

Pivotal



Spring Boot Primer

Getting Started with Spring Boot to build Cloud Native Apps.

Pivotal

Agenda

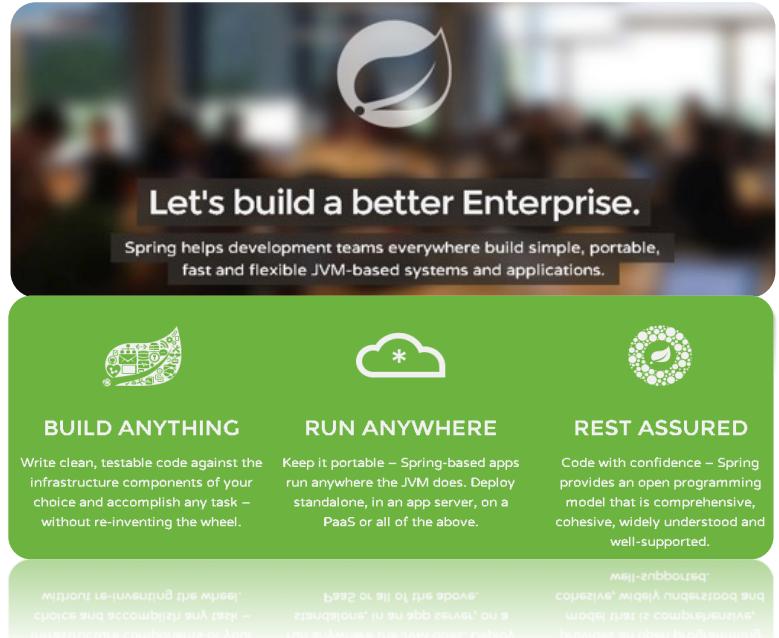
- Challenges building non-Boot applications
- What is Spring Boot?
- Capabilities
- Demos Throughout



What is Spring IO?

the #1 enterprise Java application development framework

- Embodiment of best industry dev practices
- The best of Java OSS, curated & improved
- OSS, Apache 2.0 licensed
- Proven, mature, since 2004
- Well understood and documented
- Millions of downloads
- Used heavily by Netflix and F2000



The screenshot shows the Spring IO homepage. At the top is a dark banner with the Spring logo (a stylized leaf) and the text "Let's build a better Enterprise." Below this is a sub-banner with the text "Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications." The main content area is divided into three green sections: "BUILD ANYTHING" (with a gear icon), "RUN ANYWHERE" (with a cloud icon), and "REST ASSURED" (with a circular icon). Each section has a brief description and a horizontal scroll bar at the bottom.

Let's build a better Enterprise.

Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications.

BUILD ANYTHING

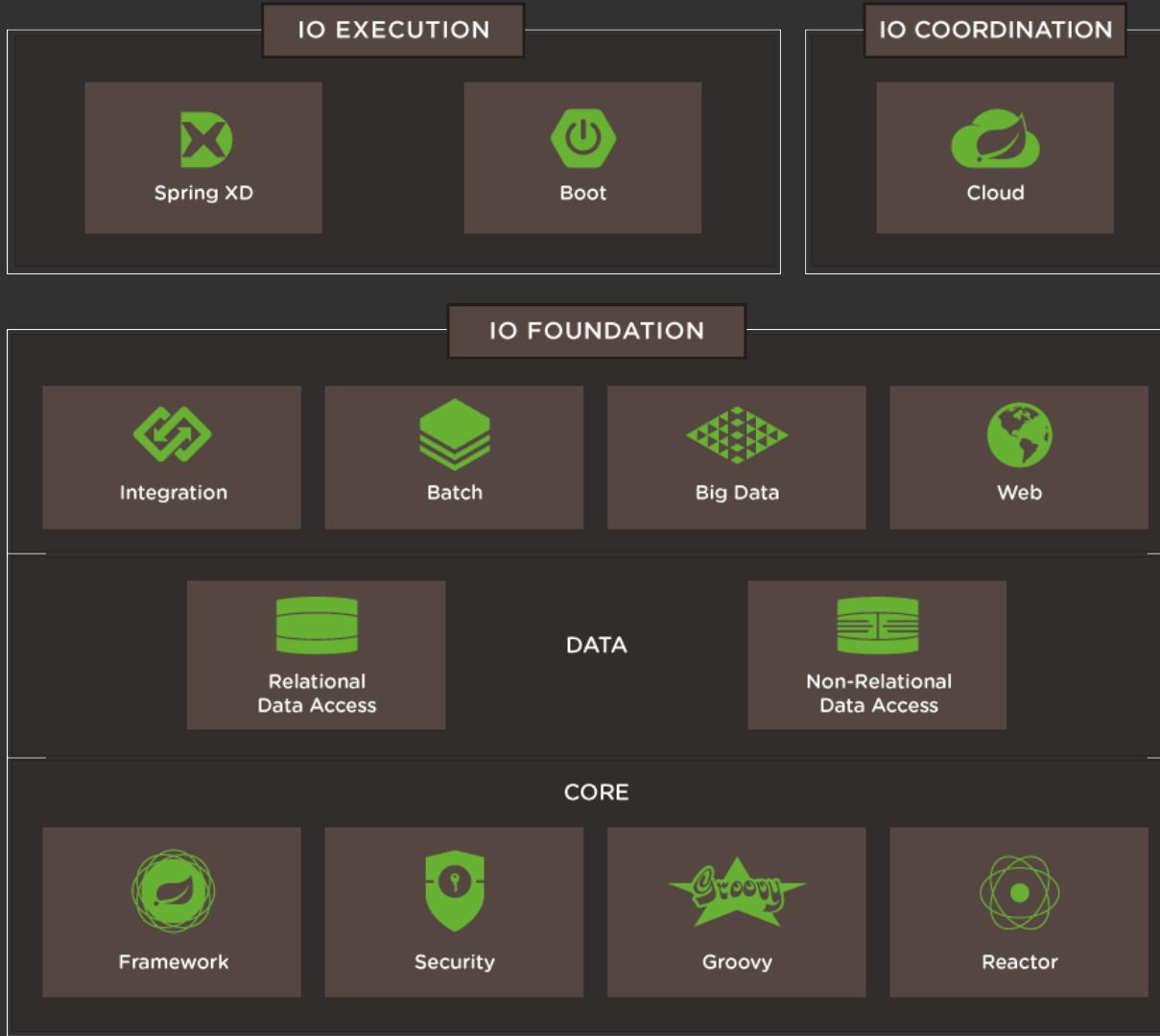
Write clean, testable code against the infrastructure components of your choice and accomplish any task – without re-inventing the wheel.

RUN ANYWHERE

Keep it portable – Spring-based apps run anywhere the JVM does. Deploy standalone, in an app server, on a PaaS or all of the above.

REST ASSURED

Code with confidence – Spring provides an open programming model that is comprehensive, cohesive, widely understood and well-supported.



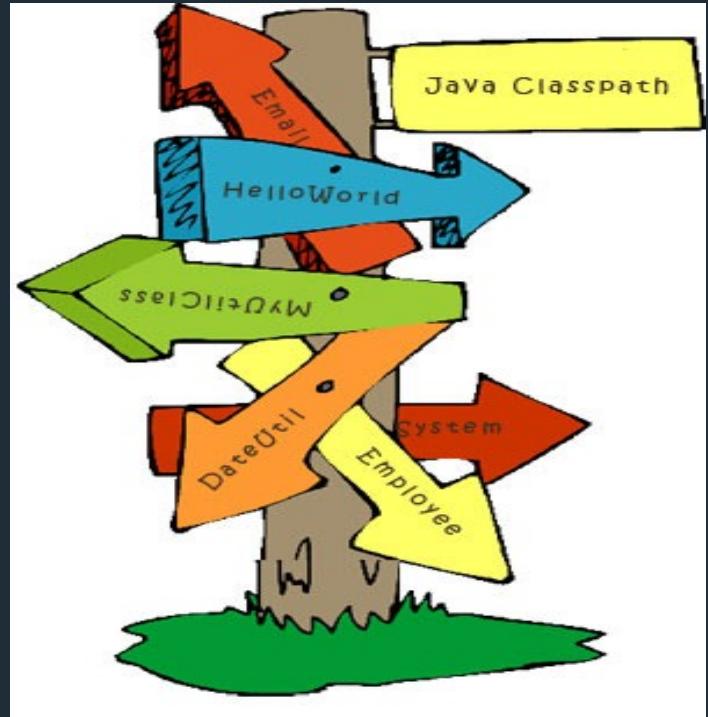
Challenges writing non-Boot applications

Version incompatibility of libraries

```
Exception in thread "main" java.lang.NoClassDefFoundError: javax/transaction/SystemException
  at java.lang.Class.forName0(Native Method)
  at java.lang.Class.forName(Class.java:344)
  at org.jboss.logging.Logger$1.run(Logger.java:2554)
  at java.security.AccessController.doPrivileged(Native Method)
  at org.jboss.logging.Logger.getMessageLogger(Logger.java:2529)
  at org.jboss.logging.Logger.getMessageLogger(Logger.java:2516)
  at org.hibernate.internal.CoreLogging.messageLogger(CoreLogging.java:28)
  at org.hibernate.internal.CoreLogging.messageLogger(CoreLogging.java:24)
  at org.hibernate.cfg.Configuration.<clinit>(Configuration.java:86)
  at com.mycompany.ContactManager.main(ContactManager.java:20)
Caused by: java.lang.ClassNotFoundException: javax.transaction.SystemException
  at java.net.URLClassLoader$1.run(URLClassLoader.java:372)
  at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
  at java.security.AccessController.doPrivileged(Native Method)
  at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
  at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
  at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
  ... 10 more
```

Challenges building non-Boot applications

- Classpath errors
 - A Jar is missing
 - There is one Jar too many
 - A class is not visible where it should be

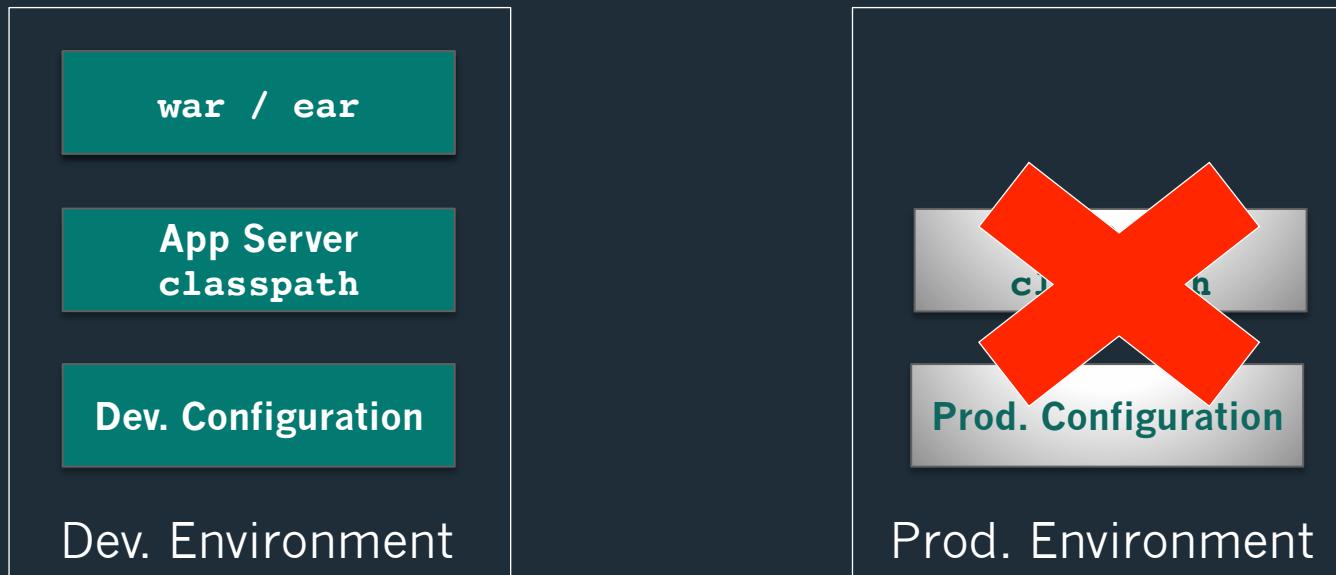


Configuration

- Maven and Spring

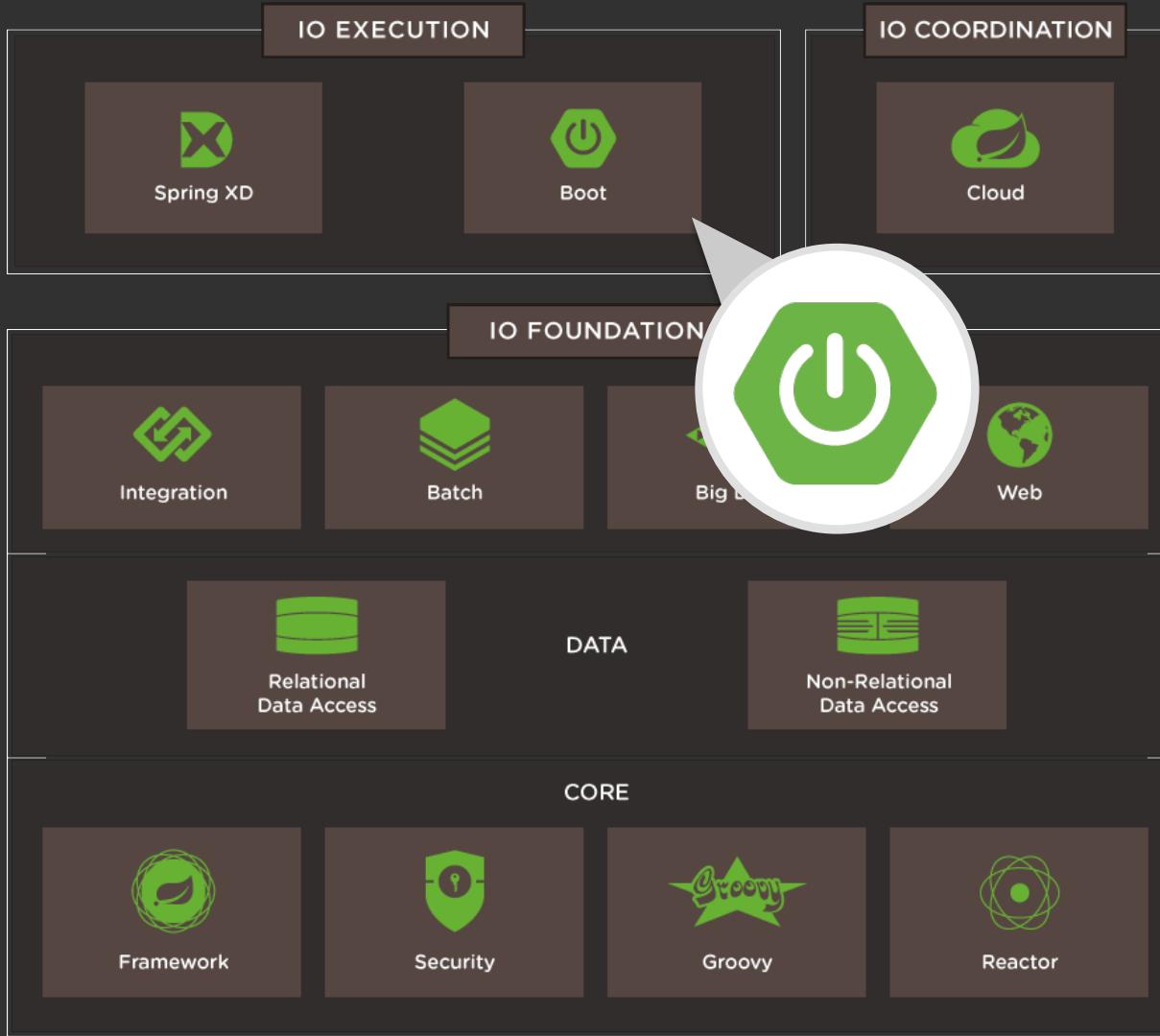
Challenges building non-Boot applications

Standard war/ear packaging causes drift across environments



What is Spring Boot?

- Spring applications typically require a lot of setup
 - Consider working with JPA. You need:
 - Datasource, TransactionManager, EntityManagerFactory ...
 - Consider a web MVC app. You need:
 - WebApplicationInitializer / web.xml, ContextLoaderListener, DispatcherServlet, ...
 - An MVC app using JPA would need all of this
- *BUT: Much of this is predictable*
 - Spring Boot can do most of this setup for you



What is Spring Boot?

An opinionated framework to simplify
bootstrapping and development of new Spring
Applications

Spring Boot



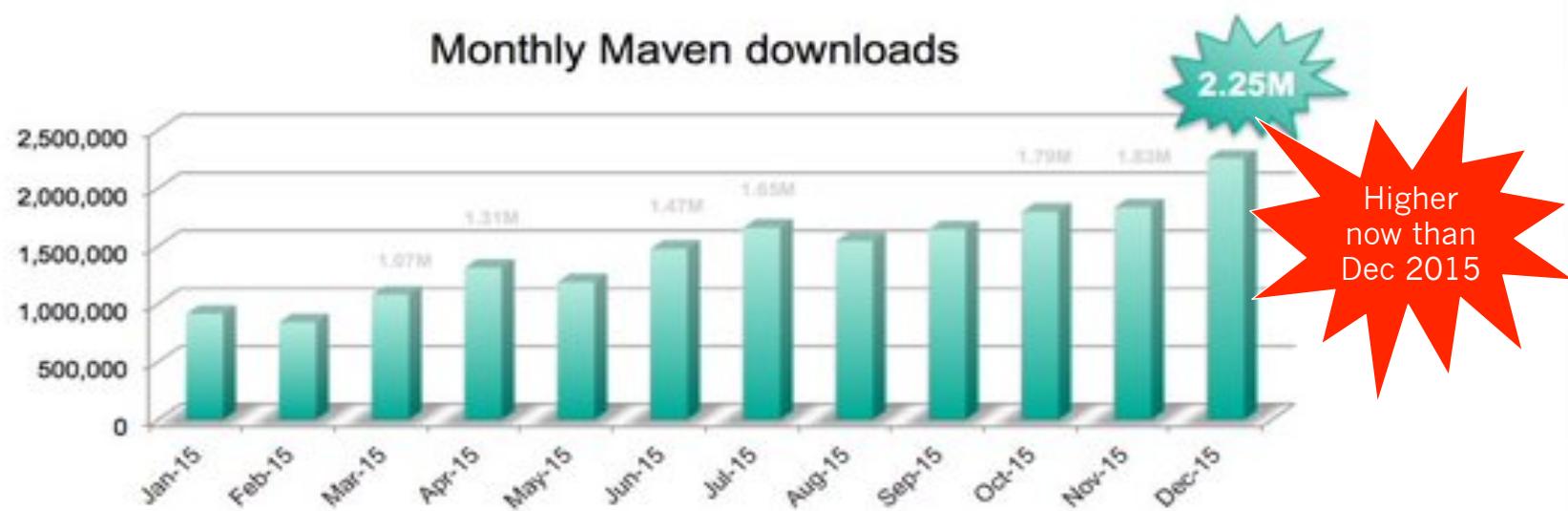
Dynamic language productivity
with maturity of enterprise Java

Cloud Native: Direct support
for Microservices, NetflixOSS
++

Fully automated app server
configuration and deployment

Production ready Ops metrics
out of the box, with a switch

Spring Boot Adoption



Capabilities

- Provides a consistent and easy way to start a project
- Manages project dependencies
- Packages deployable artifacts prevent configuration drift across environments
- Provides conditional configuration across environments
- Provides tools to improve developer productivity
- Keeps you from writing explicit configuration
- Exposes endpoints to manage and monitor applications in production



D E M O



CF

Spring Initializr

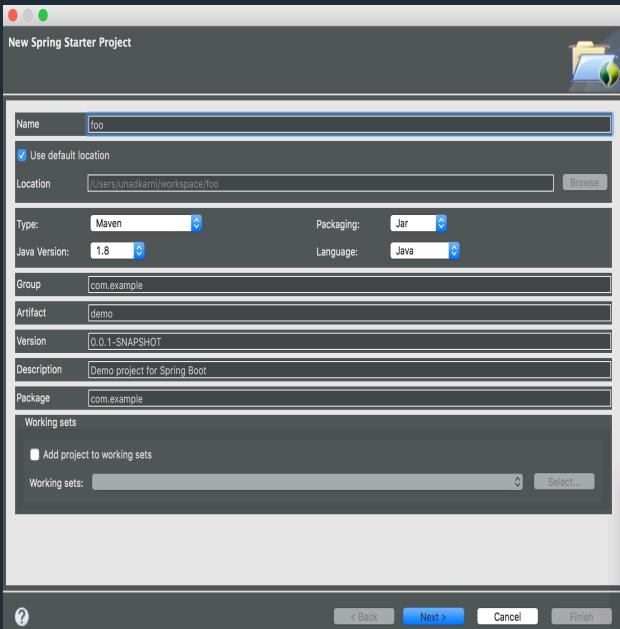
A configurable service to consistently and easily generate a quickstart project

Spring Initializr

- Generates a Spring Boot project structure.
- Provides a Maven/Gradle build specification.
- Doesn't generate application code.
- You can customize the Spring Initializr
 - <https://github.com/spring-io/initializr/>

Supported Interfaces

1. IDE



2. Web-based interface

The screenshot shows the Spring Initializr web interface. It has a header 'SPRING INITIALIZR bootstrap your application now'. Below it, a central area says 'Generate a Maven Project with Spring Boot 1.3.1'. On the left is a 'Project Metadata' section with 'Artifact coordinates' (Group: com.example, Artifact: demo) and 'Dependencies' section with 'Add Spring Boot Starters and dependencies to your application'. A search bar shows 'web'. On the right is a 'Dependencies' section listing 'Web' (selected), 'Rest Repositories', 'Vaadin', 'WS', and 'Jersey (JAX-RS)'. A large green 'Generate Project' button is at the bottom. A note at the bottom says 'Don't know what to look for? Want more options? Switch to the full version.'

3. Spring Boot CLI

The screenshot shows a terminal window with the following session:

```
UtkarshkarniMBP:workspace unadkarni$ spring init --dependencies=web,data-jpa
Using service at https://start.spring.io
Project extracted to '/Users/unadkarni/workspace/my-project'
UtkarshkarniMBP:workspace unadkarni$ tree my-project
my-project
├── mvnw
├── mvnw.cmd
└── pom.xml
└── src
    ├── main
    │   └── java
    │       └── com
    │           └── example
    │               └── DemoApplication.java
    ├── resources
    │   ├── application.properties
    │   ├── static
    │   └── templates
    └── test
        └── java
            └── com
                └── example
                    └── DemoApplicationTests.java
```

At the bottom, the terminal displays '12 directories, 6 files'.

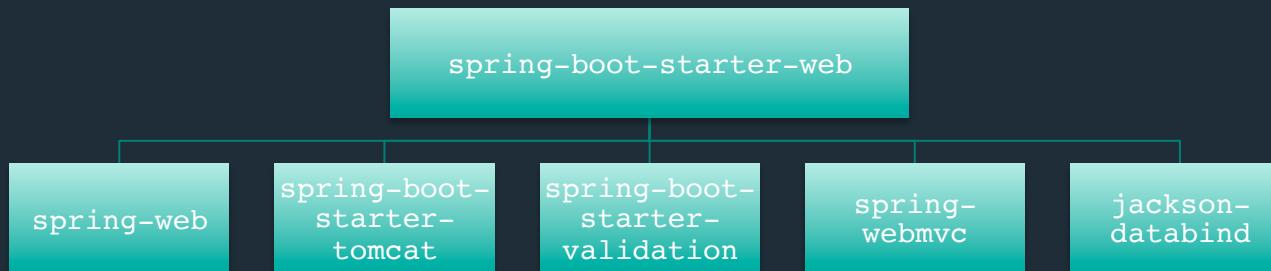
DEMO



CF

Spring Boot starters

- Are virtual packages deployed to Maven central
- They pull in other dependencies while containing no code of their own



Project pom.xml

- Set parent of your pom.xml as spring-boot-starter-parent
 - Inherit preset version numbers

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>demo</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.1.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
```

Starters vs Current Practice

Starter versions determined by version of Spring Boot used.
No explicit version numbers. Guaranteed to be compatible.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Concise. Boot pom.xml without version numbers.

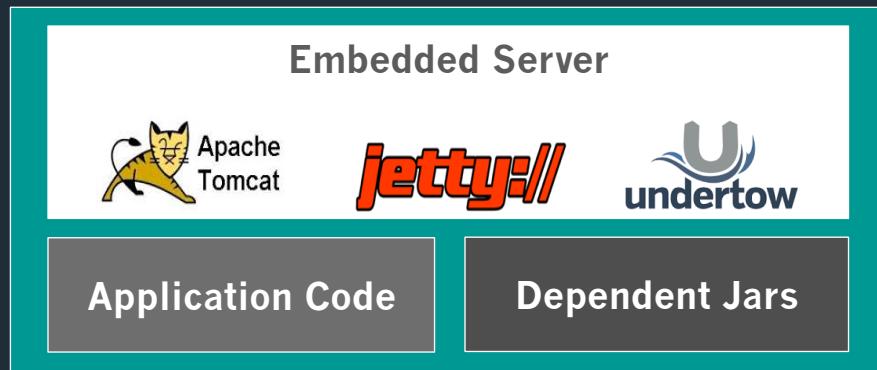
Are the version numbers compatible with each other?
Is the list of dependencies complete?

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.1.6.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf-spring4</artifactId>
    <version>2.1.4.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
    <version>1.9.2.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.0.6.Final</version>
  </dependency>
  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.190</version>
  </dependency>
</dependencies>
```

Verbose. Non-boot pom.xml with version numbers.

Packaging – Make Jar not War

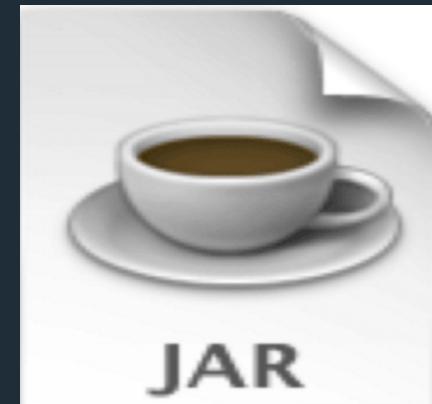
Boot's Maven / Gradle plugins produce an executable “*fat jar*”



```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Benefits of executable *fat jars*

- No installation of application servers
- No setting classpath
- Promotes consistency across environments
- Cloud-friendly
- Can be started as Unix services using
 - init.d
 - systemd



Profiles

Segregate parts of the application configuration
and make it available in certain environments via

- Annotations
- Properties file

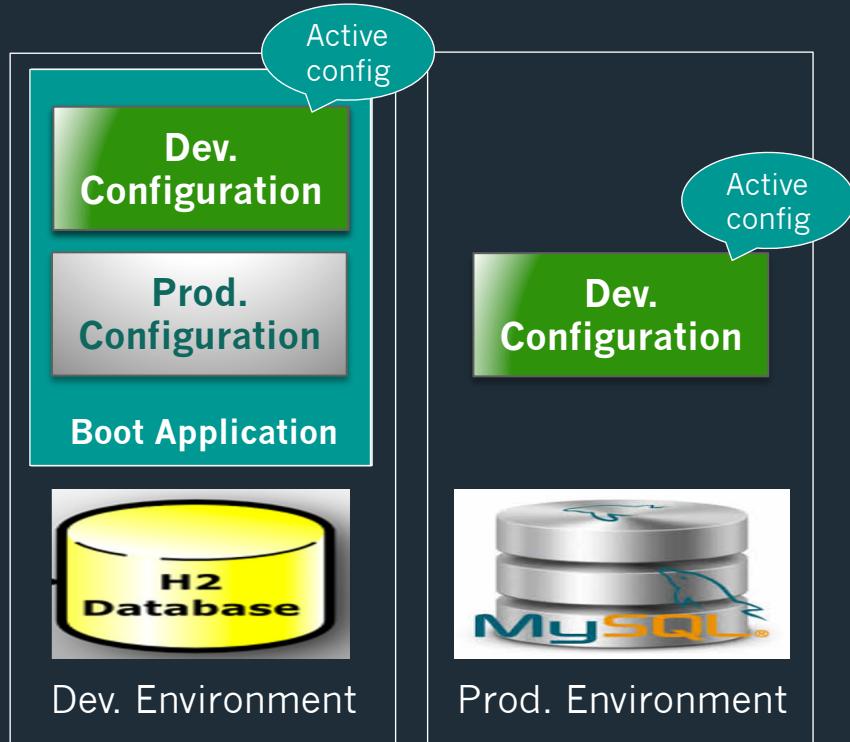
@Profile

Mark @Component or @Configuration with @Profile to limit when it is loaded.

```
@Configuration  
@Profile("development")  
public class DevelopmentConfiguration {  
    // The @Profile requires that the "development"  
    // profile be active at runtime  
    // for this configuration to be applied.  
    // Activate this profile in  
    // properties file with spring.profile.active=development  
    // command line with --spring.profiles.active=development  
}
```

```
@Configuration  
@Profile("production")  
public class ProductionConfiguration {  
    // The @Profile requires that the "production"  
    // profile be active at runtime  
    // for this configuration to be applied.  
    // Activate this profile in  
    // properties file with spring.profile.active=production  
    // command line with --spring.profiles.active=production  
}
```

Profiles



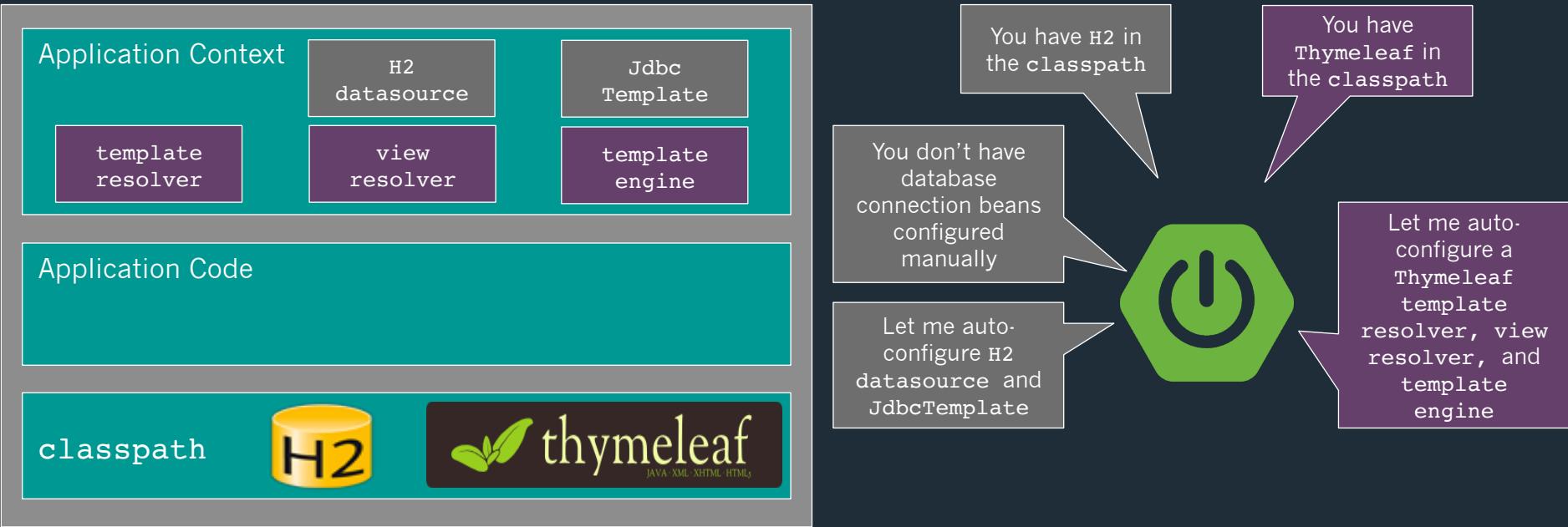
Checkout code from source control
Build an executable jar
Configuration activated based on active profile
Deploy anywhere
No Code Changes
No Configuration Changes



Pivotal™

Auto-Configuration

Automatically configures application based on the `classpath`



Override Auto-Configuration

Explicitly exclude auto-configuration

Application Context

template
resolver

view
resolver

template
engine

Application Code

```
@EnableAutoConfiguration(exclude= {DataSourceAutoConfiguration.class})
```

classpath



You have H2 in
your classpath

You want to
explicitly exclude
auto-configuration
of the DataSoure

I'll step aside.

You have
Thymeleaf in
your classpath

Let me auto-
configure a
Thymeleaf
template
resolver, view
resolver, and
template
engine





DEMO



CF

Actuator

Offers production-ready features such as monitoring and metrics to improve operator efficiency

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Actuator

- Services exposed through
 - HTTP endpoints for Spring MVC Applications
 - shell interface
 - JMX Beans



Types of Actuator Endpoints

Actuator endpoints can be organized into three categories

Configuration

- /beans
- /autoconfig
- /env
- /configprops
- /controller

Metrics

- /metrics
- /trace
- /dump

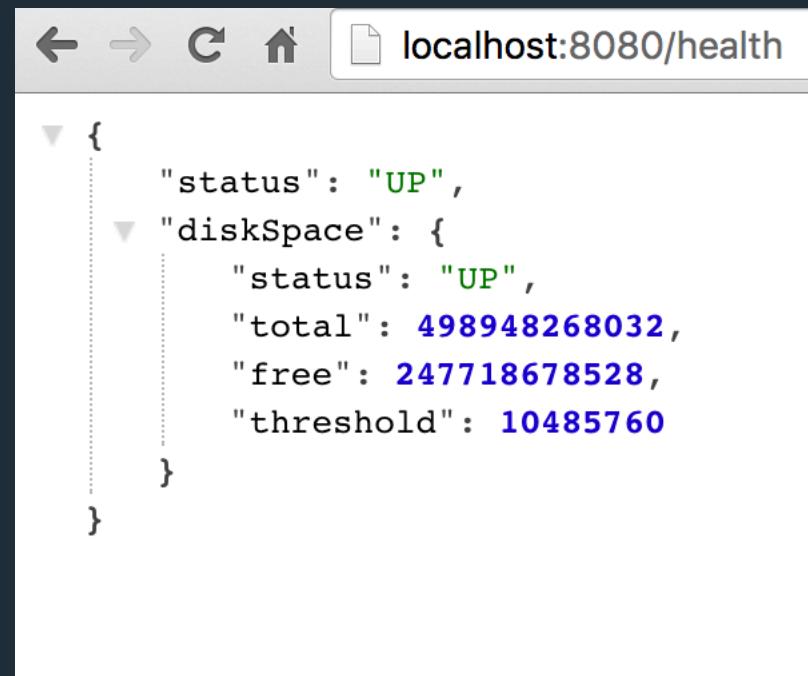
Miscellaneous

- /shutdown
- /info

/health

Reports health metrics for the application

- Each check returns a Health object
 - status
 - UP
 - DOWN
 - UNKNOWN
 - OUT_OF_SERVICE
 - Possibly with details
- Plugin own health definitions



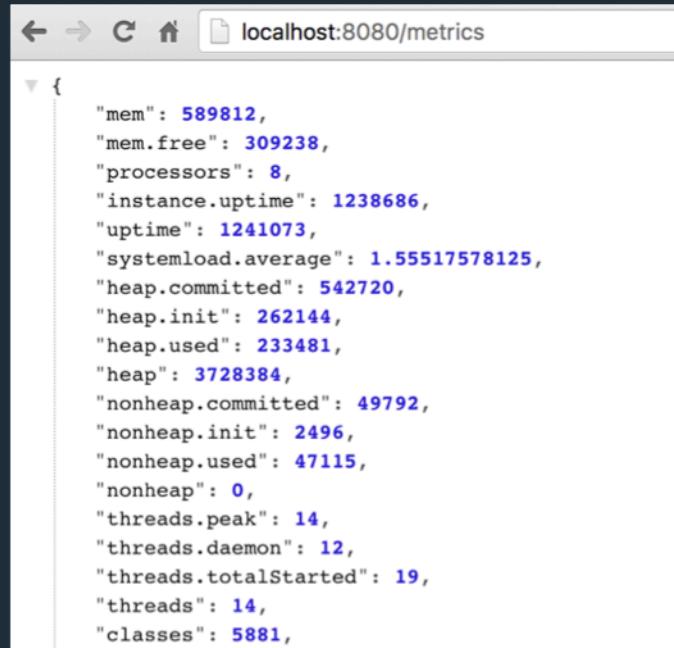
A screenshot of a web browser window displaying the JSON output of the /health endpoint. The URL in the address bar is "localhost:8080/health". The JSON response is as follows:

```
{  
  "status": "UP",  
  "diskSpace": {  
    "status": "UP",  
    "total": 498948268032,  
    "free": 247718678528,  
    "threshold": 10485760  
  }  
}
```

/metrics

Reports application metrics such as

- Memory usage and
- HTTP request counters



A screenshot of a web browser window displaying the JSON output of the `/metrics` endpoint. The URL in the address bar is `localhost:8080/metrics`. The JSON response contains various system and application metrics, many of which are highlighted in blue, likely indicating they are links to more detailed information or metrics.

```
{  
    "mem": 589812,  
    "mem.free": 309238,  
    "processors": 8,  
    "instance.uptime": 1238686,  
    "uptime": 1241073,  
    "systemload.average": 1.55517578125,  
    "heap.committed": 542720,  
    "heap.init": 262144,  
    "heap.used": 233481,  
    "heap": 3728384,  
    "nonheap.committed": 49792,  
    "nonheap.init": 2496,  
    "nonheap.used": 47115,  
    "nonheap": 0,  
    "threads.peak": 14,  
    "threads.daemon": 12,  
    "threads.totalStarted": 19,  
    "threads": 14,  
    "classes": 5881,  
}
```

DEMO



CF

Spring Boot & Cloud Foundry

Pivotal™

DEMO



CF

HATEOS

Spring Data Rest

DEMO



CF





Generate a with Spring Boot

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web

Full-stack web development with Tomcat and Spring MVC

Rest Repositories

Exposing Spring Data repositories over REST via spring-data-rest-webmvc

Vaadin

Vaadin

WS

Contract-first SOAP service development with Spring Web Services

Jersey (JAX-RS)

the Jersey RESTful Web Services framework

Don't know what to look for? Want more options? [Switch to the full version.](#)

PCF **Metrix**
Pivotal**BigDataSuite**
Pivotal**DataScience**

