2022

单一网络多集群实践

汇报人：杨小飞 时间：2022.10

高效协作 勇于担当

积极创新 用户至上

目录CONTENTS

中通云发展背景

S

截止2019年前

独立的devops平台，绝大部分流量跑在虚拟机。

W

2019年开始

开始容器化进程，自一开始便使用了bgp打通了各集群间的网络，支持虚机容器混部。

O

截至2022年中旬

公司现有业务绝大部分跑在容器上，大的生产集群6个，总共pods数量18000多个，node节点700多。持续推进集群治理以保障环境的稳定和资源充分利用。

T

未来

南北流量治理；全域容器化；提升可观测能力，有状态应用的支持。

2019年前中通云现状

2018 H1

预研K8s

2019 H1

k8s
1.14.X
1.15.X

wayne上线

k8s资源管理微服务

2019年 H2

ZKE上线

开发、测试上容器

2020 H1

生产上容器

2020 ——

不断推进容器化
多集群、cilium

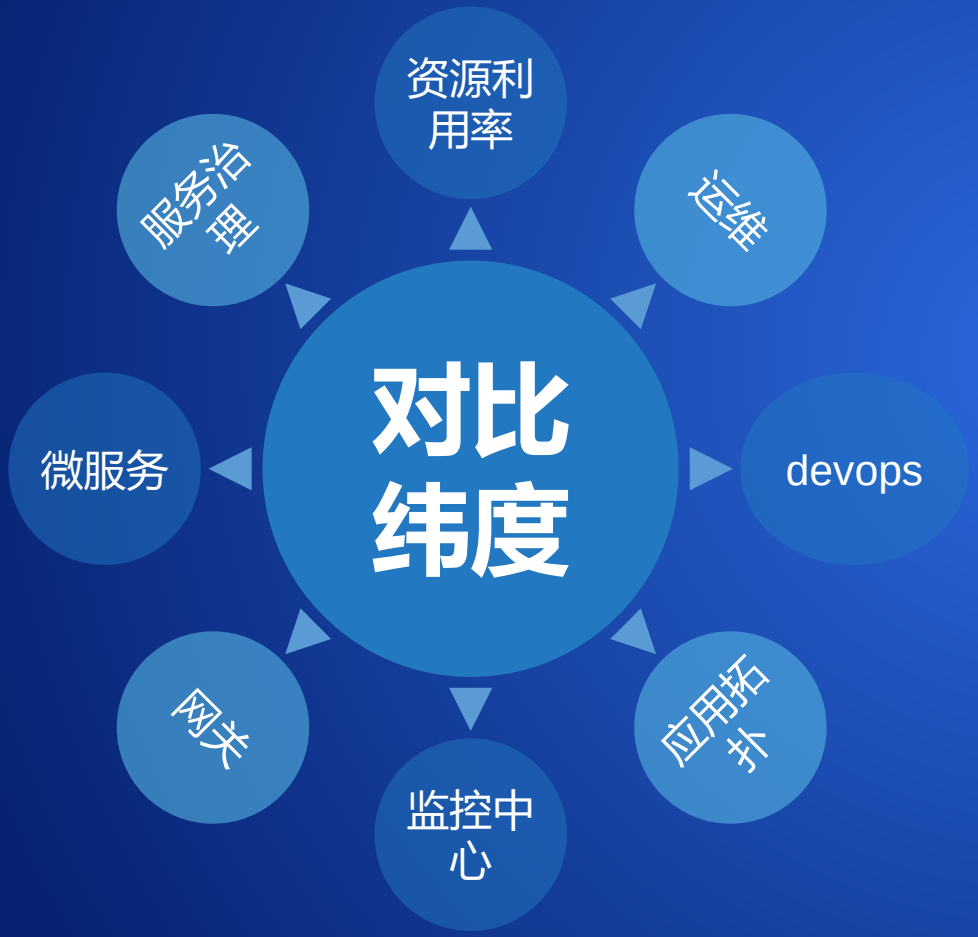
未来

统一网关、离线混部等

容器化前中通云背景


01
主要是跑在虚拟机，部分有状态服务运行在物理机。
02
应用生命周期管理迟缓
03
南北流量全部依赖于静态配置
04
伸缩缓慢
05
大量僵尸应用缺乏治理。
06
资源利用率低。

容器化前后对比



1. **资源利用率从15%提升到百分之40%**
2. **应用全生命周期管理更快便捷，快速。**
3. **弹性伸缩能力**

容器化遇到的挑战

安全要求
所有的外网访问都需要走申请

对接devops平台
有独立的devops平台，需要从传统的虚机发布添加对于容器和混部的支持。

资源吃紧
由于容器化不断推进，而资源回收滞后，集群资源禁止持续了很长时间。

容器内故障现场难以保留
运行在容器的应用故障，重启或者回滚就丢失了现场。

单集群部署存在风险
单集群的故障影响面太大。

固定ip的需求
少量应用对于固定ip存在需求，比如“雪花算法”的应用。

原生调度器不足
基于request和limit的调度无法完全体现各个节点的资源情况。

节点均衡
集群各个节点调度不均衡，为了平衡需要根据实际负载二次均衡。

从单集群到多集群

单集群支持内容

基于真实负载的调度器拓展

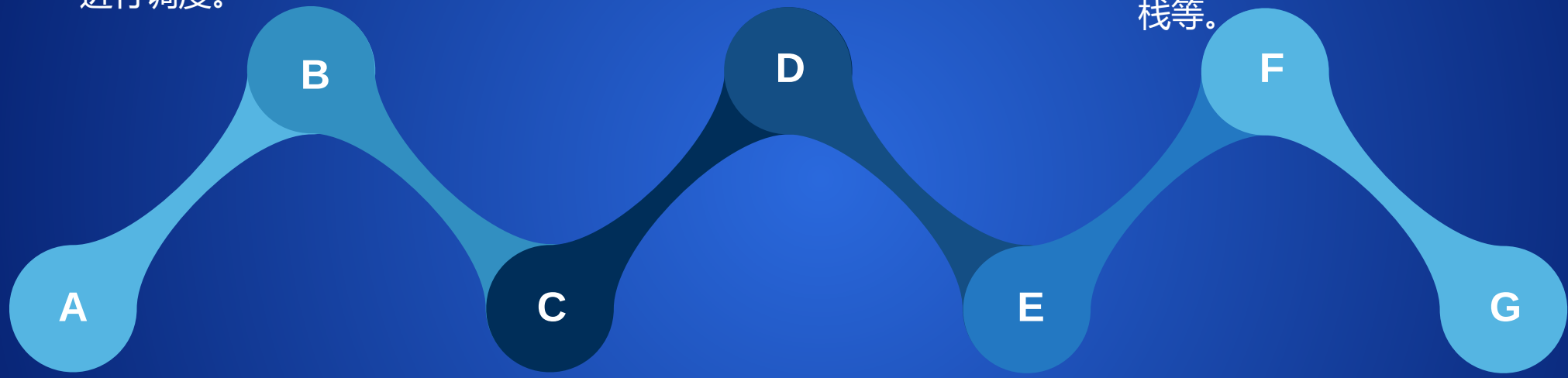
自研了基于原生的拓展调度器，基于采集到节点注解的负载信息进行调度。

固定ip支持

采用ipam固定ip方法建立专用固定ip集群。

容器现场保留

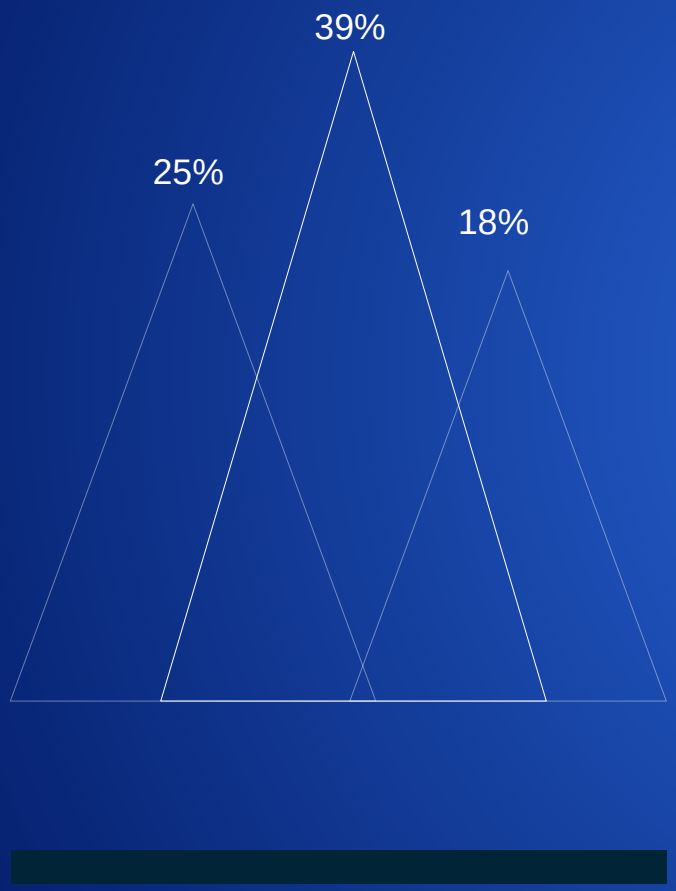
有别于官方的临时容器方案，通过宿主机cri能力切断故障现场的容器网络，然后踢出工作负载管理，再用钩子dump堆栈等。



节点二次均衡

即使使用了自研的拓展调度器，个别情况下还是会出现负载不均匀的情况。因此自研了二次均衡的job，来确保集群各个节点负载均匀。

自研的拓展调度器



改造内容

由采集器将节点真实负载写入到节点注解上，然后调度器从中获取信息之后调节节点的score。

踩过的坑

由于node-exporter偶尔出现采集不到信息的情况，导致上报的负载为0，使得该节点被大量调度pod上去，最终打垮了该节点。

热点调度

工作负载都会设置反亲和性。

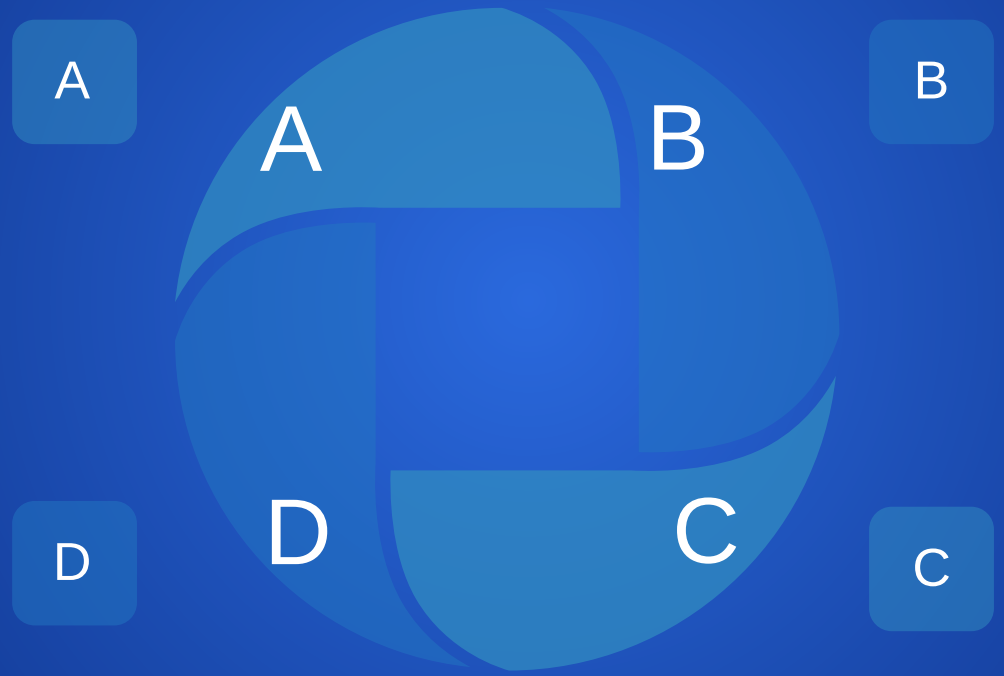
容器现场保留

需求背景

生产环境上个别应用出现java进程假死情况或者出现应用层的保持。出于快速止血的需要，一般都会进行重启或者回退版本，这使得故障现场第一时间丢失。手动dump内存等周期长，慢，且不完整。

选型考虑

官方由提供临时容器的方式，可用复刻一个现场。但是由于我们存在低K8s版本，无法全部满足需要，另外现场可能还会继续工作，如果既要他不工作又要现场保留，就会存在困难。



最终方案

由于我们内部都是无状态工作负载，所以最终采用了自带的特性：修改已由label，然后k8s会自动拉起一个新pod来替代它；同时调用容器内的dump堆栈指令保存即时的现场，再调用cri指令，切断容器的网络。

二次节点均衡

背景

单集群内长期存在各个节点负载不均匀的情况，即使上了自研的基于真实负载的拓展调度器也未解决问题。
高负载的节点会使得大量的运行上的应用高延迟，影响很大。等待触发k8s的驱逐机制非常危险且不确定。
长期的解决方式是由运维手动删除一些负载较高的节点上的pod。

解决方案

在内部，我们调研了descheduler，并且基于它改造了部分内容，根据真实的负载水平驱逐pod。但是内部我们并未采用它，我们只需要其中更简洁、确定的功能。最终我们自研了二次均衡的job。

哪些pod适合被驱逐?

1.单实例多副本中的其中一个。
2.非网关等核心应用，支持namespace、app、node节点白名单机制。
3.cpu在2C~4C之间，低CPU驱逐没有意义，高CPU驱逐对于应用影响比较大。

多集群基本能力

02 Devops支持多集群发布和多集群管理

03 单集群管理控制台适配多集群管理工作负载的能力

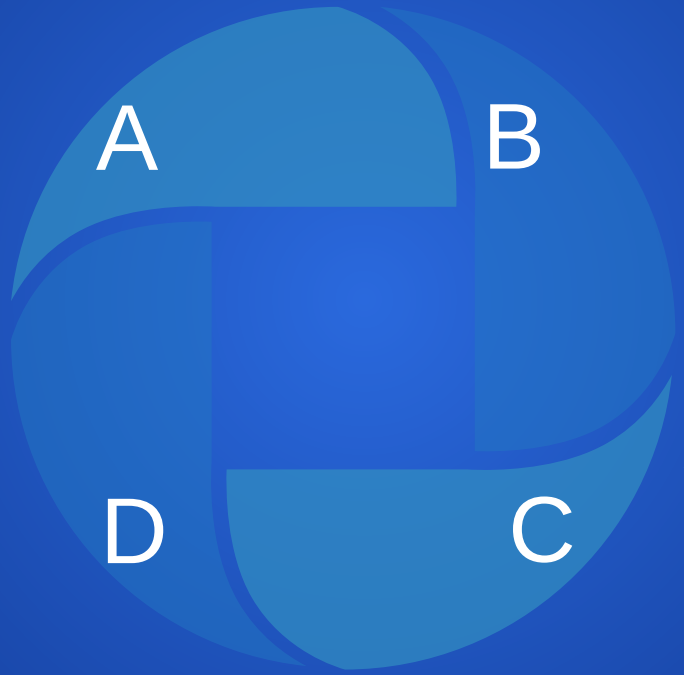
01 跨集群之间的资源管理

04 多集群管理控制台-云看板

多集群支持的内容

应用发布支持多集群

由于集群间网络打通，所以只需要在多集群确保工作负载一致。



多集群ingress与统一网关对接

应用路由以及各集群ingress网关上报给统一网关。

多集群与日志平台的对接

从工作负载页面，支持快捷跳转日志平台和应用监控平台页面。

多集群与监控中心的配合

以应用为纬度，统一上报到prometheus，再由监控中心配置告警阈值。

3

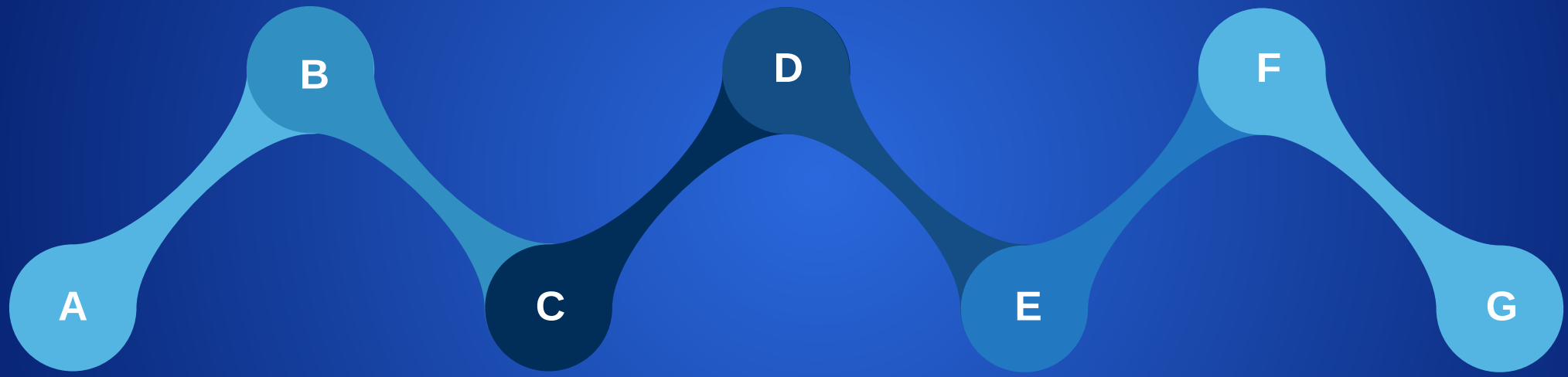
多集群治理

治理内容

应用迁徙

跨集群应用均衡

应用打分



应用迁徙

支持按照集群权重迁徙副本数

当新加入一个集群，需要对于资源进行迁徙，可根据配置，划分设置比重的副本到新集群。

支持整体迁徙

拷贝所以资源到新集群，然后停用旧集群相关资源。

应用打分

部门、产品等多维度

支持根据部门和产品纬度整体打分，输出具体打分标准和低分整改要求。

为了应用的健壮运行提供了数据支撑，各产品良性竞争。

cpu、memory等多标准

支持依据cpu使用情况和设置规则的差值比例；大副本；内存使用等多标准。

依据不同情况对于应用提供治理项和依据。

跨集群应用均衡

历史应用

支持基于副本数和HPA对于工作负载在多集群之间进行均分或者按照集群剩余request等多种方式分布或者伸缩。
且支持cronHpa，未来还将支持基于消息积压的HPA。

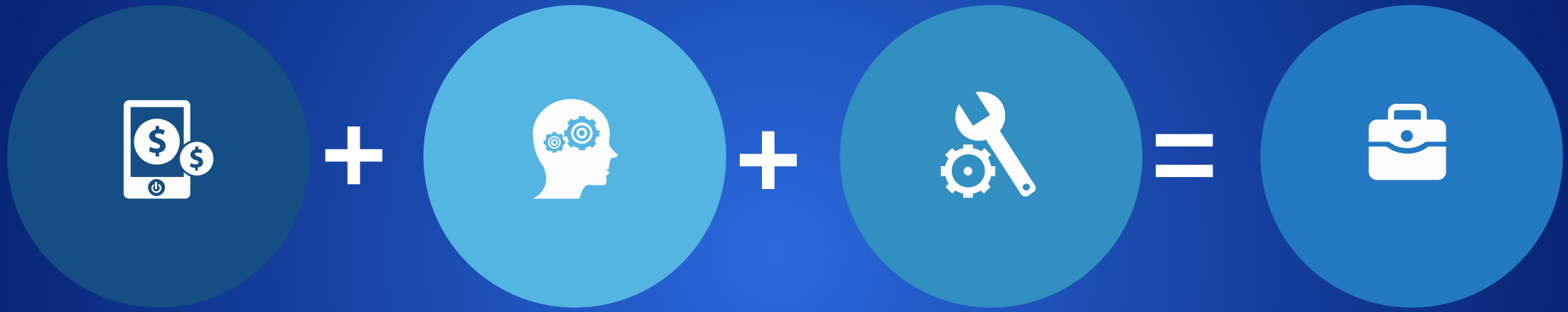
新发布应用

默认开启后，逐渐模糊单集群的概念，应用只需要关于自己的运行状态。

4

成功与总结

成功



基础设施历史包袱
推进容器化进程，回收虚拟机与物理机。

应用全生命周期管理
提供了比虚机更便捷的应用运行、发布、回滚等能力。

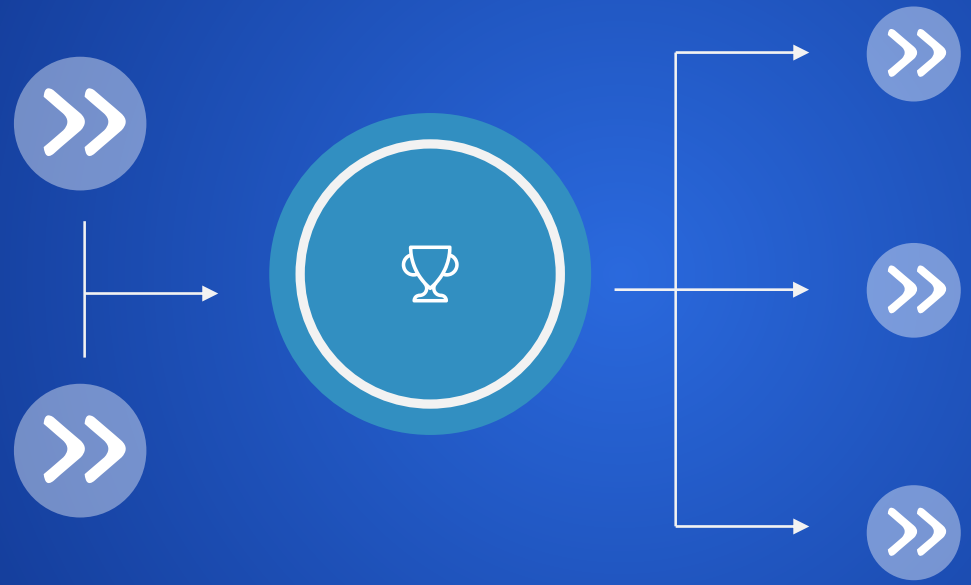
多集群控制能力层次设计
避免鸡蛋都放在一个篮子，多个生产集群，避免单集群的故障对于整体的影响。

多集群管理
目前可以非常友好的支持增加集群或者减少集群，中间周期可以大大缩短。

总结

由于各个集群之间细微的差异，比如版本，K8s核心组件和组件部署方式差异。我们在节点预留资源踩了坑。

与统一网关的结合需要更加友好。比如自动上报网关地址和端口以及域名和产品信息。



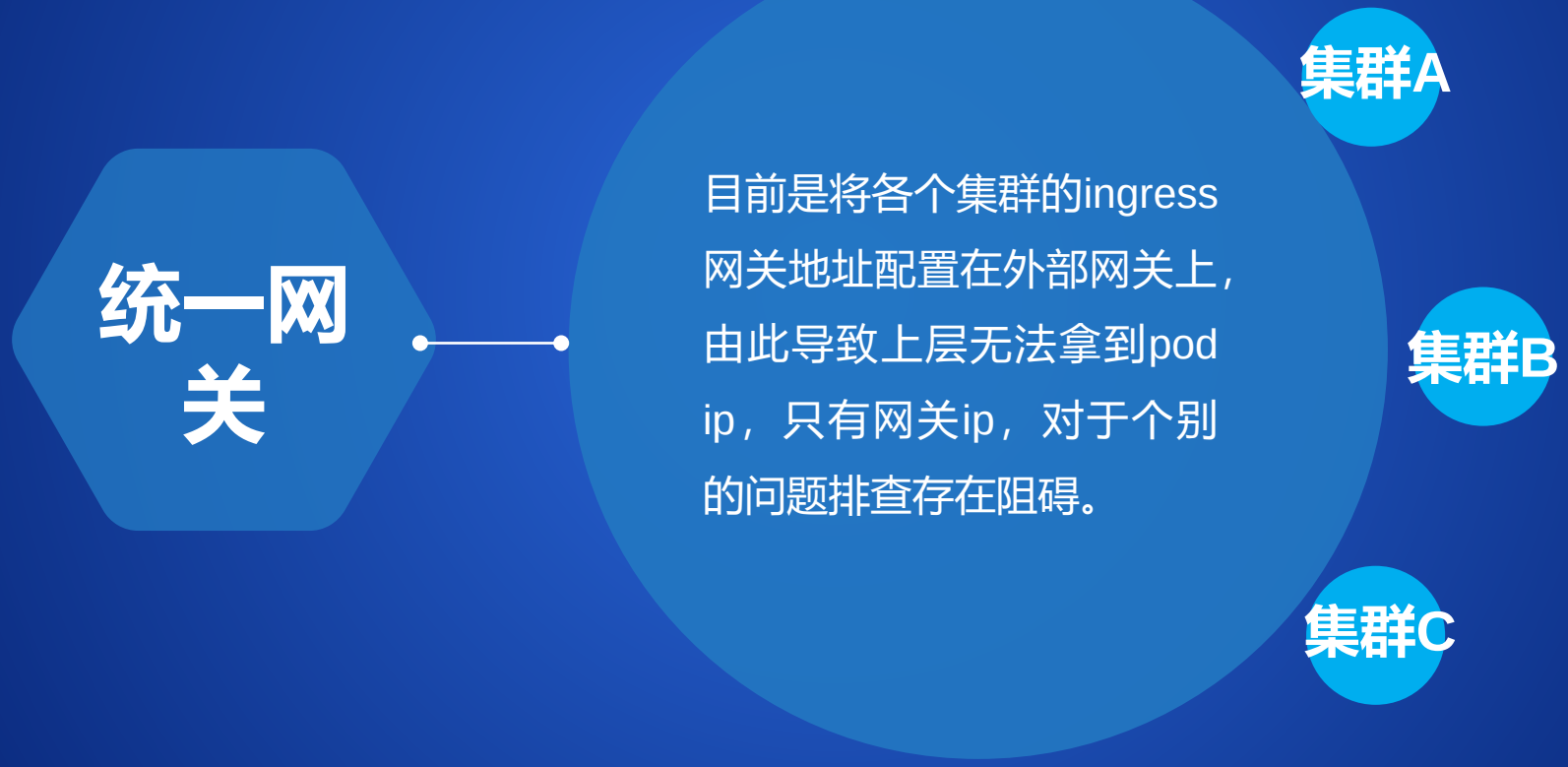
由于长久的历史原因，提供东西流量的治理存在很大的未知性。

由于安全的特殊要求，网络策略在多集群的管理下支持的不够好。

5

未来规划

多集群统一网关


统一网关
目前是将各个集群的ingress网关地址配置在外部网关上，由此导致上层无法拿到pod ip，只有网关ip，对于个别的问题排查存在阻碍。
集群A
集群B
集群C

在离混部

目前大数据侧有大量离线任务；而中通应用集群存在明显的谷峰，在低峰期存在大量可用资源。

东西服务治理

- 支持dubbo
- 充分利用现有资源，监控，zcat，网关等
- 跨部门协作
- 平滑迁徙、过度
- 多集群支持
- 虚机、容器混合

谢谢聆听 欢迎交流

高效协作　勇于担当
积极创新　用户至上