```xml
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.0.RELEASE</version>
</parent>
```

Spring Boot Dependency Management

BOM - bill of materials

Example - spring-core 4.2.3 works well with logback-core 1.1.3

Other Spring Boot Initializers

Web initializer

http://start.spring.io

Command line

Spring boot CLI

How Does Spring Boot Work?

Java

Main method entry point

Spring Application

Spring context

Spring environment

Initializers

Embedded Server

Default is Tomcat

Auto configured

```
public static void main( … )
```

◀ Starts Java and then the application

```
@SpringBootApplication
```

◀ A convenience annotation that wraps commonly used annotations with Spring Boot

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
```

◀ Spring configuration on startup

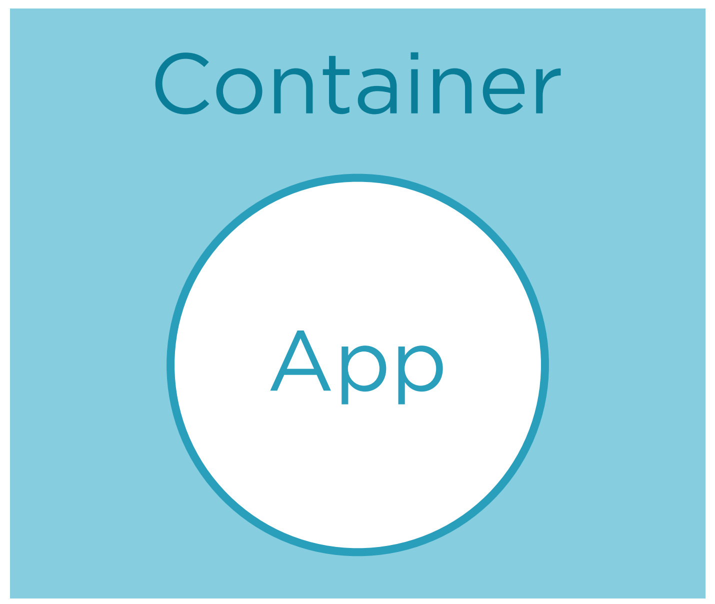
◀ Auto configures frameworks

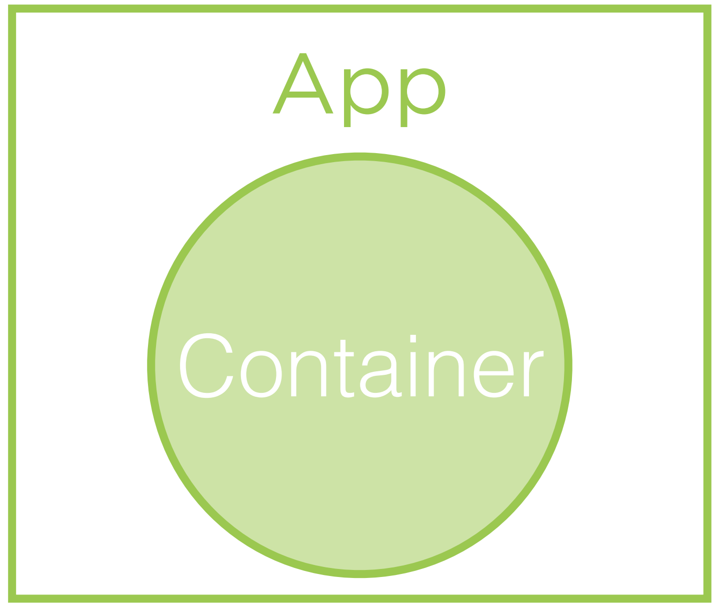
◀ Scans project for Spring components

```
SpringApplication.run( … );
```

◀ Starts Spring, creates spring context, applies annotations and sets up container

Why Move to Containerless Deployments?





Container Deployments

- Pre-setup and configuration
- Need to use files like web.xml to tell container how to work
- Environment configuration is external to your application

Application Deployments

- Runs anywhere Java is setup (think cloud deployments)
- Container is embedded and the app directs how the container works
- Environment configuration is internal to your application

Summary

Created a fully working Java web application from scratch!

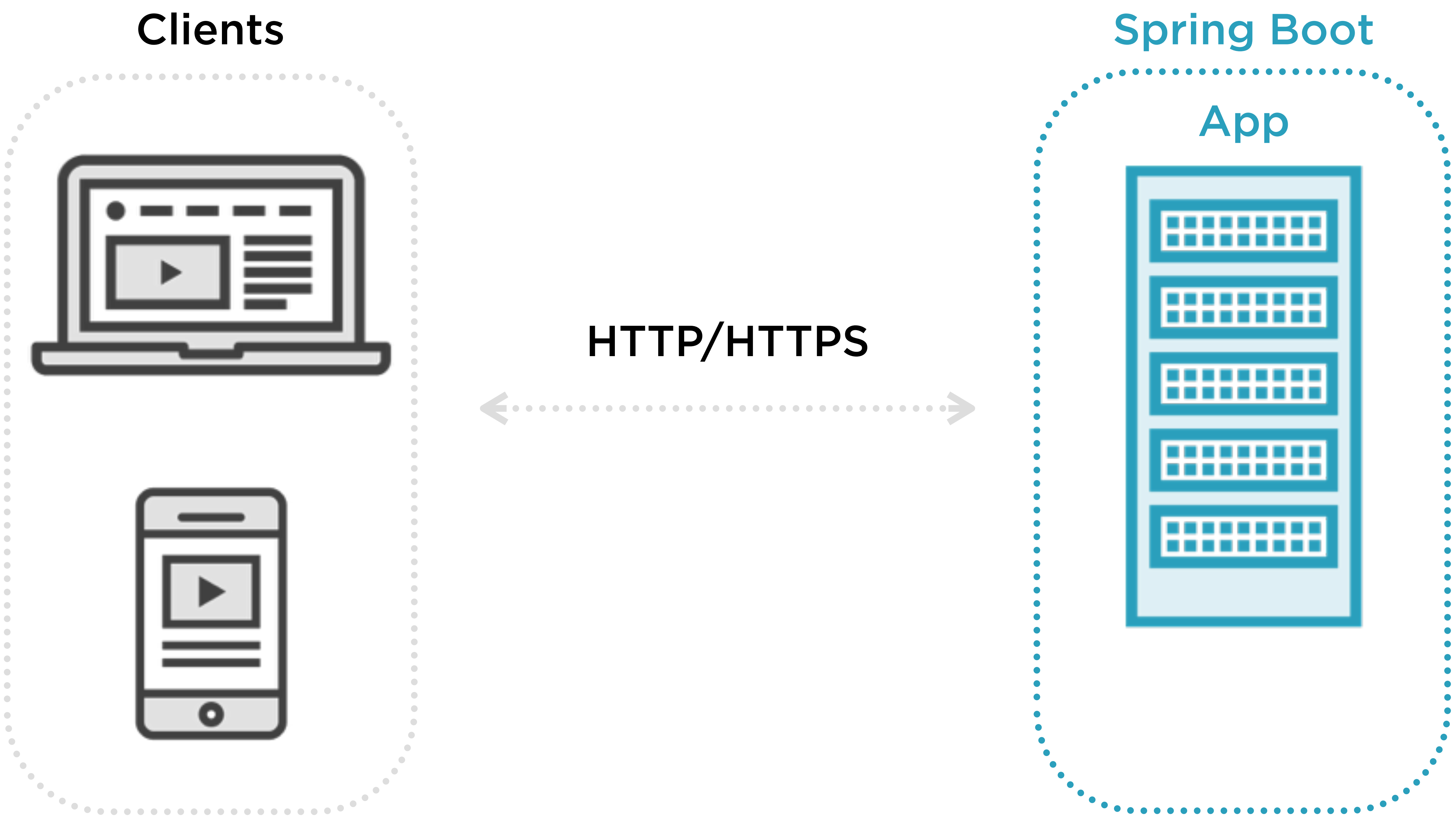
Spring Boot builds and Spring Boot bill of materials (BOM)

Spring Boot Initializers

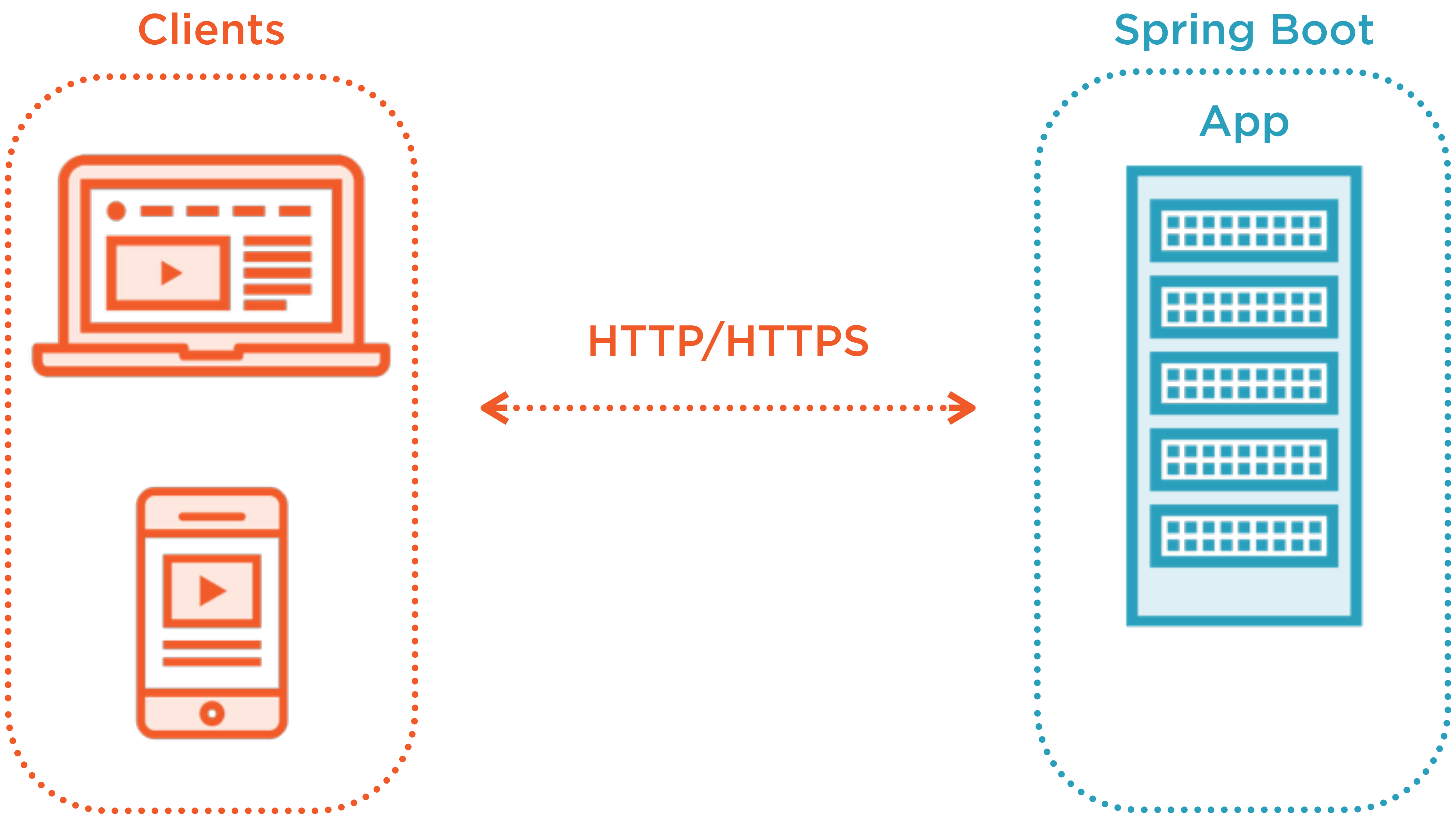
How Spring Boot really works

- Plain Java program
- Spring context initialization
- Embedded container and container less deployments

RESTful Web App



RESTful Web App



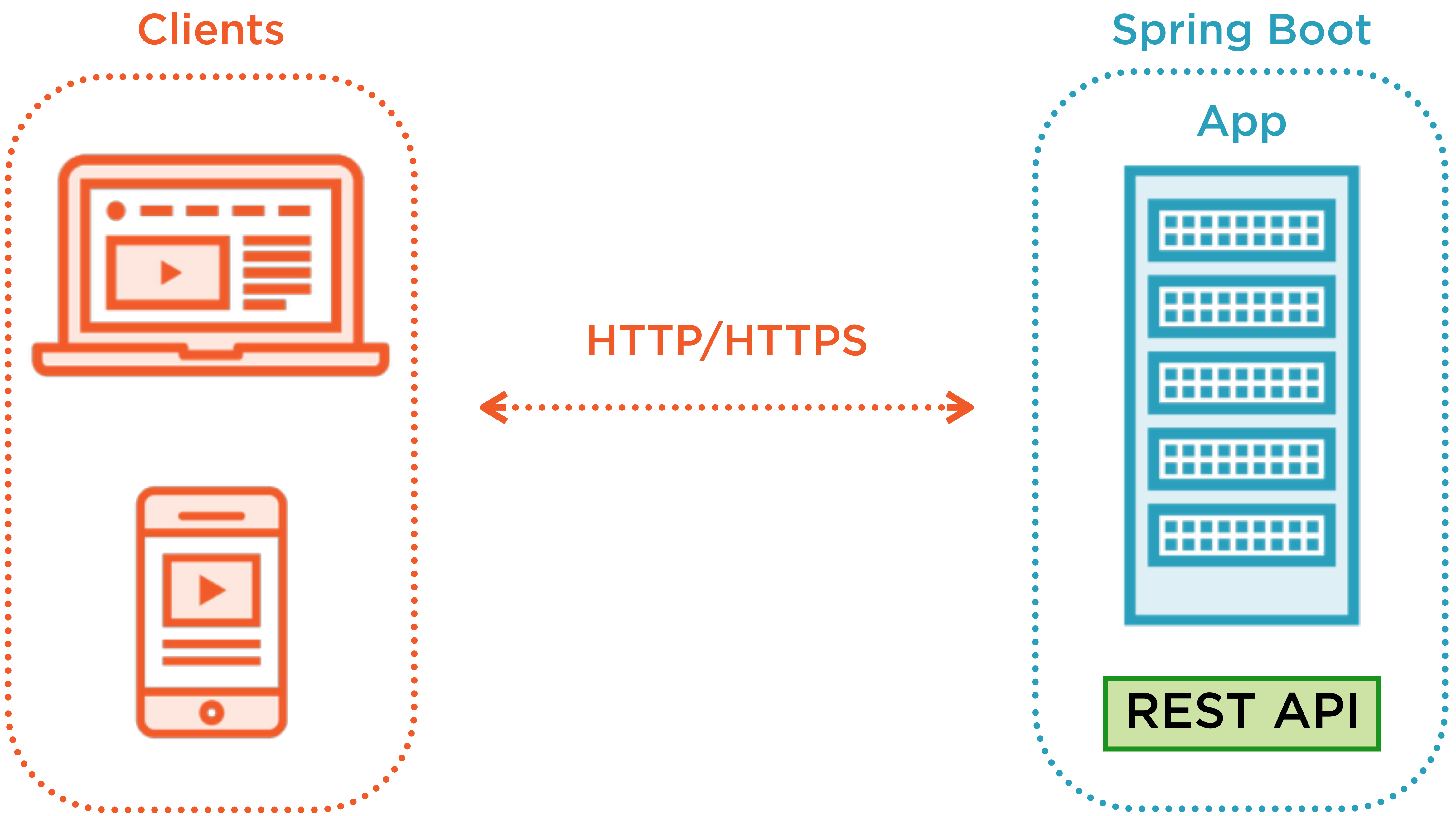
Demo

git clone https://github.com/dlbunker/ps-spring-boot-resources.git

Default static content locations

- classpath
  - /static
  - /public
  - /resources
  - /META-INF/resources

RESTful Web App



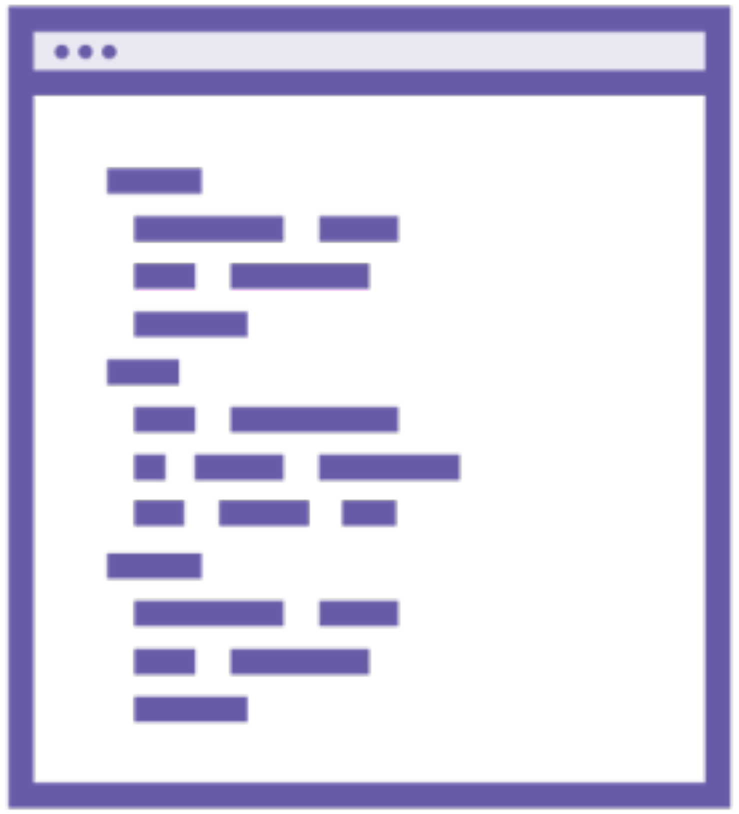
Demo

Spring MVC REST Controller

ngResource for “shipwreck”

- GET /api/v1/shipwrecks (list)
- POST /api/v1/shipwrecks (add)
- GET /api/v1/shipwrecks/{id} (view)
- PUT /api/v1/shipwrecks/{id} (udpate)
- DELETE /api/v1/shipwrecks/{id} (delete)

Spring MVC Integration Overview



spring-boot-starter-web in pom.xml

Sets up ViewResolvers

Sets up static resource serving

Sets up HttpMessageConverter

Sets up customizable hooks

Properties and Environmental Configuration

application.properties

- Place on classpath root
- YAML or Properties format

Environmental configuration

- application-{profile}.properties
- application-dev.properties
- application-prod.properties

Demo

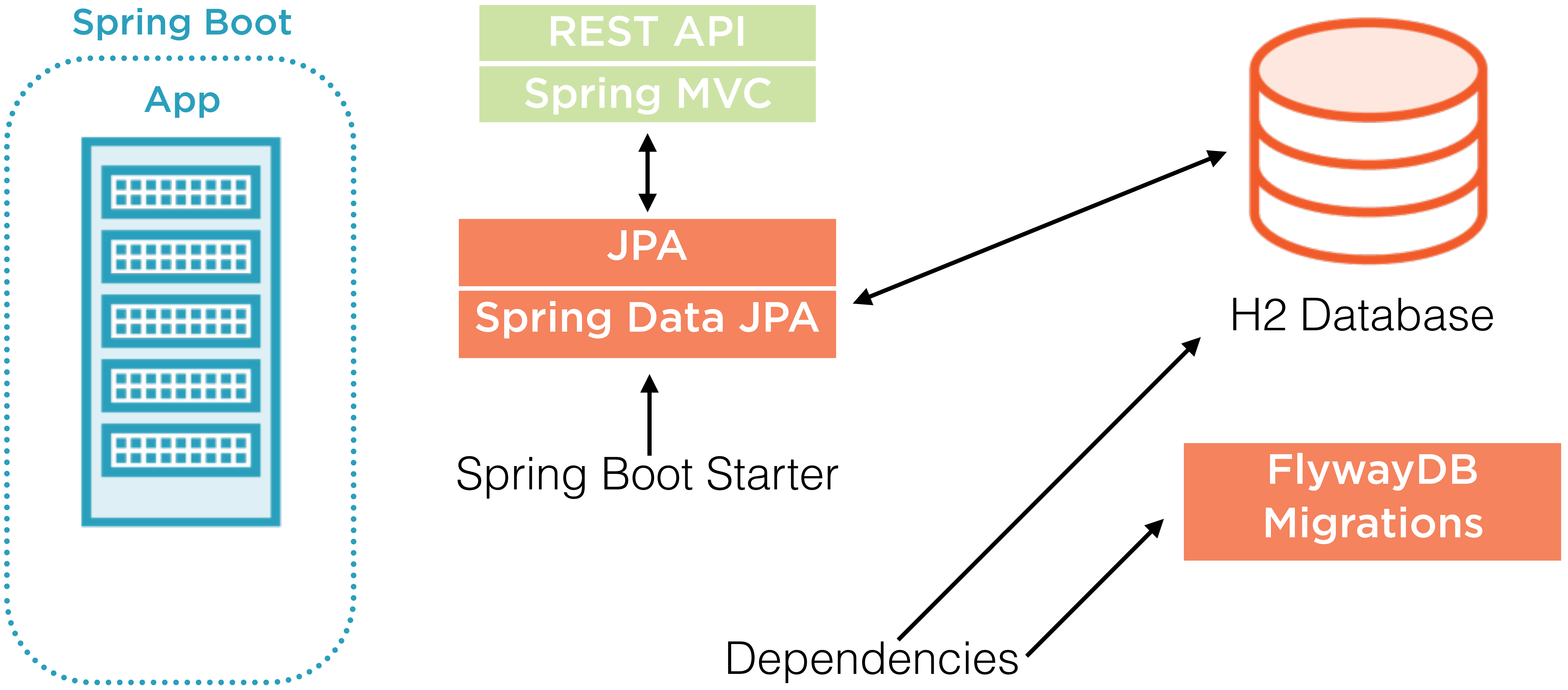
Create application.properties

Customize embedded container

Setup environment profiles

Configuring and Accessing a Data Source

Identifying Frameworks for Integration



Demo

H2 dependency

Spring Boot Starter Data JPA

Demo

DataSource configuration

DataSource pooling

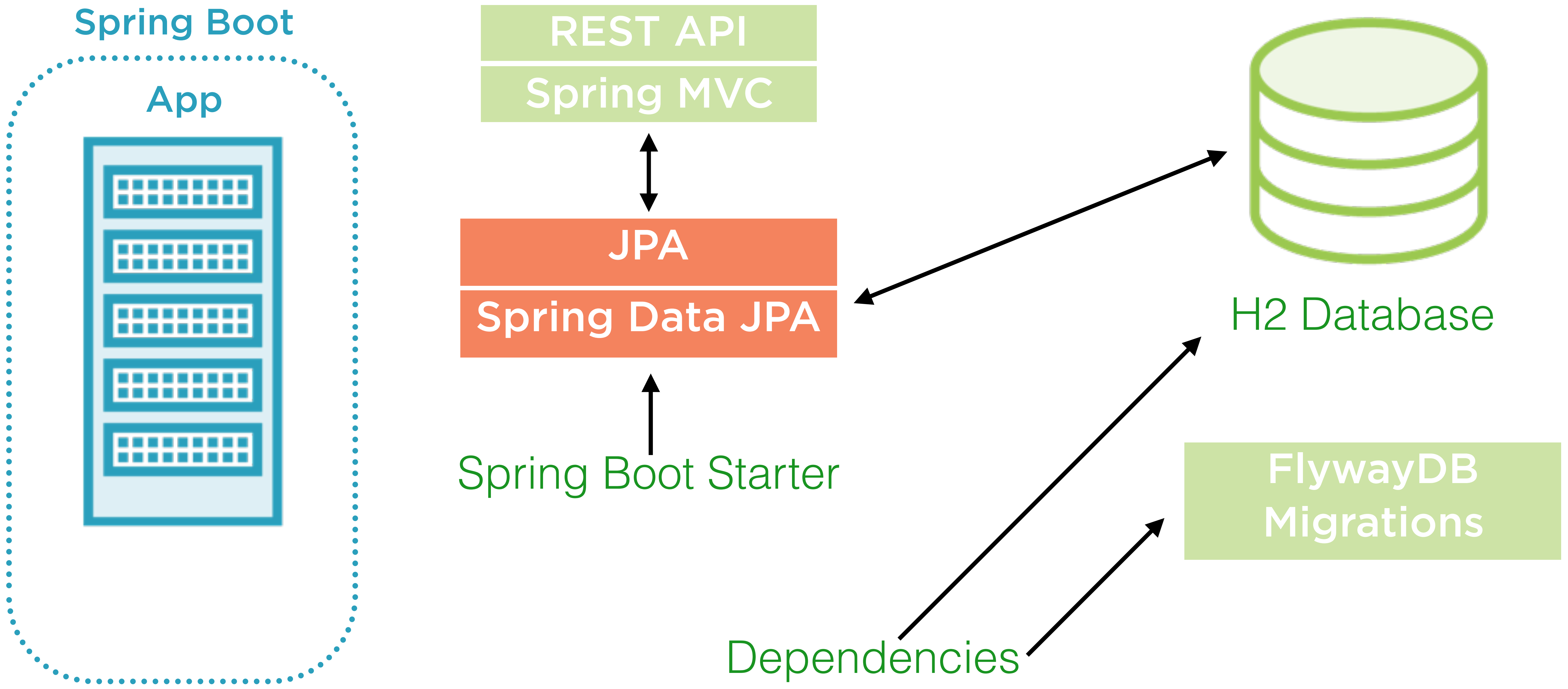
- tomcat-jdbc is default pooling strategy

```java
@Configuration
public class PersistenceConfiguration {
  @Bean
  @ConfigurationProperties(prefix="spring.datasource")
  public DataSource dataSource() {
    return DataSourceBuilder.create().build();
  }
}
```

Spring Boot Java Configuration

Define Spring beans using Java

Demo: Adding JPA and Spring Data JPA



Testing the Spring Boot Project

Getting Started with Spring Boot Testing

spring-boot-starter-test

Always start with the starter

JUnit

For all your unit testing needs (http://junit.org)

Hamcrest

Matching and assertions (http://hamcrest.org)

Mockito

Mock objects and verify (http://mockito.org)

Spring Test

Testing tools and integration testing support

Demo

Add the spring-boot-starter-test dependency

Construct a test

Running tests

Demo

Declarative, readable matching rules

Integration Testing Challenges

Traditional Spring Apps	Spring Boot Apps
Containers are difficult to test	No container, easier to start app
Spring Context needs to be available	Spring Context auto configuration
App/Test startup can be slow	App/Test startup can be slow
Database state needs to be consistent	Database state needs to be consistent

Demo

@RunWith(SpringJUnit4ClassRunner.class)

@SpringApplicationConfiguration

Demo

Web integration test == calling REST API

@WebIntegrationTest

Summary

spring-boot-starter-test dependency

JUnit, Hamcrest, Mockito, Spring Test

Basic unit test

Mocked unit test

Hamcrest result matching and assertions

Integration test

Web integration test