SpringOne Platform

Cloud Native Java with Spring Cloud Services

Roy Clarkson
@royclarkson

Craig Walls
@habuma

Pivotal

+

NETFLIX | OSS

=

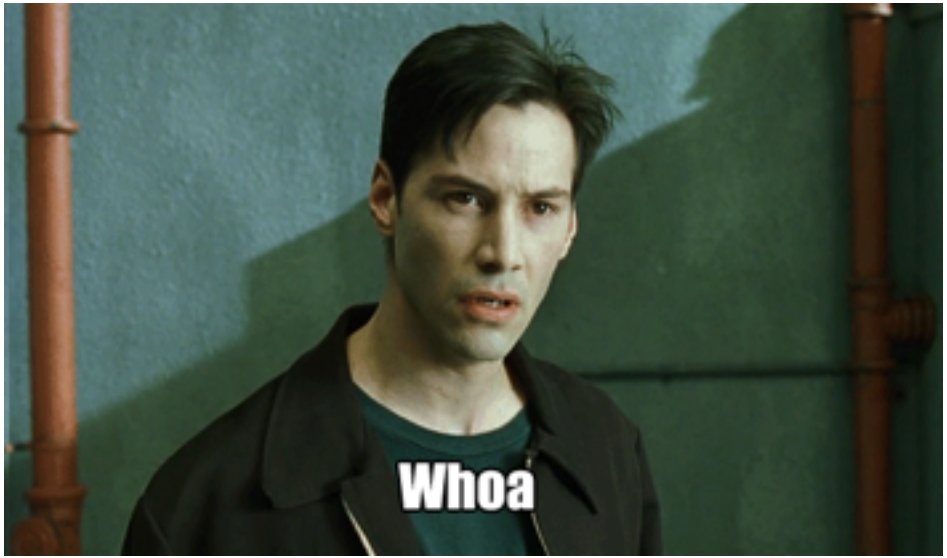
?

Pivotal
Cloud Foundry®

Spring Cloud Demo

I know Kung Fu

- Eureka Server (Spring Cloud Netflix)
- Config Server (Spring Cloud Config)
- Hystrix Dashboard (Spring Cloud Netflix)
- App Service
- App UI



- ease of use
- security
- reliability

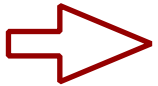
Config Server

Service Registry

Circuit Breaker Dashboard

Create a Spring Cloud server

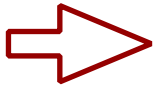
1. create a Spring Boot app
2. add Spring Cloud starters
3. add @EnableXxxxServer
4. add configuration
5. add security
6. deploy and manage



1. `cf create-service`

Create a Spring Cloud client application

1. create a Spring Boot app
2. add Spring Cloud starters
3. add configuration
   - Eureka-first or config-first bootstrapping?
4. add security

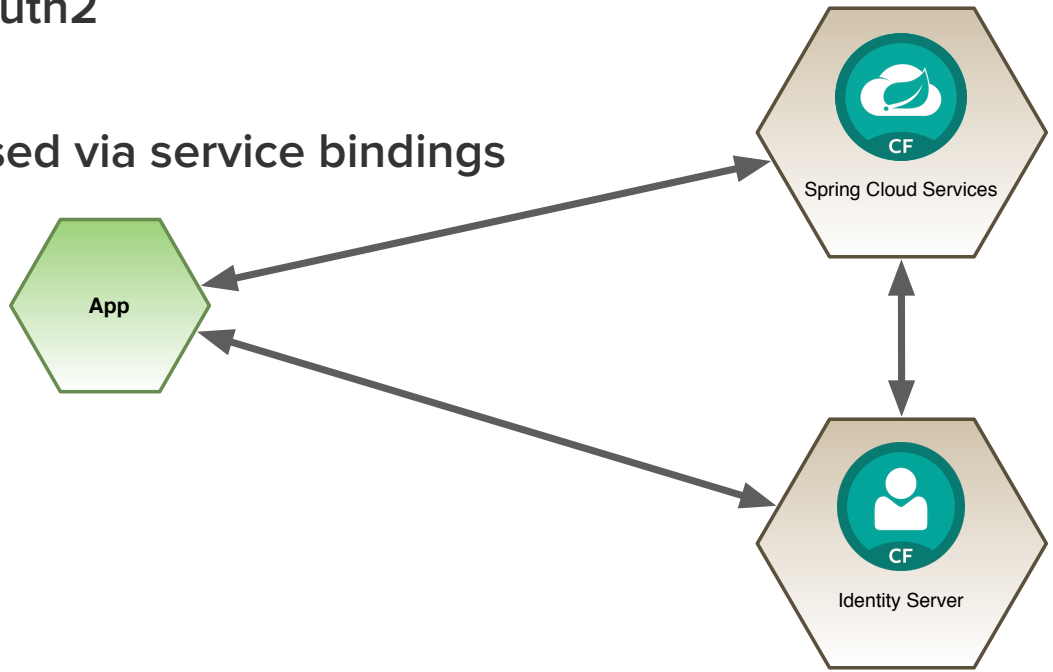


1. create a Spring Boot app
2. add Spring Cloud Services starters
3. `cf bind-service`

Secure access

Services are secured with OAuth2

OAuth2 credentials are exposed via service bindings



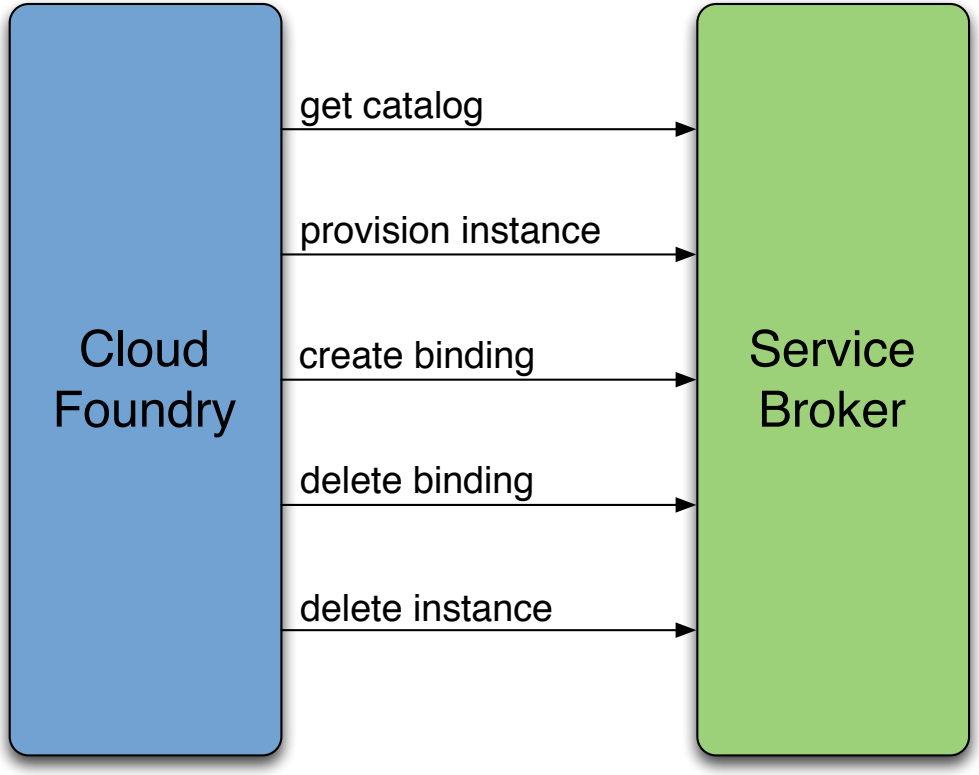
Reliability

High availability and health management are provided by the platform

Service Brokers

Service brokers extend Cloud Foundry to provide resources that can be consumed by applications

http://docs.cloudfoundry.org/services/



Marketplace

App Autoscaler

Scales bound applications in response to load

RabbitMQ

RabbitMQ is a robust and scalable high-performance multi-protocol messaging broker.

MySQL for Pivotal Cloud Foundry

MySQL service for application development and testing

Circuit Breaker

Circuit Breaker Dashboard for Spring Cloud Applications

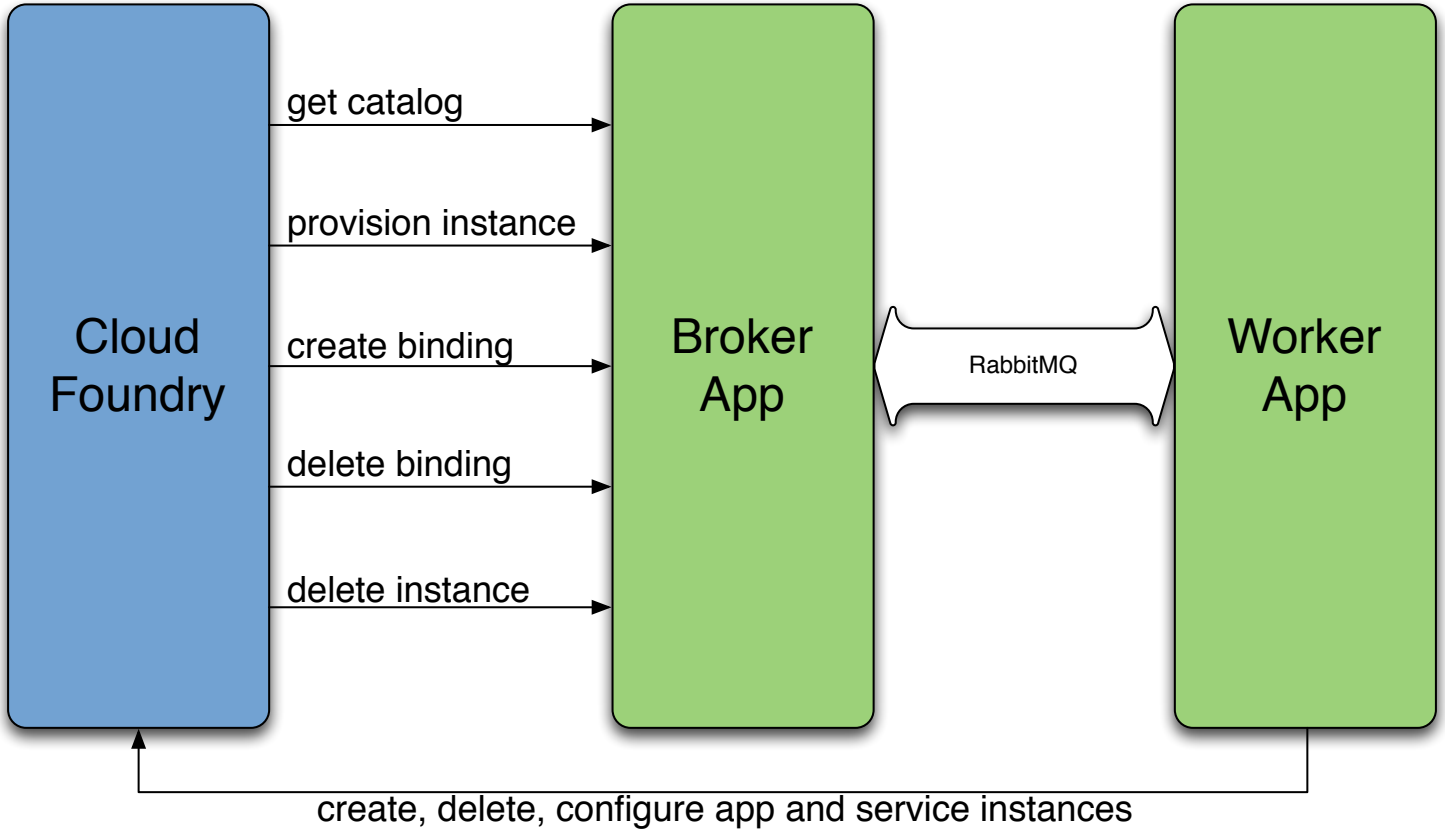
Config Server

Config Server for Spring Cloud Applications

Service Registry

Service Registry for Spring Cloud Applications

Spring Cloud Services Broker



SCS is built on Spring Projects

Spring Boot
Spring Cloud Netflix
Spring Cloud Config Server
Spring Cloud Connectors

Spring Security
Spring Security OAuth2
Spring Data JPA
Spring AMQP

Spring Boot CF Service Broker
Cloud Foundry Java Client





Config Server

Overview

Service

- deploys a Spring Cloud Config server for each service instance provisioned
- git backend
- search paths within config repo
- basic authentication to repo

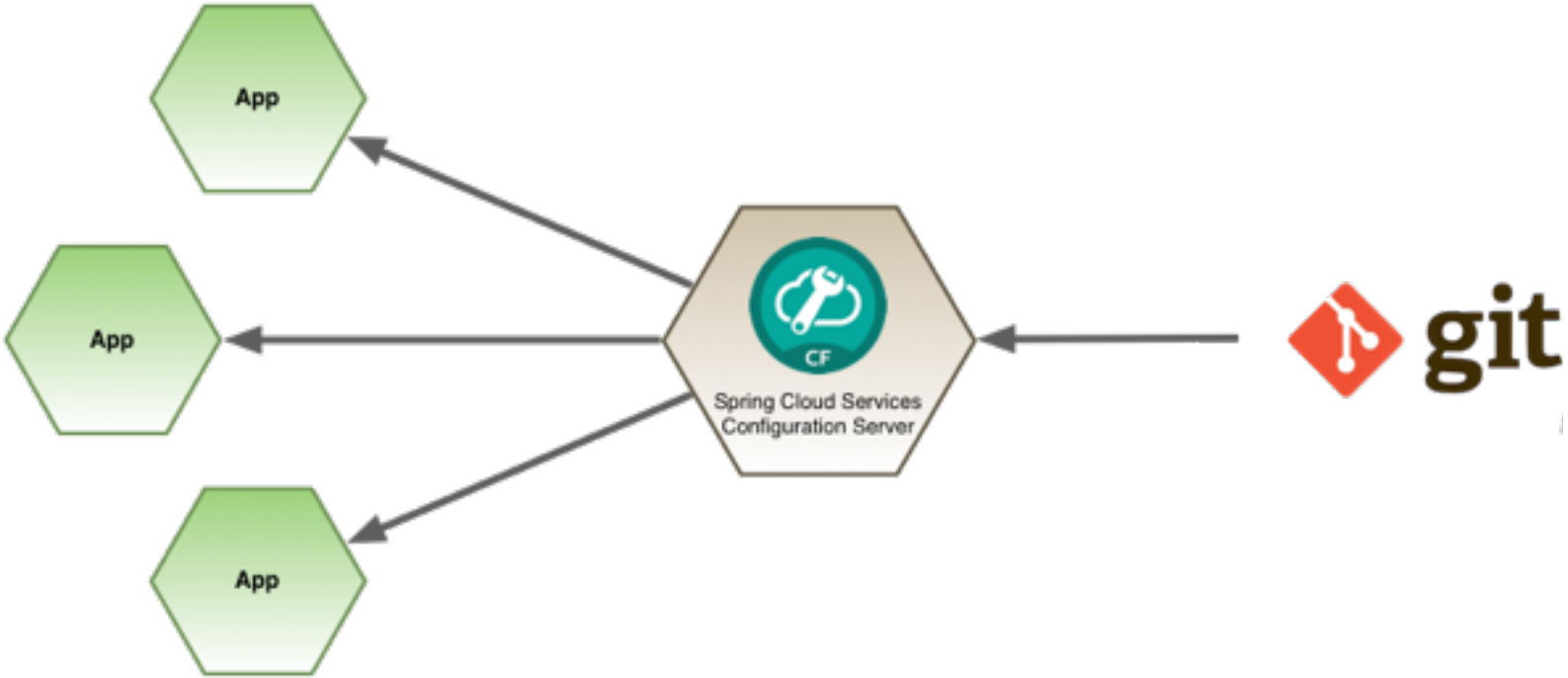
Security

- only bound applications are able to see configuration

Client Applications

- starter automatically adds a property source configured for the config server

Configuration Server flow



Dependencies

Spring Cloud:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-config-client</artifactId>
</dependency>
```

Spring Cloud Services:

```
<dependency>
  <groupId>io.pivotal.spring.cloud</groupId>
  <artifactId>spring-cloud-services-starter-config-client</artifactId>
</dependency>
```

Service Registry

Overview

Service

- deploys a Spring Cloud Netflix Eureka server for each service instance provisioned

Security

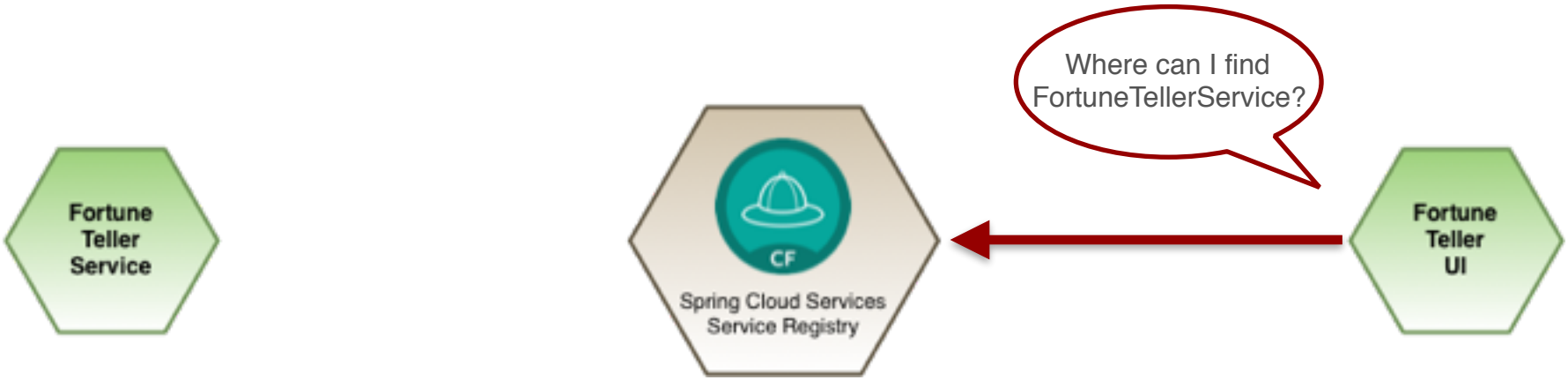
- only the application that originally creates a registration is allowed to update or remove that registration
  - prevents Man In The Middle attacks against the Registry

Client Applications

- starter automatically configures a DiscoveryClient

Service Registry Flow

Service Registry Flow



Service Registry Flow

Service Registry Flow

Dependencies

Spring Cloud:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-eureka</artifactId>
</dependency>
```

Spring Cloud Services:

```
<dependency>
  <groupId>io.pivotal.spring.cloud</groupId>
  <artifactId>spring-cloud-services-starter-service-registry</artifactId>
</dependency>
```

Circuit Breaker Dashboard

Overview

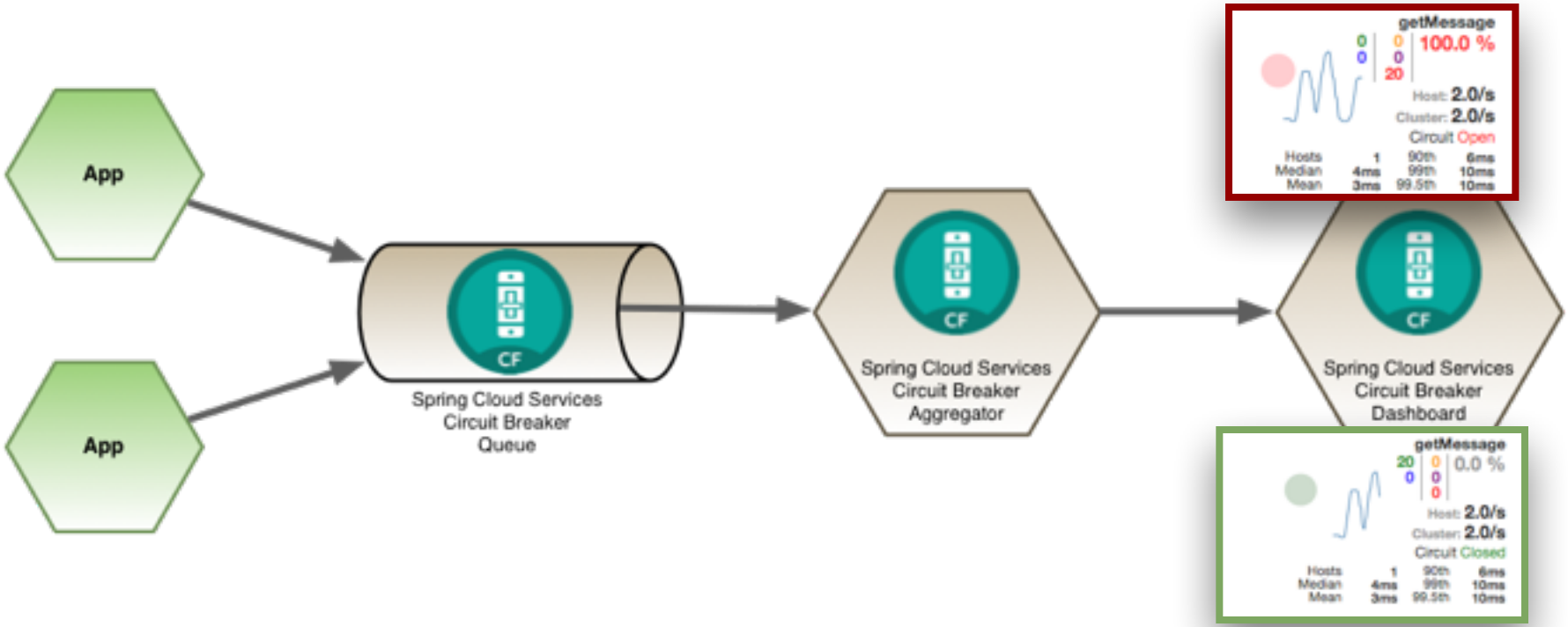
Service

- deploys for each service instance provisioned:
  - Spring Cloud Netflix Hystrix dashboard
  - Spring Cloud Netflix Turbine server
  - Pivotal RabbitMQ service instance

Client Applications

- starter automatically configures a Turbine AMQP client

Circuit Breaker Dashboard Flow



Dependencies

Spring Cloud:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-hystrix</artifactId>
</dependency>
```

Spring Cloud Services:

```
<dependency>
  <groupId>io.pivotal.spring.cloud</groupId>
  <artifactId>spring-cloud-services-starter-circuit-breaker</artifactId>
</dependency>
```

```
@Configuration
public class CloudConfig extends CloudConnectorsConfig {
    @Bean
    @HystrixConnectionFactory
    public ConnectionFactory hystrixConnectionFactory() {
        return connectionFactory().hystrixConnectionFactory();
    }
}
```

```
@Configuration
public class RabbitConfig extends AbstractCloudConfig {
    @Bean
    @Primary
    public ConnectionFactory rabbitConnectionFactory() {
        return connectionFactory().rabbitConnectionFactory();
    }
}
```

Spring Cloud Services Dependencies

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.pivotal.spring.cloud</groupId>
      <artifactId>spring-cloud-services-dependencies</artifactId>
      <version>1.1.1.RELEASE</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Brixton.SR3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Spring Cloud Services Dependencies

```
<dependencies>
  <dependency>
    <groupId>io.pivotal.spring.cloud</groupId>
    <artifactId>spring-cloud-services-starter-config-client</artifactId>
  </dependency>
  <dependency>
    <groupId>io.pivotal.spring.cloud</groupId>
    <artifactId>spring-cloud-services-starter-service-registry</artifactId>
  </dependency>
  <dependency>
    <groupId>io.pivotal.spring.cloud</groupId>
    <artifactId>spring-cloud-services-starter-circuit-breaker</artifactId>
  </dependency>
</dependencies>
```

Enabling Spring Cloud

```
@SpringBootApplication
@EnableJpaRepositories
@EnableDiscoveryClient
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Enabling Spring Cloud

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableCircuitBreaker
public class Application {

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
      return new RestTemplate();
    }

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Configuring Spring Cloud

```
spring:
  application:
    name: fortunes
```

```
spring:
  application:
    name: ui
```

What's new in SCS 1.1

- Spring Cloud Brixton Release Train and Spring Boot 1.3 Support
- Asynchronous Service Provisioning and Zero Downtime Upgrades/Updates
- Highly-Available Topologies for Config Server and Service Registry
- Enhancements to the Config Server Git Backend

https://blog.pivotal.io/pivotal-cloud-foundry/products/spring-cloud-services-1-1-now-available

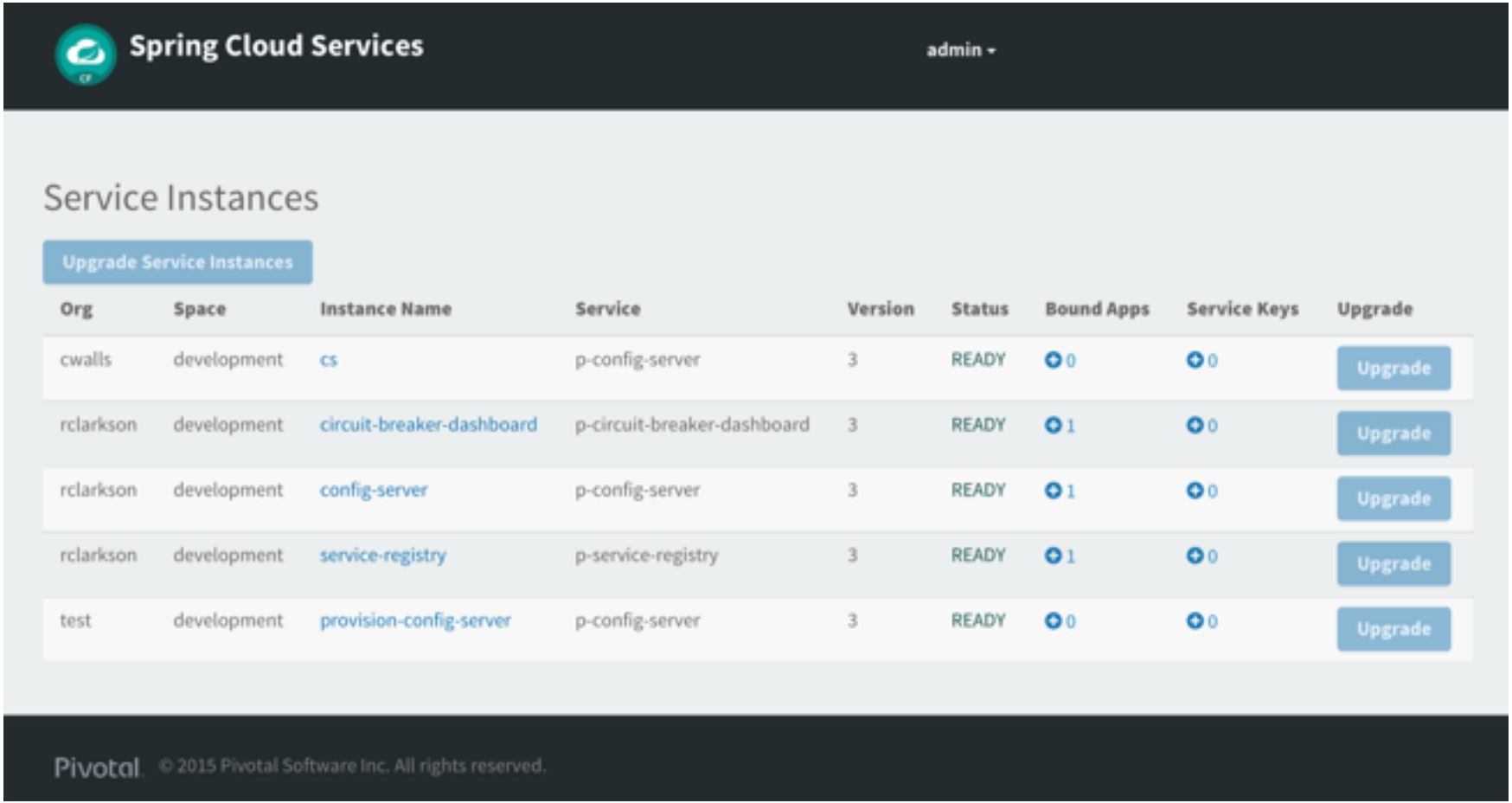
Spring Cloud Brixton and Spring Boot 1.3

```
// define a Ribbon enabled RestTemplate
@Bean
@LoadBalanced
public RestTemplate restTemplate() {
  return new RestTemplate();
}
```

```
// inject the instance into your controller
@Autowired
RestTemplate rest;
```

Zero Downtime Upgrades & Updates

- Upgrades to the SCS tile via Pivotal Cloud Foundry® Operations Manager
- Upgrades to SCS service instances



Highly Available Topologies

- Config Servers are scaled horizontally via the Cloud Foundry API.
- Service Registries are stateful components that replicate when clustered. The SCS service broker pushes the correct number of Eureka application instances and configures them for replication.
- Changes to HA configurations are executed such that no application instances experience downtime.

Config Server Enhancements

- Config Server now supports all features of the Spring Cloud Brixton Git backend, including multiple repositories, pattern matching, and placeholders.
- Git repositories can now be accessed:
  - using self-signed SSL certificates for HTTPS
  - via an HTTP or HTTPS proxy server
  - via the Git or SSH protocols (in addition to HTTP/HTTPS).

Spring Cloud Services Demo

Safe Harbor Statement

The following is intended to outline the general direction of Pivotal's offerings. It is intended for information purposes only and may not be incorporated into any contract. Any information regarding pre-release of Pivotal offerings, future updates or other planned modifications is subject to ongoing evaluation by Pivotal and is subject to change. This information is provided without warranty or any kind, express or implied, and is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions regarding Pivotal's offerings. These purchasing decisions should only be based on features currently available. The development, release, and timing of any features or functionality described for Pivotal's offerings in this presentation remain at the sole discretion of Pivotal. Pivotal has no obligation to update forward looking information in this presentation.

SpringOne Platform

Learn More. Stay Connected.

 @springcentral
spring.io/blog

 @pivotal
pivotal.io/blog

 @pivotalcf
http://engineering.pivotal.io