# CLOUD-NATIVE APPLICATION ARCHITECTURES

WITH SPRING AND CLOUD FOUNDRY

# SESSION FIVE

# Cloud-Native Architecture Patterns: Part 1

## 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

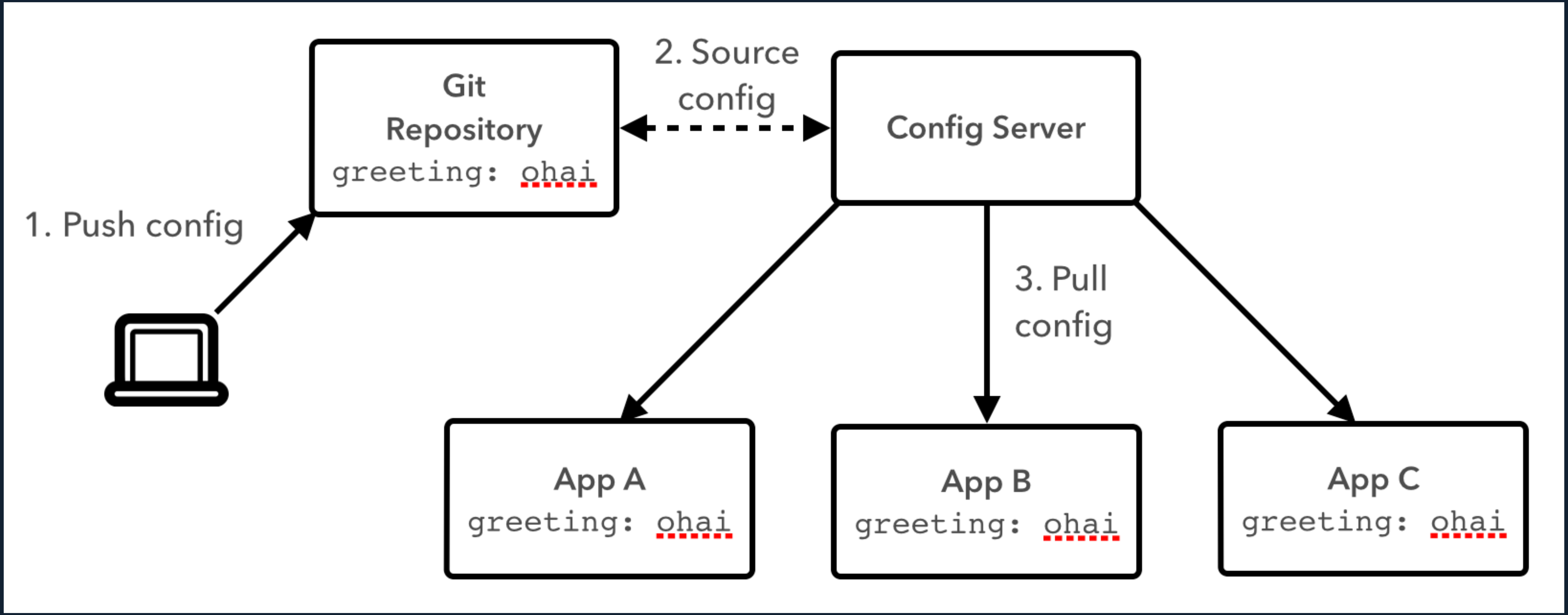
# SPRING CLOUD

## DISTRIBUTED SYSTEM PATTERNS FTW!

## CONFIGURATION MANAGEMENT

## DISTRIBUTED?

# CONFIG SERVER!


Git Repository
greeting: ohai
2. Source config
Config Server
1. Push config
3. Pull config
App A
greeting: ohai
App B
greeting: ohai
App C
greeting: ohai

## Config Server `app.groovy`

```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://github.com/mstine/config-repo.git
```

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

```
greeting: Bonjour
```

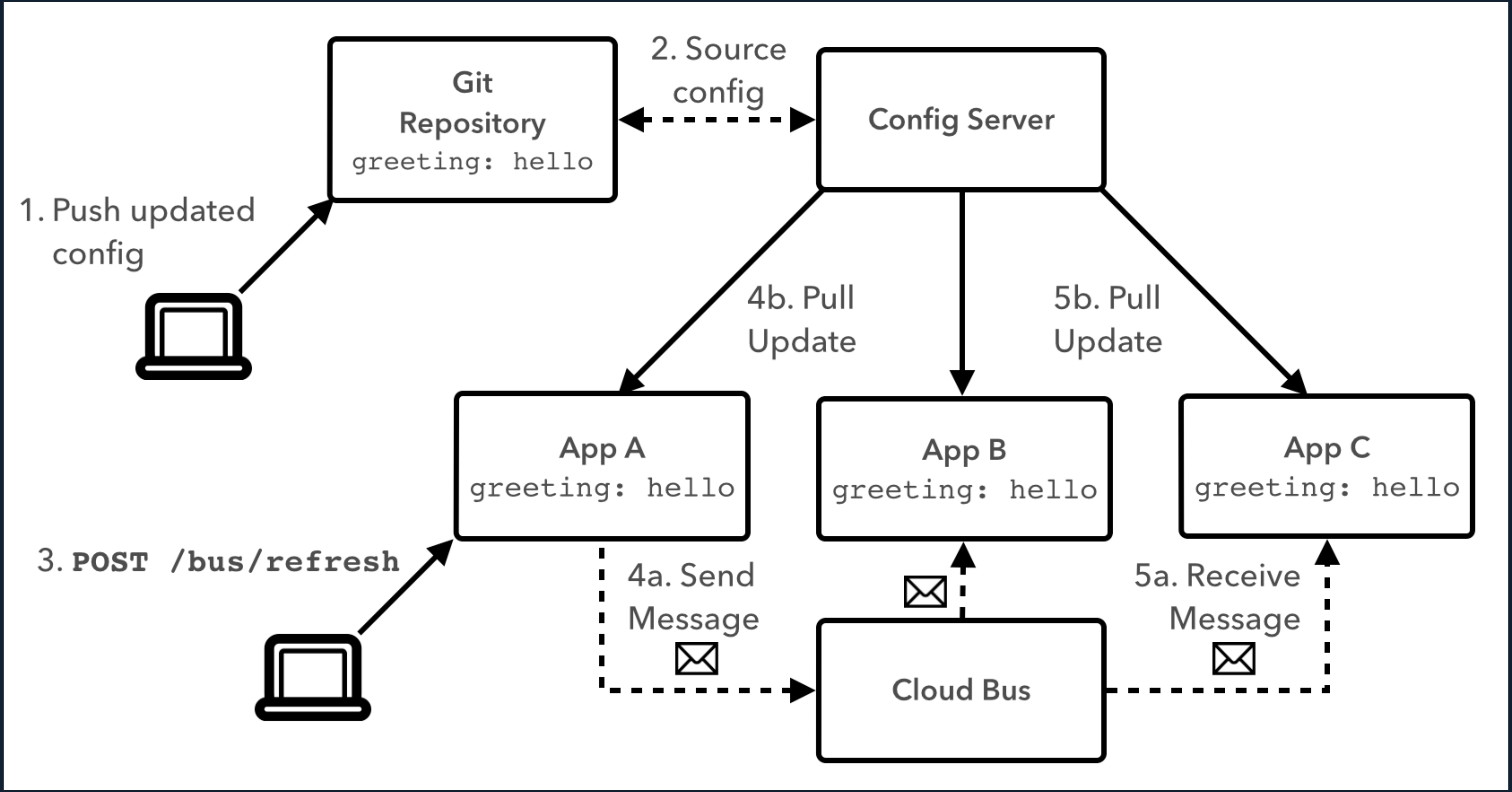
## CONFIG CLIENT app.groovy

```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

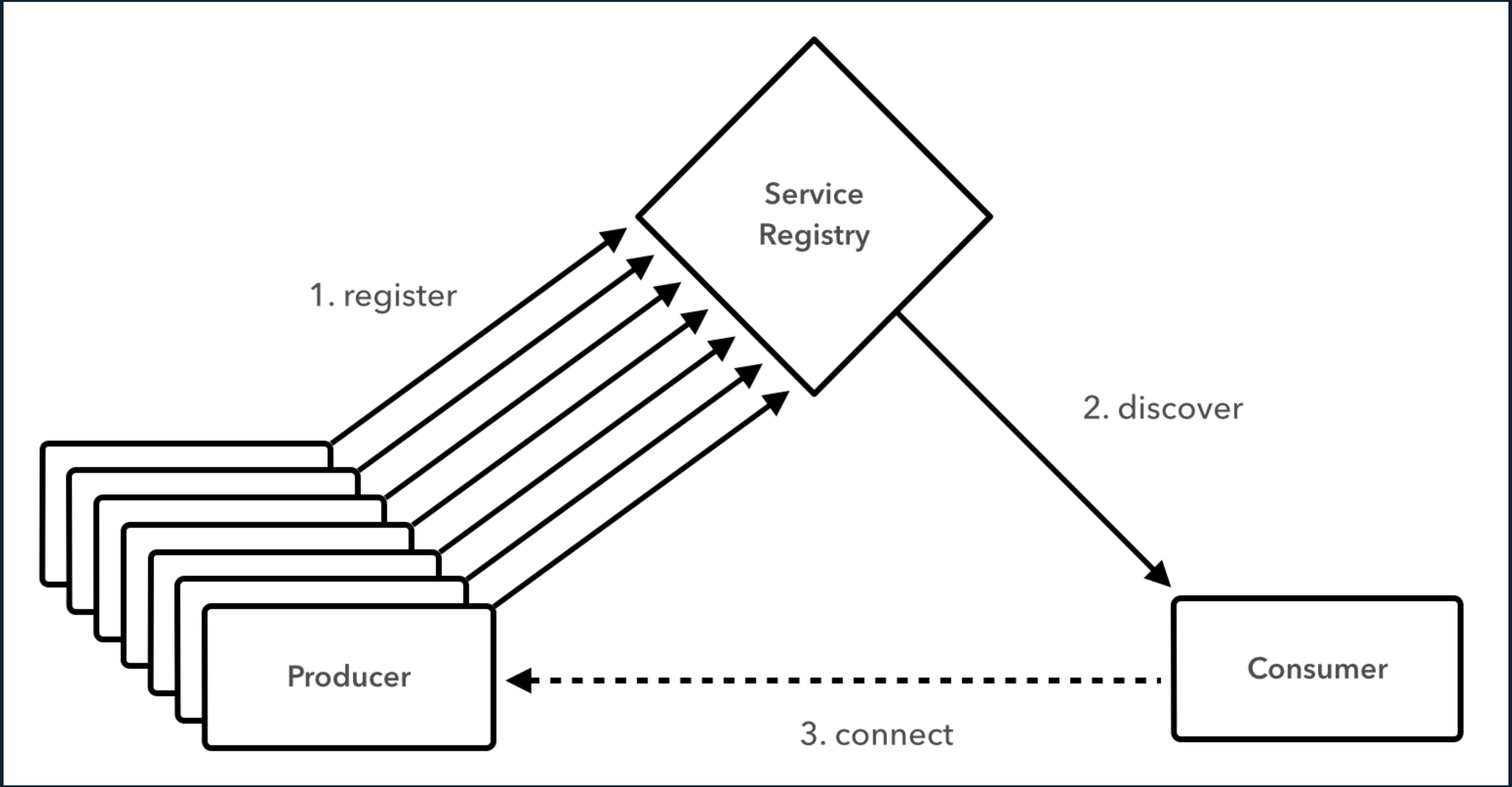
```yaml
spring:
  application:
    name: demo
```

## CLOUD BUS!


Git
Repository
greeting: hello
2. Source config
Config Server
1. Push updated config
4b. Pull Update
5b. Pull Update
App A
greeting: hello
App B
greeting: hello
App C
greeting: hello
3. POST /bus/refresh
4a. Send Message
5a. Receive Message
Cloud Bus

```
curl -X POST http://localhost:8080/bus/refresh
```

## SERVICE REGISTRATION & DISCOVERY


Service Registry
1. register
2. discover
Producer
Consumer
3. connect

NETFLIX OSS

## 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
}
```

# TO THE LABS!