Pivotal

Cloud Native Applications - Spring Cloud Services

Derrick Chua
Senior Platform Architect
tchua@pivotal.io
April 2019

Cloud Native Architectures

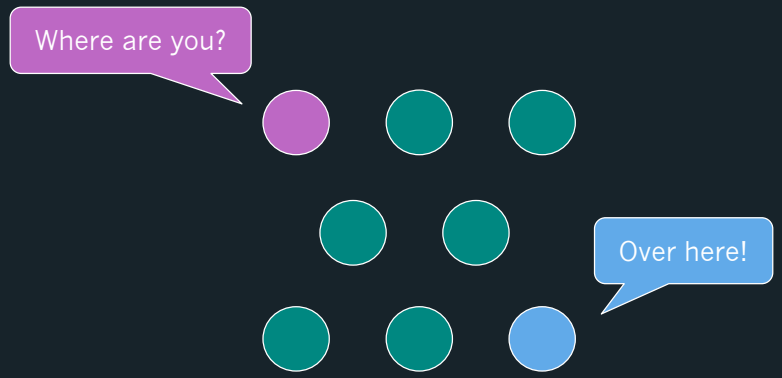
Light Side of the Cloud

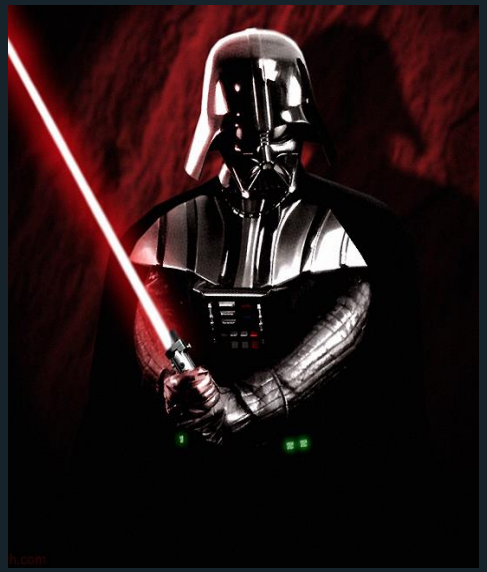
- Scalability
- High Availability
- Velocity: Continuous Delivery
- On-Demand Provisioning

Pivotal™

Cloud Native Architectures

Dark Side of the Cloud: Finding Services

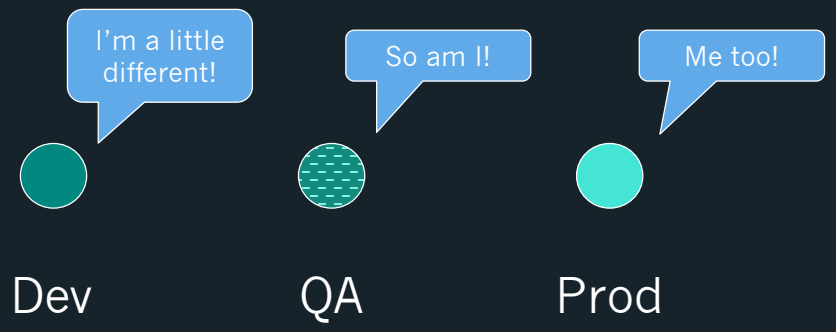


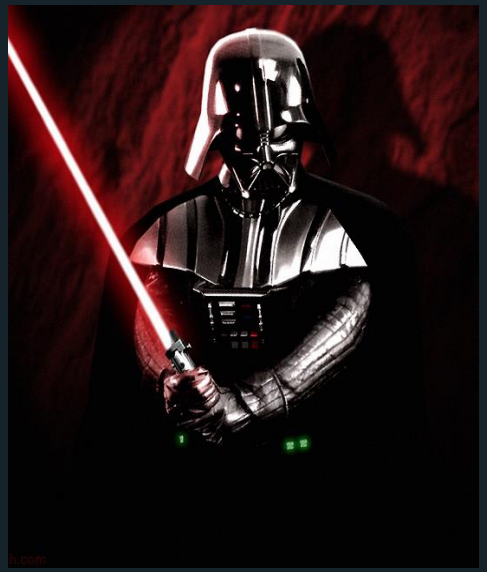


Pivotal

Cloud Native Architectures

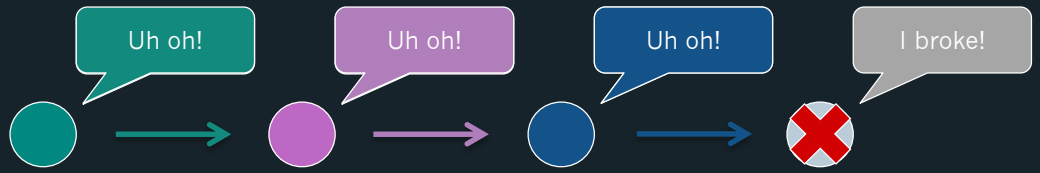
Dark Side of the Cloud: Managing Configuration Differences

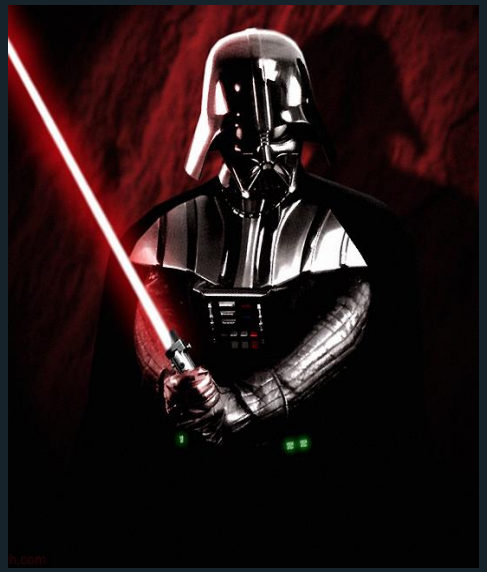




Cloud Native Architectures

Dark Side of the Cloud: Handling Failure





Pivotal

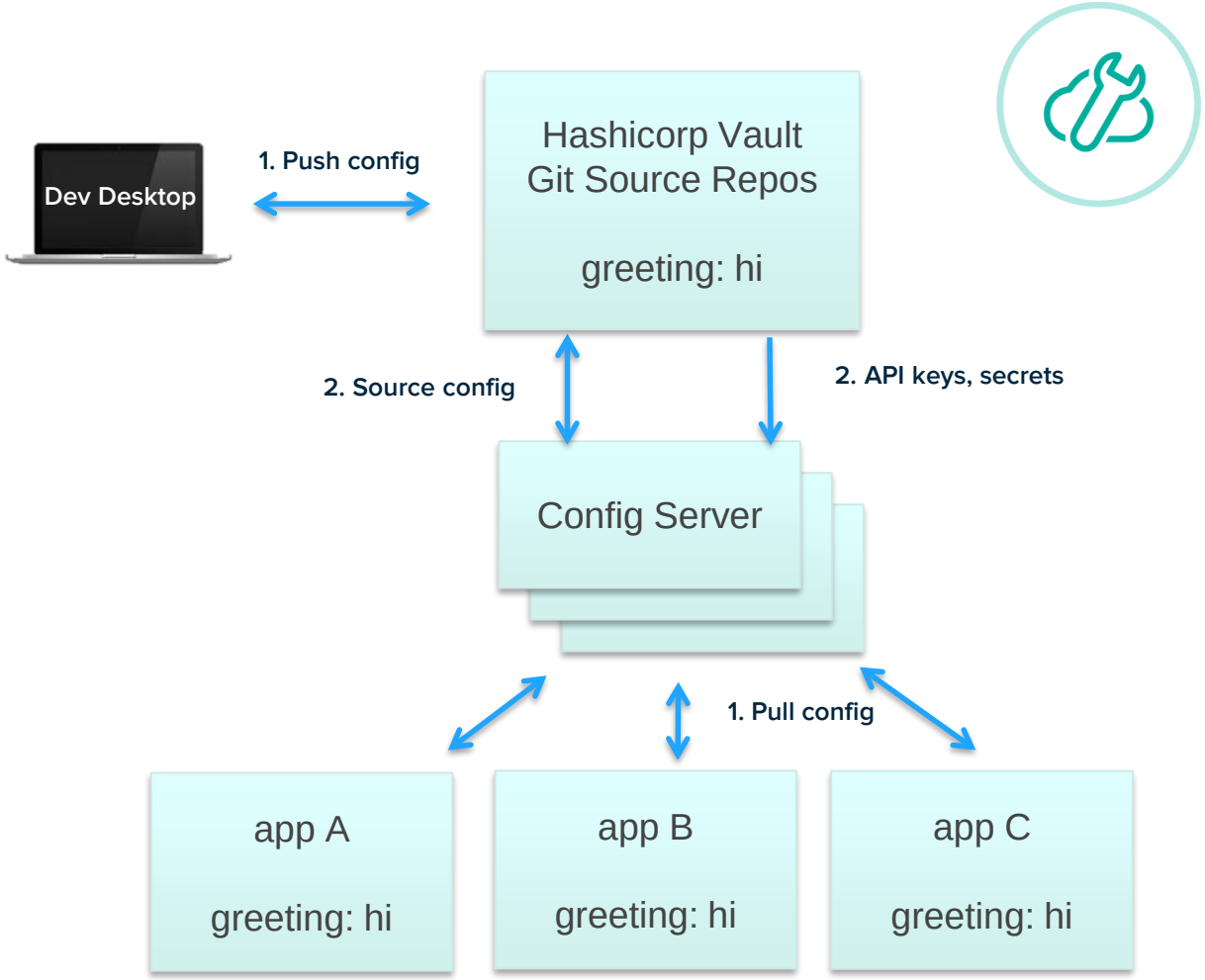
Netflix Cloud Libraries

- Netflix needed to be faster to win / disrupt
- Pioneer and vocal proponent of microservices – the key to their speed and success
- Netflix OSS supplies parts, but it's not a solution

Pivotal

Spring Cloud Config

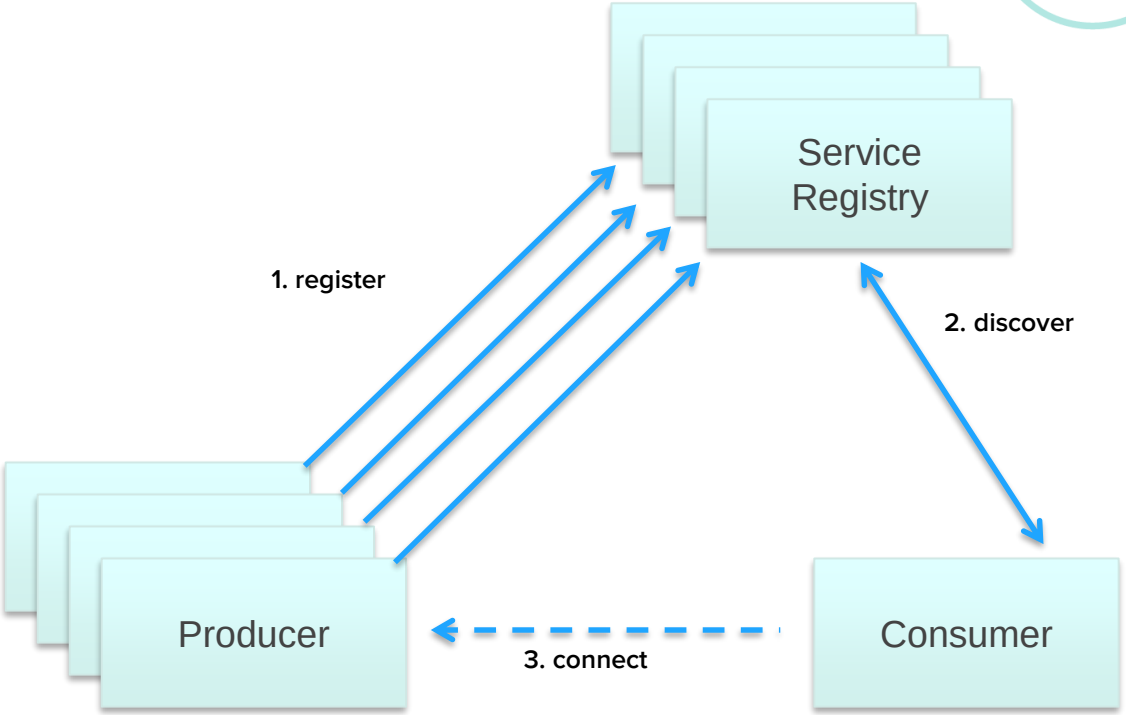
- Externalized configuration in a distributed system
- Centralized management of configuration across apps and environments



Pivotal

Spring Cloud Netflix - Eureka

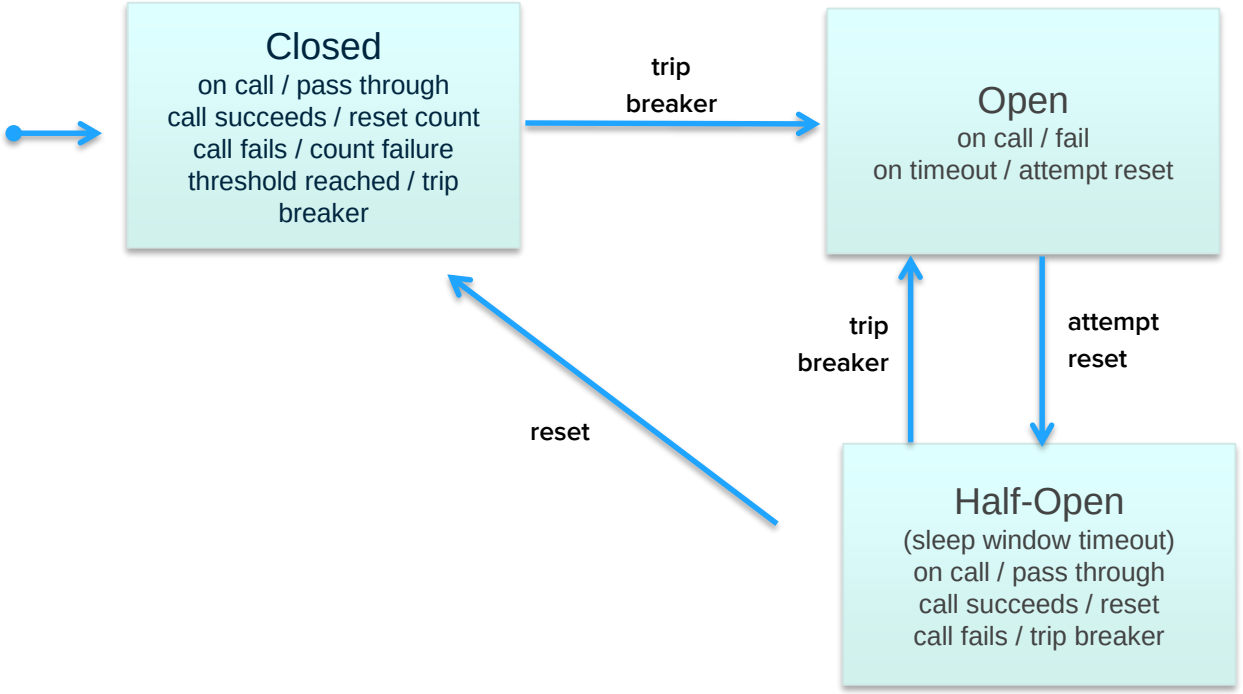
- Instance registration and discovery
- Embedded Eureka Server



Pivotal

Spring Cloud Netflix - Hystrix

Fault Tolerance Library for Distributed Systems



Pivotal

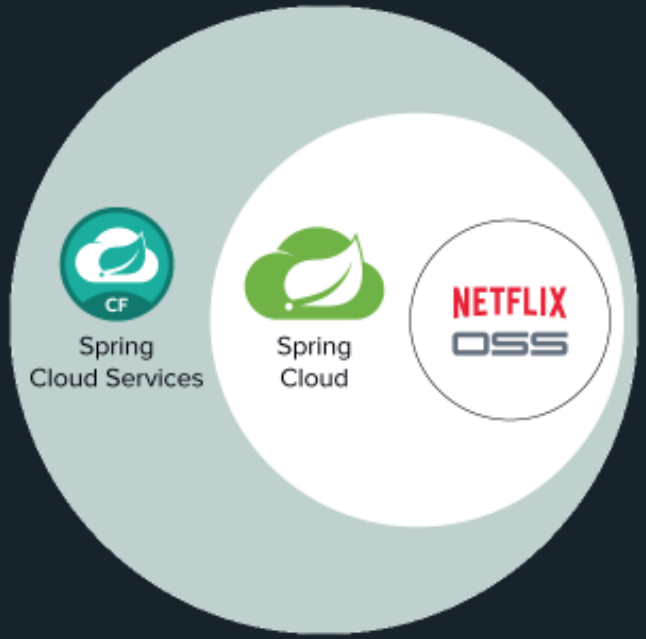
Spring Cloud Services

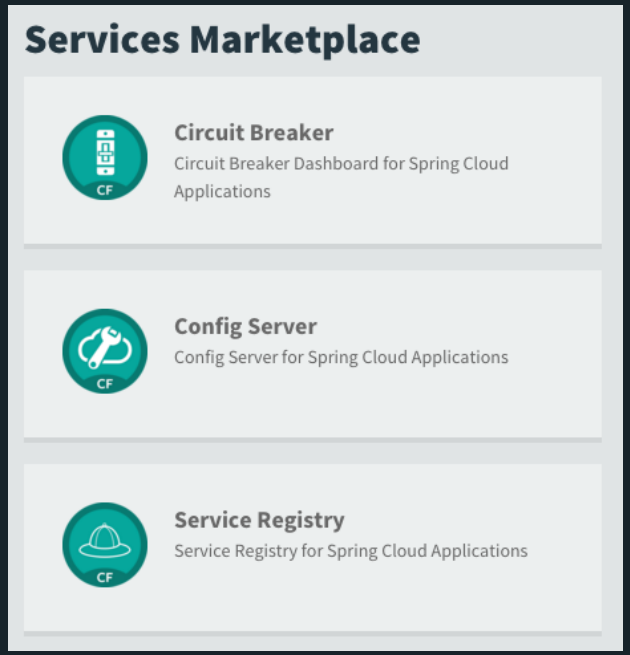
- Builds on the foundation of Spring Boot and Spring Cloud to simplify crucial operational patterns for microservices running on Pivotal Cloud Foundry.
- Packages server-side components of the popular Spring Cloud projects, including Spring Cloud Netflix (Eureka and Hystrix) and Spring Cloud Config, and makes them available as services in the Pivotal Services Marketplace

SPRING CLOUD SERVICES FOR PCF

Build, Operate, Scale and Secure Microservices on Pivotal Cloud Foundry

Spring Cloud Services





Pivotal™

Spring Cloud Services

Spring Cloud Services

Service Registry

- Service Registration and Discovery via Spring Cloud Netflix Eureka
- Registration via CF Route

Config Server

- Git URL for Config Repo provided via Service Dashboard (post-provisioning)
- Single tenant, scoped to CF space

Circuit Breaker

- Spring Netflix Hystrix
- Aggregation via AMQP (RabbitMQ)

Pivotal™

Spring Cloud & Spring Cloud Services (SCS)

Developing on the Desktop
vs.
Deploying in Production

DEV

PROD

Security: OAUTH2, TLS, PAS UAA integration, RBAC

Ops: BOSH release for Config Server, Service Registry, Circuit Breaker

Pivotal

SCS:
CF CLI Plugin

Spring Cloud Services integration for the CF Command Line Interface

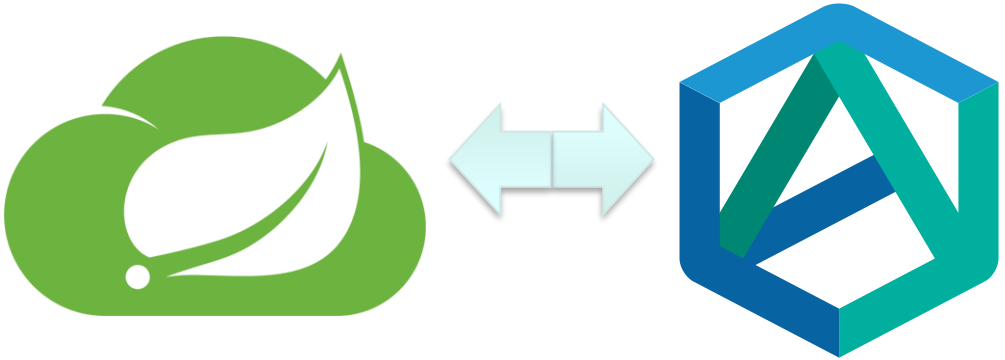
Provides SCS Dev Tools directly from CF CLI

- List apps in eureka instance

- Enable/disable Eureka registration

- Deregister service in Eureka

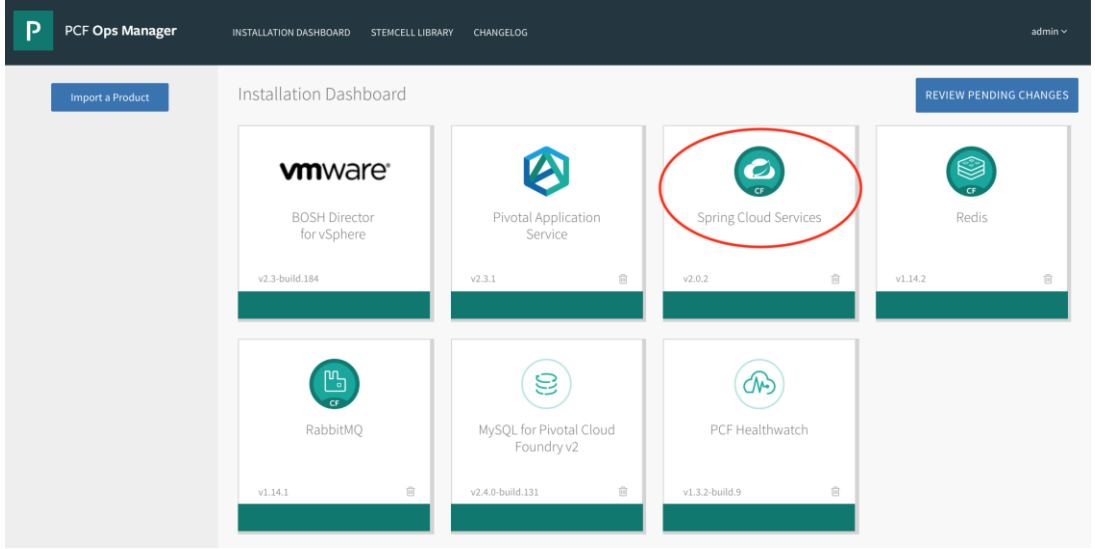
- Encrypt config server values

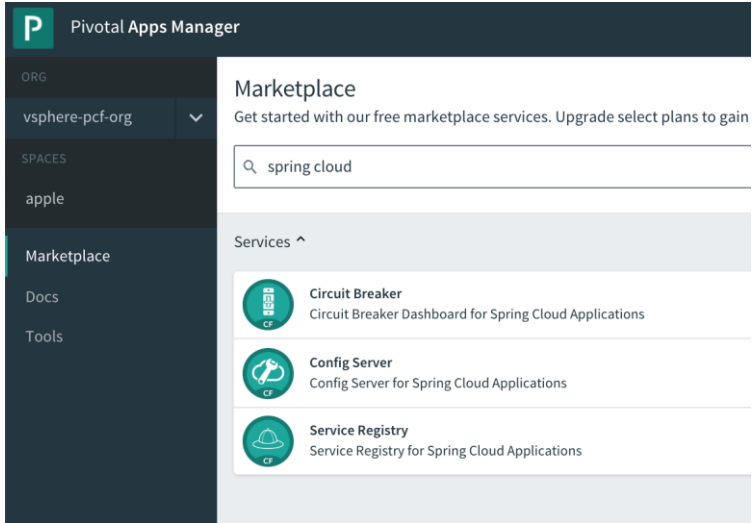


Pivotal

SCS:
PCF Services

Available on PCF once installed by the operations team via a single tile





Pivotal

Hands On

Spring Cloud Config - Local

Single annotation to create a config server

Branch: master ▾ scs-config-master / src / main / java / com / example / scsconfig / ScsConfigApplication.java Find file Copy path

Derrick Chua commit all 292bc46 a day ago

0 contributors

Executable File | 15 lines (11 sloc) | 418 Bytes Raw Blame History

```
1  package com.example.scsconfig;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.cloud.config.server.EnableConfigServer;
6
7  @SpringBootApplication
8  @EnableConfigServer
9  public class ScsConfigApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ScsConfigApplication.class, args);
13     }
14 }
```

Spring Cloud Config - Local

Git as the backing store

Branch: master ▾ scs-config-master / src / main / resources / application.yml Find file Copy path

Derrick Chua update backing git 47d3007 a day ago

0 contributors

Executable File | 9 lines (8 sloc) | 142 Bytes Raw Blame History

```yaml
server:
  port: 8888
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/derrick81/scs-config-store.git
```

Spring Cloud Netflix - Eureka - Local

Single annotation to create a Eureka server

Branch: master ▾ scs-eureka-master / src / main / java / com / example / scseureka / ScsEurekaApplication.java Find file Copy path

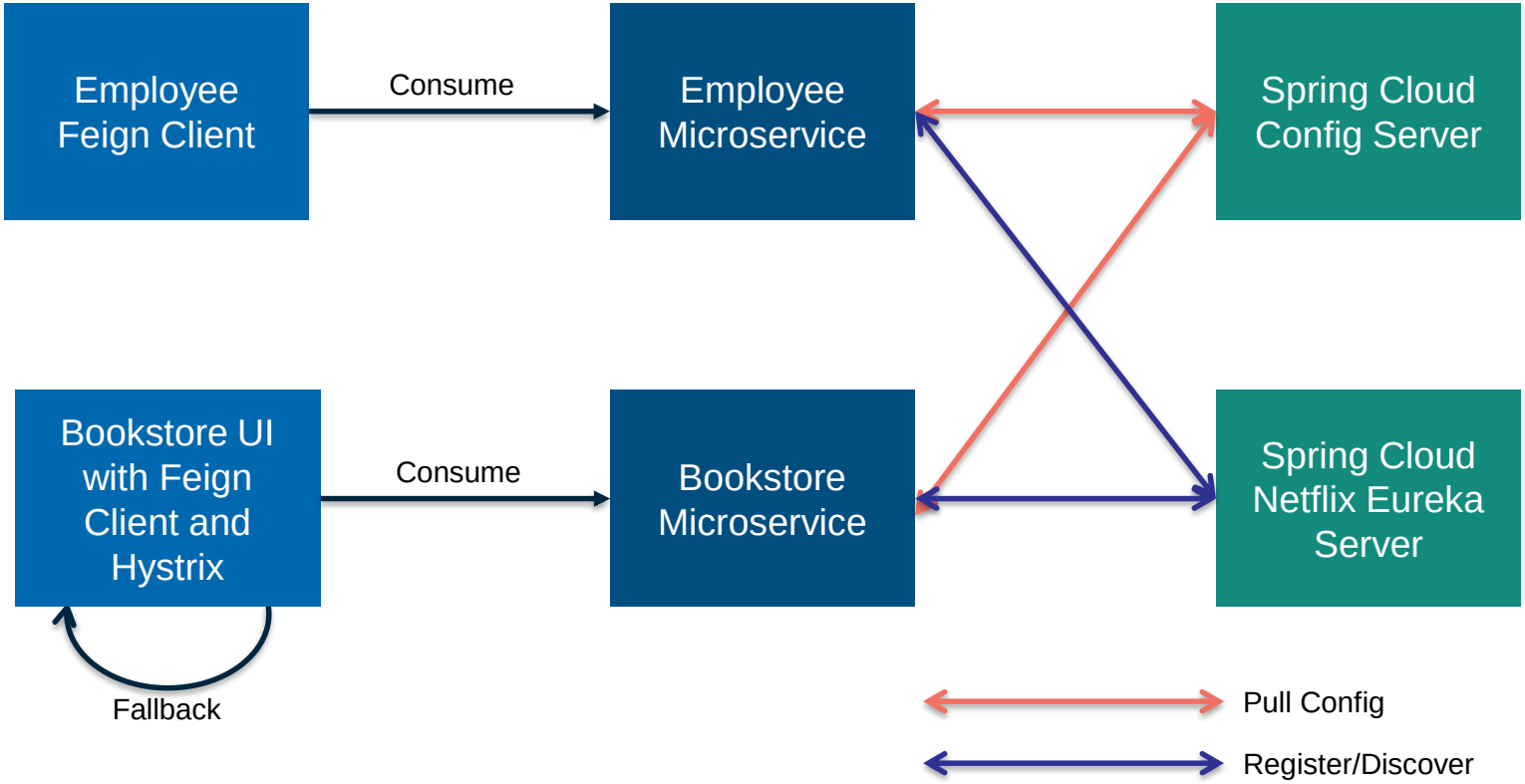
Derrick Chua commit all 4ef94ae a day ago

0 contributors

Executable File | 15 lines (11 sloc) | 426 Bytes Raw Blame History

```
1  package com.example.scseureka;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7  @SpringBootApplication
8  @EnableEurekaServer
9  public class ScsEurekaApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ScsEurekaApplication.class, args);
13     }
14 }
```

Local Hands On – Microservices and Clients



Microservice – Employee

Entity – Employee.java

```java
package com.example.employeeservice.domain;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Data
@ToString
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Employee {
    @Id
    private Long id;

    private String name;

}
```

Rest endpoint (/employees) – EmployeeRepo.java

```java
package com.example.employeeservice.repositories;

import com.example.employeeservice.domain.Employee;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.data.rest.core.annotation.RestResource;

@RepositoryRestResource (collectionResourceRel = "employees", path = "employees")
public interface EmployeeRepo extends JpaRepository<Employee, Long> {
    @RestResource(path = "name", rel = "name")
    Page<Employee> findByNameIgnoreCase(@Param("q") String name, Pageable pageable);


}
```

At runtime, Spring Data REST will create an implementation of the annotated interface automatically.

Microservice – Employee

By default, the configuration values are read on the client's startup, and not again. You can force a bean to ***refresh*** its configuration - to pull updated values from the Config Server - by annotating the class with the Spring Cloud Config @RefreshScope and then by triggering a ***refresh*** **event**.

Value injected by Config Server. Default value is "UNKNOWN-SERVICE".

Not from config server, but from Spring Boot

Spring Config demo – WhoAmI.java

```java
@RefreshScope
@RestController
public class WhoAmI {

    private EurekaClient eurekaClient;

    @Autowired
    @Lazy
    public WhoAmI(EurekaClient eurekaClient) {
        this.eurekaClient = eurekaClient;
    }

    @Value("${whoami:UNKNOWN-SERVICE}")
    private String whoami;

    @Value("${spring.application.name}")
    private String appName;

    @RequestMapping("/whoami")
    String getMessage() {
        return this.whoami;
    }

    @RequestMapping("/name")
    public String greeting() {
        return String.format("Hello from '%s'!", eurekaClient.getApplication(appName).getName());
    }
}
```

Microservice – Employee

Activates Eureka's DiscoveryClient implementation

Note: Name of service is defined in application.yaml under the property spring.application.name

Eureka demo (registration) – EmployeeServiceApplication.java

```
11  @SpringBootApplication
12  @EnableDiscoveryClient
13  public class EmployeeServiceApplication {
14
15      public static void main(String[] args) {
16          SpringApplication.run(EmployeeServiceApplication.class, args);
17      }
18
19      @Configuration
20      static class ApplicationSecurity extends WebSecurityConfigurerAdapter {
21
22          @Override
23          public void configure(WebSecurity web) throws Exception {
24              web
25                      .ignoring()
26                      .antMatchers("/**");
27          }
28      }
29  }
```

Feign Client – Employee

Activates FeignClient implementation

Feign is a declarative REST client, as you will see in the next slide.

Eureka registration & Enabling Feign – EmployeeFeignClientApplication.java

```
11  @SpringBootApplication
12  @EnableDiscoveryClient
13  @EnableFeignClients
14  public class EmployeeFeignClientApplication {
15
16      public static void main(String[] args) {
17          SpringApplication.run(EmployeeFeignClientApplication.class, args);
18      }
19
20      @Configuration
21      static class ApplicationSecurity extends WebSecurityConfigurerAdapter {
22
23          @Override
24          public void configure(WebSecurity web) throws Exception {
25              web
26                      .ignoring()
27                      .antMatchers("/**");
28          }
29      }
30  }
```

Feign Client – Employee

Rest endpoint (/emps) – InvokeEmployeeService.java

```
8   @RestController
9   public class InvokeEmployeeService {
10      private final EmployeeClient client;
11
12      @Autowired
13      public InvokeEmployeeService(EmployeeClient client) {
14          this.client = client;
15      }
16
17      @RequestMapping("/emps")
18      String getEmps() {
19          String emps = client.getEmployees();
20          return emps;
21      }
22  }
```

Actual Feign Client – EmployeeClient.java

Declares a Feign Client specifying the service to consume, complete with load balancing support

```
1   package com.example.employeefeignclient.client;
2
3   import org.springframework.cloud.openfeign.FeignClient;
4   import org.springframework.web.bind.annotation.GetMapping;
5
6   @FeignClient(name = "EMPLOYEE-SERVICE")
7   public interface EmployeeClient {
8
9       @GetMapping(value="/employees", consumes="application/json")
10      String getEmployees();
11  }
```

Target microservice path

Microservice – Bookstore

Rest endpoint (/recommended) – BookStoreRest.java

Spring MVC controller for REST calls

Maps the incoming path of the REST call

```
1   package com.example.bookstoreservice.rest;
2
3   import org.slf4j.Logger;
4   import org.slf4j.LoggerFactory;
5   import org.springframework.web.bind.annotation.RequestMapping;
6   import org.springframework.web.bind.annotation.RestController;
7
8   @RestController
9   public class BookStoreRest {
10
11      protected static Logger logger = LoggerFactory.getLogger(BookStoreRest.class);
12
13      @RequestMapping(value = "/recommended")
14      public String readingList()
15      {
16          logger.info("Invoking readingList");
17          return "Spring in Action (Manning), Cloud Native Java (O'Reilly), Learning Spring Boot (Packt)";
18      }
19  }
```

Feign + Hystrix Client – BookStore Web UI

Internal service called by controller – BookService.java

```
10  @RefreshScope
11  @Service
12  public class BookService {
13
14      protected static Logger logger = LoggerFactory.getLogger(BookService.class);
15      private final BookClient bookClient;

    @Value("${store-service:bookstore-service}")
    private String storeService;

20      public BookService(BookClient bookClient) {
21          this.bookClient = bookClient;
22      }
23
24      @HystrixCommand(fallbackMethod = "reliable")
        public String readingList() {
26          return bookClient.whichbook();

29      public String reliable() {
30          logger.info("In fallback method, something is WRONG!!!!!");
31          return "Cloud Native Java (O'Reilly)";
32      }
33  }
```

Spring Cloud Config

Hystrix Fallback

Actual Feign Client – BookClient.java

```
1   package com.example.bookstoreuiresttemplatehystrix.service;
2
3   import org.springframework.cloud.openfeign.FeignClient;
4   import org.springframework.web.bind.annotation.RequestMapping;
5
6   @FeignClient("BOOKSTORE-SERVICE")
7   public interface BookClient {
8       @RequestMapping("/recommended")
9       String whichbook();
10  }
11
```

Target microservice name

Target microservice path

Feign + Hystrix Client – BookStore Web UI

MVC controller – BookController.java

```
1   package com.example.bookstoreuiresttemplatehystrix.controller;
2
3   import com.example.bookstoreuiresttemplatehystrix.service.BookService;
4   import org.springframework.stereotype.Controller;
5   import org.springframework.ui.Model;
6   import org.springframework.web.bind.annotation.RequestMapping;
7
8   @Controller
9   public class BookController {
10
11      private BookService bookService;
12
13      public BookController(BookService bookService) {
14          this.bookService = bookService;
15      }
16
17      @RequestMapping("/book")
18      public String greeting(Model model) {
19          model.addAttribute("recommendedbook", bookService.readingList());
20          return "book";
21      }
22  }
```

Internal service

PCF Services Marketplace

cf push

Employee microservice

Branch: master ▾ scs-clients-master / employee-service / manifest.yml

Derrick Chua commit all

0 contributors

Executable File | 13 lines (13 sloc) | 312 Bytes

```yaml
---
applications:
- name: employee-service
  memory: 1024M
  instances: 2
  random-route: true
  path: ./target/employee-service-0.0.1-SNAPSHOT.jar
  services:
    - config-server
    - service-registry
  env:
    JAVA_OPTS: -Djava.security.egd=file:///dev/urandom
    TRUST_CERTS: api.run.haas-99.pez.pivotal.io
```

Employee feign client

Branch: master ▾ scs-clients-master / employee-feign-client / manifest.yml

Derrick Chua commit all

0 contributors

Executable File | 13 lines (13 sloc) | 318 Bytes

```
---
applications:
- name: employee-feign-client
  memory: 1024M
  instances: 1
  random-route: true
  path: ./target/employee-feign-client-0.0.1-SNAPSHOT.jar
  services:
  - config-server
  - service-registry
  env:
    JAVA_OPTS: -Djava.security.egd=file:///dev/urandom
    TRUST_CERTS: api.run.haas-99.pez.pivotal.io
```

Bookstore microservice

Branch: master ▾ scs-clients-master / bookstore-service / manifest.yml

Derrick Chua commit all

0 contributors

Executable File | 13 lines (13 sloc) | 310 Bytes

```yaml
---
applications:
- name: bookstore-service
  memory: 1024M
  instances: 1
  random-route: true
  path: ./target/bookstore-service-0.0.1-SNAPSHOT.jar
  services:
  - config-server
  - service-registry
  env:
    JAVA_OPTS: -Djava.security.egd=file:///dev/urandom
    TRUST_CERTS: api.run.haas-99.pez.pivotal.io
```

Bookstore feign hystrix web UI

Branch: master ▾ scs-clients-master / bookstore-ui-feign-hystrix / manifest.yml

Derrick Chua commit all

0 contributors

Executable File | 13 lines (13 sloc) | 335 Bytes

```yaml
---
applications:
- name: bookstore-ui-feign-hystrix
  memory: 1024M
  instances: 1
  random-route: true
  path: ./target/bookstore-ui-resttemplate-hystrix-0.0.1-SNAPSHOT.jar
  services:
  - config-server
  - service-registry
  env:
    JAVA_OPTS: -Djava.security.egd=file:///dev/urandom
    TRUST_CERTS: api.run.haas-99.pez.pivotal.io
```

Pivotal®

Transforming How The World Builds Software