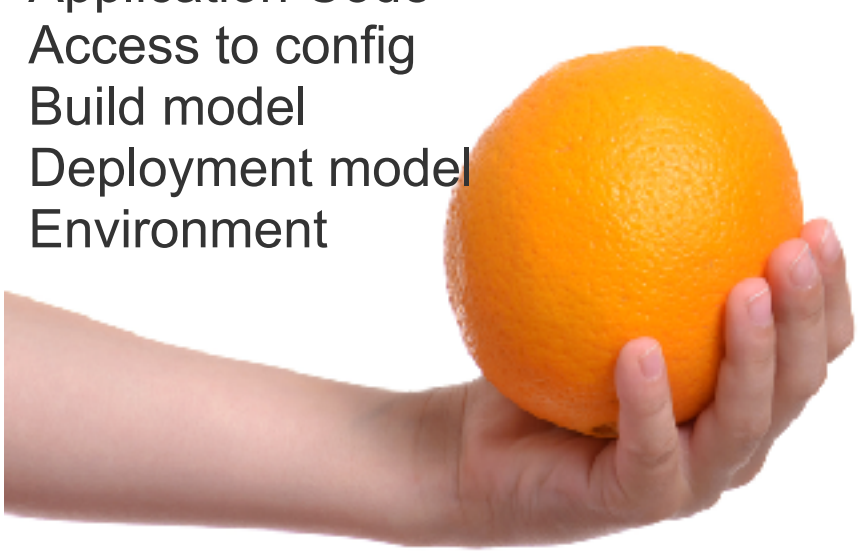
Cloud Configuration (test/local):
spring-cloud-localconfig.properties

mongodb://localhost:27017/springdays

Runtime Environments - Development vs UAT

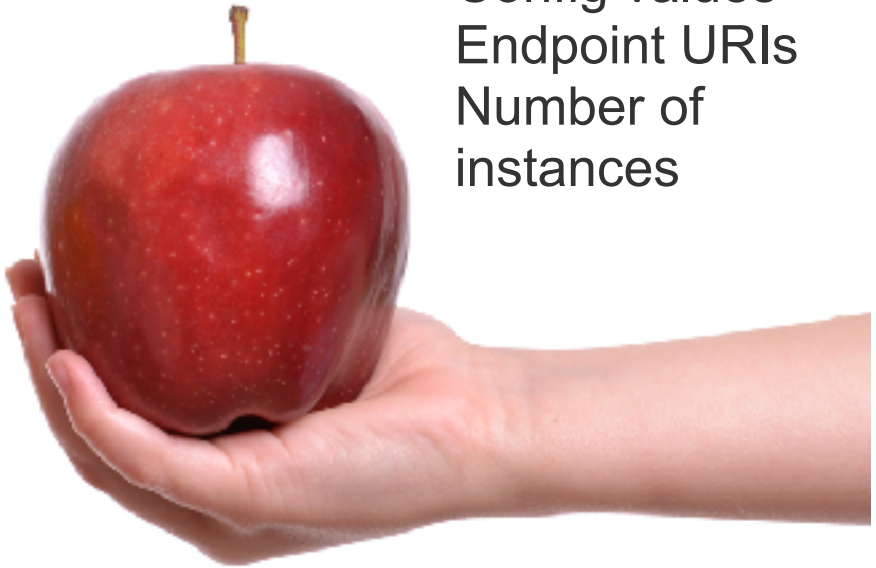
Same

Application Code
Access to config
Build model
Deployment model
Environment

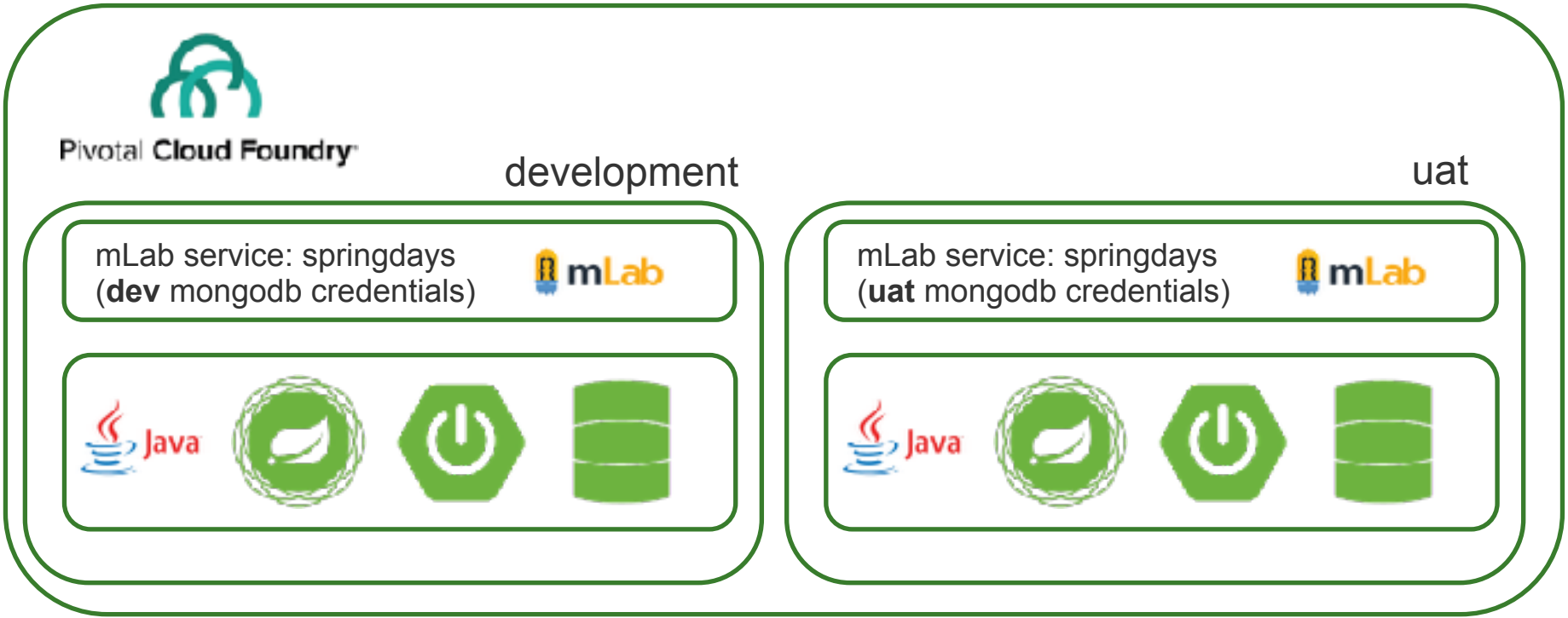


Different

Config values
Endpoint URIs
Number of instances



Deployment Environment



What are we going to do next?

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

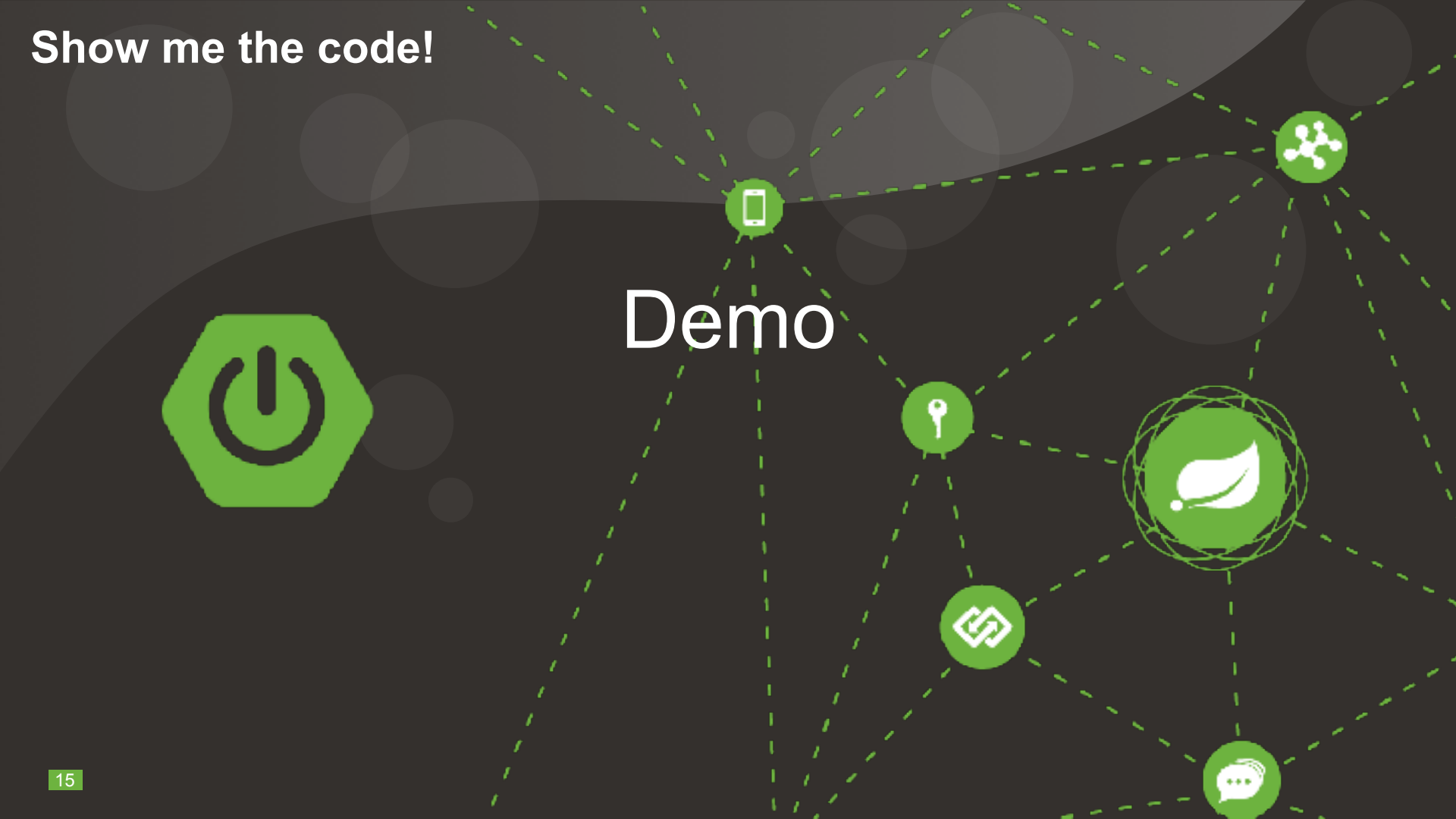
We will tell Pivotal Cloud Foundry how to execute our application using a cloud manifest (manifest.yml).

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

Prepare the Cloud Manifest (manifest.yml)

```
---
applications:
- name: spring-days-demo
  buildpack: java_buildpack
  path: build/libs/spring-days-demo-0.1.0-SNAPSHOT.jar
  services:
    - springdays
```

Show me the code!



Resources

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

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

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

Source code - https://github.com/neilshannon/spring-days-demo