

CNA

# Patterns — part one

Session 06

# Patterns on-deck

- Consistent and Distributed Configuration
- Service Registration and Discovery

# 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**

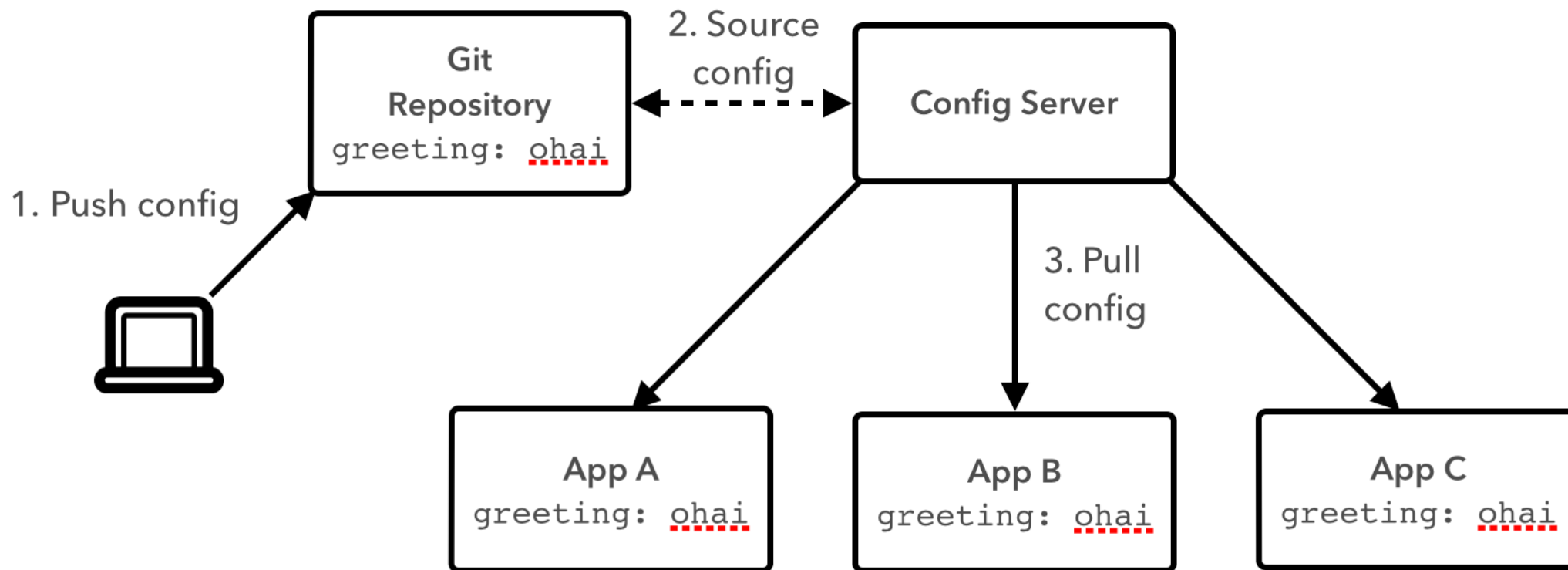
# Spring Cloud Distributed Patterns

# Configuration Management



# Distributed?

# Config Server



# Config Server (app.groovy)

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

# Config server – application.yml

```
server:~  
  port: 8888~  
~  
spring:~  
  cloud:~  
    config:~  
      server:~  
        git:~  
          uri: https://bitbucket.com/caxqueiroz/configserver-data-public.git
```

<https://bitbucket.org/caxqueiroz/configserver-data-public/src/798c0a25b54fbc597ca417211d99a5a5433c4016/demo.yml>

**Greeting: Guten Tag**

# 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

}
```

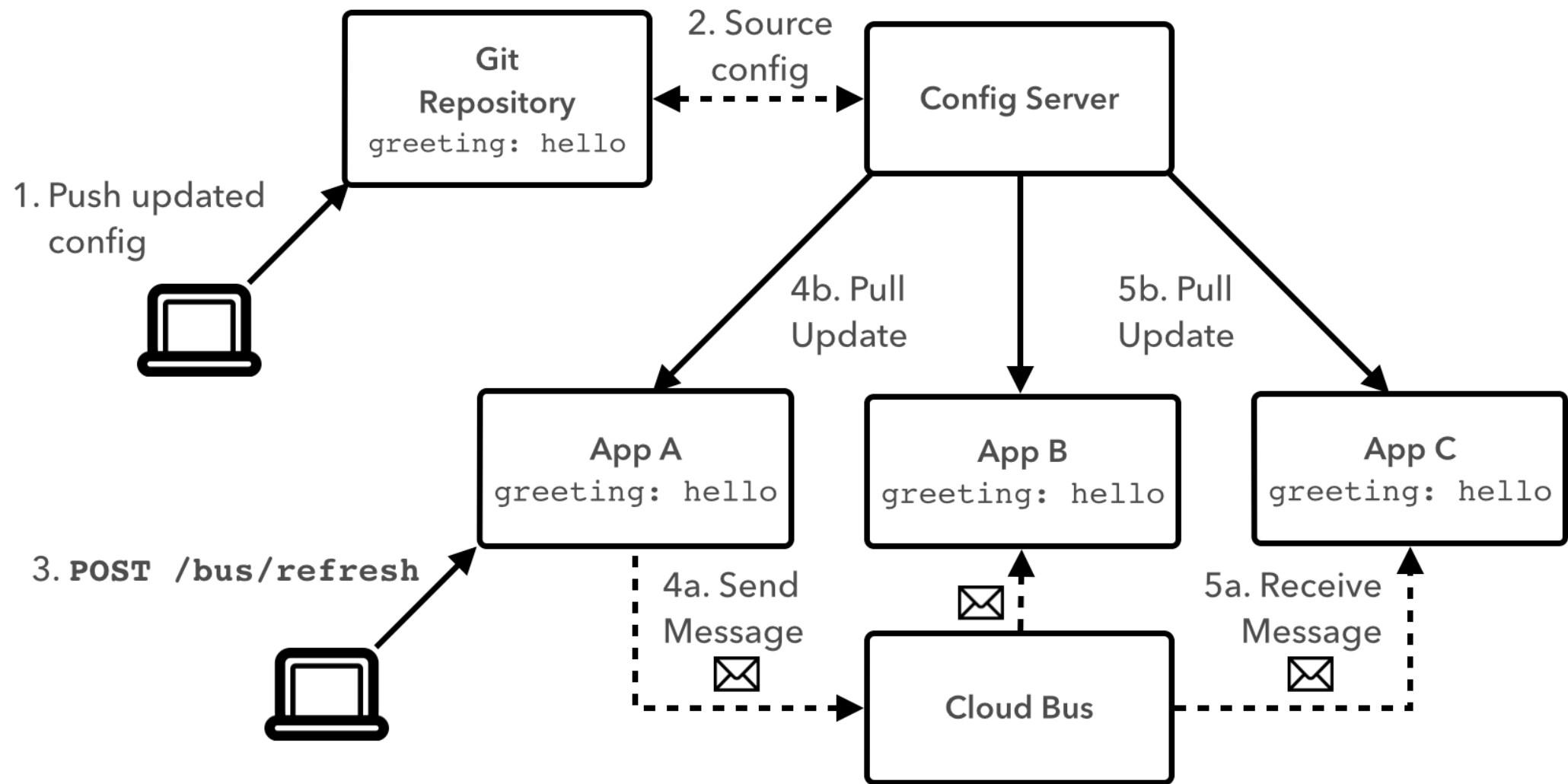
# Config client (bootstrap.yml)

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

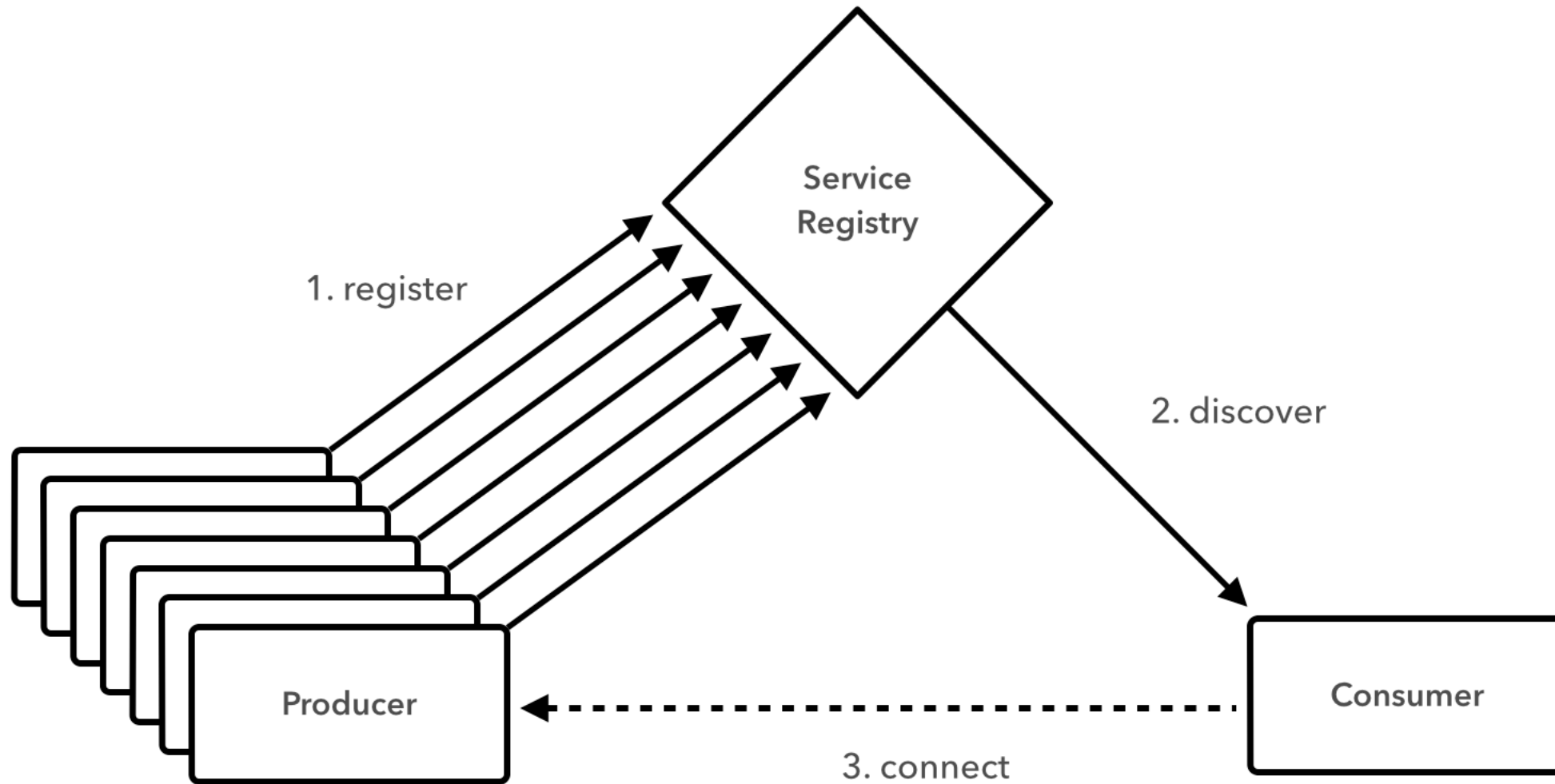


# Cloud Bus





# Service Registration & Discovery



# Eureka

# 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
}
```



Labs!!