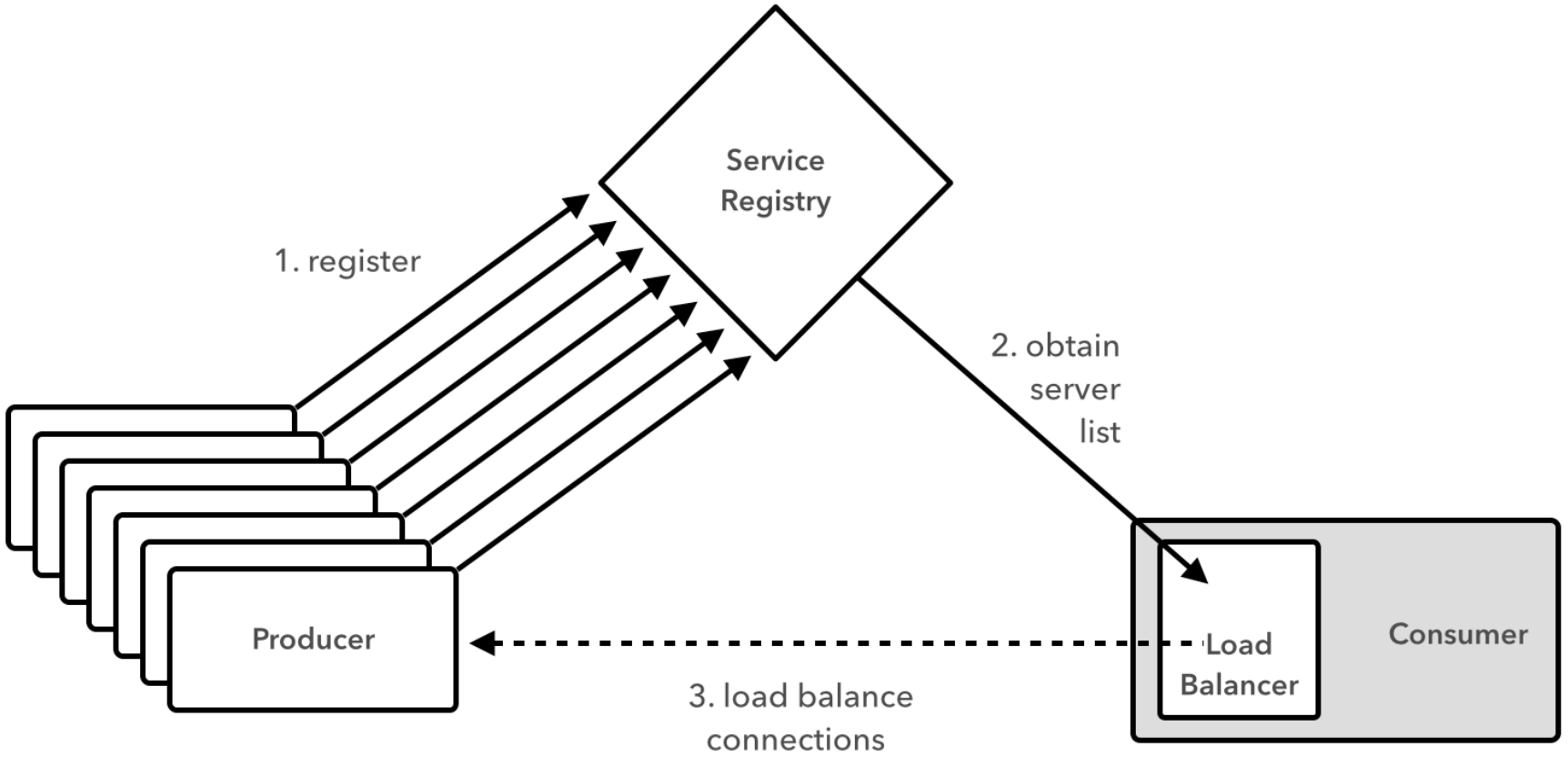
CNA
Patterns – Part two

Patterns on-deck

- Routing/Load Balancing
- Fault Tolerance

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}}"
}
```

Pivotal

Consumer with Ribbon-Enabled *RestTemplate*

```
@Autowired
RestTemplate restTemplate

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

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

Pivotal

Feign Client

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

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

Consumer with Feign Client

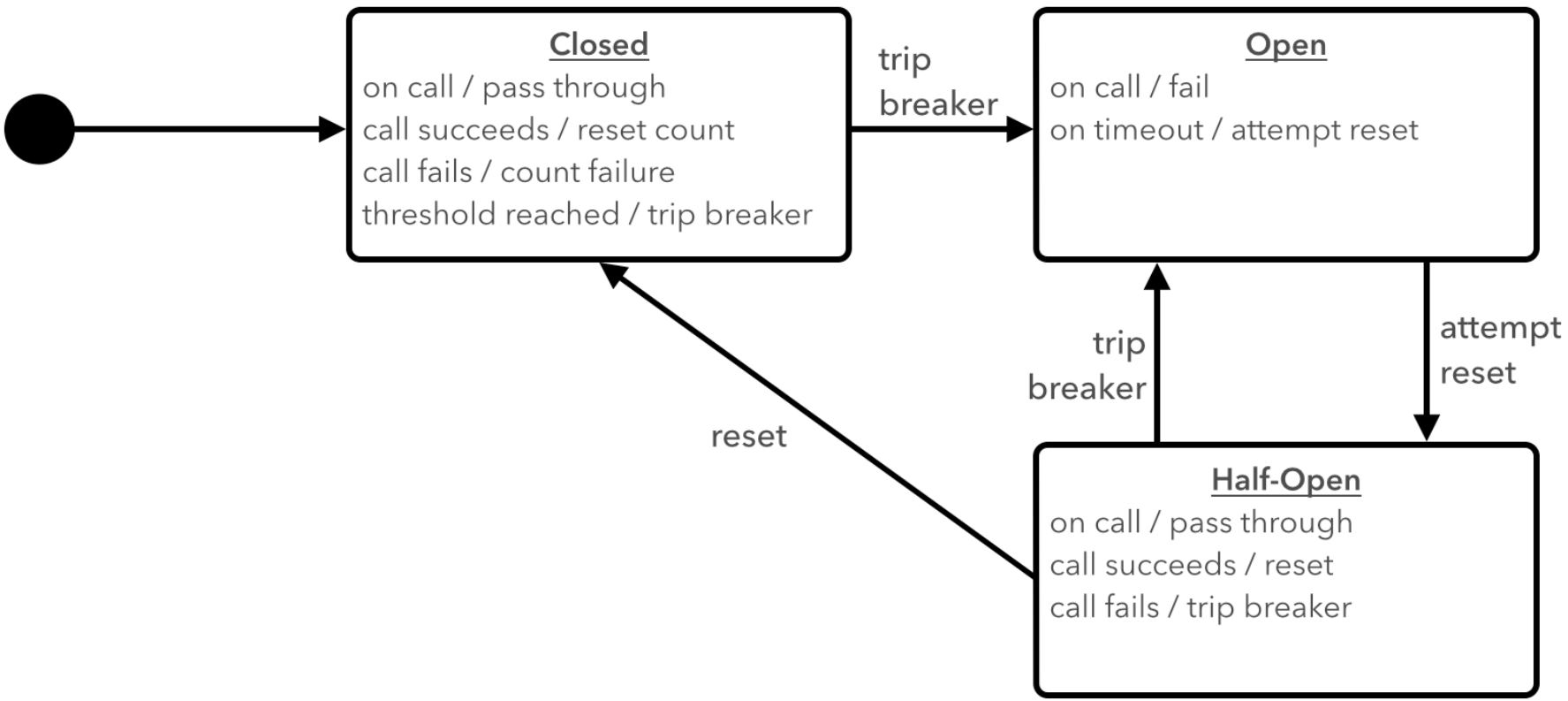
```
@SpringBootApplication
@FeignClientScan
@EnableDiscoveryClient
@RestController
public class Application {

  @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);
  }
}
```

Pivotal

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}}"
  }

}
```

Pivotal

Producer Client

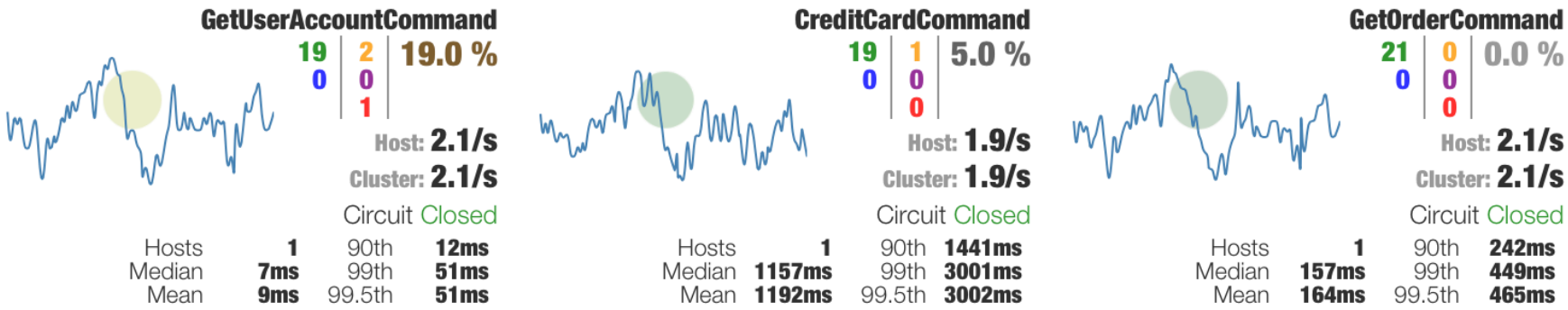
```
@Component
public class ProducerClient {

  @Autowired
  RestTemplate restTemplate

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

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

Pivotal

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 {
}
```

Labs!!