Building Distributed Systems with Netflix OSS and Spring Cloud

ME

Matt Stine (@mstine)
Senior Product Manager
Pivotal
http://www.mattstine.com
matt.stine@gmail.com

There seems to be some hype...

Define: Microservice

"Loosely coupled service oriented architecture with bounded contexts..."
Adrian Cockcroft

SPRING BOOT

A MICROFRAMEWORK FOR MICROSERVICES

It Can Get Pretty Small...

```
@RestController
class ThisWillActuallyRun {
  @RequestMapping("/")
  String home() {
    "Hello World!"
  }
}
```

DEMO

WITH SPRING DATA REST!

```java
@Entity
@Table(name = "city")
public class City implements Serializable {

  @Id
  @GeneratedValue
  private Long id;

  @Column(nullable = false)
  private String name;

  @Column(nullable = false)
  private String county;

  //...

}
```

WITH SPRING DATA REST!

```java
@RepositoryRestResource(collectionResourceRel = "cities", path = "cities")
public interface CityRepository extends PagingAndSortingRepository<City, Long> {}
```

WITH SPRING DATA REST!

```java
@SpringBootApplication
@EnableJpaRepositories
@Import(RepositoryRestMvcConfiguration.class)
public class Application {

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

WITH SPRING DATA REST!


```
{
  "_links" : {
    "next" : {
      "href" : "http://localhost:8080/cities?page=1&size=20"
    },
    "self" : {
      "href" : "http://localhost:8080/cities{?page,size,sort}",
      "templated" : true
    }
  },
  "_embedded" : {
    "cities" : [ {
      "name" : "HOLTSVILLE",
      "county" : "SUFFOLK",
      "stateCode" : "NY",
      "postalCode" : "00501",
      "latitude" : "+40.922326",
      "longitude" : "-072.637078",
```

DEMO

WRITING A SINGLE SERVICE IS

NICE...

BUT NO MICROSERVICE IS AN ISLAND

Challenges of Distributed Systems

- Configuration Management
- Service Registration & Discovery
- Routing & Load Balancing
- Fault Tolerance (Circuit Breakers!)
- Monitoring
- Concurrent API Aggregation & Transformation

NETFLIX

OSS

SPRING CLOUD

DISTRIBUTED SYSTEM PATTERNS FTW!

CONFIGURATION MANAGEMENT

Spring Environment

» Properties

» Profiles

app.groovy

```groovy
@RestController
class BasicConfig {

  @Value('${greeting}')
  String greeting

  @RequestMapping("/")
  String home() {
    "${greeting} World!"
  }
}
```

application.yml

```
greeting: Hello
```

DEMO

Boot Priority

1. Command Line Args
2. JNDI
3. Java System Properties
4. OS Environment Variables
5. Properties Files
6. @PropertySource
7. Defaults

DEMO

PROFILES

application.yml

```yaml
greeting: Hello

---

spring:
  profiles: spanish
greeting: Hola
```

DEMO

DISTRIBUTED?

CONFIG SERVER!

Config Server `app.groovy`

```
@Grab("org.springframework.cloud:spring-cloud-starter-bus-amqp:1.0.0.RC1")
@Configuration
@EnableAutoConfiguration
@EnableConfigServer
class ConfigServer {
}
```

CONFIG SERVER

application.yml

```yaml
server:
  port: 8888

spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/mstine/config-repo.git
```

https://github.com/mstine/config-repo/blob/master/demo.yml

```
greeting: Bonjour
```

Config Client `app.groovy`

```
@Grab("org.springframework.cloud:spring-cloud-starter-bus-amqp:1.0.0.RC1")
@RestController
class BasicConfig {

  @Autowired
  Greeter greeter

  @RequestMapping("/")
  String home() {
    "${greeter.greeting} World!"
  }
}

@Component
@RefreshScope
class Greeter {

  @Value('${greeting}')
  String greeting

}
```

CONFIG CLIENT bootstrap.yml

```yaml
spring:
  application:
    name: demo
```

DEMO

CLOUD BUS!

```
curl -X POST http://localhost:8888/bus/refresh
```

DEMO

SERVICE REGISTRATION & DISCOVERY

EUREKA

PRODUCER CONSUMER

Eureka Service Registry

```
@GrabExclude("ch.qos.logback:logback-classic")
@EnableEurekaServer
class Eureka {
}
```

Producer

```
@EnableDiscoveryClient
@RestController
public class Application {

  int counter = 0

  @RequestMapping("/")
  String produce() {
    "{\"value\": ${counter++}}"
  }
}
```

CONSUMER

```
@EnableDiscoveryClient
@RestController
public class Application {

  @Autowired
  DiscoveryClient discoveryClient

  @RequestMapping("/")
  String consume() {
    InstanceInfo instance = discoveryClient.getNextServerFromEureka("PRODUCER", false)

    RestTemplate restTemplate = new RestTemplate()
    ProducerResponse response = restTemplate.getForObject(instance.homePageUrl, ProducerResponse.class)

    "{\"value\": ${response.value}}"
  }
}

public class ProducerResponse {
  Integer value
}
```

DEMO

ROUTING & LOAD BALANCING

RIBBON

CONSUMER WITH LOAD BALANCER

```
@Autowired
LoadBalancerClient loadBalancer

@RequestMapping("/")
String consume() {
  ServiceInstance instance = loadBalancer.choose("producer")
  URI producerUri = URI.create("http://${instance.host}:${instance.port}");

  RestTemplate restTemplate = new RestTemplate()
  ProducerResponse response = restTemplate.getForObject(producerUri, ProducerResponse.class)

  "{\"value\": ${response.value}}"
}
```

DEMO

CONSUMER WITH RIBBON-ENABLED RestTemplate

```
@Autowired
RestTemplate restTemplate

@RequestMapping("/")
String consume() {
  ProducerResponse response = restTemplate.getForObject("http://producer", ProducerResponse.class)

  "{\"value\": ${response.value}}"
}
```

DEMO

FEIGN CLIENT

```java
@FeignClient("producer")
public interface ProducerClient {

  @RequestMapping(method = RequestMethod.GET, value = "/")
  ProducerResponse getValue();
}
```

CONSUMER WITH FEIGN CLIENT

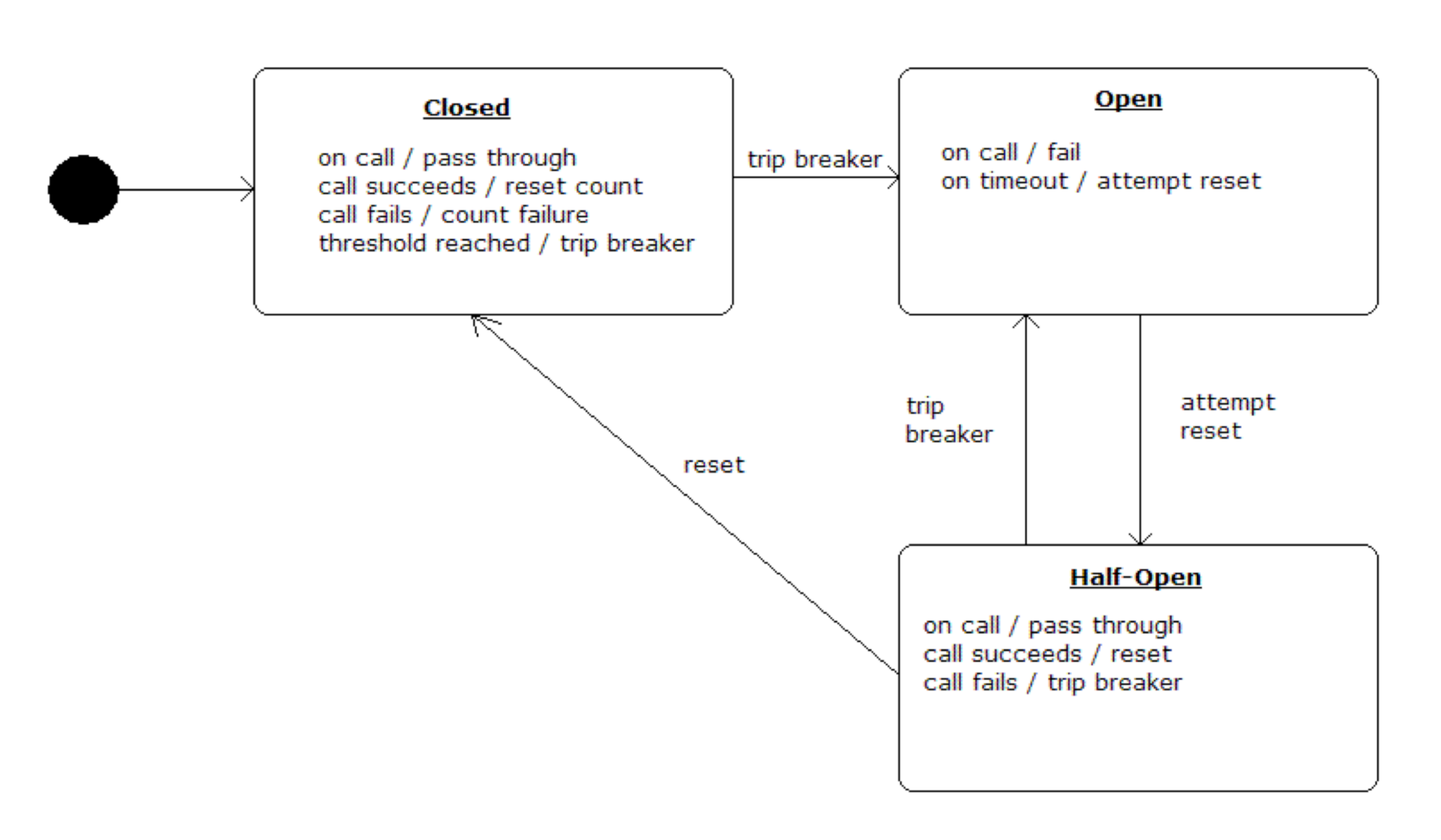
```java
@SpringBootApplication
@FeignClientScan
@EnableDiscoveryClient
@RestController
public class Application {

  @Autowired
  ProducerClient client;

  @RequestMapping("/")
  String consume() {
    ProducerResponse response = client.getValue();

    return "{\"value\": " + response.getValue() + "}";
  }

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

DEMO

FAULT TOLERANCE

HYSTRIX

CIRCUIT BREAKER



CONSUMER app.groovy

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

  @Autowired
  ProducerClient client

  @RequestMapping("/")
  String consume() {
    ProducerResponse response = client.getProducerResponse()

    "{\"value\": ${response.value}}"
  }

}
```

Producer Client

```
@Component
public class ProducerClient {

  @Autowired
  RestTemplate restTemplate

  @HystrixCommand(fallbackMethod = "getProducerFallback")
  ProducerResponse getProducerResponse() {
    restTemplate.getForObject("http://producer", ProducerResponse.class)
  }

  ProducerResponse getProducerFallback() {
    new ProducerResponse(value: 42)
  }
}
```

DEMO

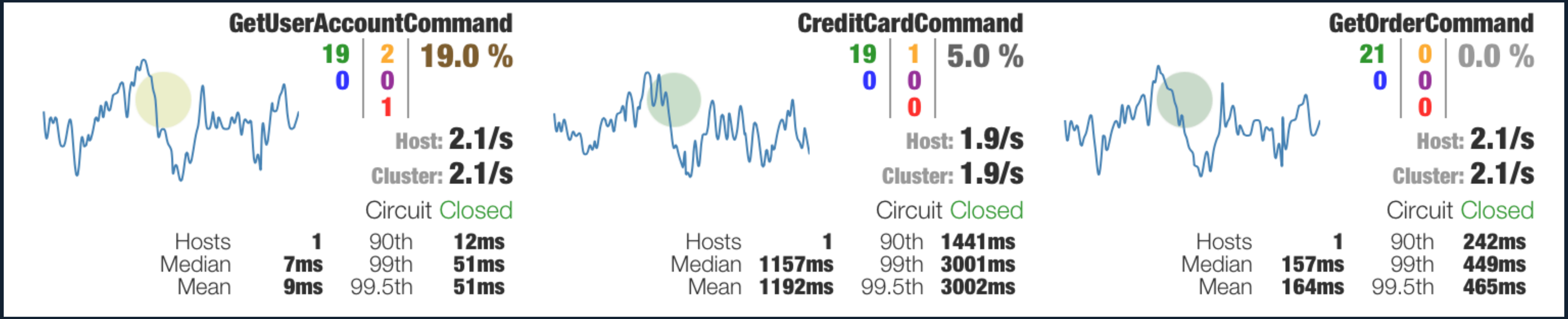
MONITORING

DEMO

`http://localhost:8082/`

HYSTRIX DASHBOARD

HYSTRIX DASHBOARD



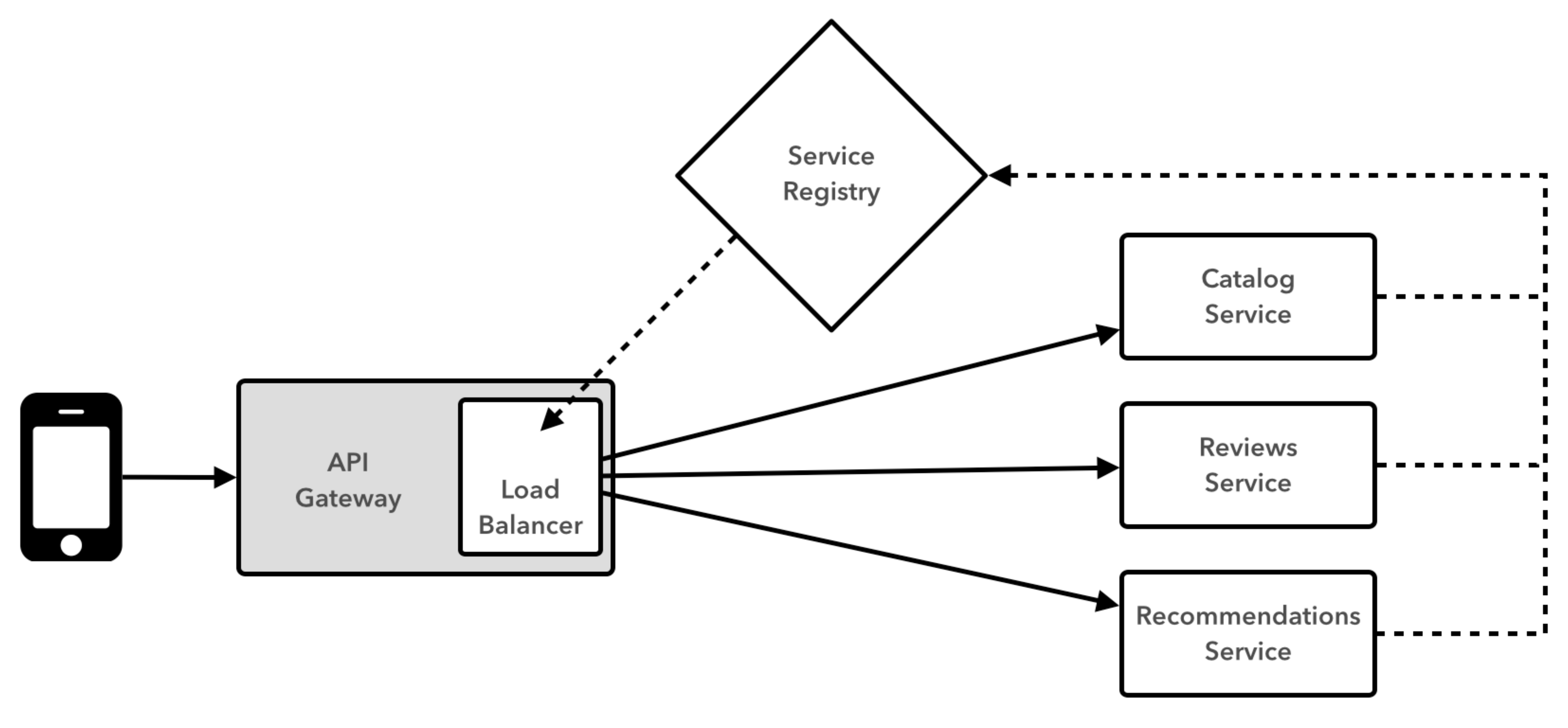
HYSTRIX DASHBOARD

```
@Grab("org.springframework.cloud:spring-cloud-starter-hystrix-dashboard:1.0.0.RC1")

import org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard

@EnableHystrixDashboard
class HystrixDashboard {
}
```

DEMO

CONCURRENT API AGGREGATION & TRANSFORMATION

API Gateway Pattern



RXJAVA

Movie Catalog Service

```
@RequestMapping(value = "/catalog/movies/{mlId}", method = RequestMethod.GET)
public Movie movie(@PathVariable String mlId) {
  return movieRepository.findByMlId(mlId);
}
```

Movie Catalog Service


```
{
  id: 1001,
  title: "GoldenEye (1995)",
  mlId: "2",
  genres: [
  {
    id: 1001,
    mlId: "1",
    name: "Action"
  },
  {
    id: 1002,
    mlId: "2",
    name: "Adventure"
  },
  {
    id: 1016,
    mlId: "16",
    name: "Thriller"
  }
  ]
}
```

Movie Review Service

```
@RequestMapping(value = "/reviews/reviews/{mlId}", method = RequestMethod.GET)
public Iterable<Review> reviews(@PathVariable String mlId) {
  return reviewRepository.findByMlId(mlId);
}
```

Movie Review Service


```
[
{
  id: "54b85cbe004e0464177e90e4",
  mlId: "2",
  userName: "mstine",
  title: "GoldenEye (1995)",
  review: "Pretty good...",
  rating: 3
},
{
  id: "54b85cbe004e0464177e90e5",
  mlId: "2",
  userName: "starbuxman",
  title: "GoldenEye (1995)",
  review: "BOND BOND BOND!",
  rating: 5
},
{
  id: "54b85cbf004e0464177e90e8",
  mlId: "2",
  userName: "littleidea",
  title: "GoldenEye (1995)",
  review: "Good show!",
  rating: 4
}
]
```

MOVIE RECOMMENDATIONS SERVICE

```
public interface MovieRepository extends GraphRepository<Movie> {
  Movie findByMlId(String mlId);

  @Query("MATCH (movie:Movie) WHERE movie.mlId = {0} MATCH movie<-[:LIKES]-slm-[:LIKES]->recommendations " +
  "RETURN distinct recommendations")
  Iterable<Movie> moviesLikedByPeopleWhoLiked(String mlId);
}
```

MOVIE RECOMMENDATIONS SERVICE

```java
@RequestMapping(value = "/recommendations/forMovie/{mlId}", method = RequestMethod.GET)
public Iterable<Movie> recommendedMoviesForMovie(@PathVariable String mlId) {
  return movieRepository.moviesLikedByPeopleWhoLiked(mlId);
}
```

Movie Recommendations Service

```
@RequestMapping(value = "/recommendations/forMovie/{mlId}", method = RequestMethod.GET)
public Iterable<Movie> recommendedMoviesForMovie(@PathVariable String mlId) {
  return movieRepository.moviesLikedByPeopleWhoLiked(mlId);
}
```

Movie Recommendations Service


```
[
{
  id: 6,
  mlId: "1",
  title: "Toy Story (1995)"
},
{
  id: 1,
  mlId: "4",
  title: "Get Shorty (1995)"
},
{
  id: 2,
  mlId: "5",
  title: "Copycat (1995)"
},
{
  id: 0,
  mlId: "3",
  title: "Four Rooms (1995)"
}
]
```

API GATEWAY

CATALOG INTEGRATION SERVICE

```java
@Service
public class CatalogIntegrationService {

  @Autowired
  RestTemplate restTemplate;

  @HystrixCommand(fallbackMethod = "stubMovie")
  public Observable<Movie> getMovie(final String mlId) {
    return new ObservableResult<Movie>() {
      @Override
      public Movie invoke() {
        return restTemplate.getForObject("http://catalog-service/catalog/movies/{mlId}", Movie.class, mlId);
      }
    };
  }

  private Movie stubMovie(final String mlId) {
    Movie stub = new Movie();
    stub.setMlId(mlId);
    stub.setTitle("Interesting...the wrong title. Sssshhhh!");
    return stub;
  }
}
```

Reviews Integration Service

```java
@Service
public class ReviewsIntegrationService {

  @Autowired
  RestTemplate restTemplate;

  @HystrixCommand(fallbackMethod = "stubReviews")
  public Observable<List<Review>> reviewsFor(String mlId) {
    return new ObservableResult<List<Review>>() {
      @Override
      public List<Review> invoke() {
        ParameterizedTypeReference<List<Review>> responseType = new ParameterizedTypeReference<List<Review>>() {
        };
        return restTemplate.exchange("http://reviews-service/reviews/reviews/{mlId}", HttpMethod.GET, null, responseType, mlId).getBody();
      }
    };
  }

  private List<Review> stubReviews(String mlId) {
    Review review = new Review();
    review.setMlId(mlId);
    review.setRating(4);
    review.setTitle("Interesting...the wrong title. Sssshhhh!");
    review.setReview("Awesome sauce!");
    review.setUserName("joeblow");
    return Arrays.asList(review);
  }

}
```

RECOMMENDATIONS INTEGRATION SERVICE

```java
@Service
public class RecommendationsIntegrationService {
  @Autowired
  RestTemplate restTemplate;

  @HystrixCommand(fallbackMethod = "stubRecommendations")
  public Observable<List<Movie>> getRecommendations(final String mlId) {
    return new ObservableResult<List<Movie>>() {
      @Override
      public List<Movie> invoke() {
        ParameterizedTypeReference<List<Movie>> responseType = new ParameterizedTypeReference<List<Movie>>() {
        };
        return restTemplate.exchange("http://recommendations-service/recommendations/forMovie/{mlId}", HttpMethod.GET, null, responseType, mlId).getBody();
      }
    };
  }

  private List<Movie> stubRecommendations(final String mlId) {
    Movie one = new Movie();
    one.setMlId("25");
    one.setMlId("A movie which doesn't exist");
    Movie two = new Movie();
    two.setMlId("26");
    two.setMlId("A movie about nothing");
    return Arrays.asList(one, two);
  }
}
```

CONCURRENTLY AGGREGATE AND TRANSFORM

```
@RequestMapping("/movie/{mlId}")
public DeferredResult<MovieDetails> movieDetails(@PathVariable String mlId) {
  Observable<MovieDetails> details = Observable.zip(

  catalogIntegrationService.getMovie(mlId),
  reviewsIntegrationService.reviewsFor(mlId),
  recommendationsIntegrationService.getRecommendations(mlId),

  (movie, reviews, recommendations) -> {
    MovieDetails movieDetails = new MovieDetails();
    movieDetails.setMlId(movie.getMlId());
    movieDetails.setTitle(movie.getTitle());
    movieDetails.setReviews(reviews);
    movieDetails.setRecommendations(recommendations);
    return movieDetails;
  }

  );
  return toDeferredResult(details);
}
```

DEMO

THANKS!

Matt Stine (@mstine)

» Spring Cloud: http://cloud.spring.io

» This Presentation: https://github.com/mstine/nfjs_2015/tree/master/DistributedSystemsWithSpringCloud

» SpringBox-Cloud: https://github.com/mstine/microservices-lab/tree/master/springbox-cloud

Image Credits

- http://i.imgur.com/atz81.jpg
- http://theroomermill.net/wp-content/uploads/2014/06/island-house.jpg
- Circuit Breaker: Nygard, Michael. Release It!