CLOUD-NATIVE APPLICATION ARCHITECTURES

WITH SPRING AND CLOUD FOUNDRY

SESSION SIX

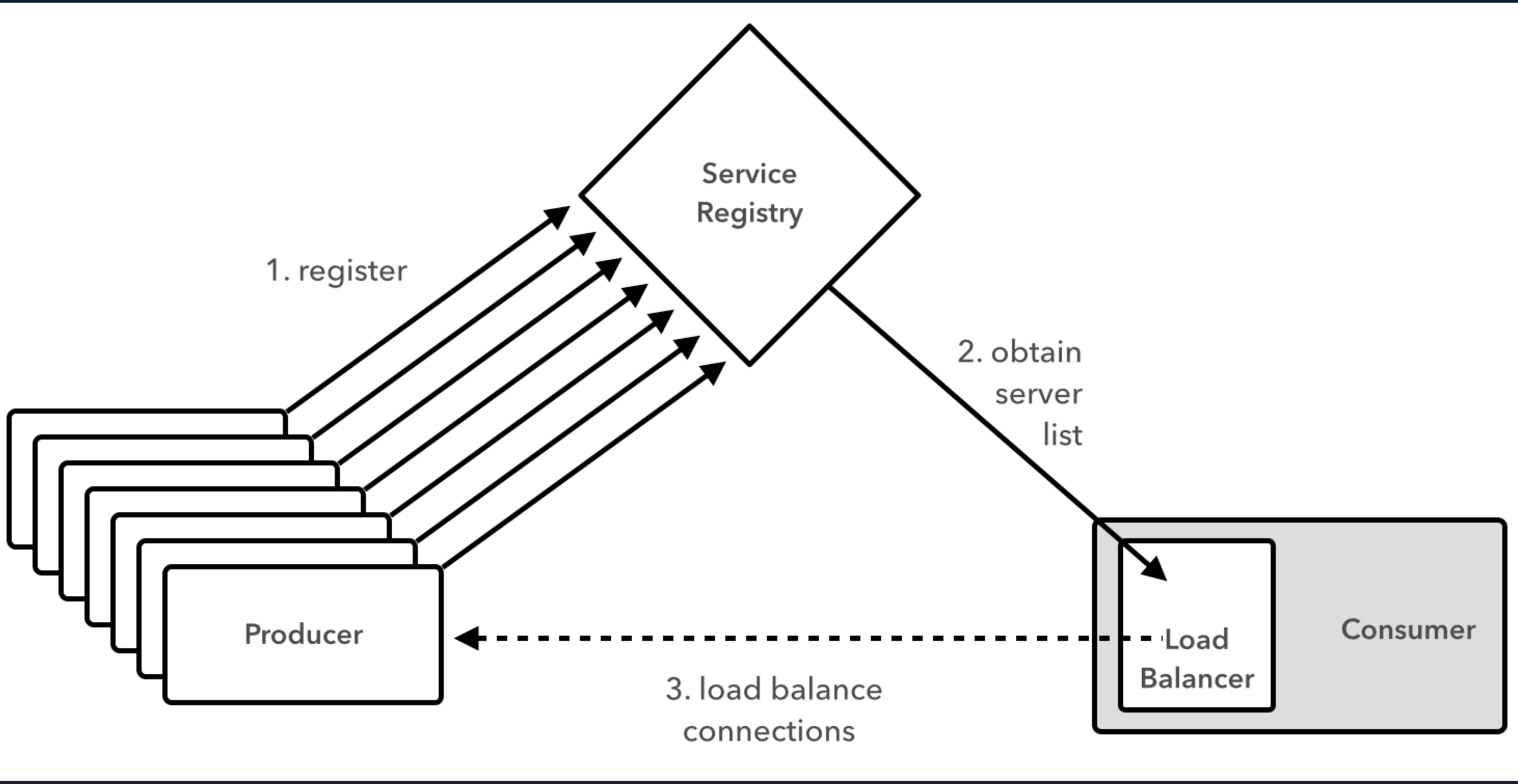
CLOUD-NATIVE ARCHITECTURE PATTERNS: PART 2

PATTERNS ON-DECK

» Routing/Load Balancing

» Fault Tolerance

ROUTING & LOAD BALANCING


Service
Registry
1. register
2. obtain
server
list
Producer
Load
Balancer
Consumer
3. load balance
connections


NETFLIX
OSS
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}}"
}
```

CONSUMER WITH RIBBON-ENABLED RestTemplate

```
@Autowired
RestTemplate restTemplate

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

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

FEIGN CLIENT

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

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

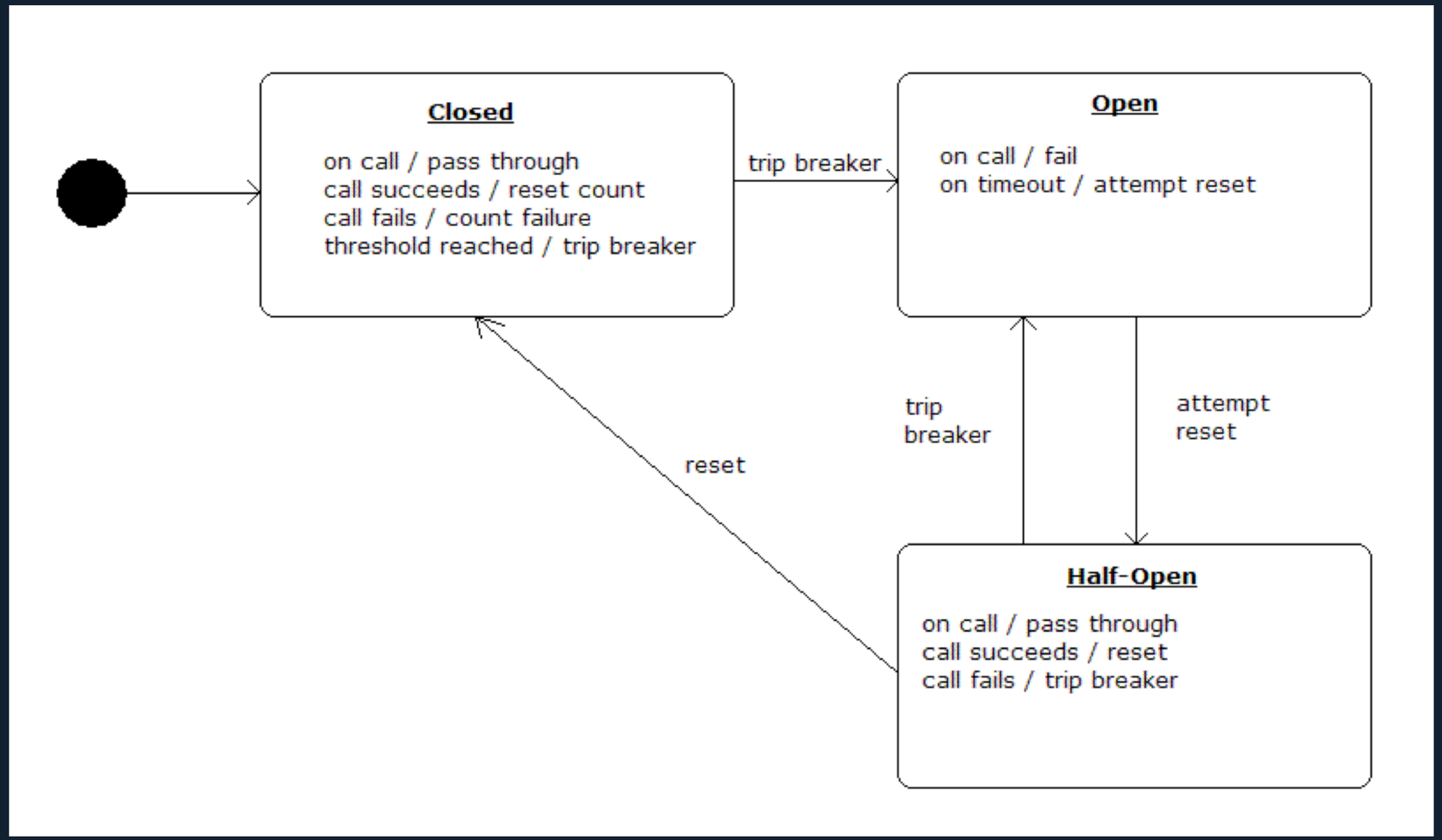
CONSUMER WITH FEIGN CLIENT

```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);
  }
}
```

FAULT TOLERANCE


NETFLIX
OSS
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}}"
  }

}
```

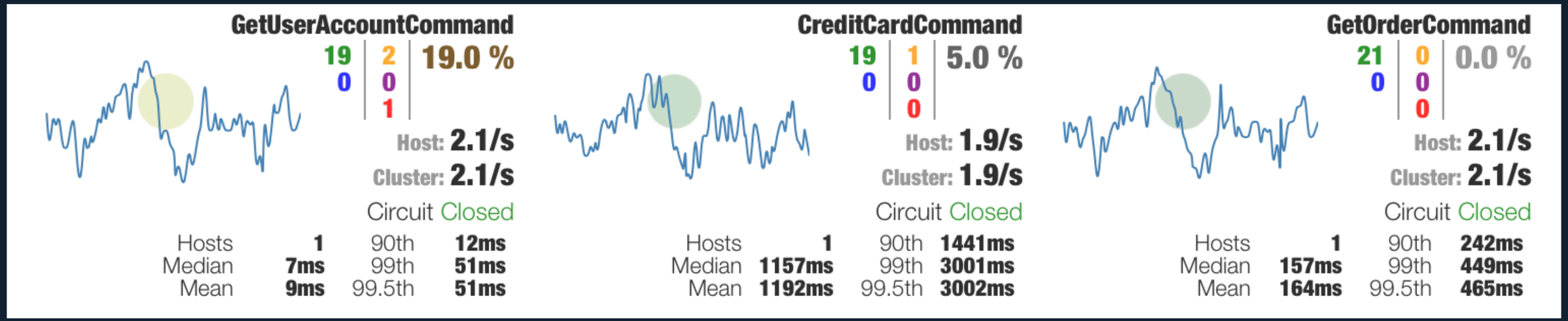
PRODUCER CLIENT

```
@Component
public class ProducerClient {

  @Autowired
  RestTemplate restTemplate

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

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

MONITORING

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 {
}
```

TO THE LABS!