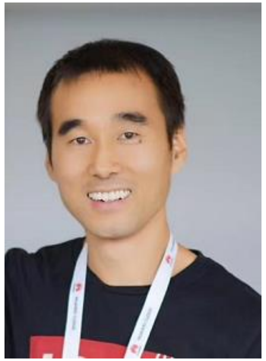
Best practice: from spring cloud to Istio

zhangchaomeng@huawei.com

#IstioCon

About me



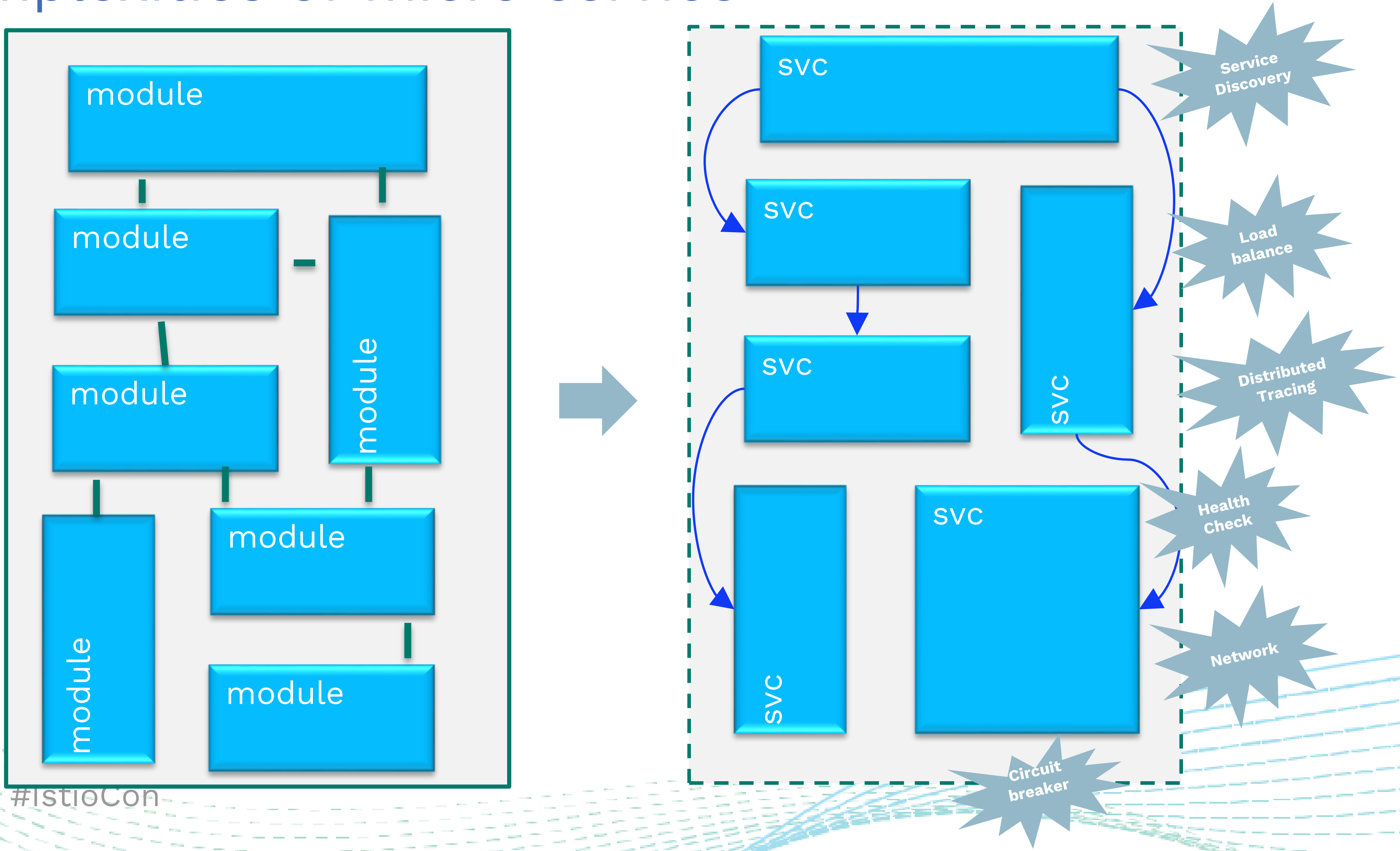
Chaomeng Zhang is the Chief Architect of Huawei Cloud ASM (Application Service Mesh) service, which is based on Istio and Kubernetes.

Chaomeng has been working on cloud native technologies for more than 6 years, including Kubernetes, microservices, service catalog, APM, devops and service mesh for now. He is an Istio community member, author of one bestselling Chinese Istio book "Cloud Native Service Mesh Istio". He is also an experienced speaker of many cloud native and open source conferences, including KubeCon, Cloud Native Days, Service mesh meetup, k8smeetup.

Agenda

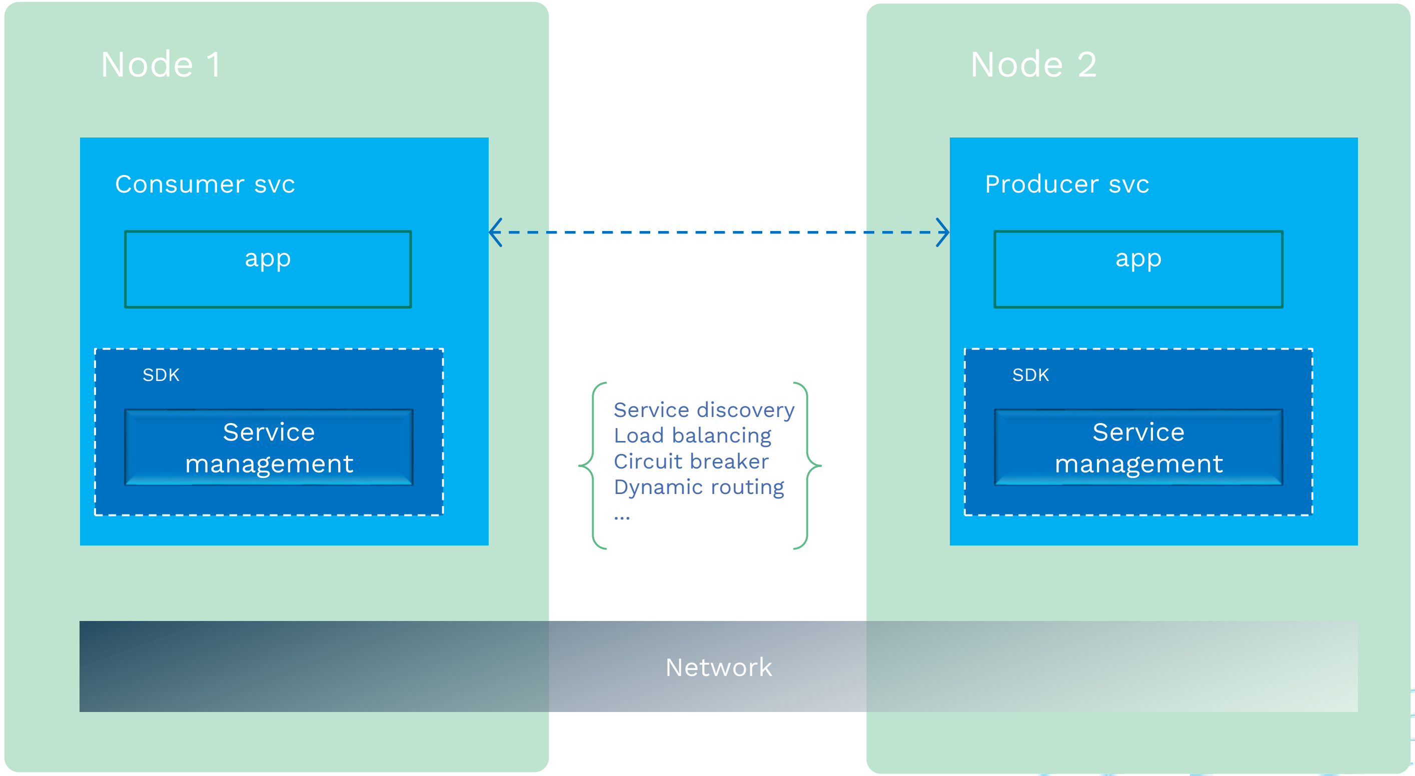
- Concepts
- Problems
- Solutions
- Practice

Complexities of micro service

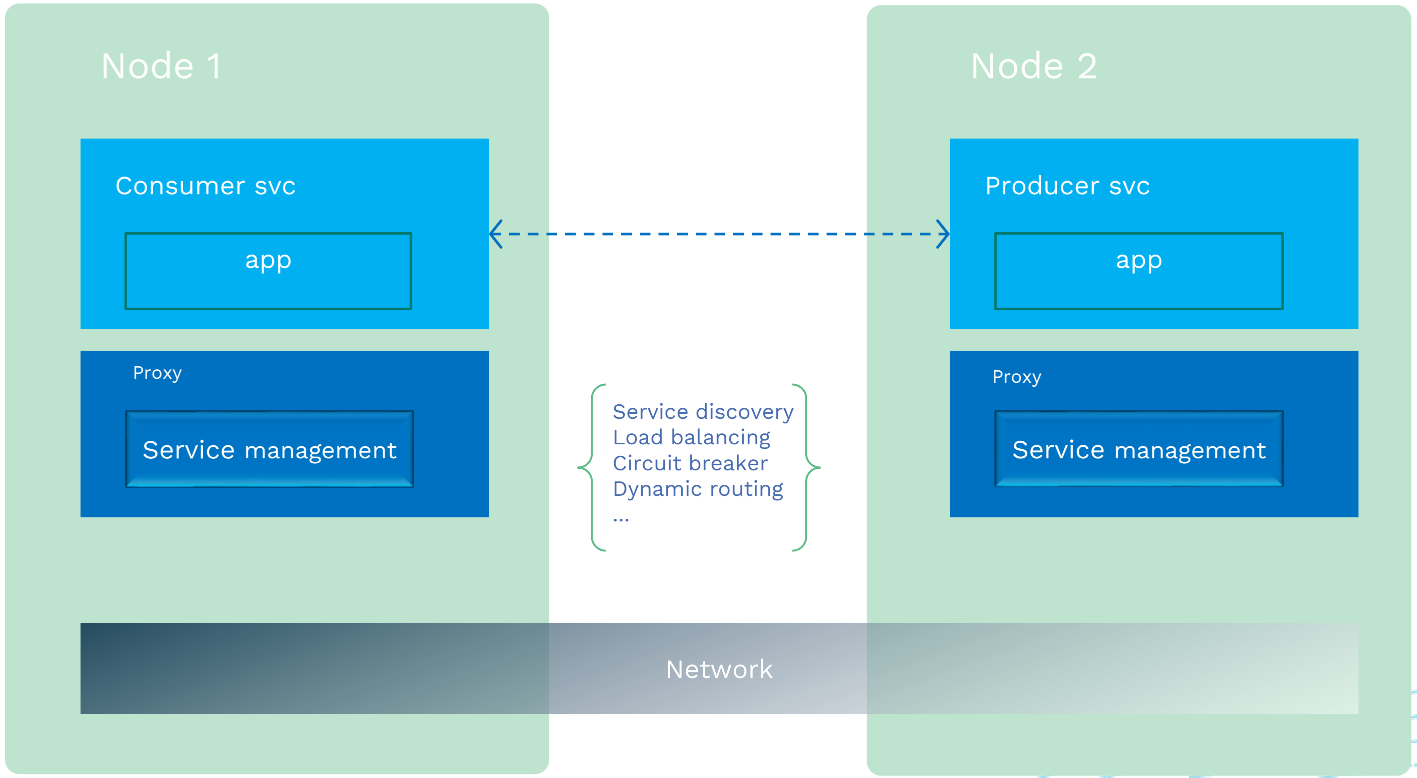


#IstioCon

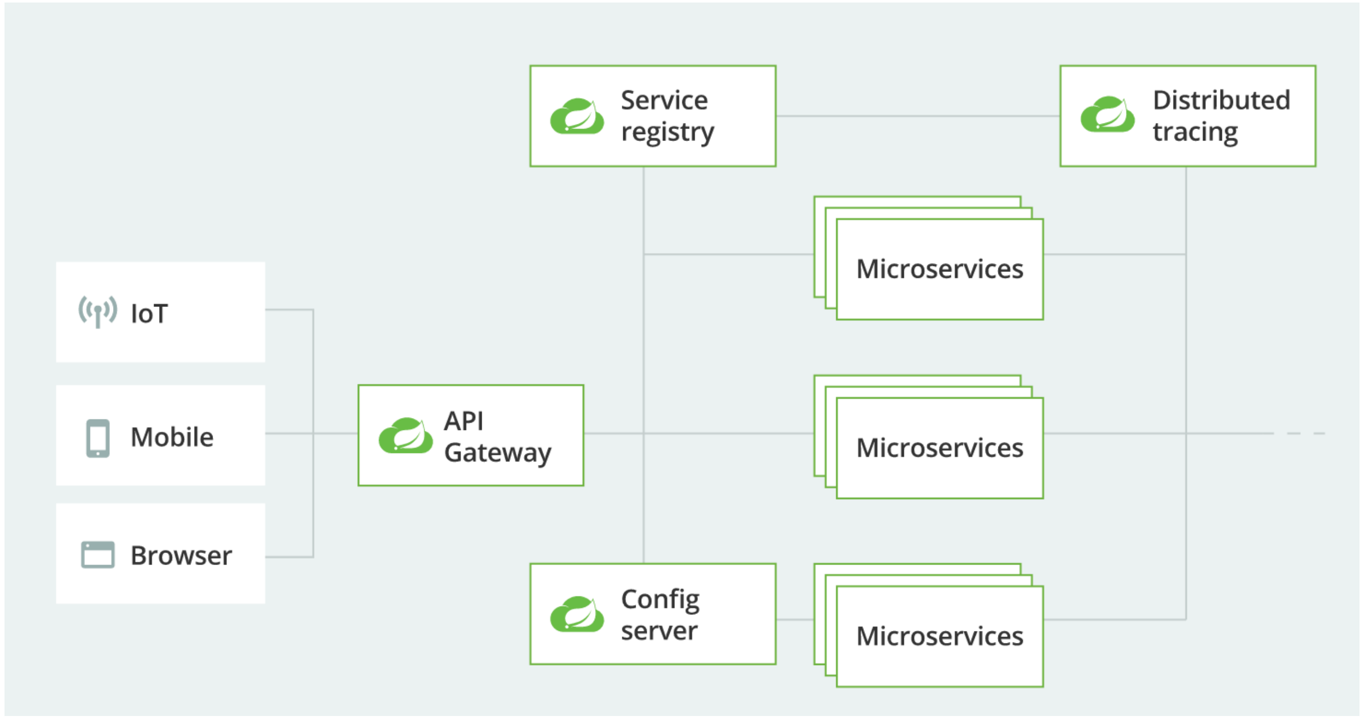
Microservice SDK



Service Mesh

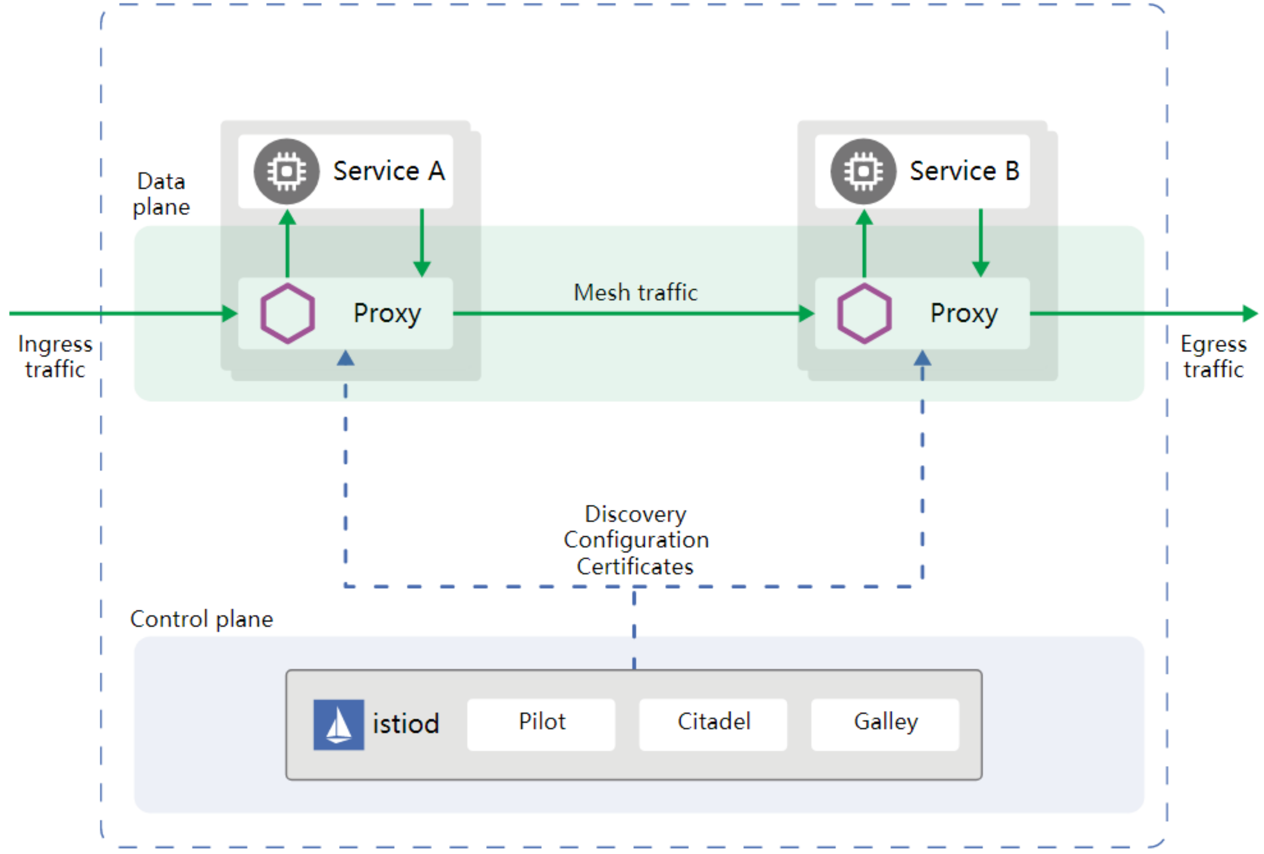


Spring cloud



- Spring Cloud Bus
- Spring Cloud Circuit Breaker
- Spring Cloud CLI
- Spring Cloud for Cloud Foundry
- Spring Cloud - Cloud Foundry Service Broker
- Spring Cloud Cluster
- Spring Cloud Commons
- Spring Cloud Config
- Spring Cloud Connectors
- Spring Cloud Consul
- Spring Cloud Contract
- Spring Cloud Function
- Spring Cloud Gateway
- Spring Cloud GCP
- Spring Cloud Kubernetes
- Spring Cloud Netflix
- Spring Cloud Open Service Broker
- Spring Cloud OpenFeign
- Spring Cloud Pipelines
- Spring Cloud Schema Registry
- Spring Cloud Security
- Spring Cloud Skipper
- Spring Cloud Sleuth
- Spring Cloud Stream
- Spring Cloud Stream App Starters
- Spring Cloud Stream Applications
- Spring Cloud Task
- Spring Cloud Task App Starters
- Spring Cloud Vault
- Spring Cloud Zookeeper
- Spring Cloud App Broker

Istio



Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.

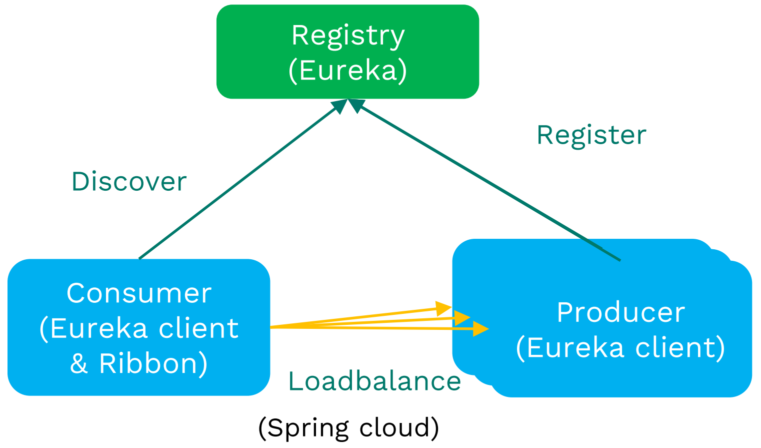
Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.

A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.

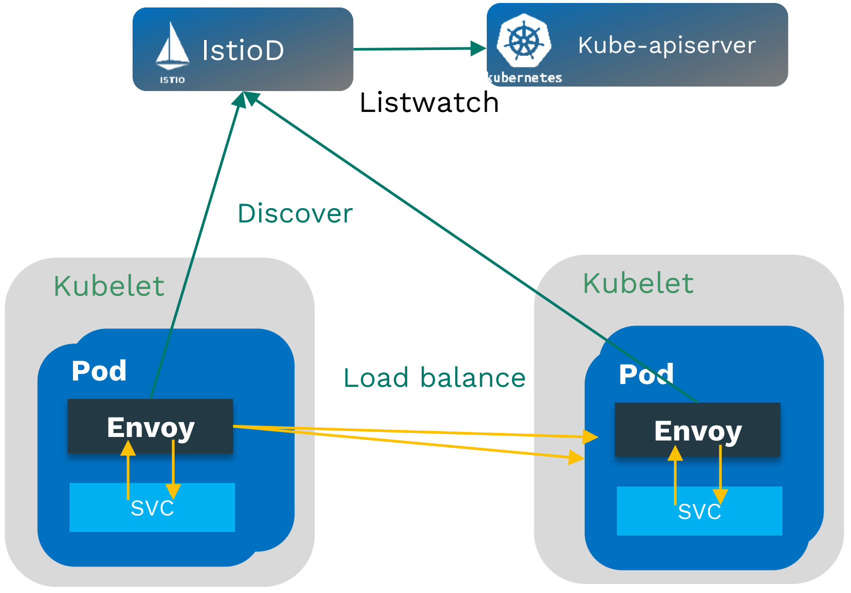
Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.

Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Service Discovery & Load balance



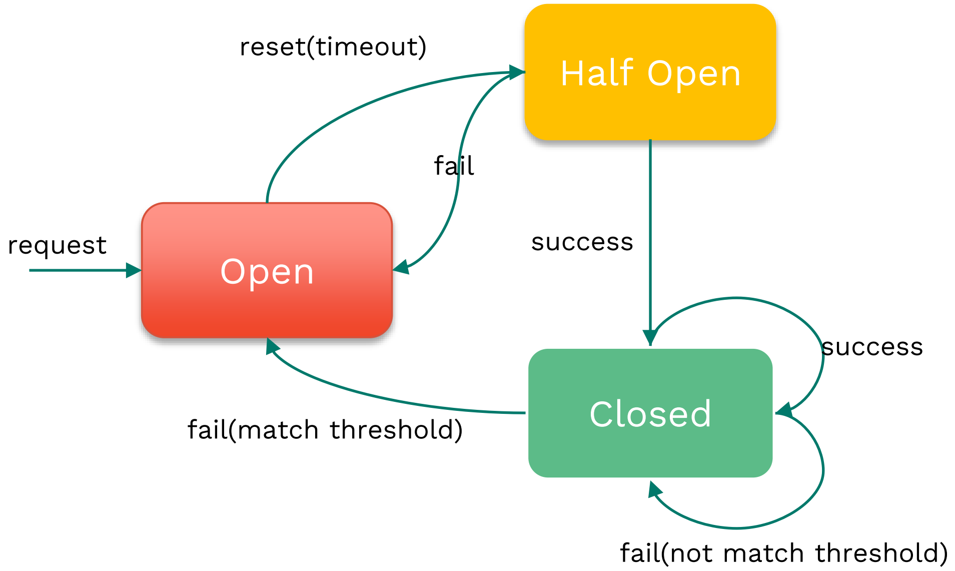
(Spring cloud)



(Istio)

	Spring cloud	Istio
Service Registry	Service register to Eureka	Not need
Service discovery	Consumer calls eureka get instance list	Pilot listwatch kube-apiserver service and endpoints
Load balance	Ribbon in SDK selet instance	Proxy select instance
Location	In process	Out of process

Circuit breaker

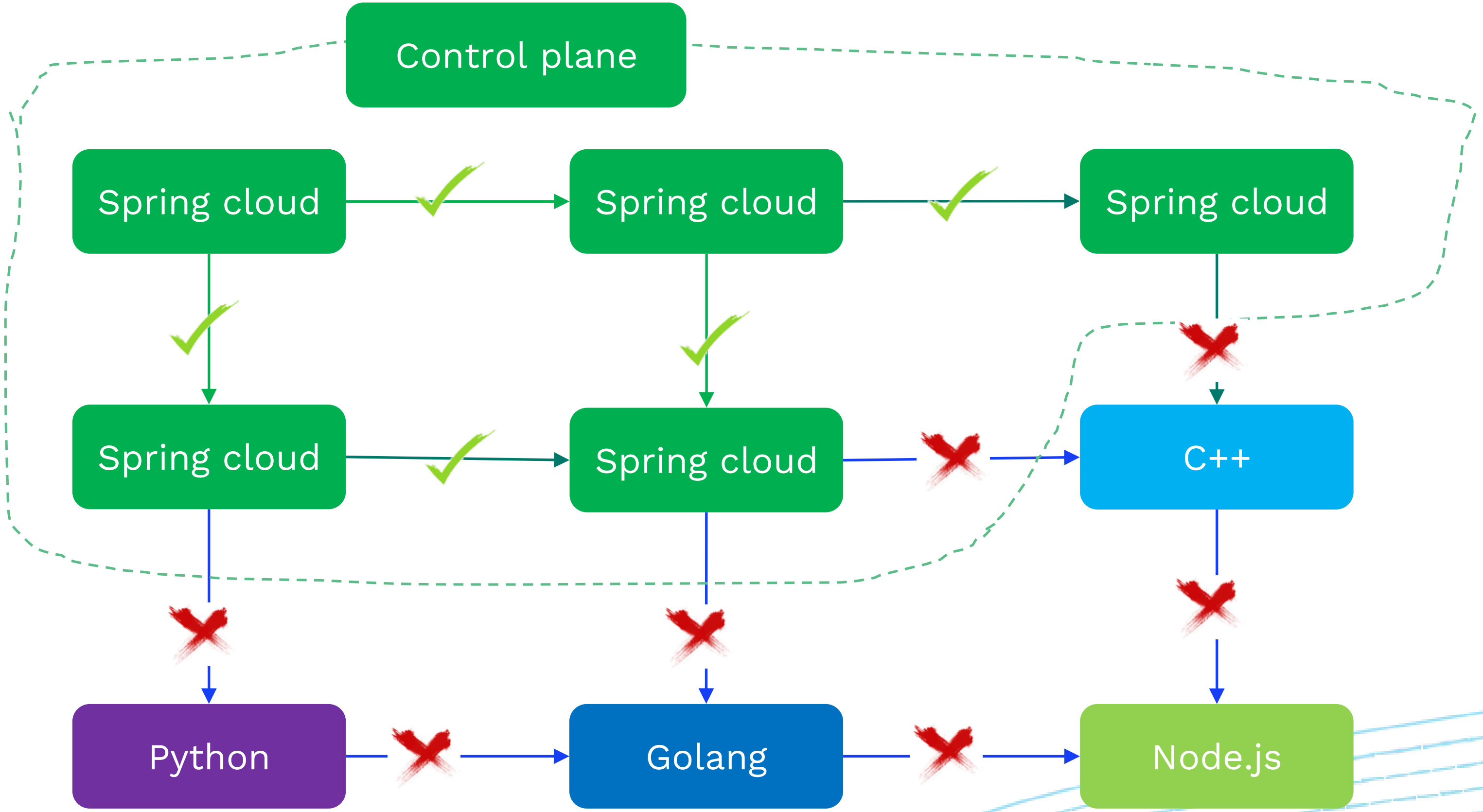


Compare	Hystrix	Istio
Method	White box	Black box
Actions	Coding fallback	Only configuration
With application	Wraped with HystrixCommand	Non-invasive, by proxy
Function	Circuit breaker, threadpool	Outlier detection, threadpool
Protect functionality	Micro service call, and other potentially risky functionality	Service call over the network with fault and latency tolerance

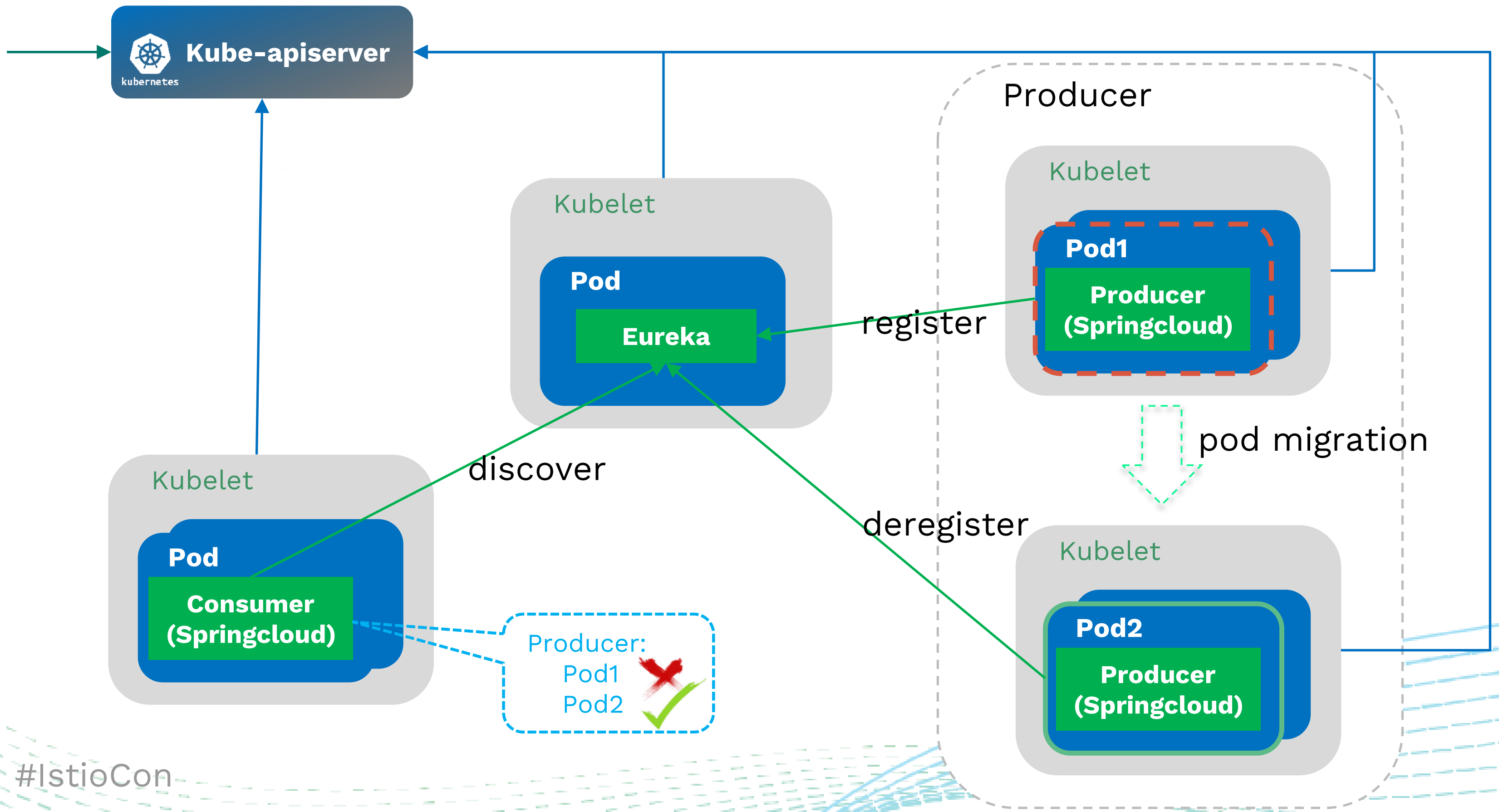
Agenda

- Concepts
- Problems
- Solutions
- Practice

Problem 1: Multi language

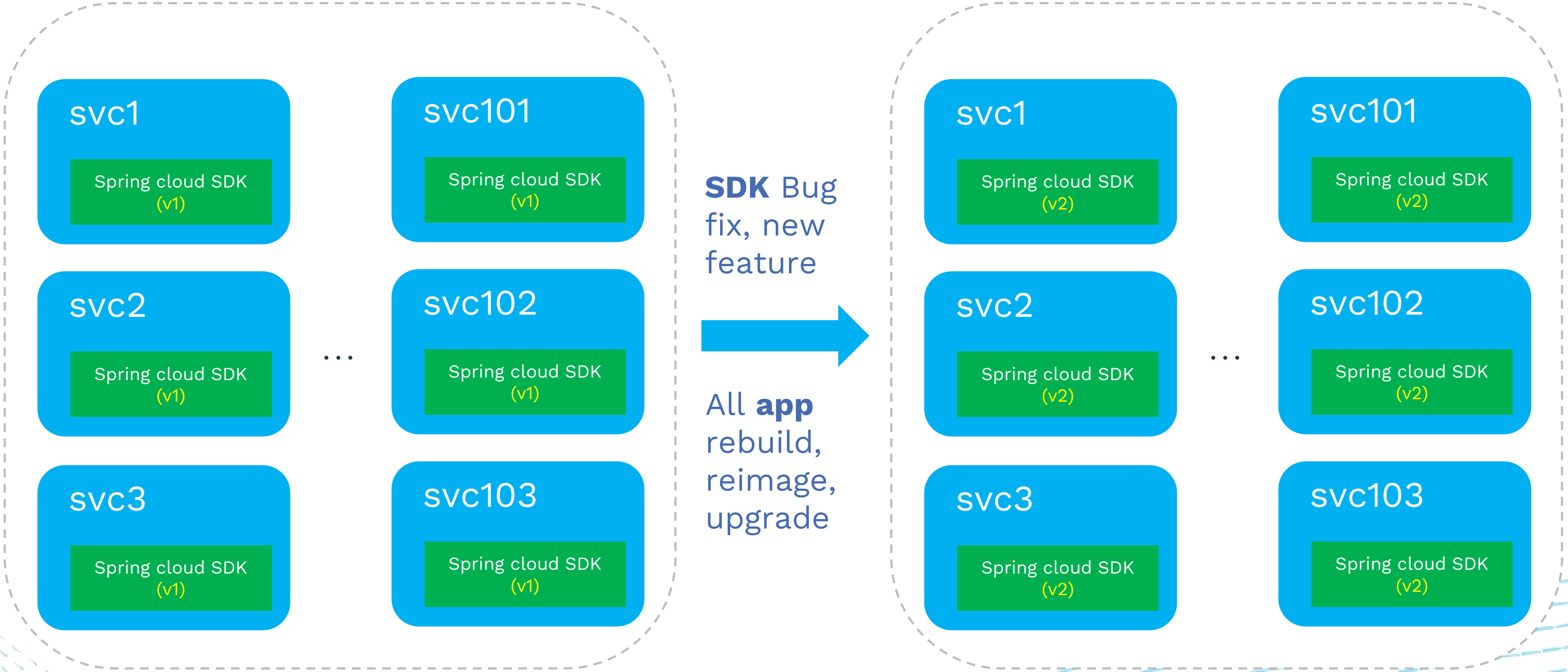


Problem 2: Discovery latency when Spring cloud running inside Kubernetes

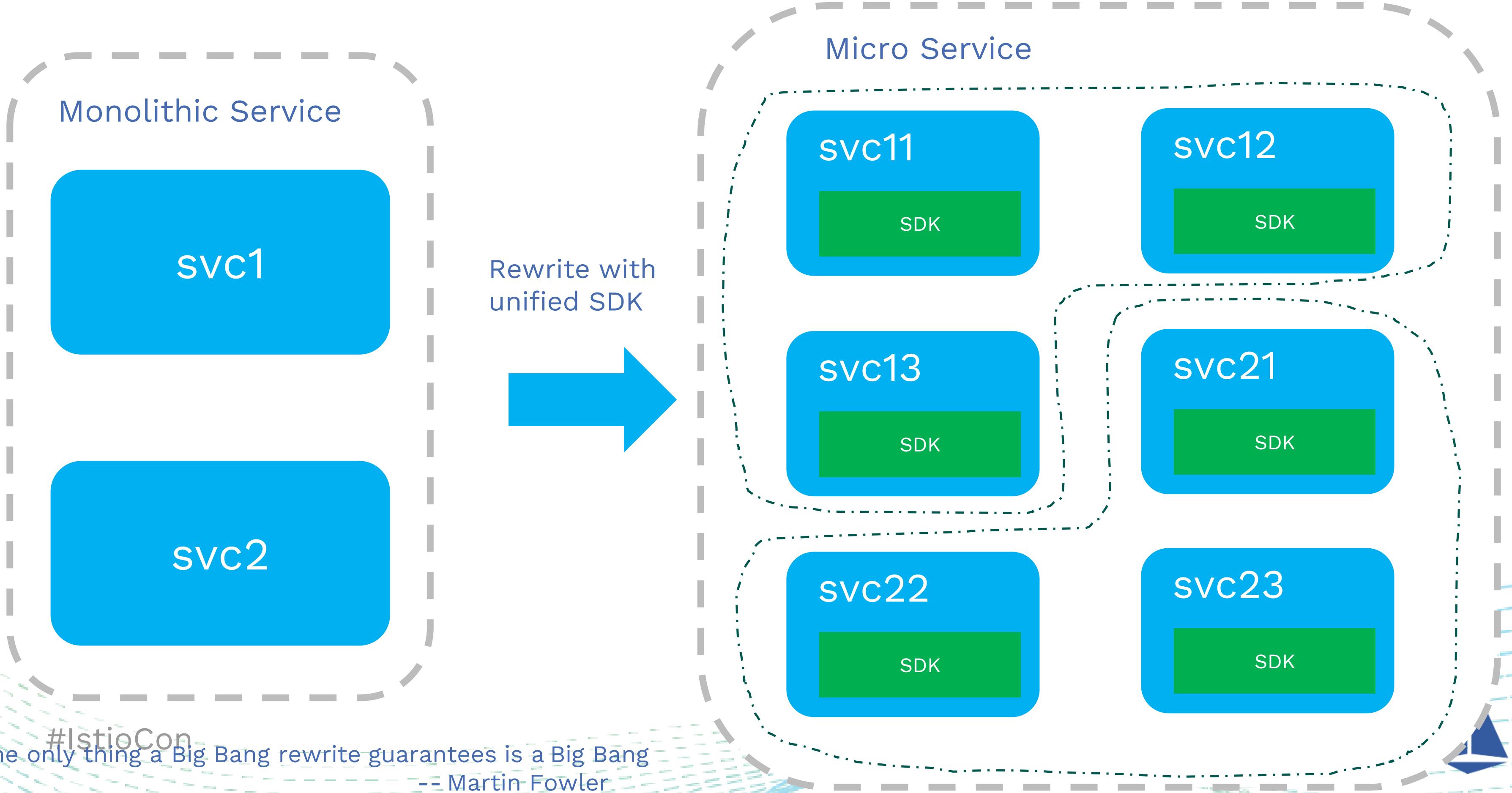


#IstioCon

Problem 3: Upgrade all application in case of service management changing



Problem 4: Gradually migrate from a monolith to micro services



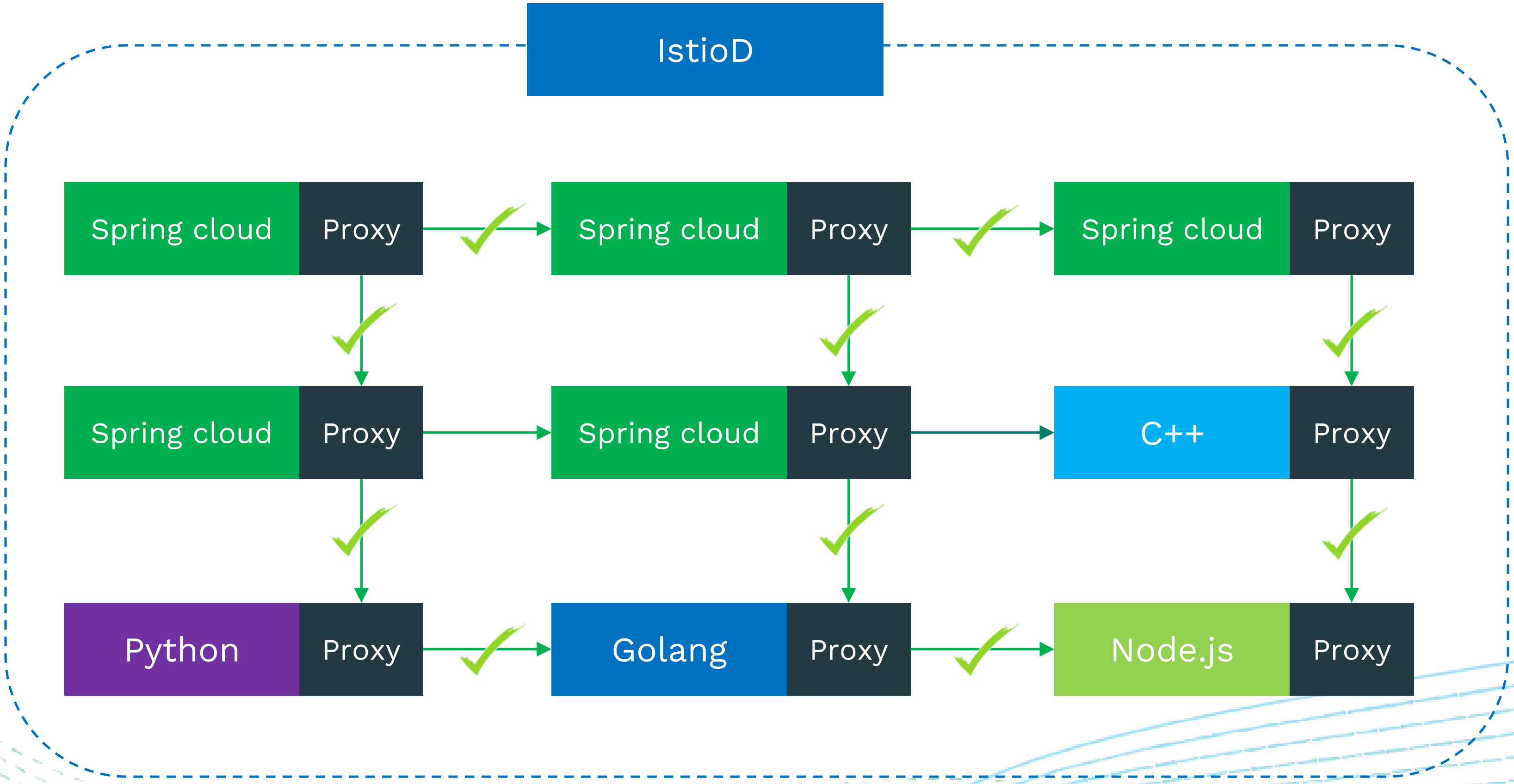
#IstioCon

The only thing a Big Bang rewrite guarantees is a Big Bang
-- Martin Fowler

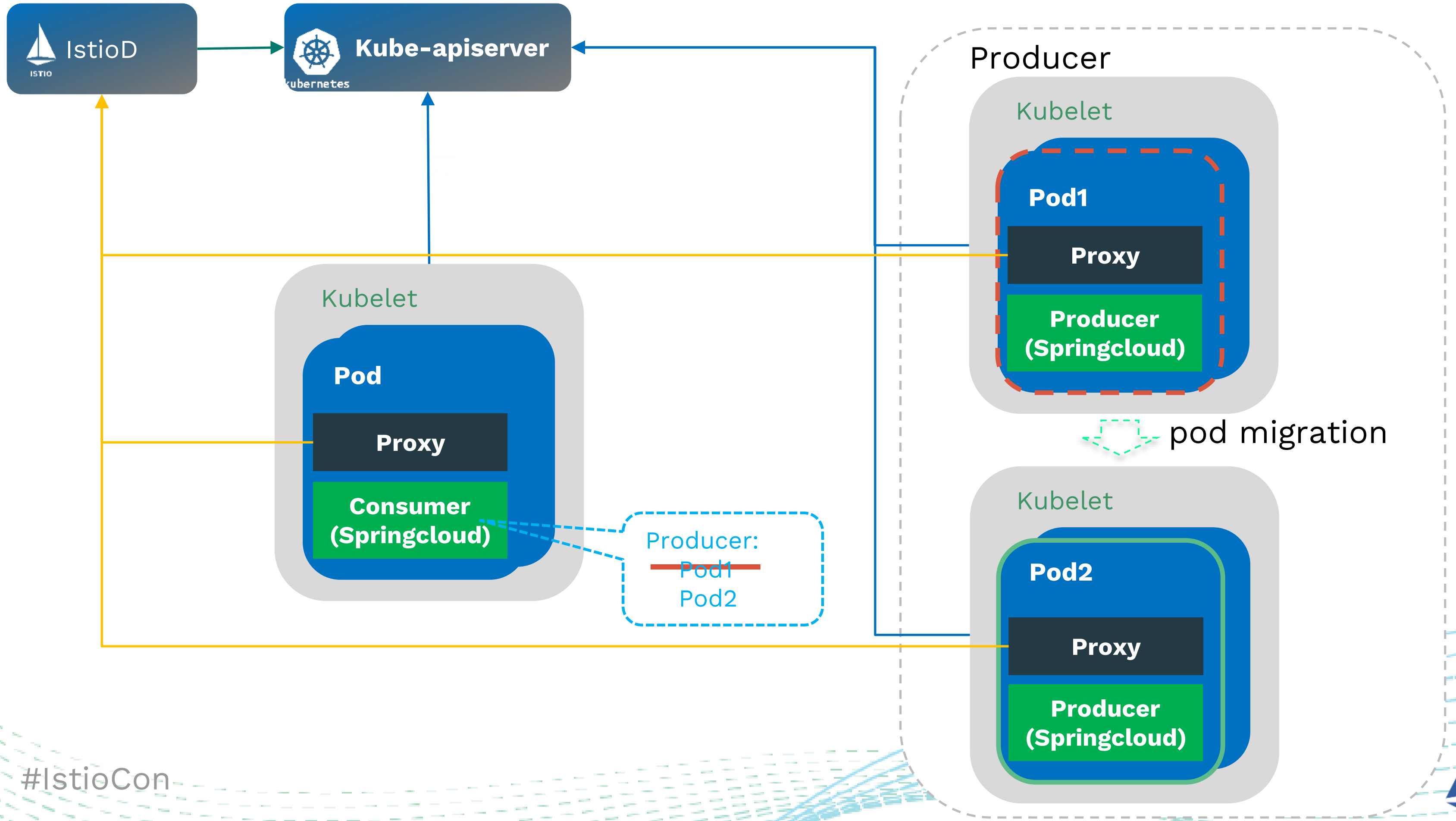
Agenda

- Concepts
- Problems
- Solutions
- Practice

Solution 1: Multi language

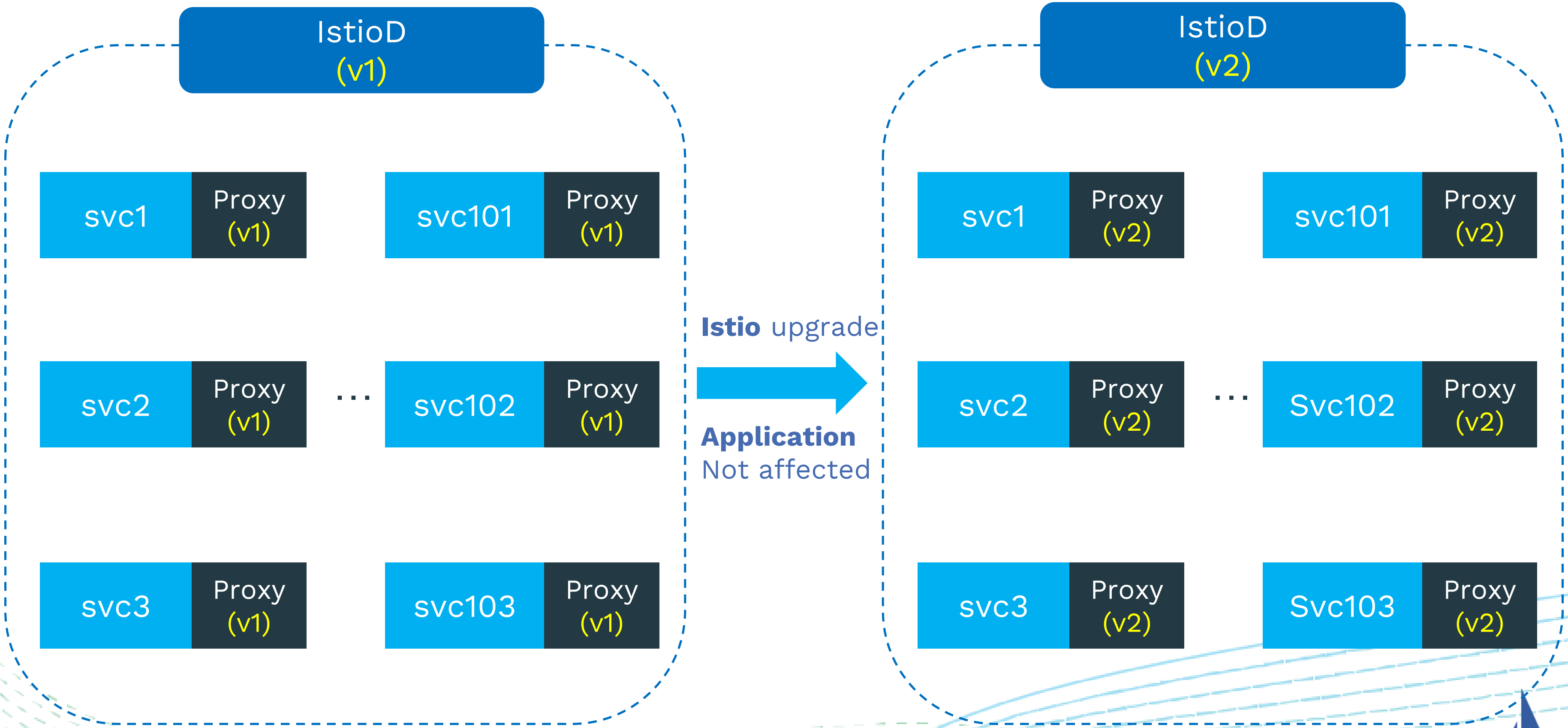


Solution 2: Native Kubernetes service discovery

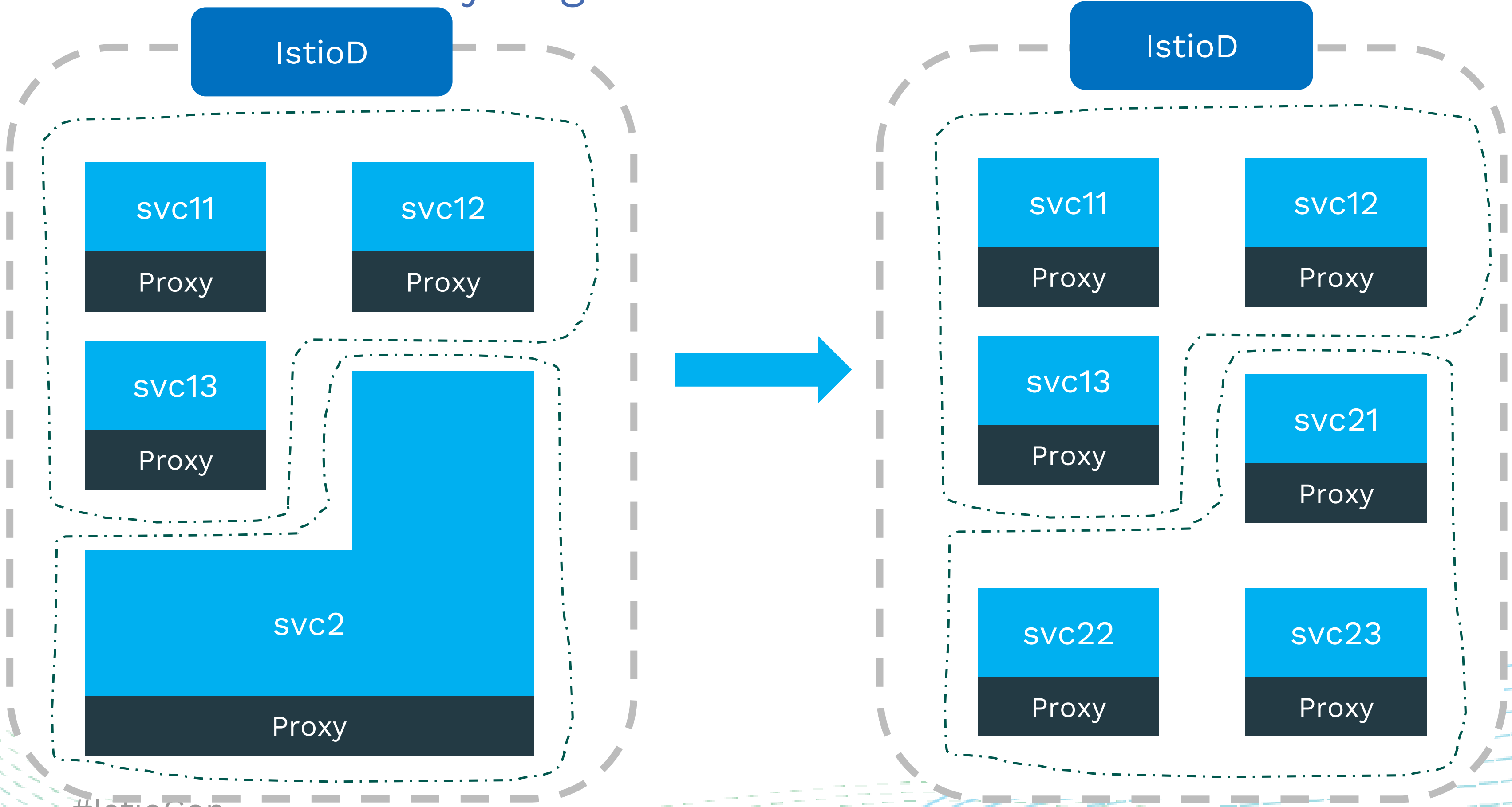


#IstioCon

Solution 3: Application NOT affected in case of Service management upgrading



Solution 4: Gradually migrate from a monolith to micro services



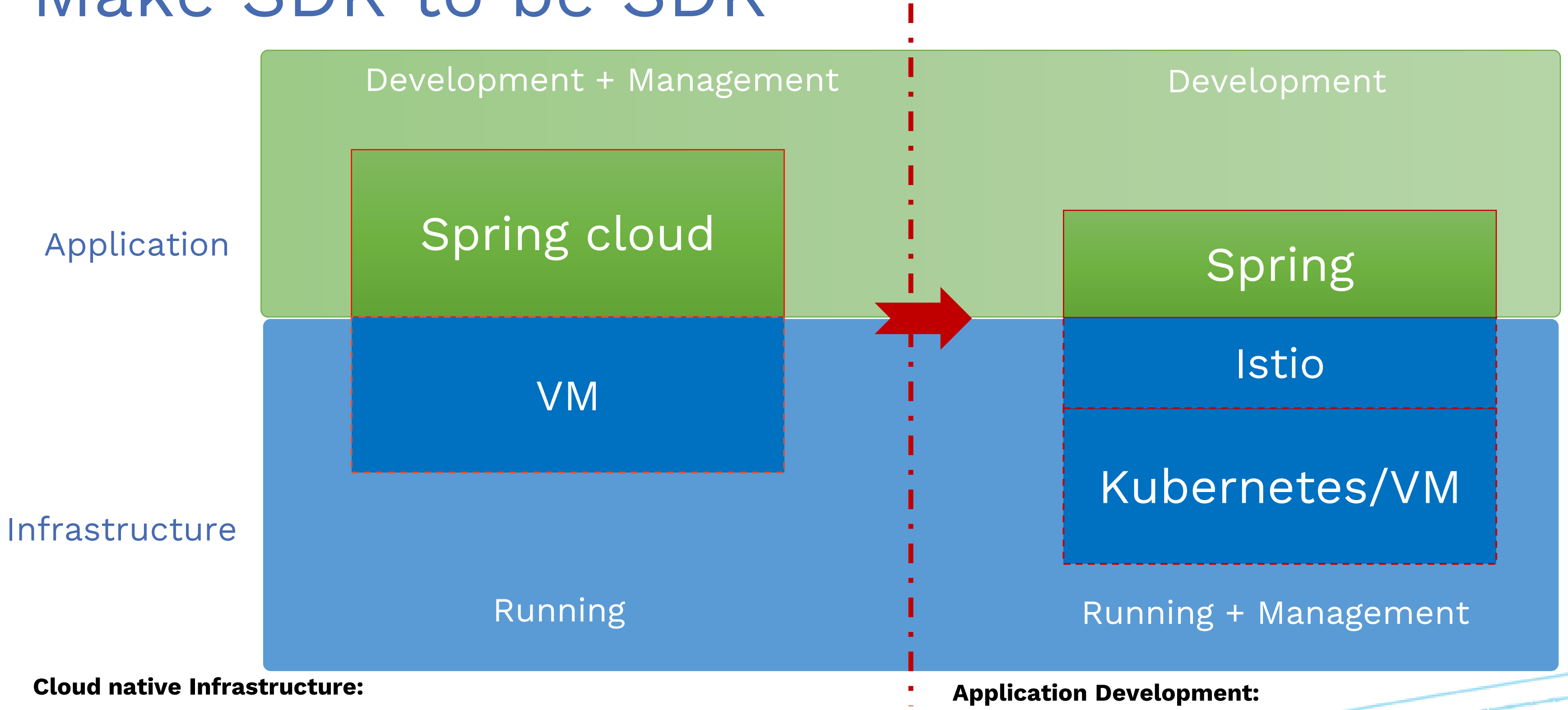
Old Monolith also equally managed by Istio as well as new micro service

Agenda

- Concepts
- Problems
- Solutions
- Practice

#IstioCon

Make SDK to be SDK



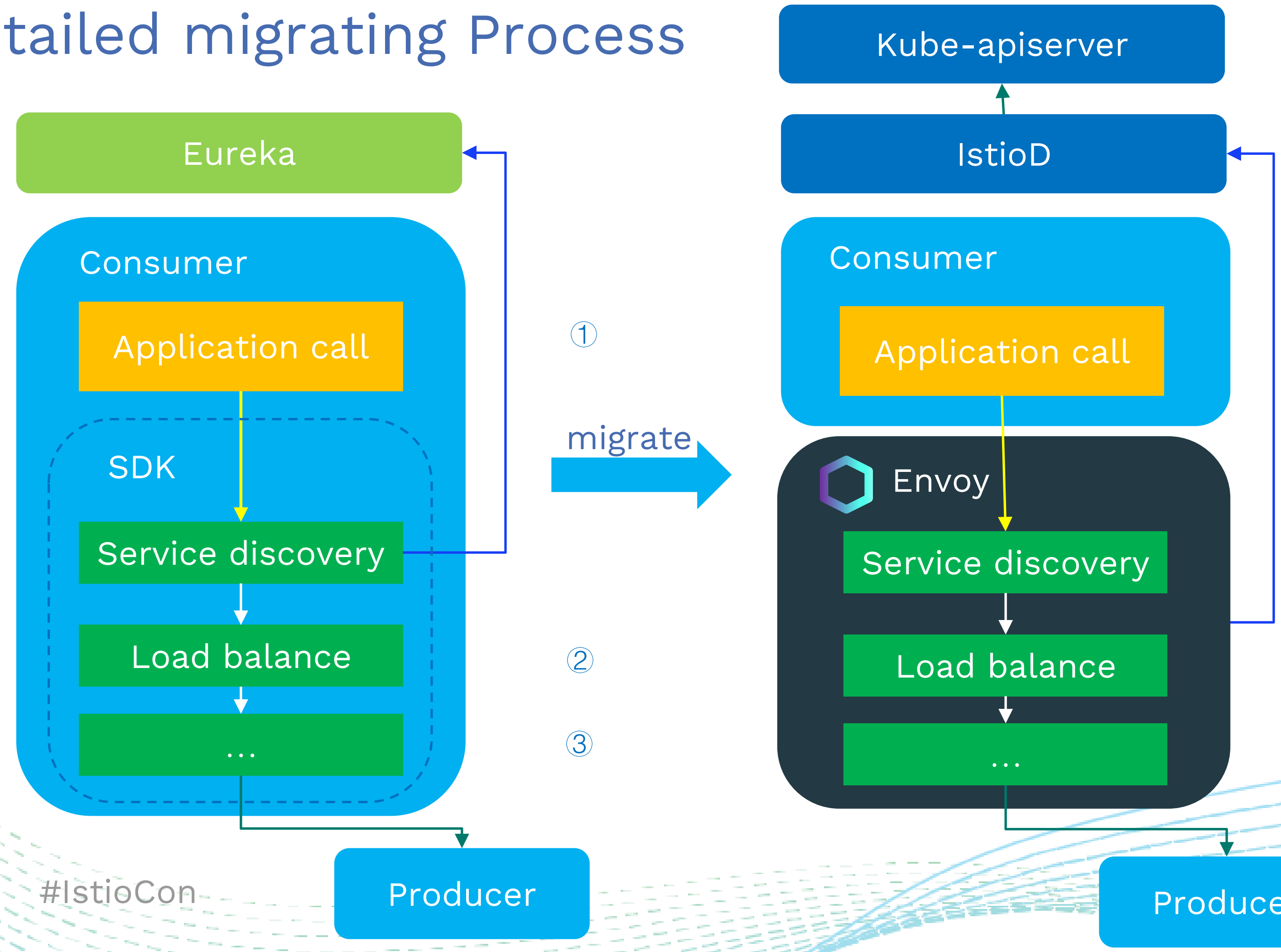
Cloud native Infrastructure:

- **Kubernetes**: Flexible application deployment, management and scaling
- **Istio**: Non-intrusive traffic management, security and observability

Application Development:

Spring Boot: Create production-grade applications that you can "just run".

Detailed migrating Process



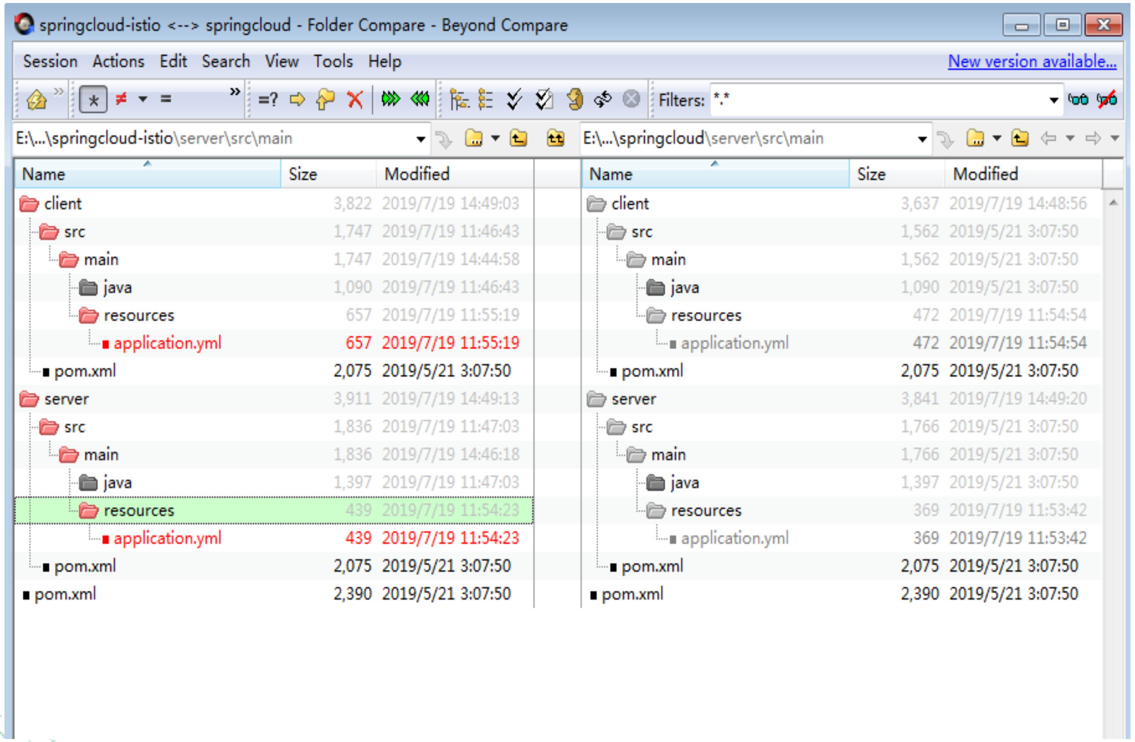
① Discard SDK service registry

② Bypass sdk service discovery and Loadbalancer, call producer directly by Kubernetes service name

③ Replace SDK’s service management logic with mesh gradually.

#IstioCon

Bypass SDK by changing configuration


springcloud-istio <--> springcloud - Folder Compare - Beyond Compare
Session Actions Edit Search View Tools Help
New version available...
Filters: *.*

E:\...\springcloud-istio\server\src\main

E:\...\springcloud\server\src\main

Name	Size	Modified	Name	Size	Modified
client	3,822	2019/7/19 14:49:03	client	3,637	2019/7/19 14:48:56
src	1,747	2019/7/19 11:46:43	src	1,562	2019/5/21 3:07:50
main	1,747	2019/7/19 14:44:58	main	1,562	2019/5/21 3:07:50
java	1,090	2019/7/19 11:46:43	java	1,090	2019/5/21 3:07:50
resources	657	2019/7/19 11:55:19	resources	472	2019/7/19 11:54:54
application.yml	657	2019/7/19 11:55:19	application.yml	472	2019/7/19 11:54:54
pom.xml	2,075	2019/5/21 3:07:50	pom.xml	2,075	2019/5/21 3:07:50
server	3,911	2019/7/19 14:49:13	server	3,841	2019/7/19 14:49:20
src	1,836	2019/7/19 11:47:03	src	1,766	2019/5/21 3:07:50
main	1,836	2019/7/19 14:46:18	main	1,766	2019/5/21 3:07:50
java	1,397	2019/7/19 11:47:03	java	1,397	2019/5/21 3:07:50
resources	439	2019/7/19 11:54:23	resources	369	2019/7/19 11:53:42
application.yml	439	2019/7/19 11:54:23	application.yml	369	2019/7/19 11:53:42
pom.xml	2,075	2019/5/21 3:07:50	pom.xml	2,075	2019/5/21 3:07:50
pom.xml	2,390	2019/5/21 3:07:50	pom.xml	2,390	2019/5/21 3:07:50

application.yaml

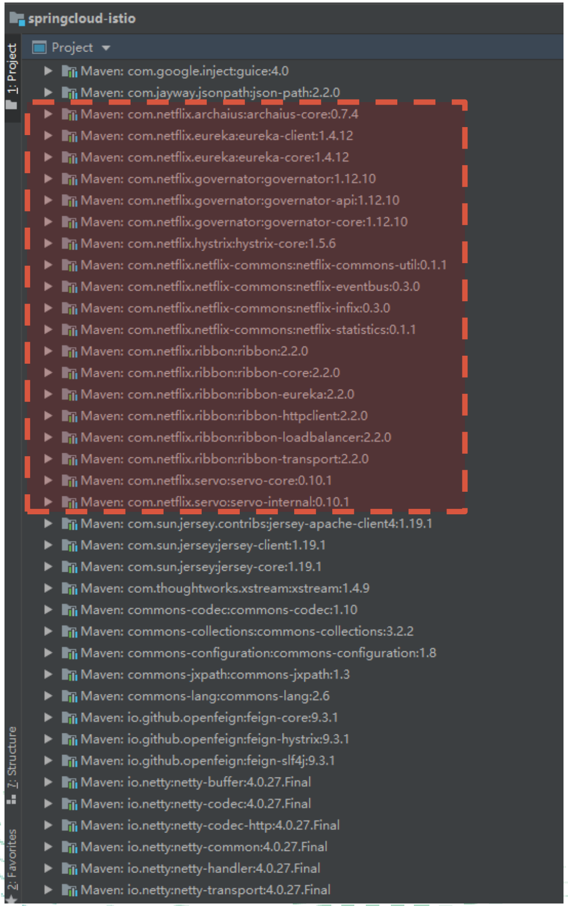
```
# disable eureka discovery
#eureka:
#  client:
#    serviceUrl:
#      defaultZone:
http://10.133.249.158:8761/eureka/
#  instance:
#    leaseRenewalIntervalInSeconds:
10


# ribbon static instance set to
kubernetes service name and port
producer:
  ribbon:
    listOfServers: producer:7111
```

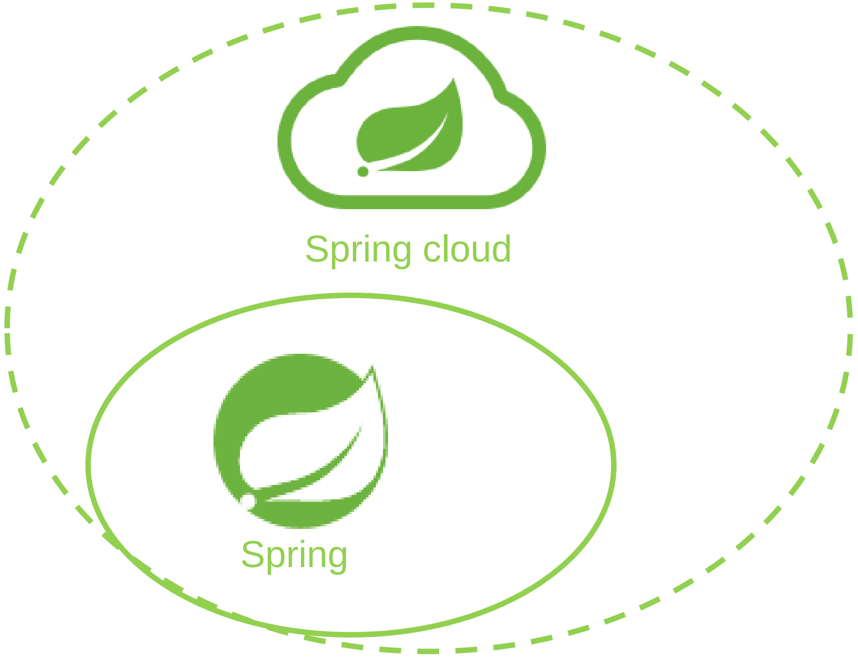
#IstioCon

(Similar modification to annotation based config)

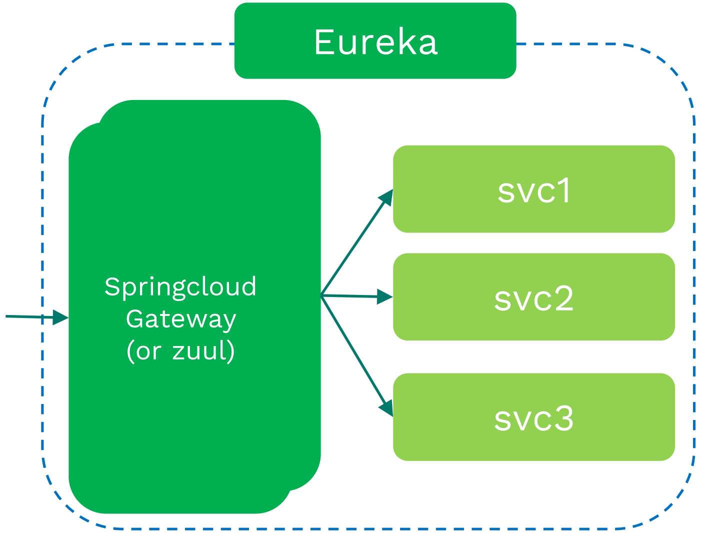
PASSTHROUGH: Offload CLOUD in Spring cloud

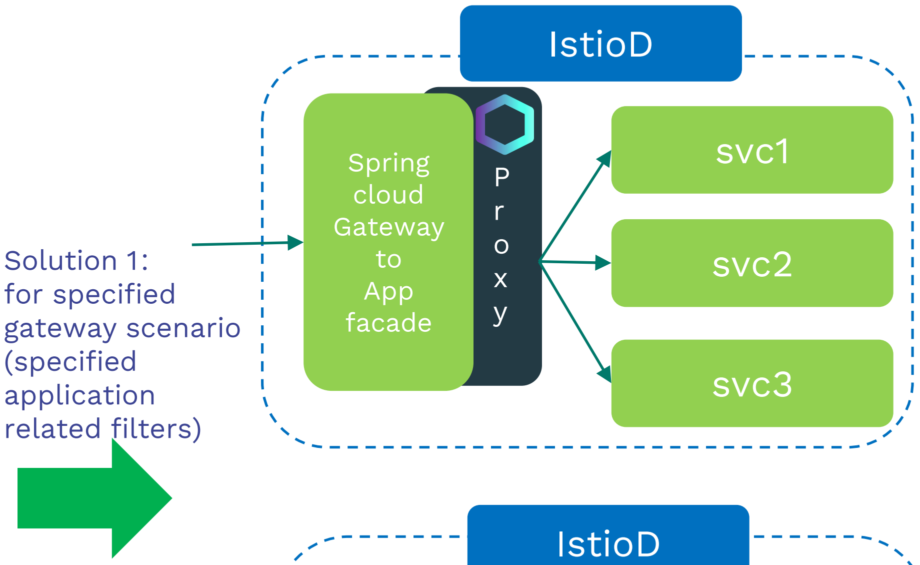


```
Maven: com.google.inject:guice:4.0
Maven: com.jayway.jsonpath:json-path:2.2.0
Maven: com.netflix.archaius:archaius-core:0.7.4
Maven: com.netflix.eureka:eureka-client:1.4.12
Maven: com.netflix.eureka:eureka-core:1.4.12
Maven: com.netflix.governator:governator:1.12.10
Maven: com.netflix.governator:governator-api:1.12.10
Maven: com.netflix.governator:governator-core:1.12.10
Maven: com.netflix.hystrix:hystrix-core:1.5.6
Maven: com.netflix.netflix-commons:netflix-commons-util:0.1.1
Maven: com.netflix.netflix-commons:netflix-eventbus:0.3.0
Maven: com.netflix.netflix-commons:netflix-infix:0.3.0
Maven: com.netflix.netflix-commons:netflix-statistics:0.1.1
Maven: com.netflix.ribbon:ribbon:2.2.0
Maven: com.netflix.ribbon:ribbon-core:2.2.0
Maven: com.netflix.ribbon:ribbon-eureka:2.2.0
Maven: com.netflix.ribbon:ribbon-httpclient:2.2.0
Maven: com.netflix.ribbon:ribbon-loadbalancer:2.2.0
Maven: com.netflix.ribbon:ribbon-transport:2.2.0
Maven: com.netflix.servo:servo-core:0.10.1
Maven: com.netflix.servo:servo-internal:0.10.1
Maven: com.sun.jersey.contribs:jersey-apache-client4:1.19.1
Maven: com.sun.jersey:jersey-client:1.19.1
Maven: com.sun.jersey:jersey-core:1.19.1
Maven: com.thoughtworks.xstream:xstream:1.4.9
Maven: commons-codec:commons-codec:1.10
Maven: commons-collections:commons-collections:3.2.2
Maven: commons-configuration:commons-configuration:1.8
Maven: commons-jxpath:commons-jxpath:1.3
Maven: commons-lang:commons-lang:2.6
Maven: io.github.openfeign:feign-core:9.3.1
Maven: io.github.openfeign:feign-hystrix:9.3.1
Maven: io.github.openfeign:feign-slf4j:9.3.1
Maven: io.netty:netty-buffer:4.0.27.Final
Maven: io.netty:netty-codec:4.0.27.Final
Maven: io.netty:netty-codec-http:4.0.27.Final
Maven: io.netty:netty-common:4.0.27.Final
Maven: io.netty:netty-handler:4.0.27.Final
Maven: io.netty:netty-transport:4.0.27.Final
```

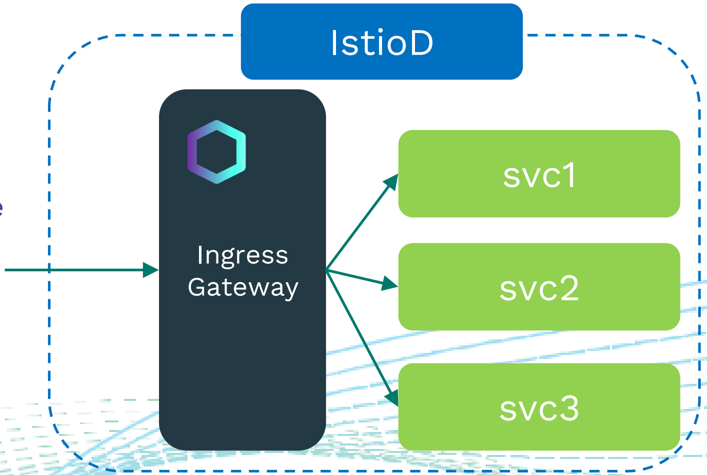


Micro service Gateway migration





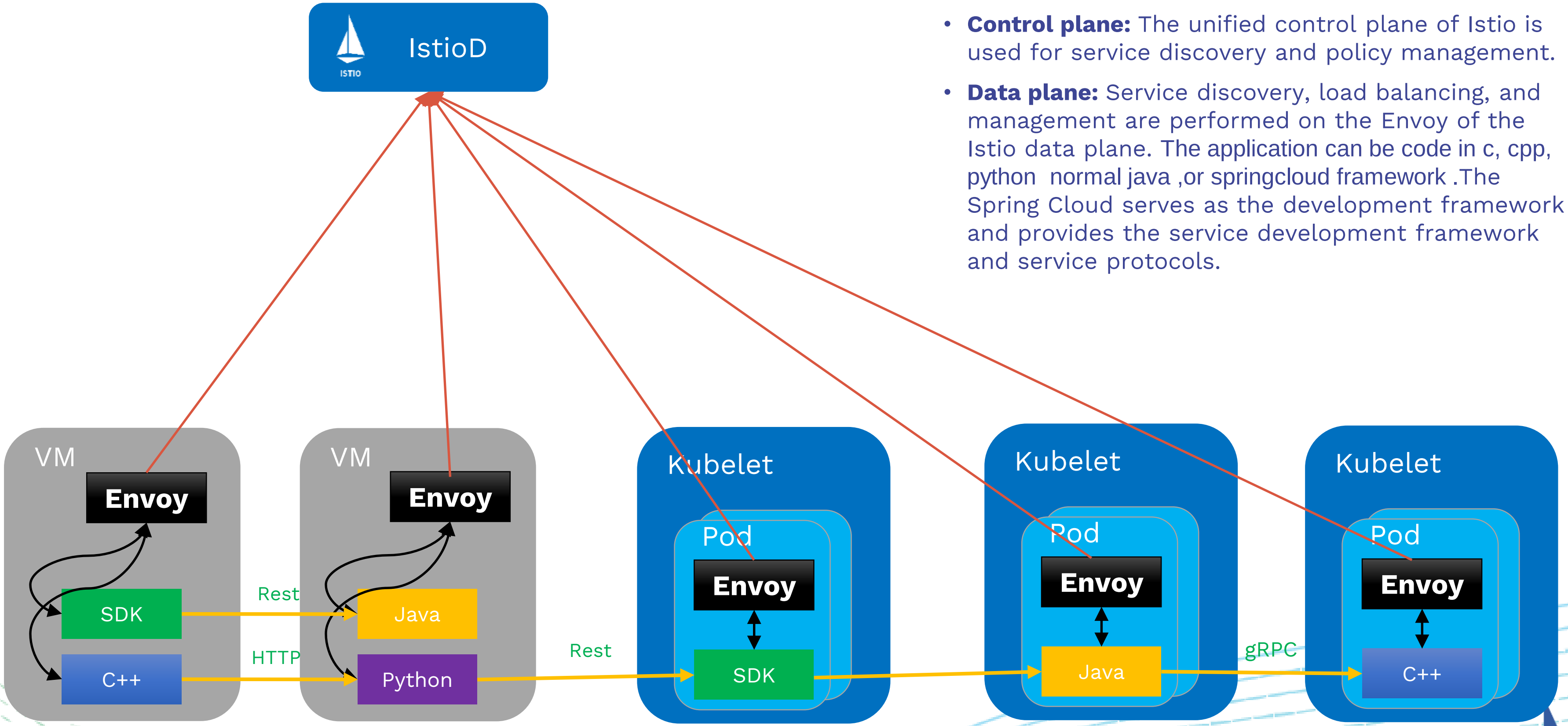
Solution 1:
for specified gateway scenario (specified application related filters)



Solution 2:
for general gateway scenario(service mapping, security)

#IstioCon

Multi language, Multi framework, Multi Env.

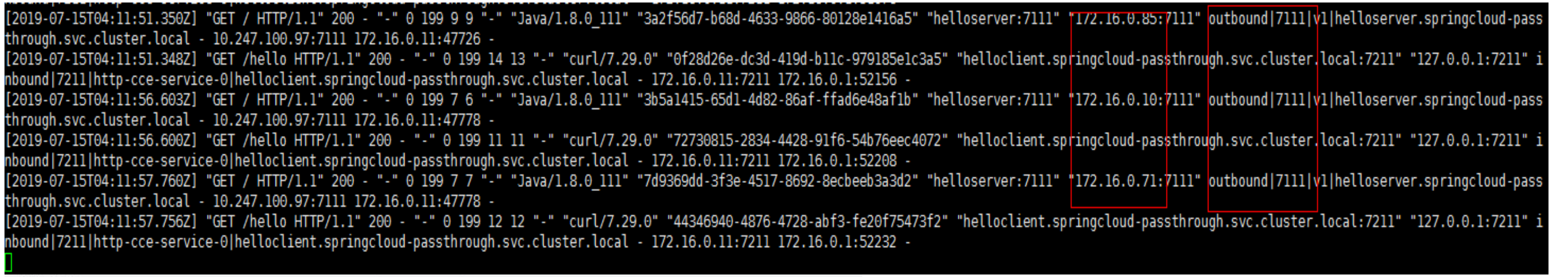


- **Control plane:** The unified control plane of Istio is used for service discovery and policy management.
- **Data plane:** Service discovery, load balancing, and management are performed on the Envoy of the Istio data plane. The application can be code in c, cpp, python normal java ,or springcloud framework .The Spring Cloud serves as the development framework and provides the service development framework and service protocols.

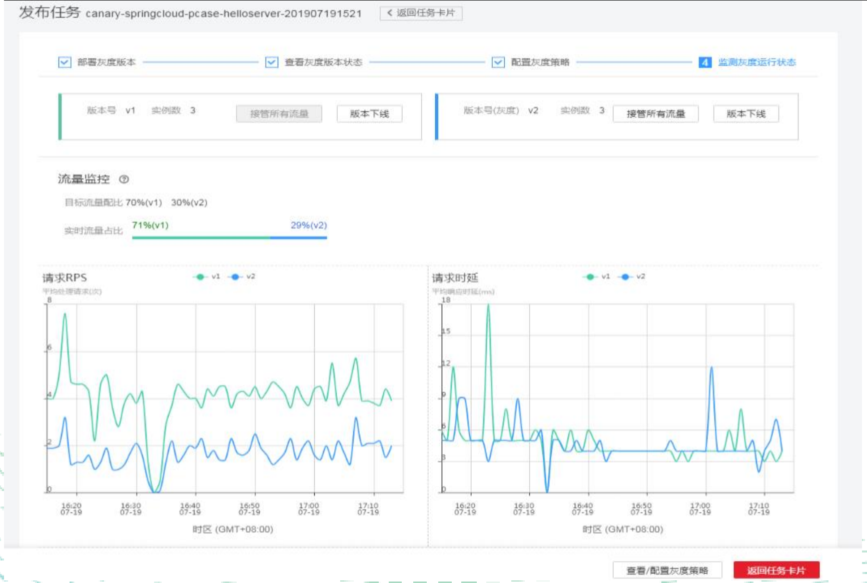
#IstioCon

Example: Istio canary for SpringCloud app

kubectl logs helloclient-6fcc9cb8c9-qz5ng -c istio-proxy -nspringcloud-passthrough -f



```
[2019-07-15T04:11:51.350Z] "GET / HTTP/1.1" 200 - "-" 0 199 9 9 "-" "Java/1.8.0_111" "3a2f56d7-b68d-4633-9866-80128e1416a5" "helloserver:7111" "172.16.0.85:7111" outbound|7111|v1|helloserver.springcloud-pass
through.svc.cluster.local - 10.247.100.97:7111 172.16.0.11:47726 -
[2019-07-15T04:11:51.348Z] "GET /hello HTTP/1.1" 200 - "-" 0 199 14 13 "-" "curl/7.29.0" "0f28d26e-dc3d-419d-b11c-979185e1c3a5" "helloclient.springcloud-passthrough.svc.cluster.local:7211" "127.0.0.1:7211" i
nbound|7211|http-cce-service-0|helloclient.springcloud-passthrough.svc.cluster.local - 172.16.0.11:7211 172.16.0.1:52156 -
[2019-07-15T04:11:56.603Z] "GET / HTTP/1.1" 200 - "-" 0 199 7 6 "-" "Java/1.8.0_111" "3b5a1415-65d1-4d82-86af-ffad6e48af1b" "helloserver:7111" "172.16.0.10:7111" outbound|7111|v1|helloserver.springcloud-pass
through.svc.cluster.local - 10.247.100.97:7111 172.16.0.11:47778 -
[2019-07-15T04:11:56.600Z] "GET /hello HTTP/1.1" 200 - "-" 0 199 11 11 "-" "curl/7.29.0" "72730815-2834-4428-91f6-54b76eec4072" "helloclient.springcloud-passthrough.svc.cluster.local:7211" "127.0.0.1:7211" i
nbound|7211|http-cce-service-0|helloclient.springcloud-passthrough.svc.cluster.local - 172.16.0.11:7211 172.16.0.1:52208 -
[2019-07-15T04:11:57.760Z] "GET / HTTP/1.1" 200 - "-" 0 199 7 7 "-" "Java/1.8.0_111" "7d9369dd-3f3e-4517-8692-8ecbeeb3a3d2" "helloserver:7111" "172.16.0.71:7111" outbound|7111|v1|helloserver.springcloud-pass
through.svc.cluster.local - 10.247.100.97:7111 172.16.0.11:47778 -
[2019-07-15T04:11:57.756Z] "GET /hello HTTP/1.1" 200 - "-" 0 199 12 12 "-" "curl/7.29.0" "44346940-4876-4728-abf3-fe20f75473f2" "helloclient.springcloud-passthrough.svc.cluster.local:7211" "127.0.0.1:7211" i
nbound|7211|http-cce-service-0|helloclient.springcloud-passthrough.svc.cluster.local - 172.16.0.11:7211 172.16.0.1:52232 -
```



发布任务 canary-springcloud-pcase-helloserver-201907191521

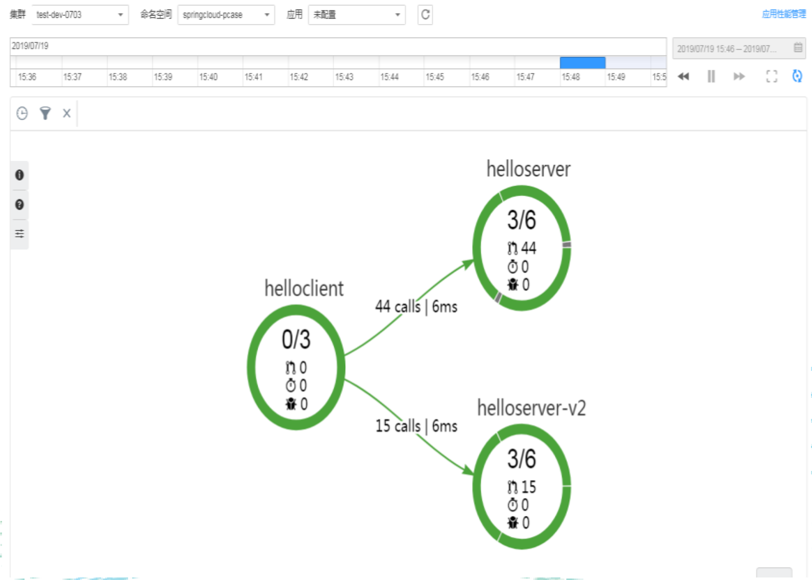
版本号 v1　实例数 3

版本号(灰度) v2　实例数 3

流量监控

目标流量配比 70%(v1)　30%(v2)

实时流量占比 71%(v1)　29%(v2)

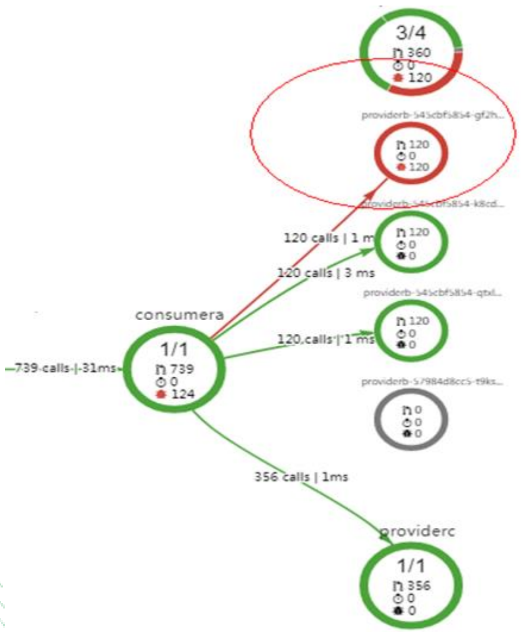


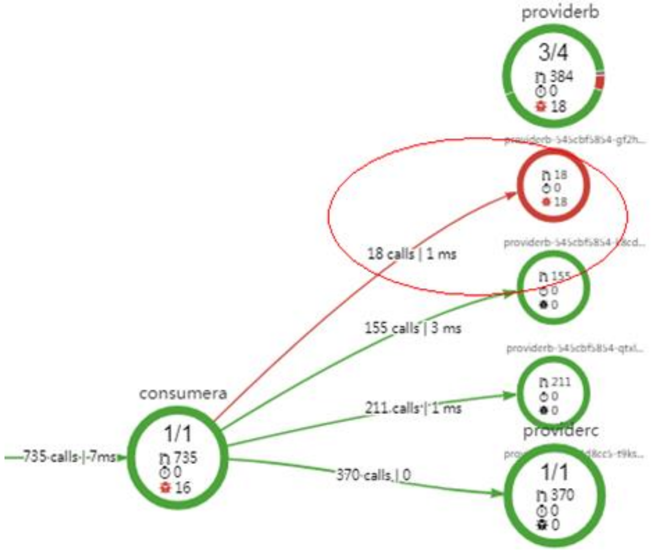
集群 test-dev-0703　命名空间 springcloud-pcase　应用 未配置

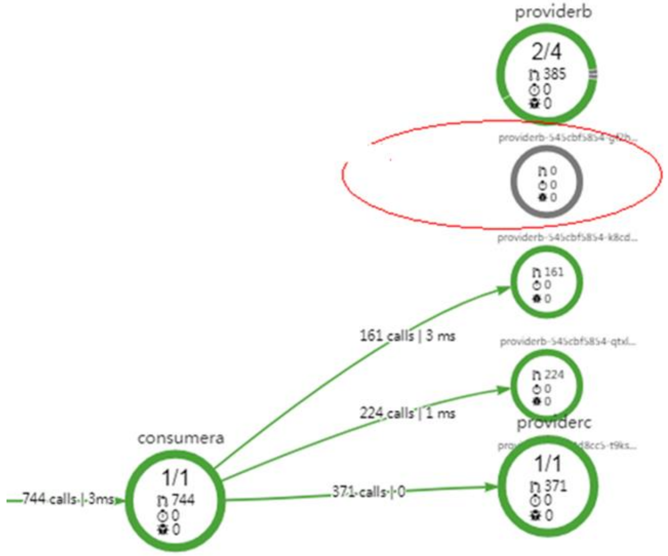
2019/07/19　2019/07/19 15:46 – 2019/07...

Example: Istio circuit breaker help isolate unhealthy Springcloud instance

The traffic on the unhealthy instance is gradually reduced until it is completely isolated. As a whole, only healthy instances of the service receive traffic.







① Unhealthy instance triggers circuit breaker

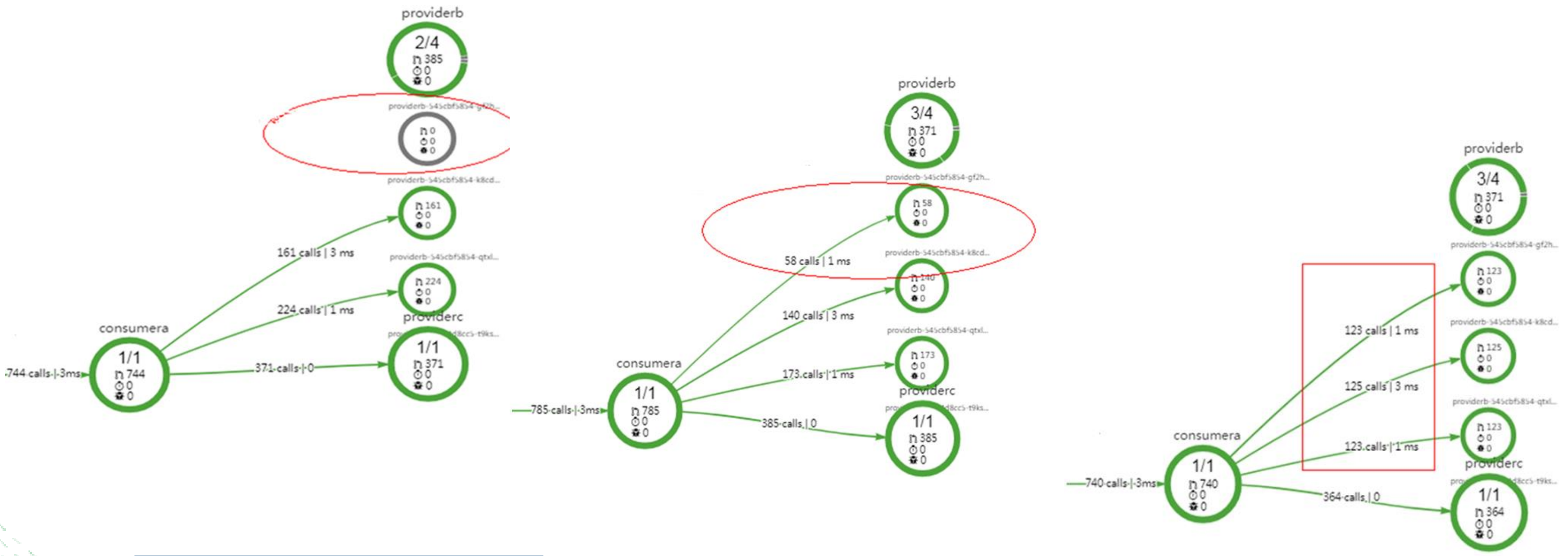
② Traffic of the unhealthy instance decreases

③ The unhealthy instance is isolated

#Is

Example: Istio circuit breaker help isolate unhealthy Springcloud instance

When the unhealthy instance is normal back, under circuit breaker policy, traffic will be **automatically** distributed to it .



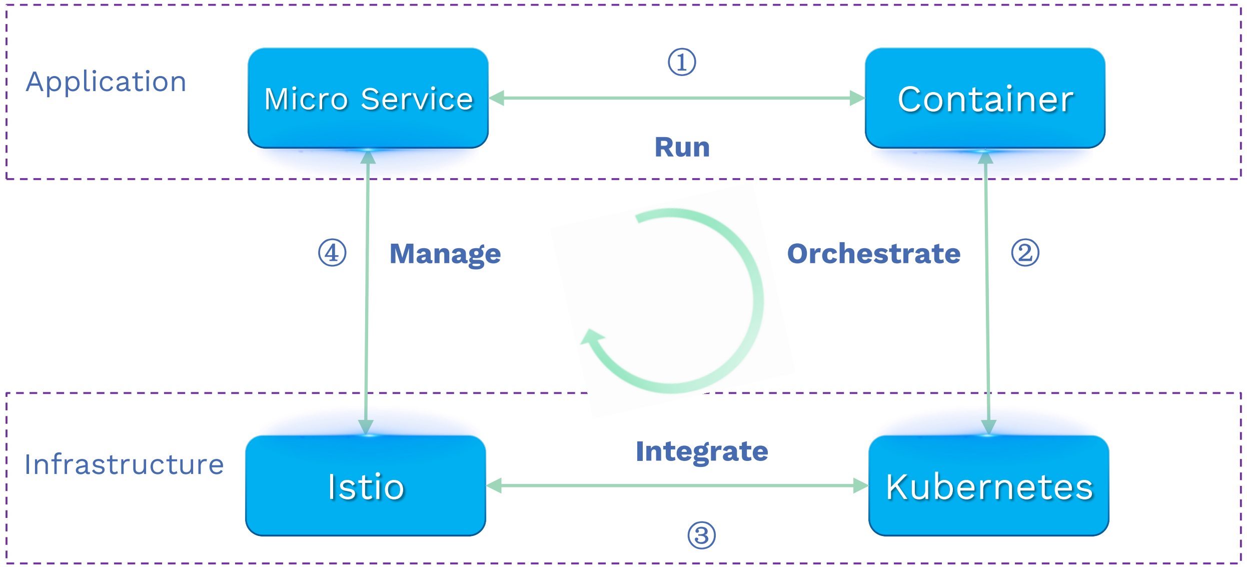
① There is no traffic on the recovered failed instance

② A small amount of traffic starts to be received on the recovered failed instance

③ The recovered failed instance receives the same traffic as other instances

#Is

Summary: Micro service, Container, Kubernetes， Istio



① Containers and microservices share the same lightweight and agile features.
② The use of Kubernetes for container orchestration is already the current standard.
③ Istio and Kubernetes are closely combined to provide an end-to-end microservice running & management platform.
④ Istio becomes the trend of microservice management.

Thank you!

@idouba