DEPLOYING MICROSERVICES TO

CLOUD FOUNDRY™

ME

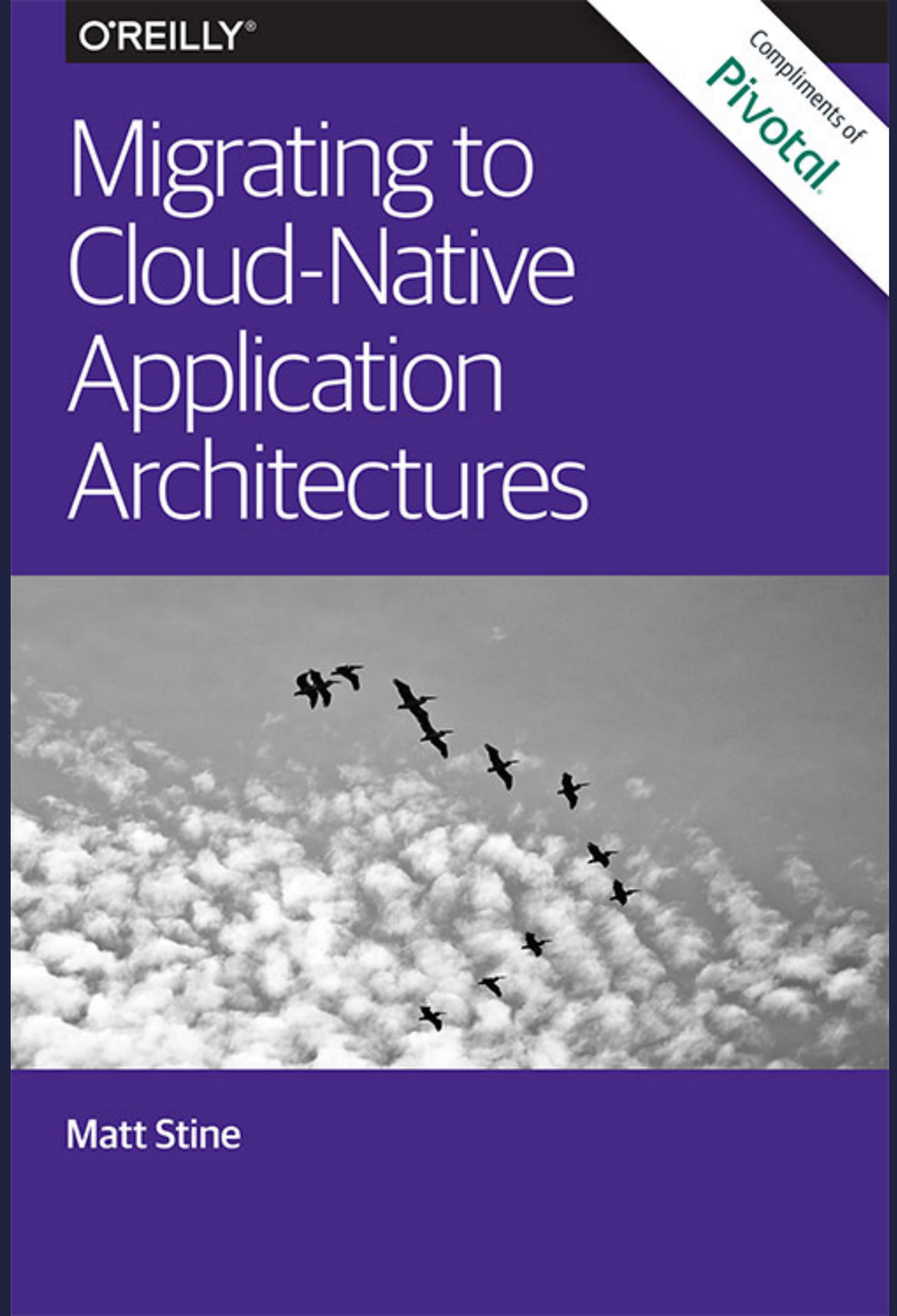
Matt Stine @mstine
Principal Software Engineer
Pivotal
mstine@pivotal.io

I WROTE A LITTLE CLOUD BOOK...

FREE - Compliments of Pivotal

http://bit.ly/cloud-native-book

ATTACK THE PIVOTAL BOOTH TO GET A PRINT COPY!



Now that you have Cloud Foundry, what are you going to do with it?

IF YOU'RE SMART, START WITH A MONOLITH.

The Twelve-Factor App

MAKE IT 12 FACTOR

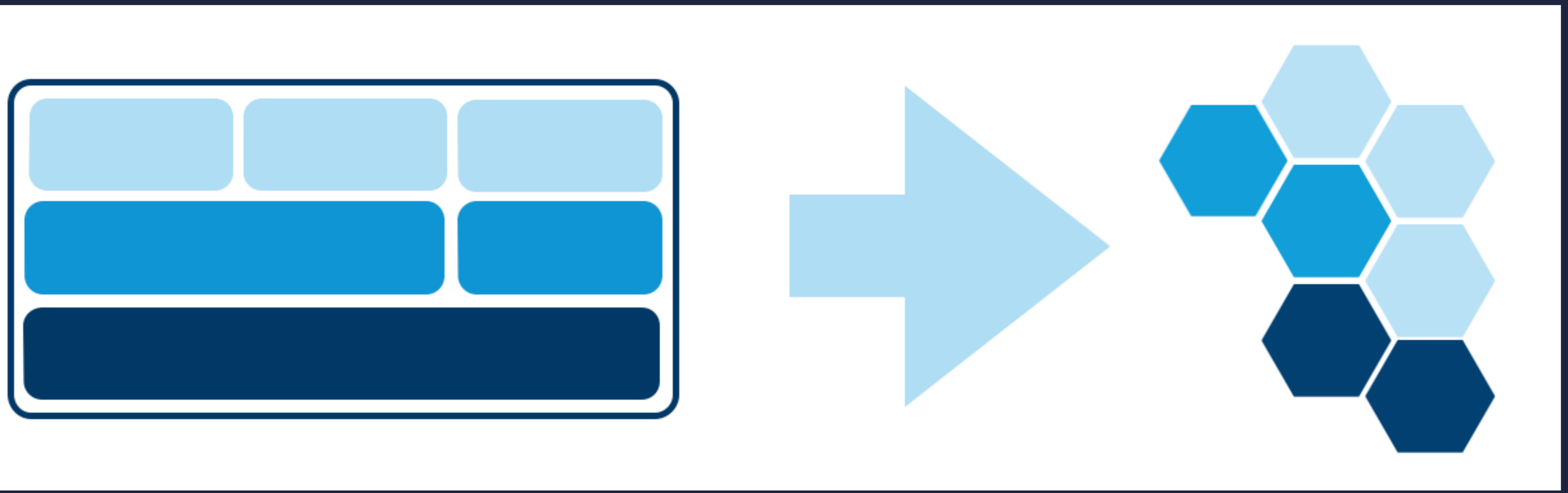
Introduction

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any


EVENTUALLY THINGS GET BIGGER...



...AND IT BECOMES TIME TO DECOMPOSE.


MICROSERVICES?
http://www.etsy.com/shop/sharpwriter
HEUSER 2012

Loosely coupled service oriented architecture with bounded contexts...

– Adrian Cockcroft

YOU MUST BE THIS TALL TO USE MICROSERVICES

- RAPID PROVISIONING
- BASIC MONITORING
- RAPID APPLICATION DEPLOYMENT
- DEVOPS CULTURE

http://martinfowler.com/bliki/MicroservicePrerequisites.html

A SYMBIOTIC RELATIONSHIP

PLATFORM FEATURES

- Environment Provisioning
  - On-Demand Scaling
  - Failover/Resilience
  - Routing/Load Balancing
- Data Service Operations (BOSH)


NO MICROSERVICE
IS AN ISLAND

SOME CHALLENGES OF DISTRIBUTED SYSTEMS

- Configuration Management
- Service Registration & Discovery
- Routing & Load Balancing
- Fault Tolerance
- Monitoring

We need a representation of the composite system!

THE BIG A APP

```
---
memory: 512M
instances: 1
no-route: true
services:
  - cloudamqp-autoscale
applications:
- name: worker-process
  path: worker-process/build/libs/worker-process.jar
- name: producer-process
  path: producer/build/libs/producer.jar
- name: autoscaler-process
  path: autoscaler/build/libs/autoscaler.jar
- name: autoscale-monitor
  no-route: false
  host: autoscale-monitor-${random-word}
  path: monitor/build/libs/monitor.jar
```

STATIC VS. DYNAMIC

Like BOSH, but for microservices?

DECENTRALIZED
AUTONOMOUS
CAPABILITY
TEAMS / SERVICES


You write it, you run it!


NETFLIX
OSS

SPRING CLOUD

DISTRIBUTED SYSTEM PATTERNS FTW!

APP/SERVICE LEVEL PATTERNS

DEPLOYING/RUNNING APPS

VS.

COMPOSING FAULT TOLERANT SYSTEMS

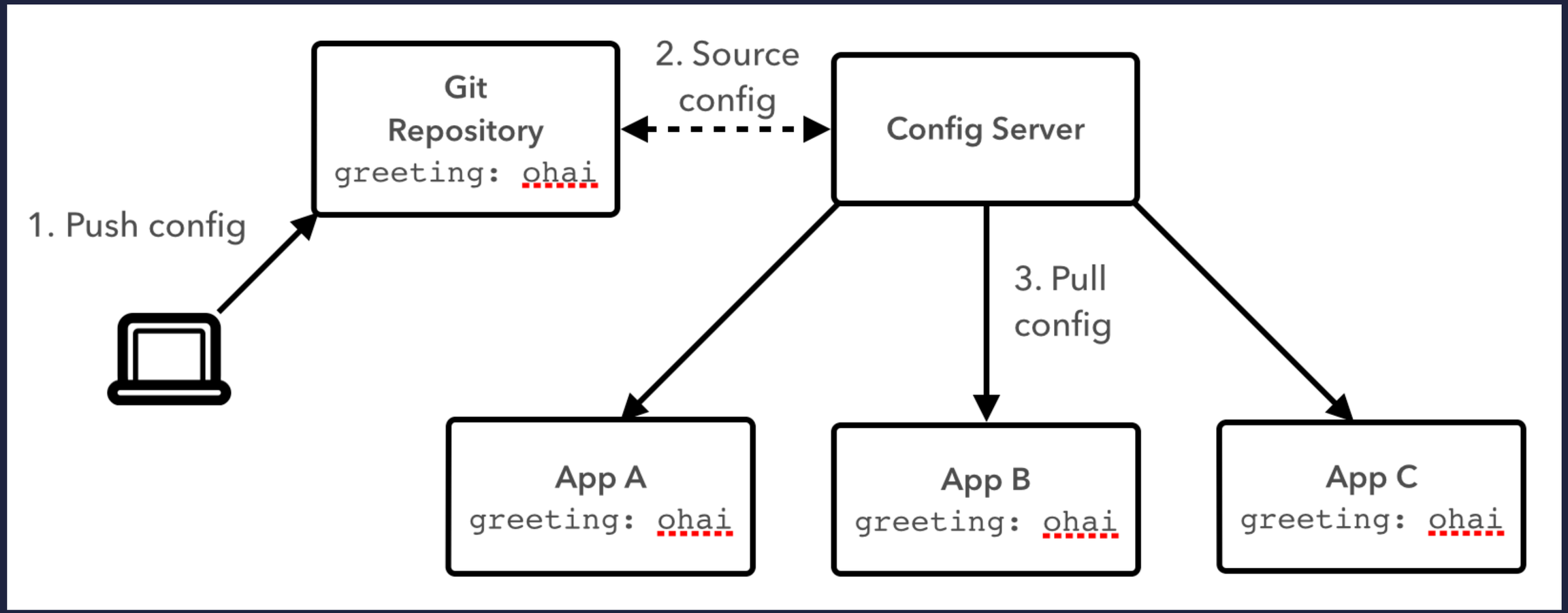
SPRING CLOUD + NETFLIX OSS

- Config Server / Cloud Bus
- Eureka (Service Registry)
- Ribbon (Load Balancer)
- Hystrix (Circuit Breakers)
- Zuul (Intelligent Routing)

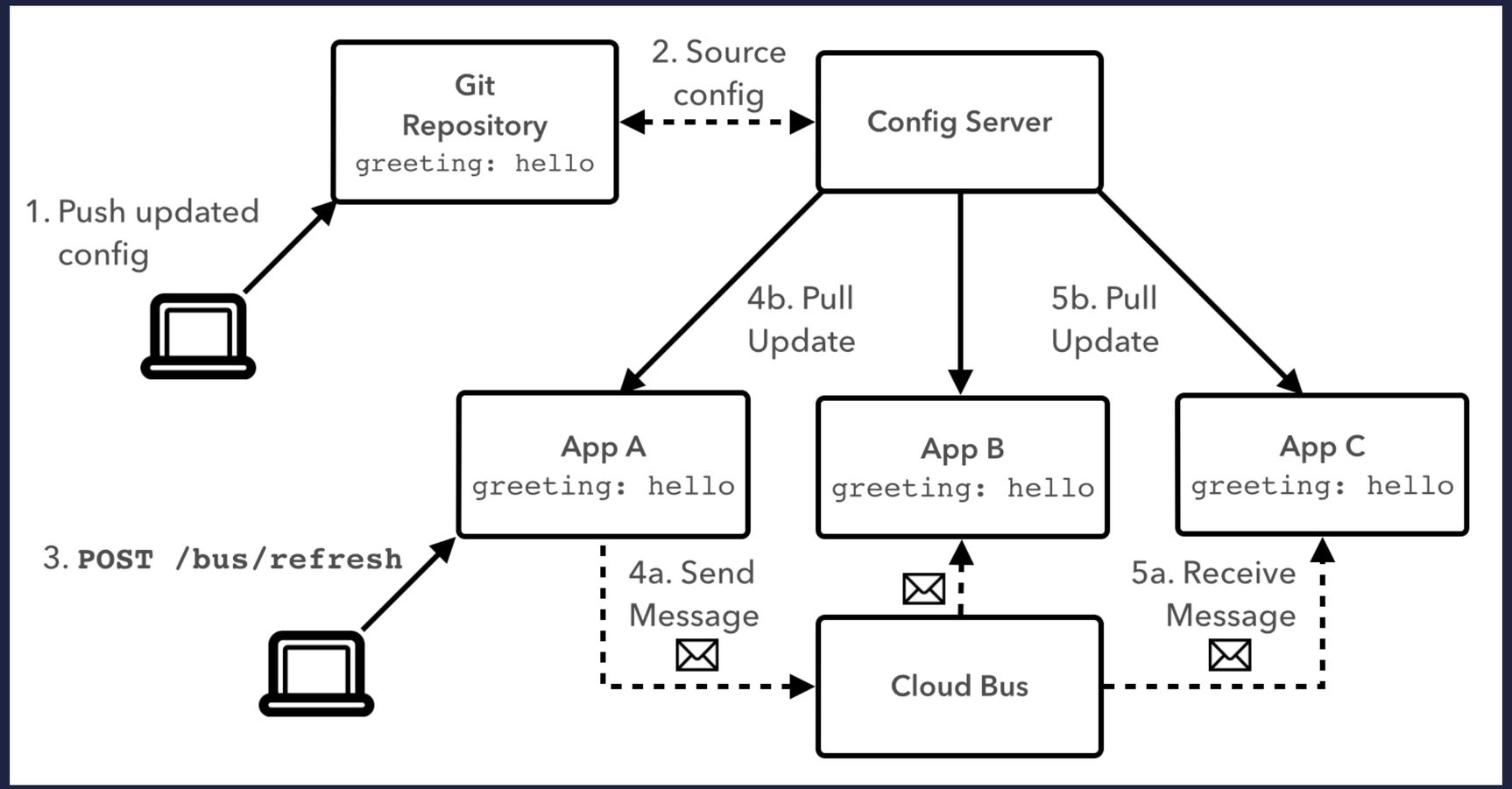
WORKS ON LATTICE TOO!

https://lattice.cf

CONFIG SERVER

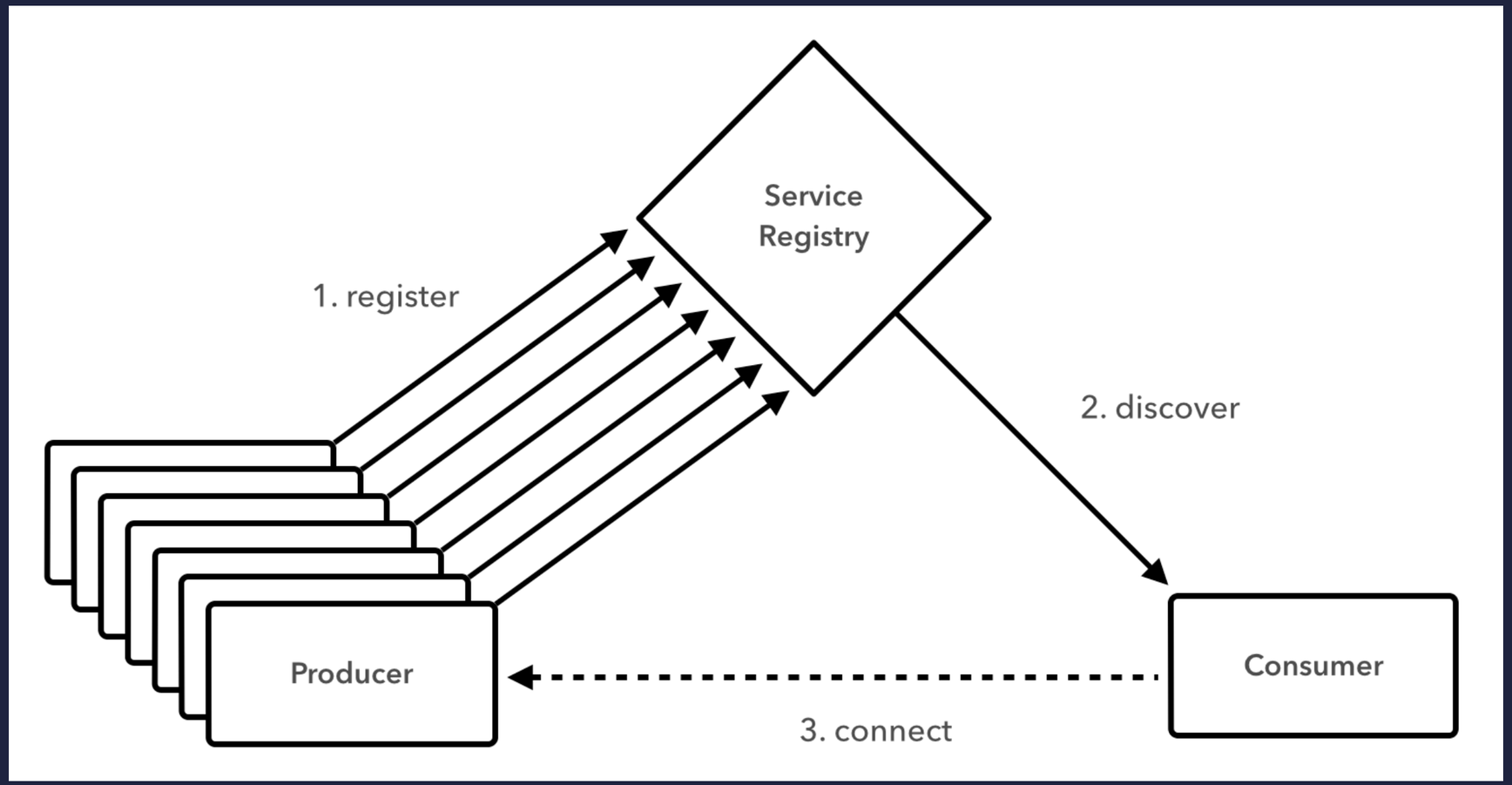


CONFIG SERVER + CLOUD BUS



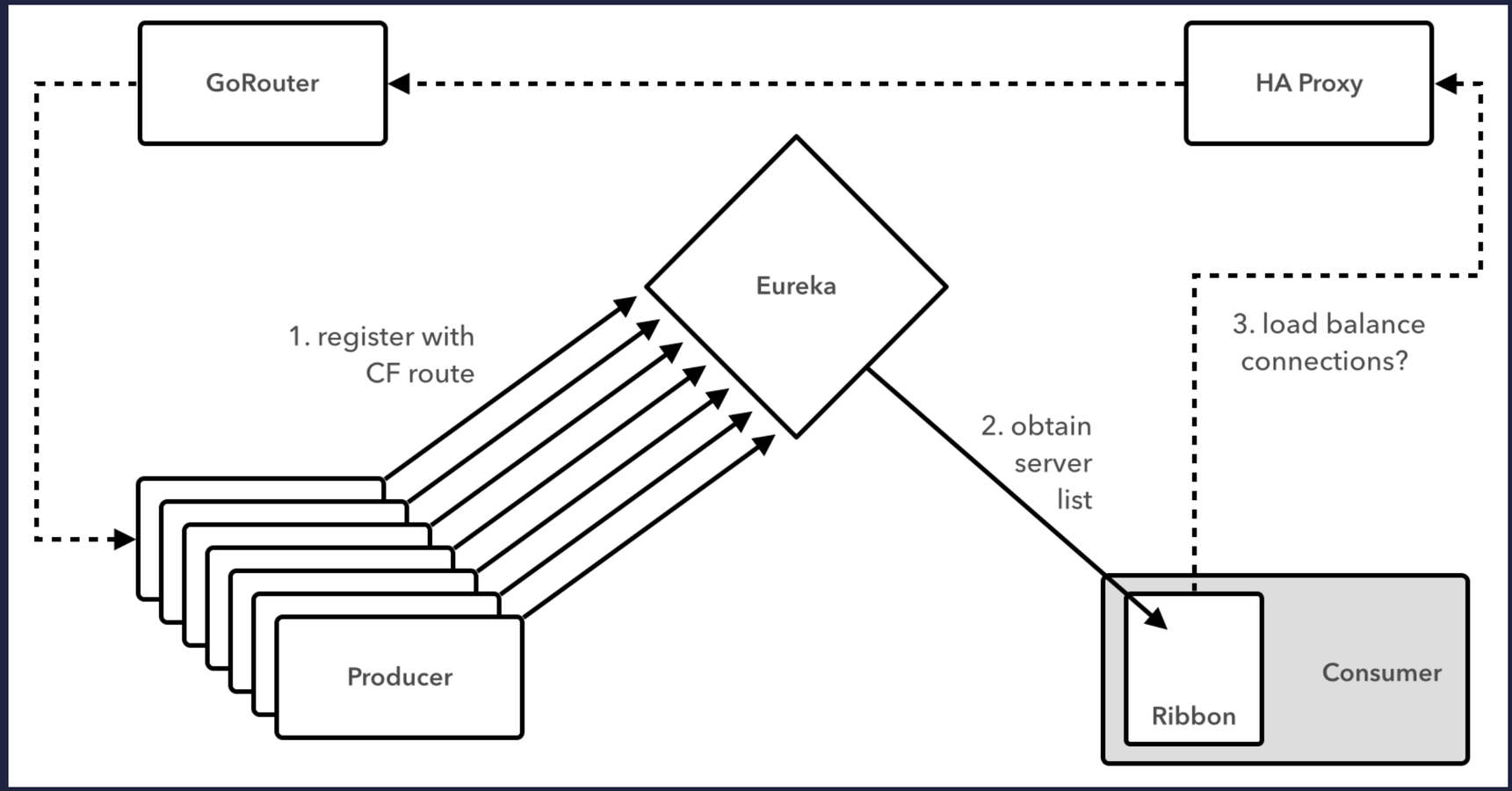
DEMO

EUREKA (SERVICE REGISTRY)



DEMO

BEFORE CF_RELEASE V204



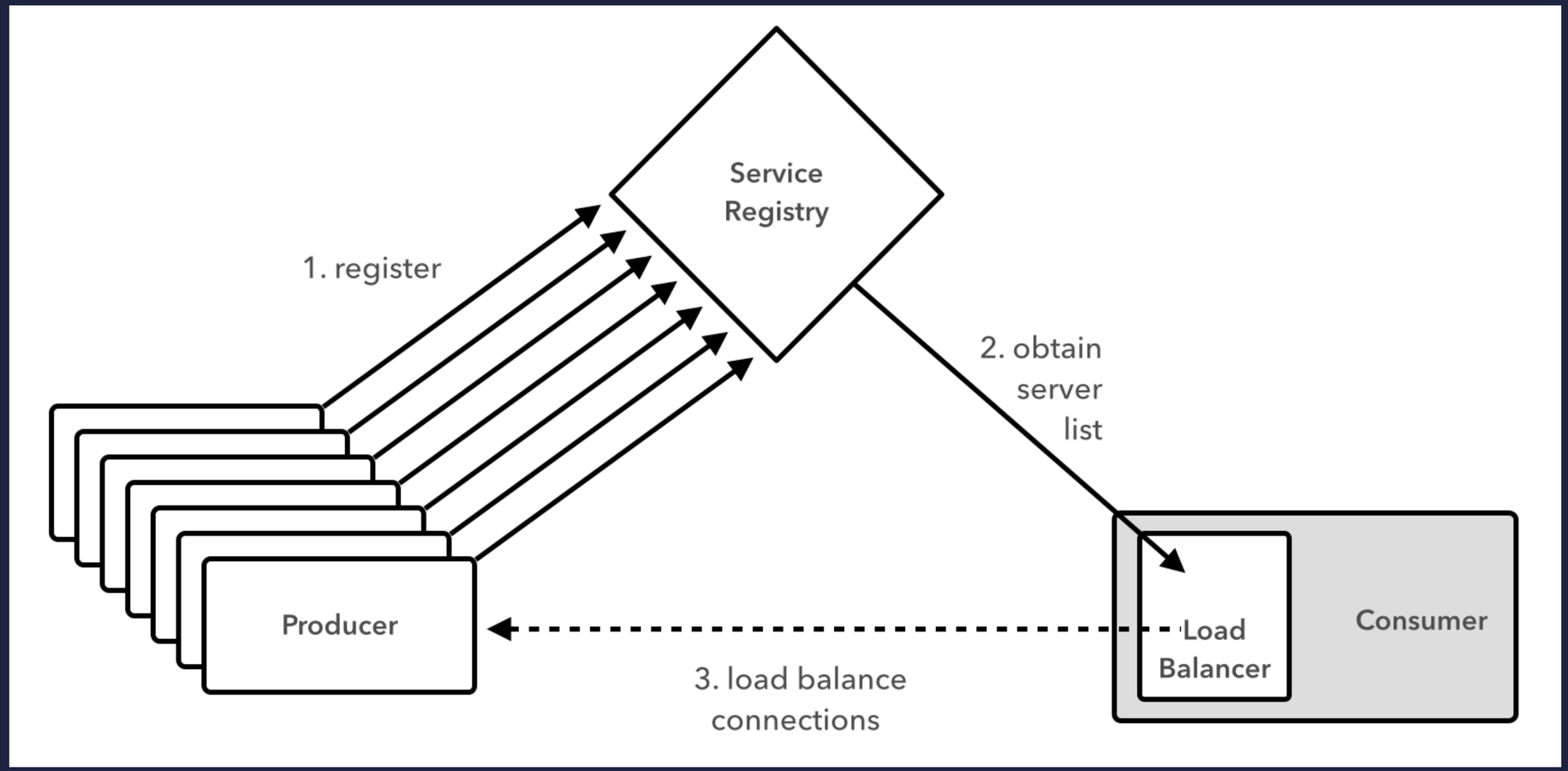
CF_RELEASE V195

```
"CF_INSTANCE_INDEX"=>"0",
"CF_INSTANCE_IP"=>"1.2.3.4",
"CF_INSTANCE_PORT"=>"5678",
"CF_INSTANCE_ADDR"=>"1.2.3.4:5678",
"CF_INSTANCE_PORTS"=>[{external:80,internal:5678}]
```

CF_RELEASE V204

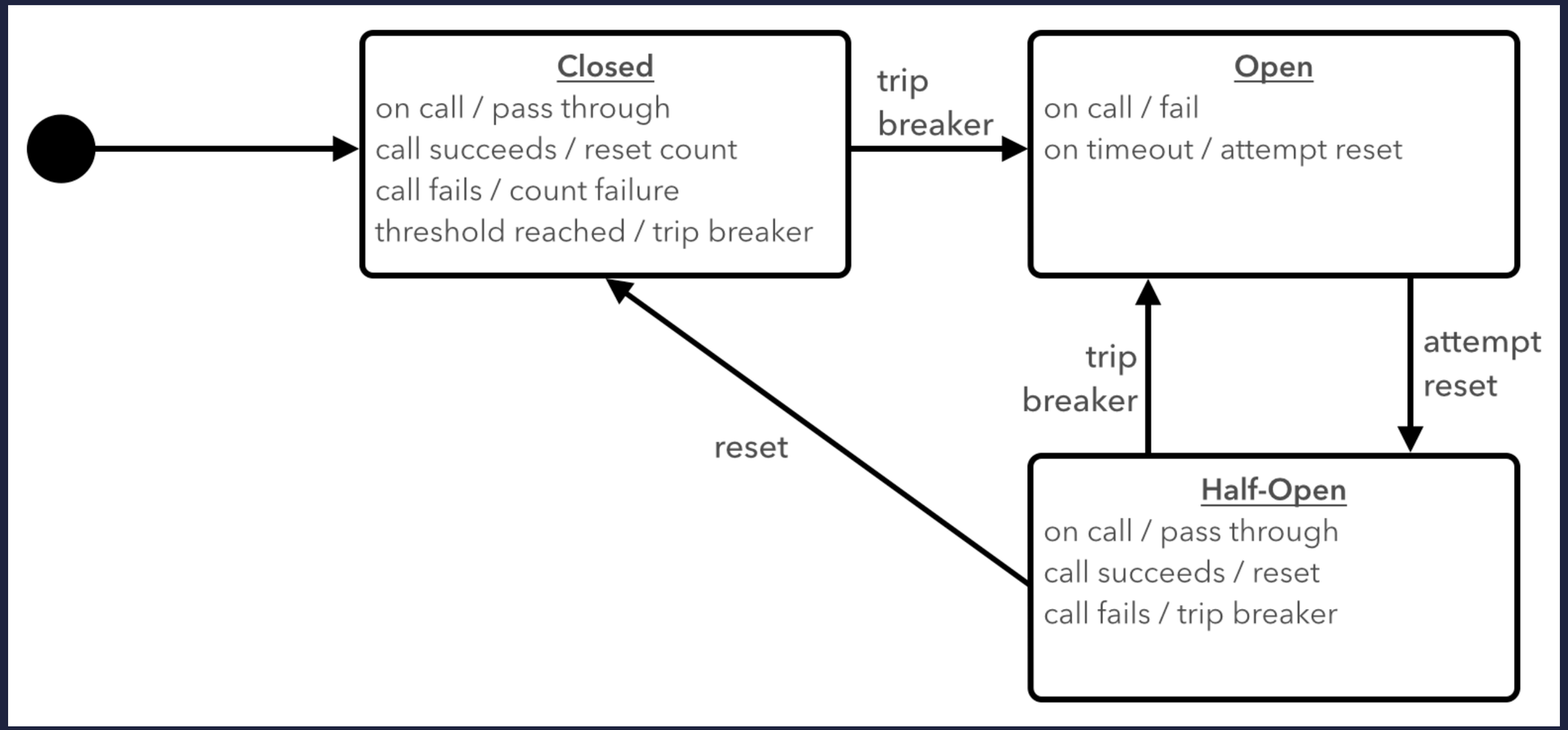
```
properties:
  dea_next:
    allow_host_access: true
```

AFTER CF_RELEASE V204



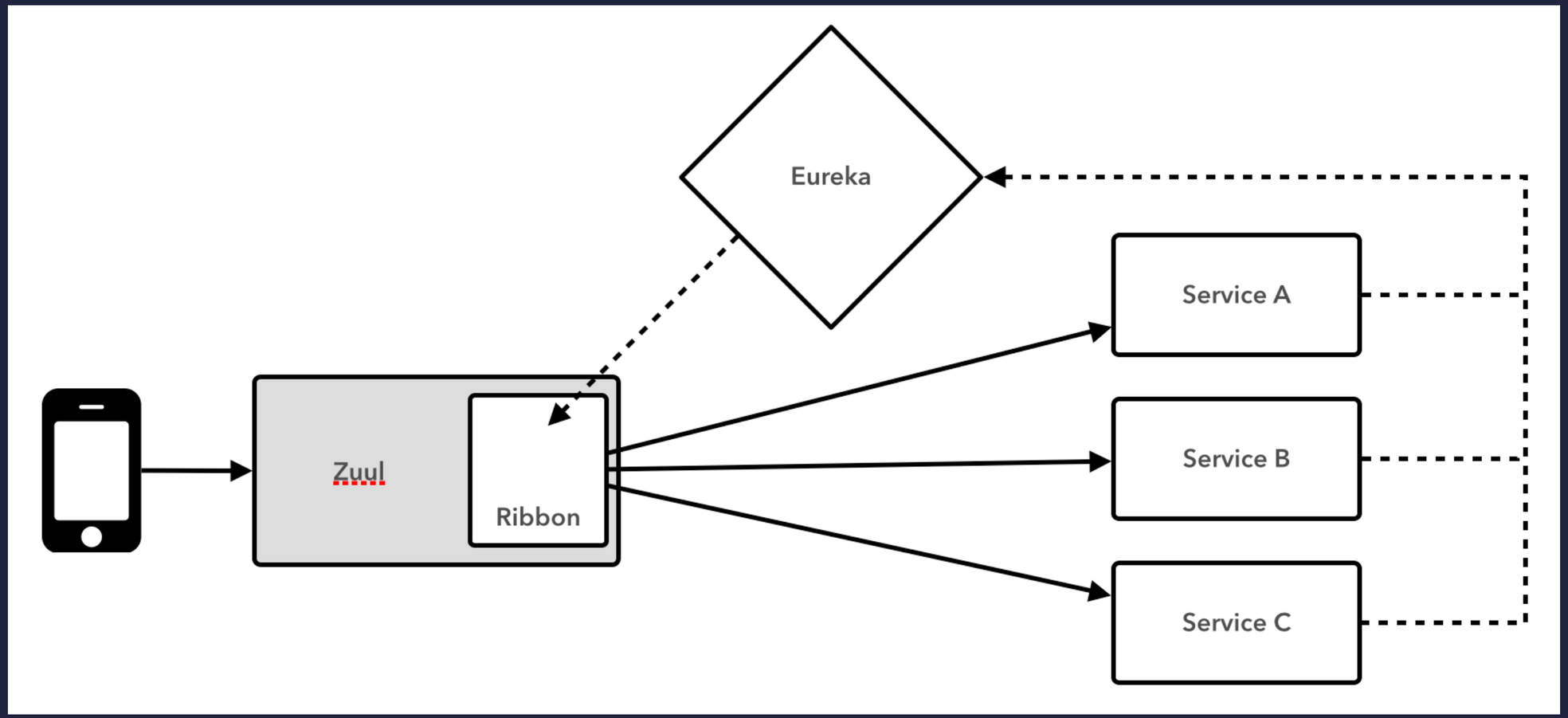
DEMO

HYSTRIX (CIRCUIT BREAKER)



DEMO

ZUUL (INTELLIGENT ROUTING)



DEMO

SPRING CLOUD FUTURES

- Alternative Stacks (Consul, Zookeeper, etcd, JRugged, ...)
- Distributed Request Tracing/Correlation (i.e. Dapper/Zipkin)
- Stateful Patterns (Leader Election, Locks, State Machine)
- Developer Workflow Improvements
- Switches (Canaries, Feature Toggles, Surgical Routing, ...)

LEARN MOAR

- http://cloud.spring.io
- http://spring.io/blog/2015/04/06/lattice-and-spring-cloud-resilient-sub-structure-for-your-cloud-native-spring-applications
- http://lattice.cf
- http://netflix.github.io

THANK YOU

- THIS TALK: https://github.com/mstine/2015-cfsummit-deploying-ms-to-cf
- DEMOS: https://github.com/mstine/intro-spring-cloud-workshop/tree/lattice