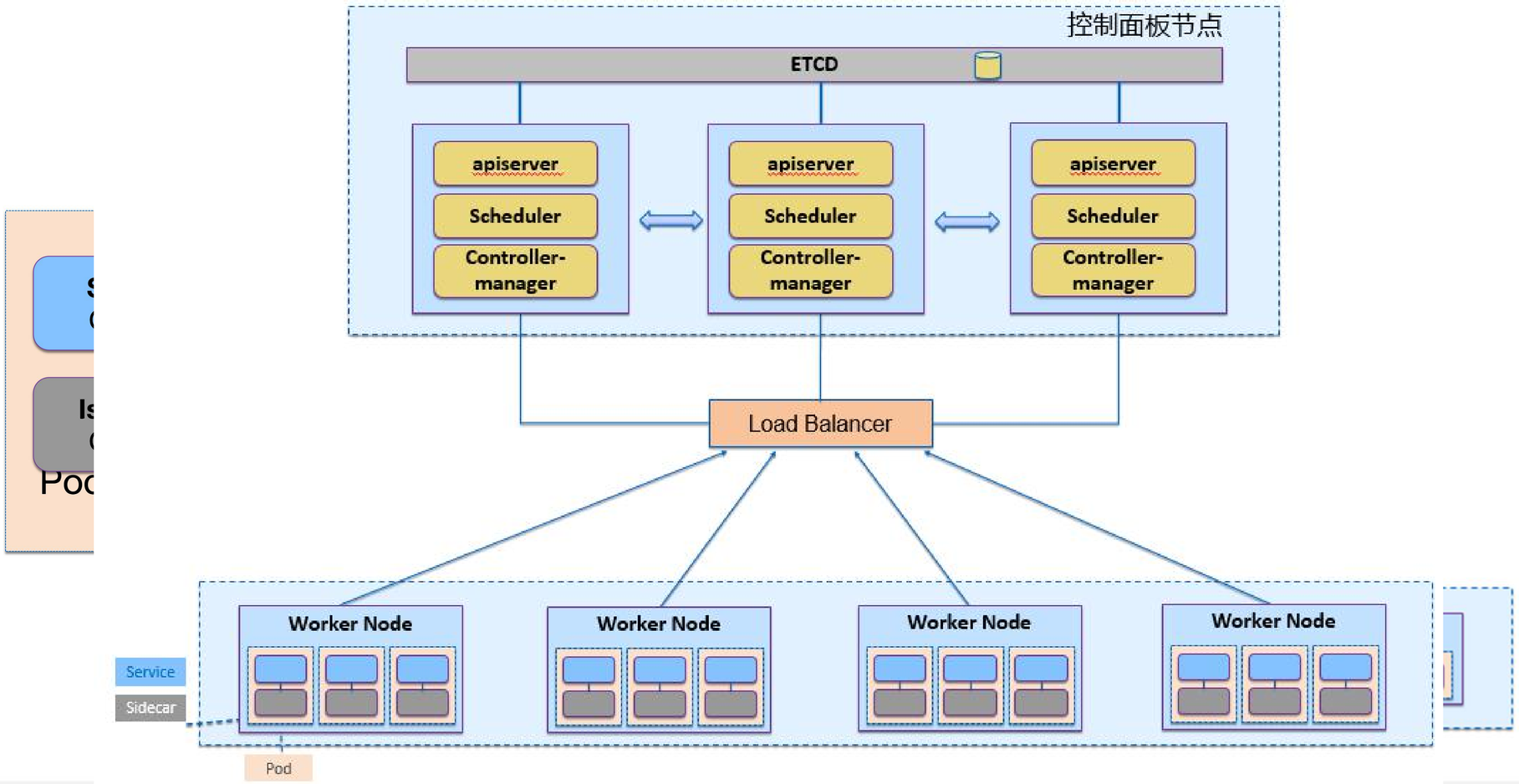
国投瑞银微服务平台升级Service Mesh实践

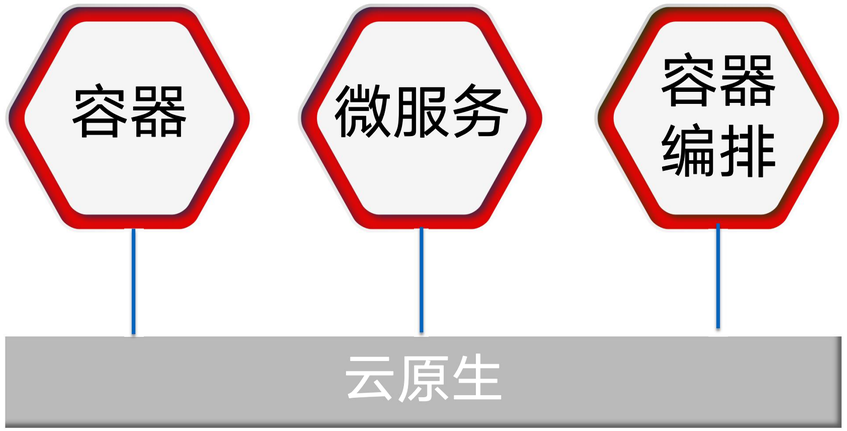
国投瑞银基金信息技术部 马学宁

目录

CONTENTS

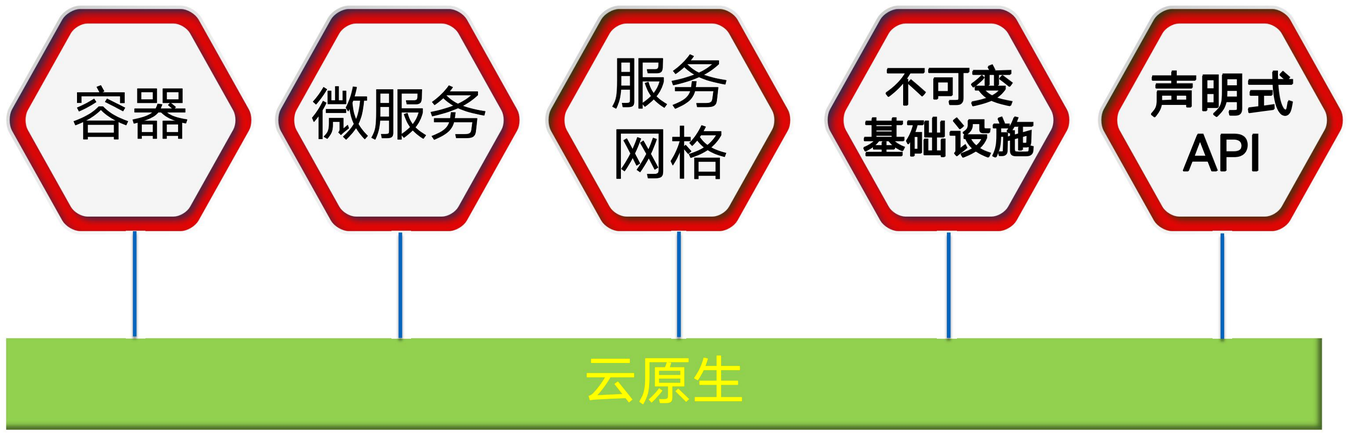
升级现有微服务架构


控制面板节点
ETCD
apiserver
Scheduler
Controller-manager
apiserver
Scheduler
Controller-manager
apiserver
Scheduler
Controller-manager
Load Balancer
Worker Node
Worker Node
Worker Node
Worker Node
Service
Sidecar
Pod



CNCF 2015云原生定义

- 容器：服务能从底层架构中分离出来，实现完全的可移植性
- Docker Swarm：只能对Docker进行编排
- 命令式部署

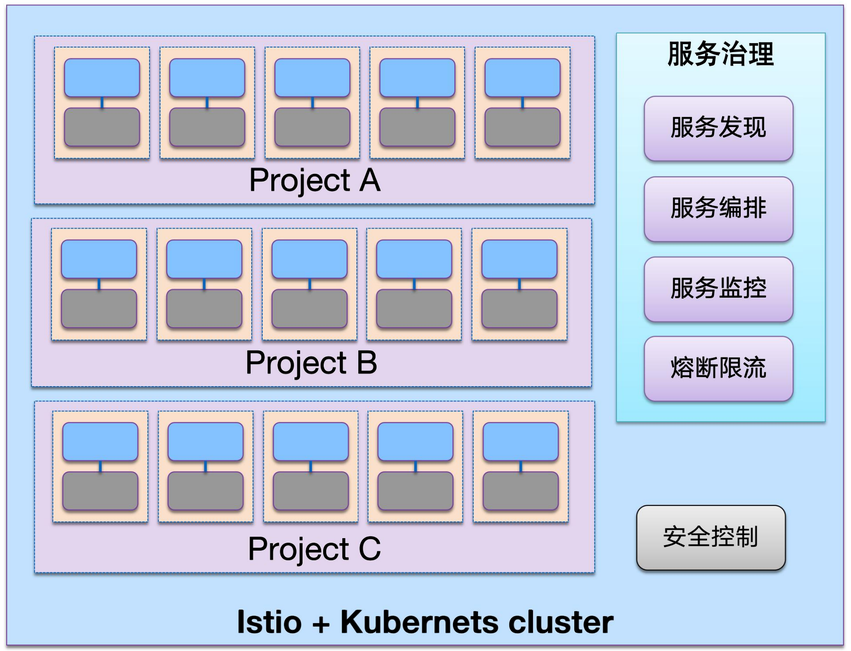


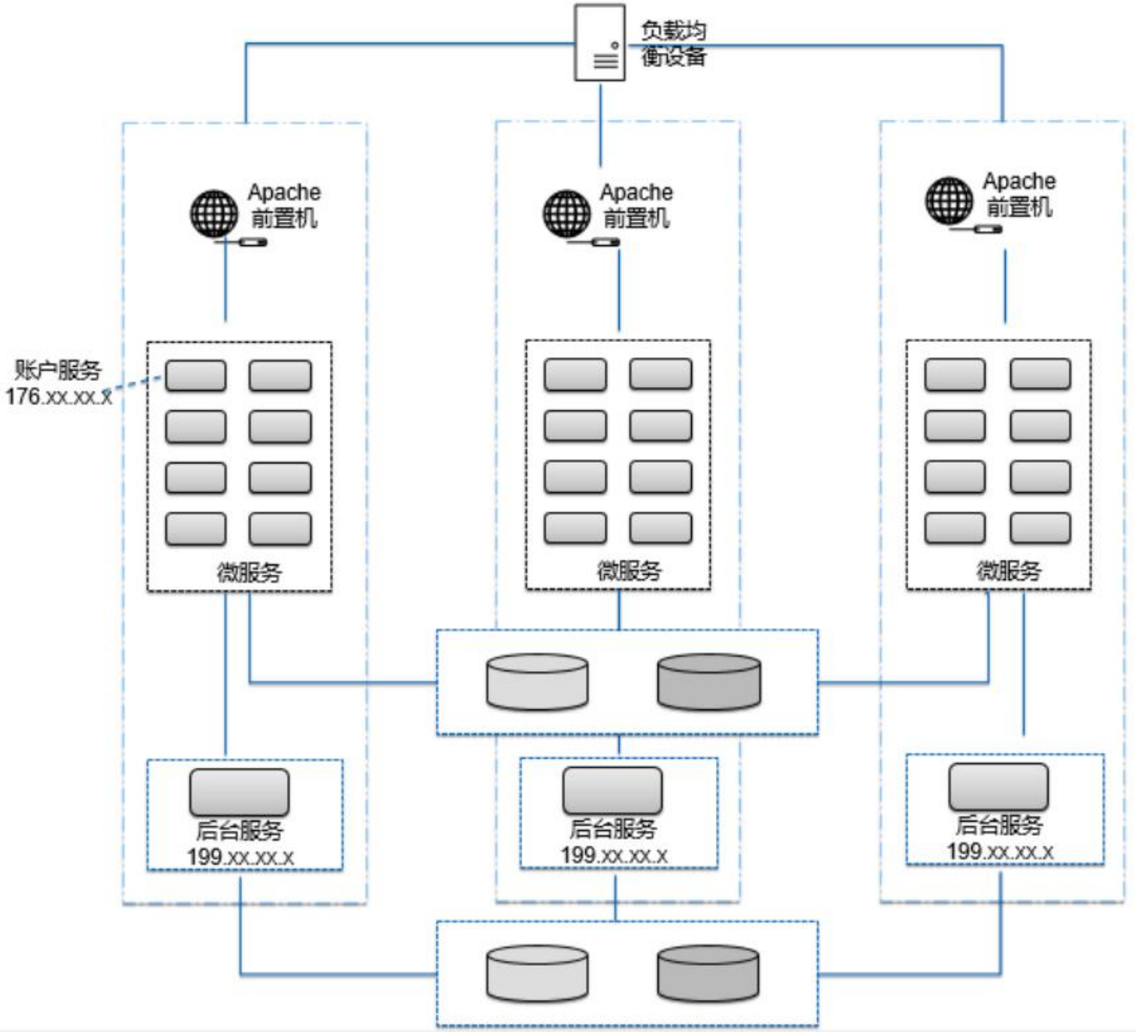
CNCF最新云原生定义

- 服务网格（Service Mesh）：下一代微服务框架，它实现了服务间通信的基础设施层，通过该层可以服务进行最精细的管控，解决微服务架构的痛点需求
- 声明式API：将部署的定义和具体操作拆分开来，声明定义了期望的状态
- 不可变基础设施：自包含、自描述可以完全在不同环境中迁移的实例，容器镜像

- 敏捷开发，提升研发效率
- 不中断业务持续更新，持续交付、加速新技术落地应用
- 自动化运维，动态资源管理，能够让集群资源得到最高效的利用，降低运维成本
- 高可靠，弹性伸缩，易扩展，故障隔离保护，故障自动转移
- 异构语言/框架的统一治理

第三方业务系统接入部署的统一规划与管理





目录

CONTENTS

集群构建方式选择

托管	只解决了集群部署问题，可以任意添加工作节点，保障托管集群扩展性，可用性。 缺乏灵活性和自由，技术锁定
PaaS平台产品	内置Kubernets，提供应用开发工具和运行环境，能够快速的构建、测试和部署应用，集成了gitlab，jenkins，监控，容器管理
第三方服务商	如果Kubernets对你来说学习比较艰难，服务商提供的功能所带来的限制可以接受，同时能及时响应你的需求
自建	有大量的功能可以配置，这种配置使它非常灵活和强大，可以完全控制应用平台

目录

CONTENTS

平台升级核心模块对比

Jenkins

docker SWARM

Prometheus Grafana OpenSearch beats

træfik proxy

Overlay

docker nexus repository

Jenkins argo git Ops

kubernetes HELM

Prometheus Grafana OpenSearch beats

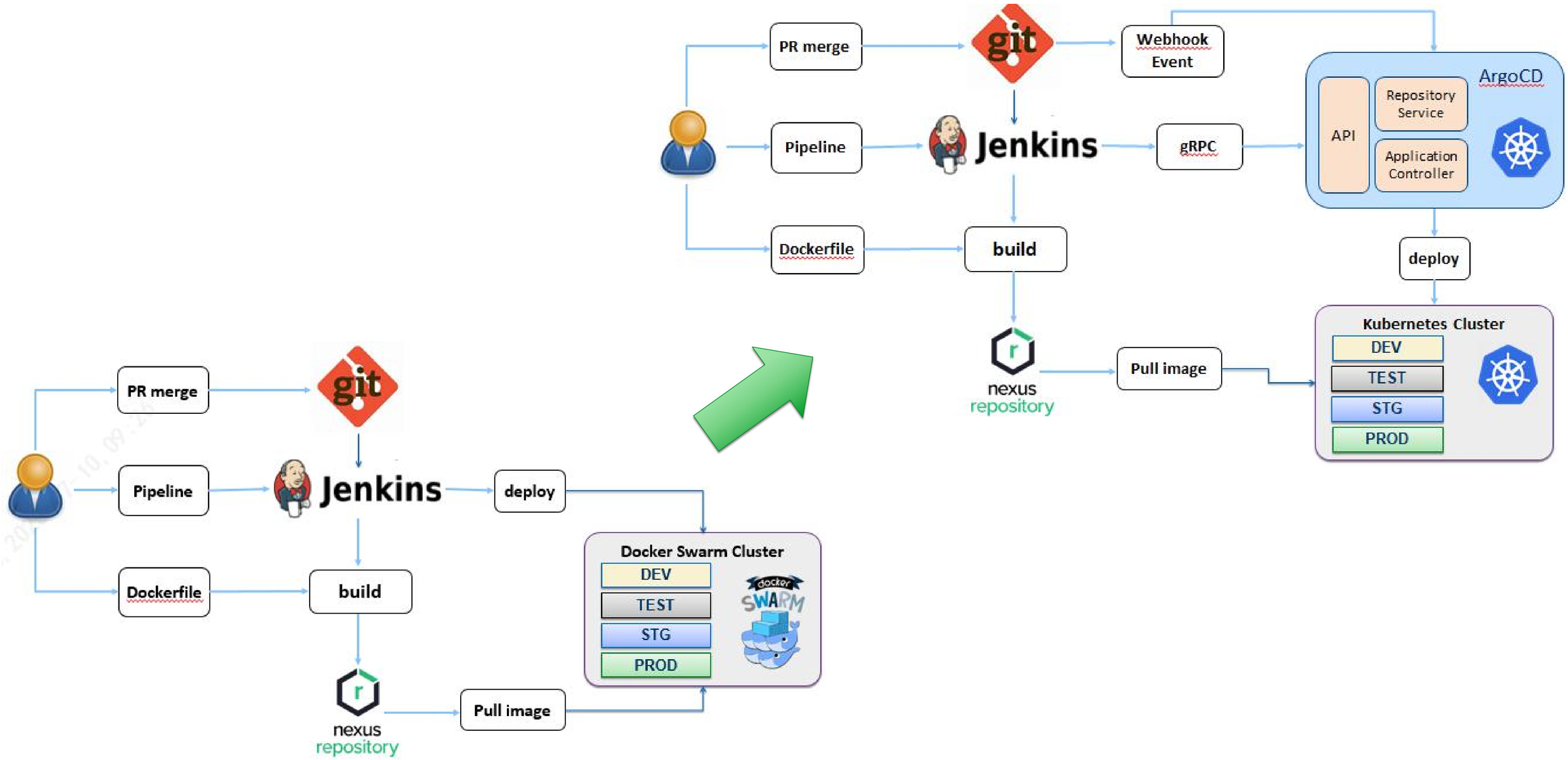
envoy Istio

CNI PROJECT CALICO

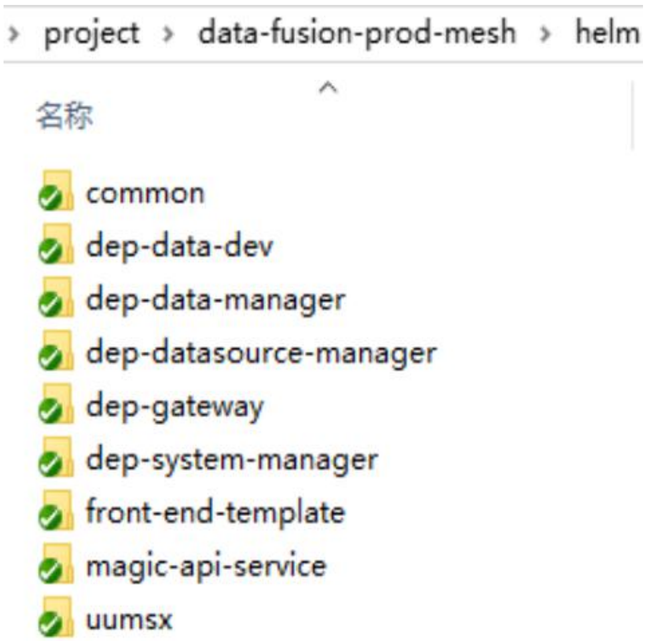
gRPC

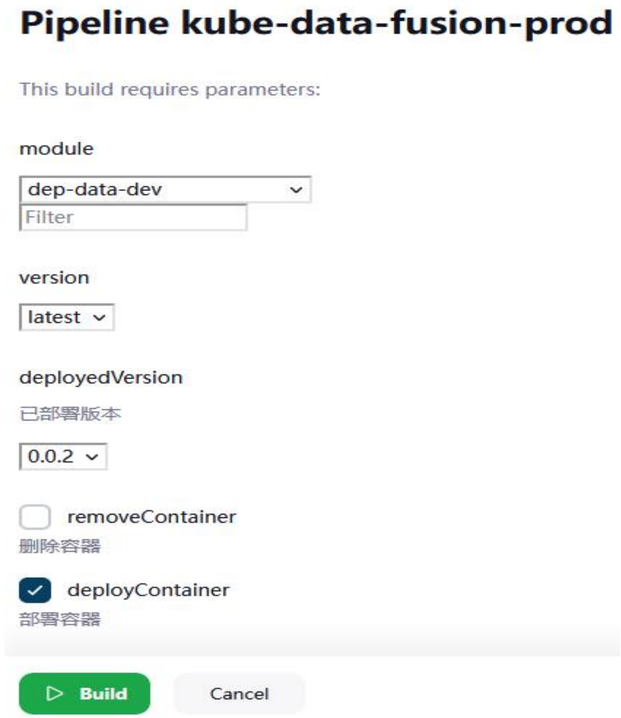
container nexus repository

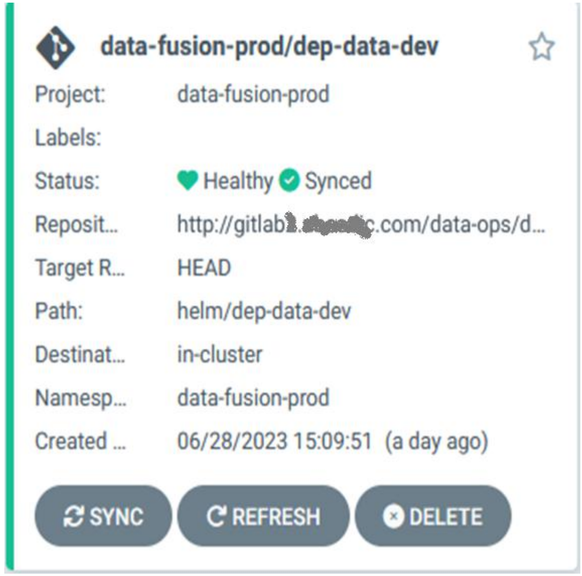
CI/CD


PR merge
git
Webhook Event
ArgoCD
API
Repository Service
Application Controller
Pipeline
Jenkins
gRPC
Dockerfile
build
deploy
nexus repository
Pull image
Kubernetes Cluster
DEV
TEST
STG
PROD
PR merge
git
Pipeline
Jenkins
deploy
Dockerfile
build
Docker Swarm Cluster
DEV
TEST
STG
PROD
nexus repository
Pull image

CI/CD


project
data-fusion-prod-mesh
helm
名称
common
dep-data-dev
dep-data-manager
dep-datasource-manager
dep-gateway
dep-system-manager
front-end-template
magic-api-service
uumsx


Pipeline kube-data-fusion-prod
This build requires parameters:
module
dep-data-dev
Filter
version
latest
deployedVersion
已部署版本
0.0.2
removeContainer
删除容器
deployContainer
部署容器
Build
Cancel


data-fusion-prod/dep-data-dev
Project: data-fusion-prod
Labels:
Status: Healthy Synced
Reposit... http://gitlab
c.com/data-ops/d...
Target R... HEAD
Path: helm/dep-data-dev
Destinat... in-cluster
Namesp... data-fusion-prod
Created ... 06/28/2023 15:09:51 (a day ago)
SYNC
REFRESH
DELETE

代理，服务发现和网格

01 路由和流量

- 4层和7层路由，支持HTTP/2, TCP, gRPC, Websockets
- 蓝绿与金丝雀部署
- 流量镜像
- 熔断，重试，限流

02 服务发现

- Docker Swarm，Kubernetes，KV

03 安全

- HTTPS，认证

04 可观察性

- 内建Dashboard
- 实时指标
- 分布式追踪

01 路由和流量

- 请求路由，流量转移，入口/出口流量控制
- 调测，故障注入，流量镜像
- 网络弹性，请求超时，重试，熔断

02 服务发现

- Kubernetes

03 安全

- 认证与授权，mTLS

04 可观察性

- 指标
- 日志
- 分布式追踪

平台升级功能对比

服务编排	Docker Swarm	Kubernetes
容器运行载体	Docker	CRI （**Containerd**， CRI-O）
配置与秘密	Config, secrets	Configmap, secrets
容器网络	Bridge,Overlay	CNI （**Calico**， Cilium）
存储集成	NFS,CIFS/Volume	CSI （**CSI-SMB**）
服务发现/流量控制/安全	Traefik	istio
持续集成/部署	Jenkins pipeline	Jenkins pipeline， **ArgoCD**、FluxCD
监控/告警/	Prometheus,AlertManager,Grafana	Prometheus,AlertManager,Grafana

平台升级小结

- Containerd沿袭使用
- 为每个服务建Helm脚本
- 周边服务如Prometheus，Grafana迁入Kubernetes
- 业务无侵入，所以代码层面基本没有改动

目录

CONTENTS

服务容器化

镜像打包

- 基础镜像（Java版本）
- docker-maven-plugin

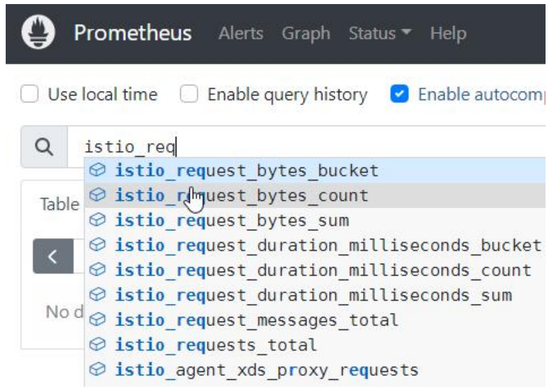

```
<baseImage>nexus.mycompany.com/openjdk-${java.version}</baseImage>
<registryUrl>https://nexus.mycompany.com</registryUrl>
<serverId>ufos-docker-hub</serverId>
</resources>
<exposes>
    <expose>9299</expose>
</exposes>
<entryPoint>["java", "${java.run.option}", "-jar", "${project.build.finalName}.jar"]
```

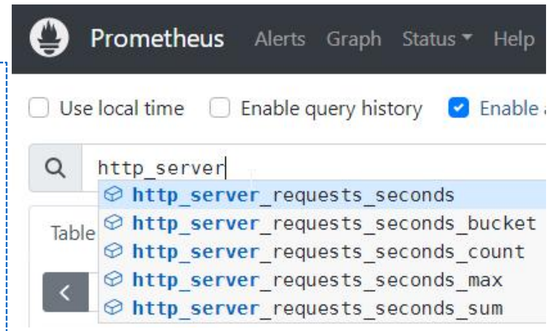
指标（Metrics）

Actuator & micrometer

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

```
management:
  endpoints:
    web:
      base-path: /metrics
      exposure:
        include: [ 'health', 'prometheus']
  endpoint:
    health:
      probes:
        enabled: true
  metrics:
    enable:
      all: false
      http: false
      process: true
```

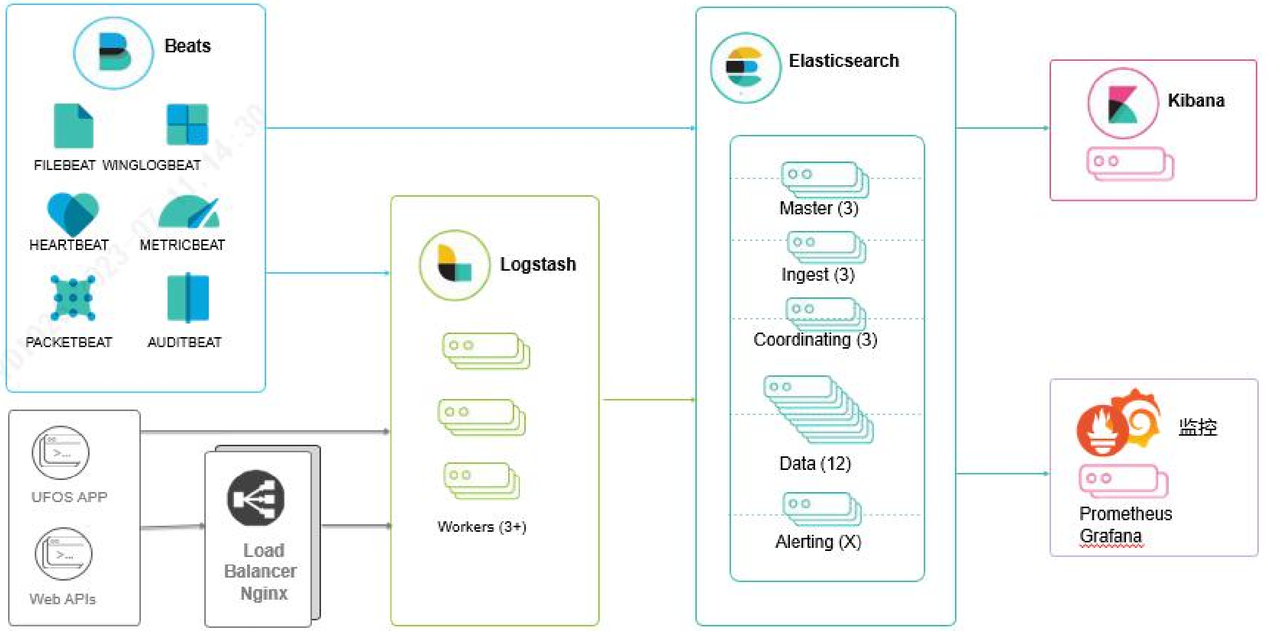





```
# HELP http_server_requests_seconds
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/
http_server_requests_seconds_sum{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/uf
http_server_requests_seconds_count{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/
http_server_requests_seconds_sum{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/uf
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/m
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/met
http_server_requests_seconds_count{exception="None",method="GET",outcome="REDIRECTION",status="302",uri
http_server_requests_seconds_sum{exception="None",method="GET",outcome="REDIRECTION",status="302",uri="
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/uf
http_server_requests_seconds_count{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/
http_server_requests_seconds_sum{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/uf
http_server_requests_seconds_count{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/
http_server_requests_seconds_sum{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/uf
http_server_requests_seconds_count{exception="None",method="GET",outcome="REDIRECTION",status="302",uri
http_server_requests_seconds_sum{exception="None",method="GET",outcome="REDIRECTION",status="302",uri="
http_server_requests_seconds_count{exception="None",method="GET",outcome="CLIENT_ERROR",status="404",ur
http_server_requests_seconds_sum{exception="None",method="GET",outcome="CLIENT_ERROR",status="404",uri=
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{exception="None",method="POST",outcome="SUCCESS",status="200",uri="/uf
```

日志（Logging）

```
<?xml version="1.0" encoding="UTF-8"?>
<included>
    <appender name="LOGSTASH-PROD" class="net.logstash.logback.appender.LogstashTcpSocketAppender">
        <destination>elk.company.com:5678</destination>
        <encoder class="net.logstash.logback.encoder.LogstashEncoder">
            <shortenedLoggerNameLength>36</shortenedLoggerNameLength>
            <customFields>{"service_name":"${hostname}"}</customFields>
            <fieldNames>
                <version>[ignore]</version>
            </fieldNames>
        </encoder>
    </appender>
</included>
```



目录

CONTENTS

第三方业务系统接入

- 规范：服务命名，服务端口，服务容器化
- 服务指标Micrometer
- 日志中心配置对接
- 服务发现对接
- 多租户和安全控制

想一想，我该如何把这些技术应用在工作实践中？

THANKS

InfoQ 极客传媒

ArchSummit

全球架构师峰会

数字化转型下的架构升级