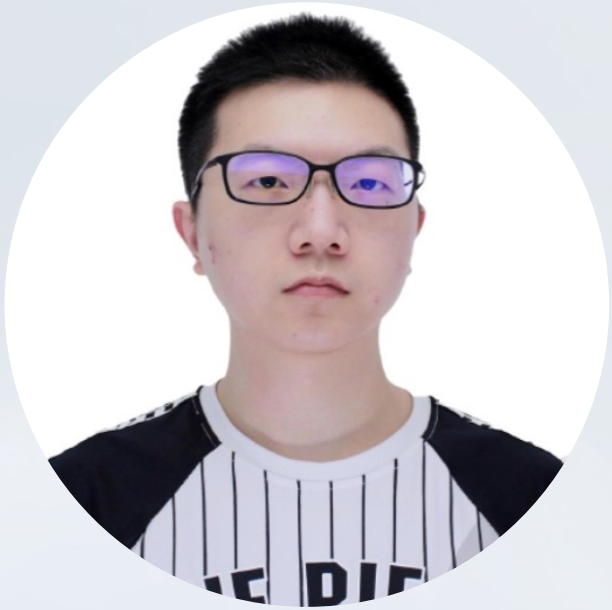
腾讯游戏GitOps发布变更方案

2024年4月

https://sre-elite.com

个人介绍



胡宇涵

腾讯游戏SRE高级工程师

工作经历：

2018 年-2020年：携程

2020 年至今：腾讯

个人简介：

具备丰富的运维开发和技术运营经验，主导了基于蓝鲸容器平台的云原生SaaS设计和开发工作，顺利帮助数十款业务实现云原生改造。目前聚焦于复杂异构业务云原生一体化方案落地以及SRE云原生服务体系构建。

目录

CONTENT

SREelite

01

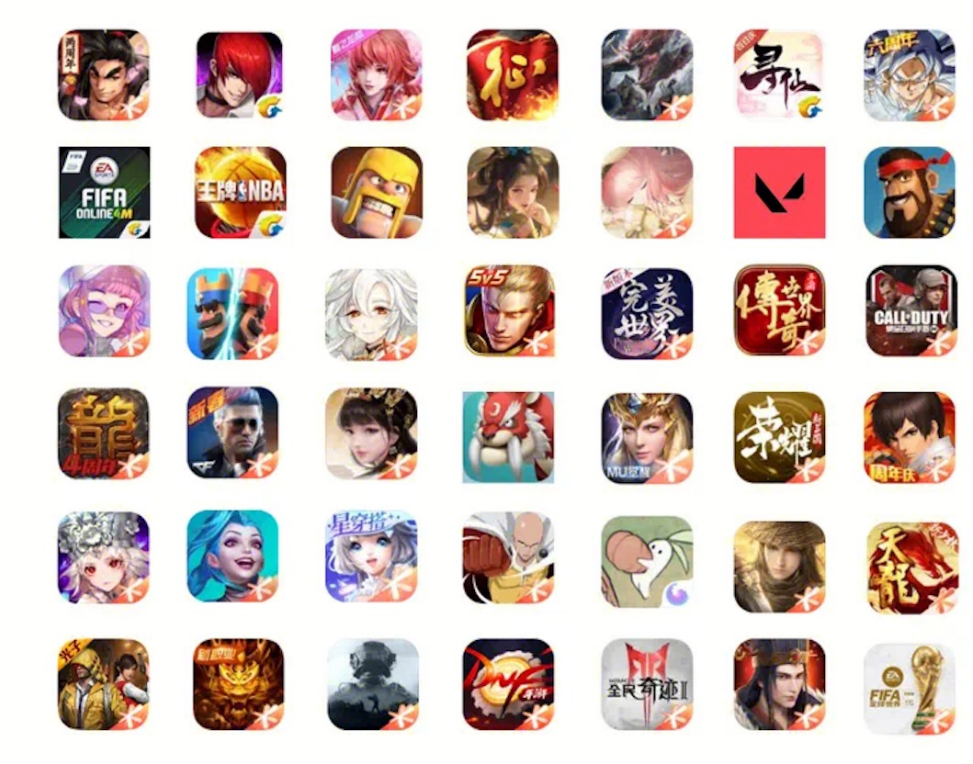
腾讯游戏SRE支撑体系

SREelite

腾讯游戏研发模式

SREelite

大规模业务 + 多样化场景



腾讯游戏业务主要分为自研和代理，部分合作研发

- 游戏开发厂商众多，开发语言多样化
- 游戏类型众多，游戏架构差异大
- 不同游戏间相互独立，业务场景上各不相同
- 每个运维同学的技术栈和运维方式也不同

问题思考：

技术团队如何在不增加人力资源的情况下服务更多业务？

腾讯游戏SRE支撑体系

SREelite

技术运营的三个阶段	SRE理论借鉴	SRE工程化思维	SRE项目落地
CI - 持续集成	Google Site Reliability Engineering	DevOps工程	云原生改造项目 X as Code GitOps持续交付
CD- 持续部署		发布工程 自动化工程	运营/开发/运维协同 发布计划/发布协调
CO- 持续运营		可观测工程 容量工程 可靠性工程 安全工程 AIOPS工程 MLOPS工程	消除琐事/On Call 稳定性:SLI/SLO 数据评估/成本控制 MTBF/MTTR/应急 辅助运营/辅助决策 数据治理/隐私合规

工程化思维与工程化工作：

1. 通常是有创新性和创造性的，着重通过设计来解决问题，解决方案越通用越好；
2. 符合长期战略，对服务进行长久性的改善的工作；
3. 有助于使团队维持同等配备情况下接手更大或更多的服务；

02

X as Code 一切皆代码

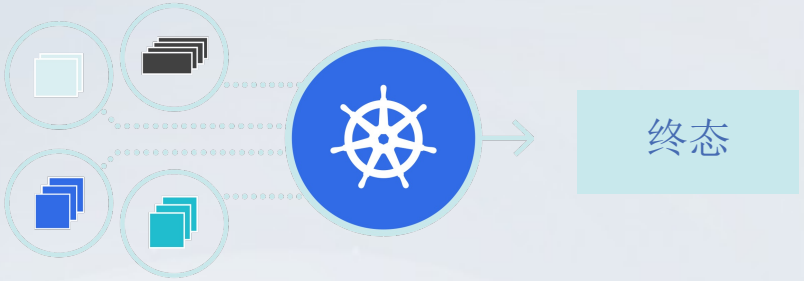
SREelite

Kubernetes带来的启发

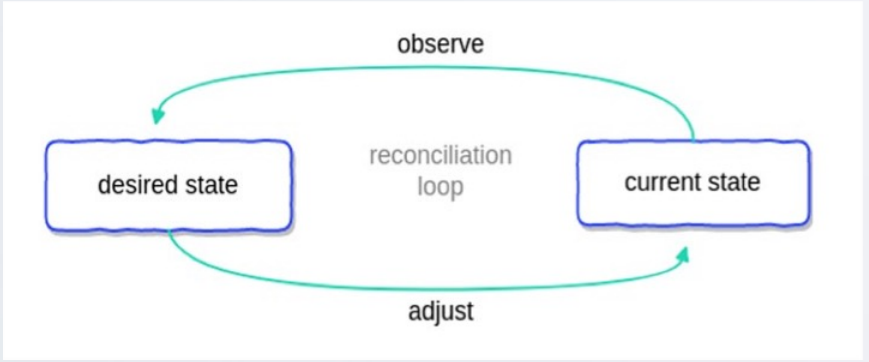
SREelite

Kubernetes 有两个核心理念，一个是声明式 API设计，一个是面向终态的控制循环。

- 声明式API是用户侧抽象的最佳表达方式



- 面向终态的控制循环是保证可靠性的最佳方式



kubectl apply （声明式）

vs

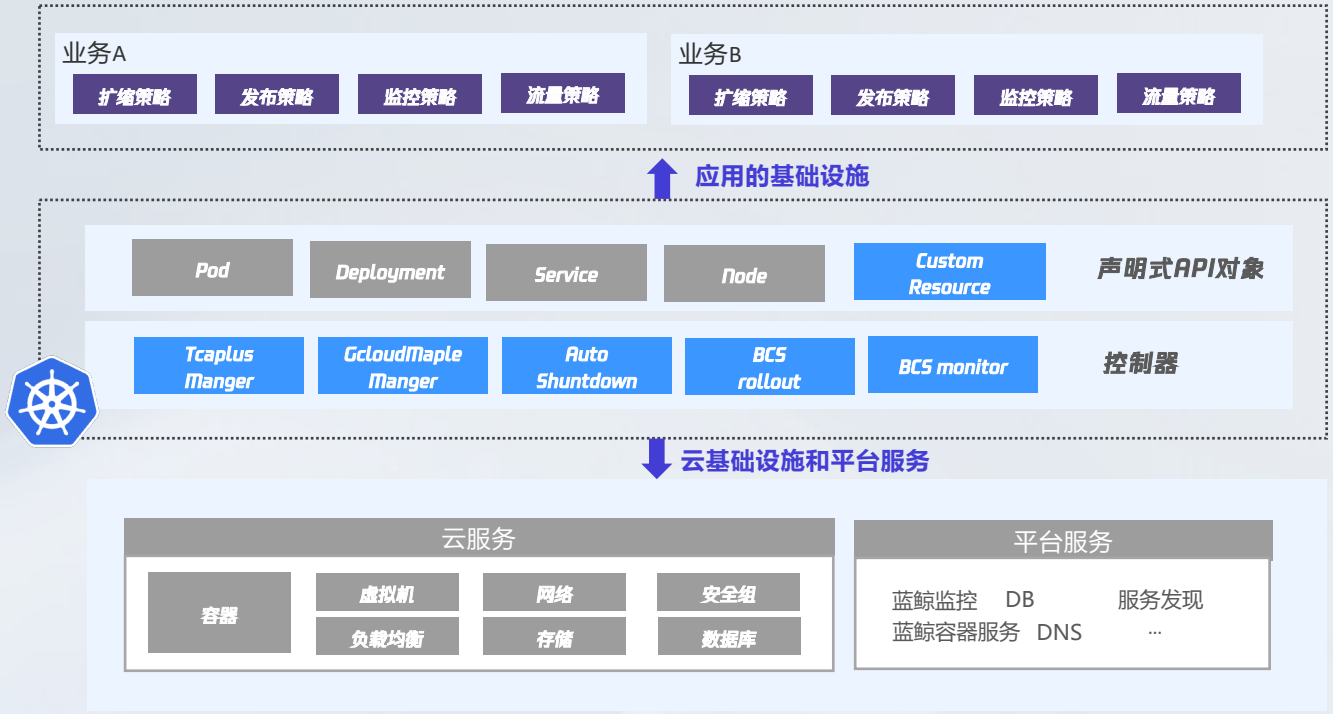
kubectl create （命令式）

声明式API让我们只需描述所需结果，如Kubernetes的apply和create。apply是面向终态的，自动处理创建或更新。这有助于开发运维围绕可版本化的YAML文件协作，无需关注执行步骤。面向终态的控制循环确保应用描述始终正确，具备自愈能力。

X as Code

SREelite

K8s 的成功和价值在于，提供了一种标准的编程接口（API），可以用来编写和使用软件定义的基础设施服务。



实践SRE文化：通过软件工程化的思维，把业务中的工作场景工程化，抽离相对通用的方法论、自动化流程等，帮助开发团队屏蔽复杂的底层基础设施，并可靠的管理应用。

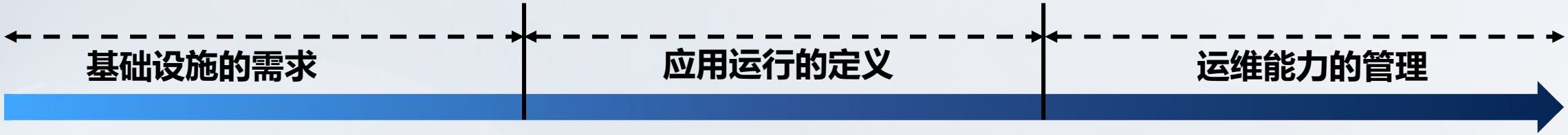
SRE能力团队

我们会把应用对基础设施的需求、应用的运行定义以及各种运维能力下沉到K8s中，基于Kubernetes CRD Operator扩展和构建一系列声明式的服务场景。这样当我们K8s中接入越多的这些能力，我们能在应用中声明的期望状态就越丰富。接下来我会介绍其中几个。

X as Code 场景案例

SREelite

- 面向应用的终态，与环境无关，实现自助服务
- 一份配置描述应用对基础设施的需求，应用的运行定义以及运维能力定义，覆盖应用全生命周期



案例1：
新增一种云资源类型-腾讯云TDMQ管理

案例2：
新增一种DB资源-TCaplusDB自动变更

案例3：
新增一种运维能力-渐进式发布

X as Code 场景案例-腾讯云TDMQ（消息队列）管理

SREelite

- 太多的公有云和系统控制台
- 需要适配不同云的实现

“Topic变更能不能更自动一些？”

- 通过声明式的方式定义应用对基础设施的需求，无需关心执行过程
- 随应用自动创建和删除

Topic CR定义

Live State | Diff | Desire State

```yaml
apiVersion: tdmq.tencentcloud.crossplane.io/v1alpha1
kind: Topic
metadata:
  labels:
    argocd.argoproj.io/instance: sn-server-test
  name: pulsar-topic-transaction-server-test
spec:
  deletionPolicy: Delete
  forProvider:
    clusterId: pulsar-r8[redacted]
    environId: server-test
    partitions: 3
    pulsarTopicType: 3
    remark: crossplane created.
    topicName: transaction
  providerConfigRef:
    name: tcc-crossplane-providerconfig
  providerRef:
    name: provider-tencentcloud
```

X as Code 场景案例-TcaplusDB的变更管理

SREelite

- DB的变更总是遗漏
- 每次需要手动到控制台手动提单操作
- 新增一个环境需要在DB控制台手动创建游戏区和建表

“DB变更能不能更自动一些？”

- 通过声明式的方式定义DB变更，将DB的表结构定义xml/tdr 配置随Helm Chart一起交付
- 自动进行DB游戏区的创建和表结构的变更
- 自动处理异常变更，并通过企业微信发送变更通知

Tcaplus CR定义

Live State | Diff | Desire State

```yaml
apiVersion: apps.power.dev/v1alpha1
kind: Tcaplus
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: '-10'
  labels:
    argocd.argoproj.io/instance: [redacted]-k8s-cdce-trunk
  name: tcaplus-k8s-cdce-trunk
  namespace: k8s-cdce-trunk
spec:
  alert:
    wechat:
      webhook: >-
        https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=a[redacted]
  autoApprove: true
  config:
    appId: '[redacted]'
    host: [redacted]
    password: [redacted]
    setId: '[redacted]'
    username: [redacted]
    zoneId: '[redacted]'
    zoneName: null
  isDelete: false
  onlyCommit: false
  rebuildInValid: true
  sourceRef:
    xml:
      configmap: tcaplus-configmap-k8s-cdce-trunk
  tcaplusHook:
    order: false
    podDelete: false
  zoneTag:
    create:
      - powerapp_busy
    delete:
      - powerapp_idle
```

X as Code 场景案例-渐进式发布

SREelite

刚刚上线的业务往往有很多小问题需要频繁修复，故对于发布能力有较高要求：

- 发布过程对用户影响小
- 支持人工确认
- 发布策略易配置、灵活性高
- 发布可靠性高
- 支持自动回滚
- 支持版本管理

传统基于标准运维的任务：

1.需要维护脚本，脚本执行报错无自愈能力

2.脚本的执行结果不确定，需要检查执行的结果

3.应用的配置有多个来源，不容易追溯历史的变更

4.服务器需要保存集群的凭据，任务迁移繁杂

Rollout声明式定义

```yaml
apiVersion: tkex.tencent.com/v1alpha1
kind: Rollout
metadata:
  name: ck-player-rollout
  namespace: ck-stg-qa
spec:
  failurePolicy: Backward
  objectRef:
    workloadRef:
      apiVersion: apps/v1
      kind: Deployment
      name: ck-player
  steps:
    - hook:
        templateName: ck-player-rollouthook
      replicas: 2
```

03

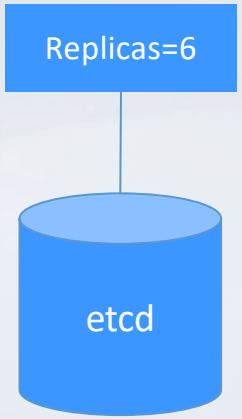
GitOps持续交付

SREelite

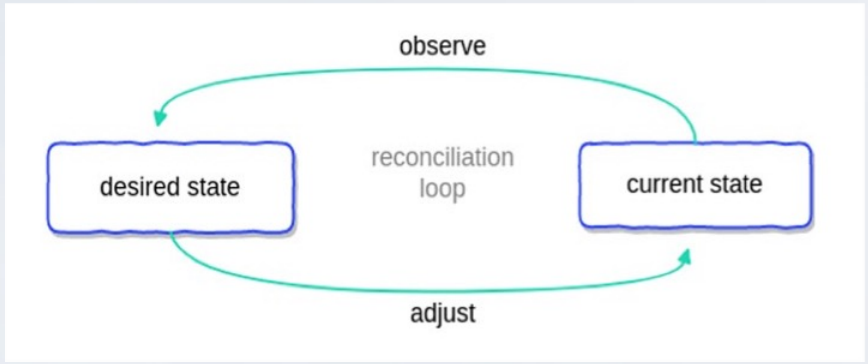
https://sre-elite.com

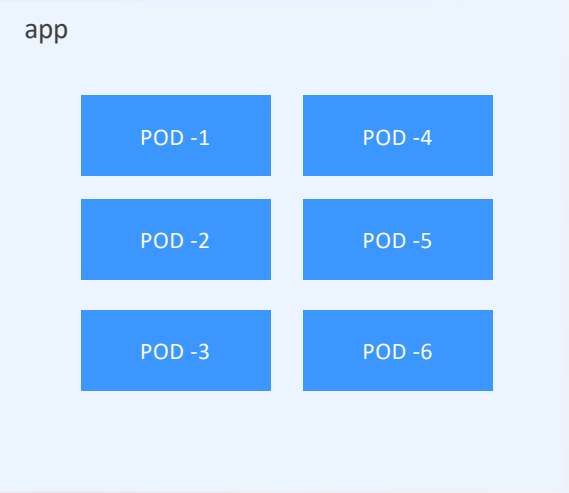
Kubernetes带来的启发：

- 面向终态的控制循环是保证可靠性的最佳方式



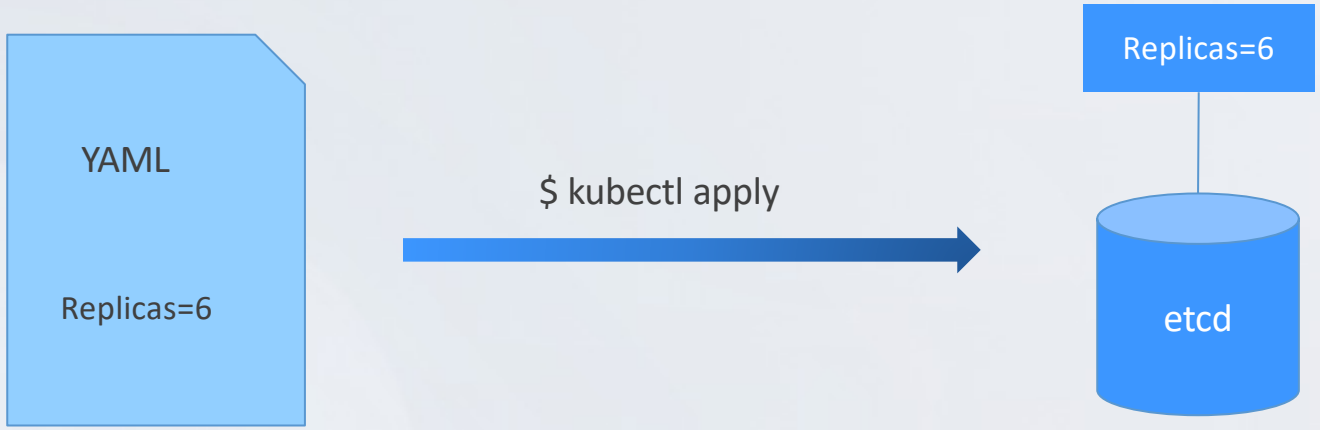
期望状态





当前状态

Replicas=6 这一个期望状态是谁来提供的？
来自于声明式YAML文件

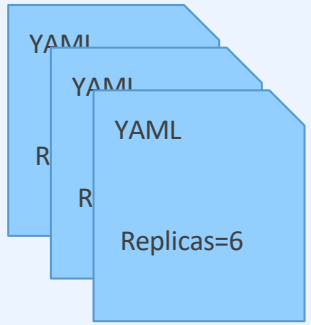


什么是GitOps

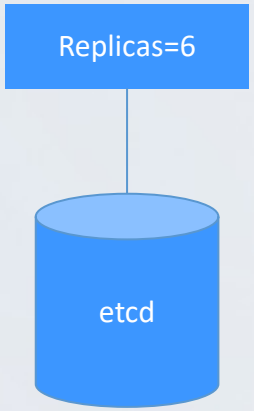
SREelite

期望状态

工蜂



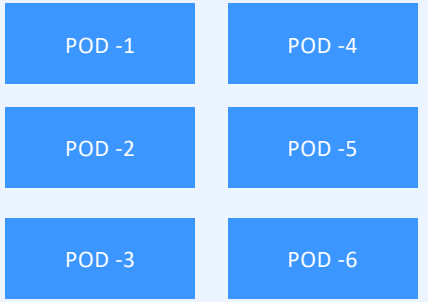
GitOps控制循环



Kubernetes控制循环

当前状态

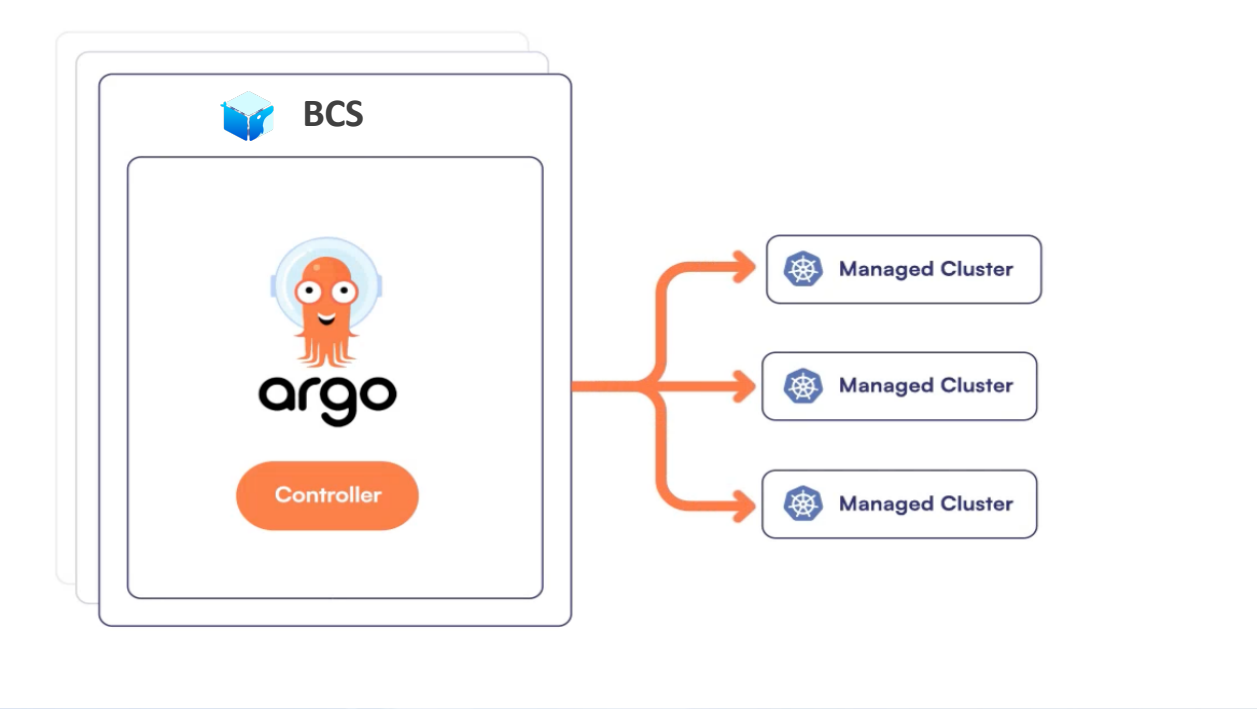
app



基于Git的访问和版本控制

- PR驱动
- 最小化权限
- Review机制
- Commit签名
- 审计和监控
- 回滚

Argo 于 2020 年 3 月 26 日被 CNCF 接受。
Argo 项目于 2022 年 12 月 6 日**从 CNCF 毕业。Argo**获得并保持了 OpenSSF**最佳实践徽章**，并于 2022 年 7 月通过了**第三方安全审计**。
超过 250 个组织正式**使用 Argo CD**，超过 400 个组织至少使用 4 个 Argo 项目中的一个。



GitOps应用交付模式

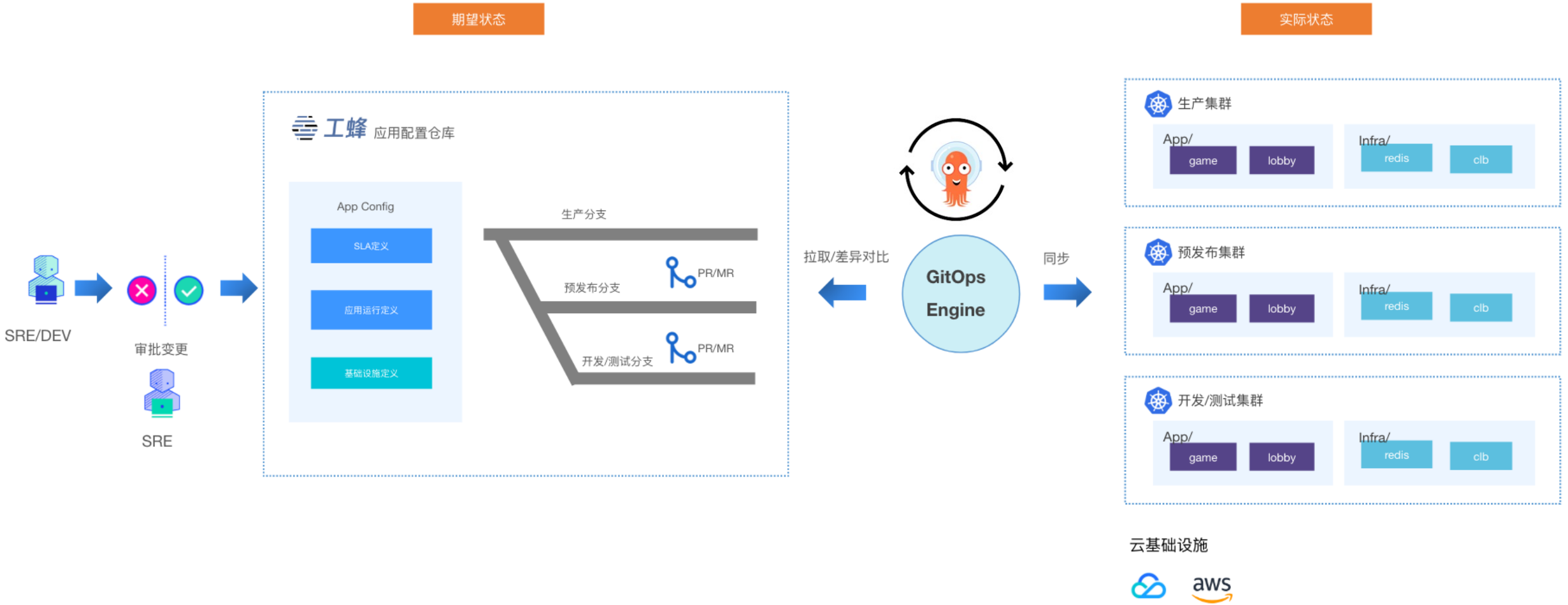
SREelite

CI持续集成

不可变防火墙

CD 持续部署

GitOps持续交付


期望状态
实际状态
SRE/DEV
审批变更
SRE
工蜂 应用配置仓库
App Config
SLA定义
应用运行定义
基础设施定义
生产分支
预发布分支
开发/测试分支
PR/MR
PR/MR
拉取/差异对比
GitOps
Engine
同步
生产集群
App/
game
lobby
Infra/
redis
clb
预发布集群
App/
game
lobby
Infra/
redis
clb
开发/测试集群
App/
game
lobby
Infra/
redis
clb
云基础设施
aws

GitOps持续交付：健康检查和同步顺序

SREelite

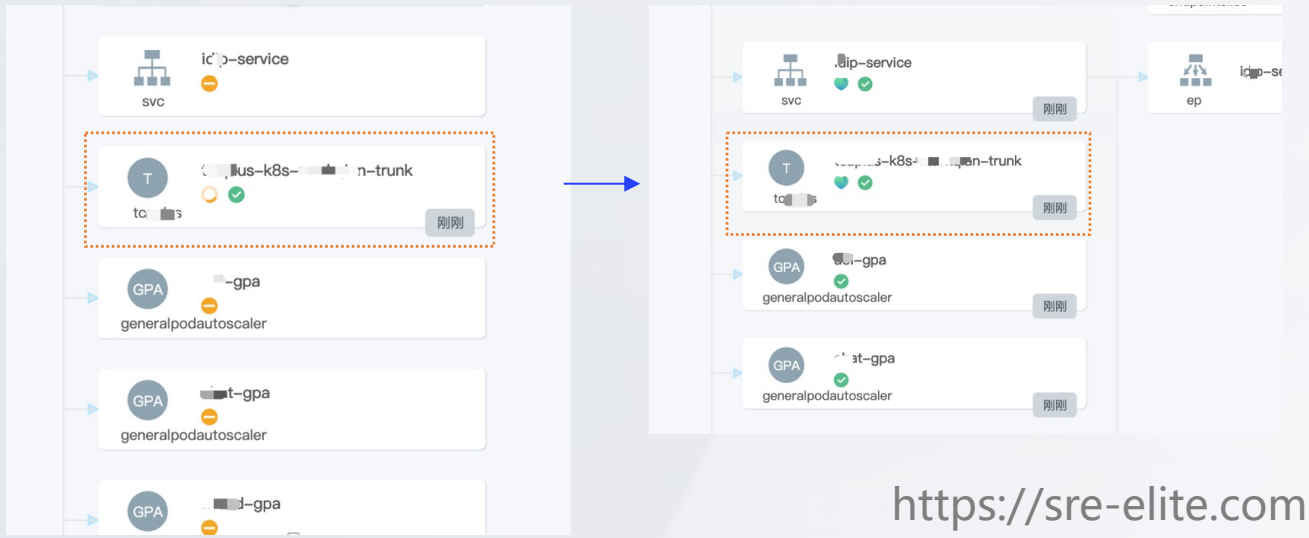
“我希望在部署和更新应用的时候能按照特定的顺序进行？”

自定义资源健康检查

```yaml
apiVersion: apps.power.dev/v1alpha1
kind: Tcaplus
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: '-10'
  labels:
    argocd.argoproj.io/instance: ngcgame-k8s-ai-agent
  name: tcaplus-k8s-ai-agent
  namespace: k8s-ai-agent
spec:
  alert:
    wechat:
```

```yaml
apiVersion: [illegible]om/v1alpha1
kind: GameDeployment
metadata:
  labels:
    app: id[illegible]p
    argocd.argoproj.io/instance: [illegible]fy-pressure-publish
    io.tencent.bcs.dev/deletion-allow: Always
  name: id[illegible]p
```

```
resource.customizations.health.apps.power.dev_T[illegible]us: |
  hs = {}
  if obj.status ~= nil then
    if obj.status.status == "Complete" then
      hs.status = "Healthy"
      hs.message = obj.status.tableInfo
      return hs
    end
  end
  hs.status = "Progressing"
  hs.message = "Waiting for tcaplus"
  return hs
```

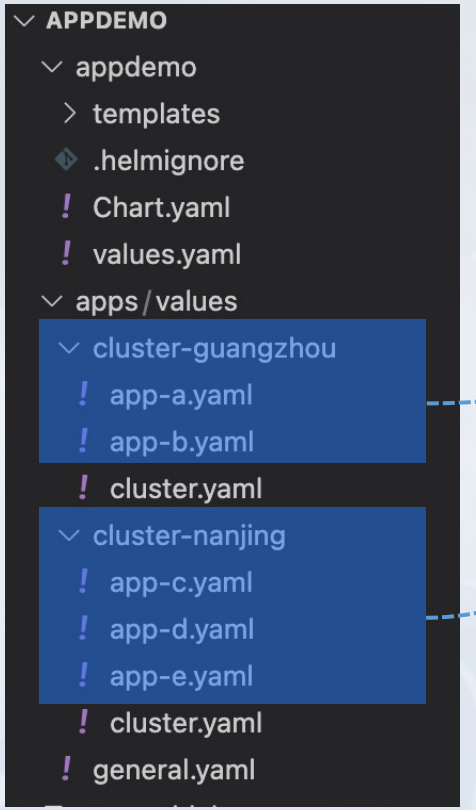


GitOps持续交付：多应用和集群的自动化部署

SREelite

“测试环境的部署模型是否也能通过Git目录结构来定义？”

Git 环境部署仓库目录结构



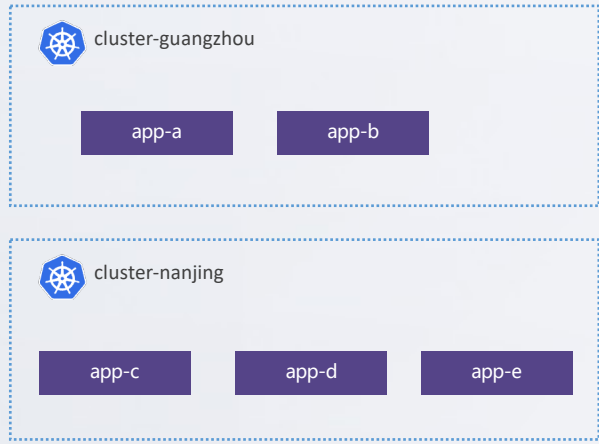
通过Git 文件生成器渲染部署来源和部署目标

sn-appset.yaml > ...

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: sn-game
  namespace: argocd
spec:
  generators:
  - git:
      repoURL: https://[redacted]g/appdemo.git
      revision: HEAD
      files:
      # 只匹配指定目录下，指定文件名开头的yaml
      - path: "apps/values/*/app*.yaml"
  template:
    metadata:
      # 从yaml中获取应用名称参数
      name: '{{fullnameOverride}}'
    spec:
      project: sn
      source:
        repoURL: https://[redacted]g/appdemo.git
        # 从yaml中获取环境部署所使用的git tag
        targetRevision: '{{versionTag}}'
        path: "appdemo"
        helm:
          valueFiles:
            - ../apps/values/general.yaml
            - ../apps/values/{{path.basename}}/cluster.yaml
            - ../{{path}}/{{path.filename}}
      destination:
        name: '{{path.basename}}'
        namespace: '{{fullnameOverride}}'

```

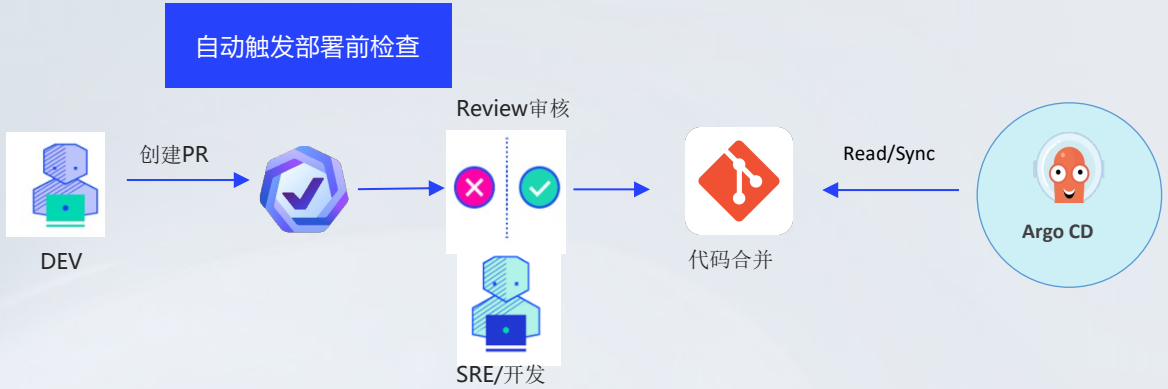
K8s集群上应用



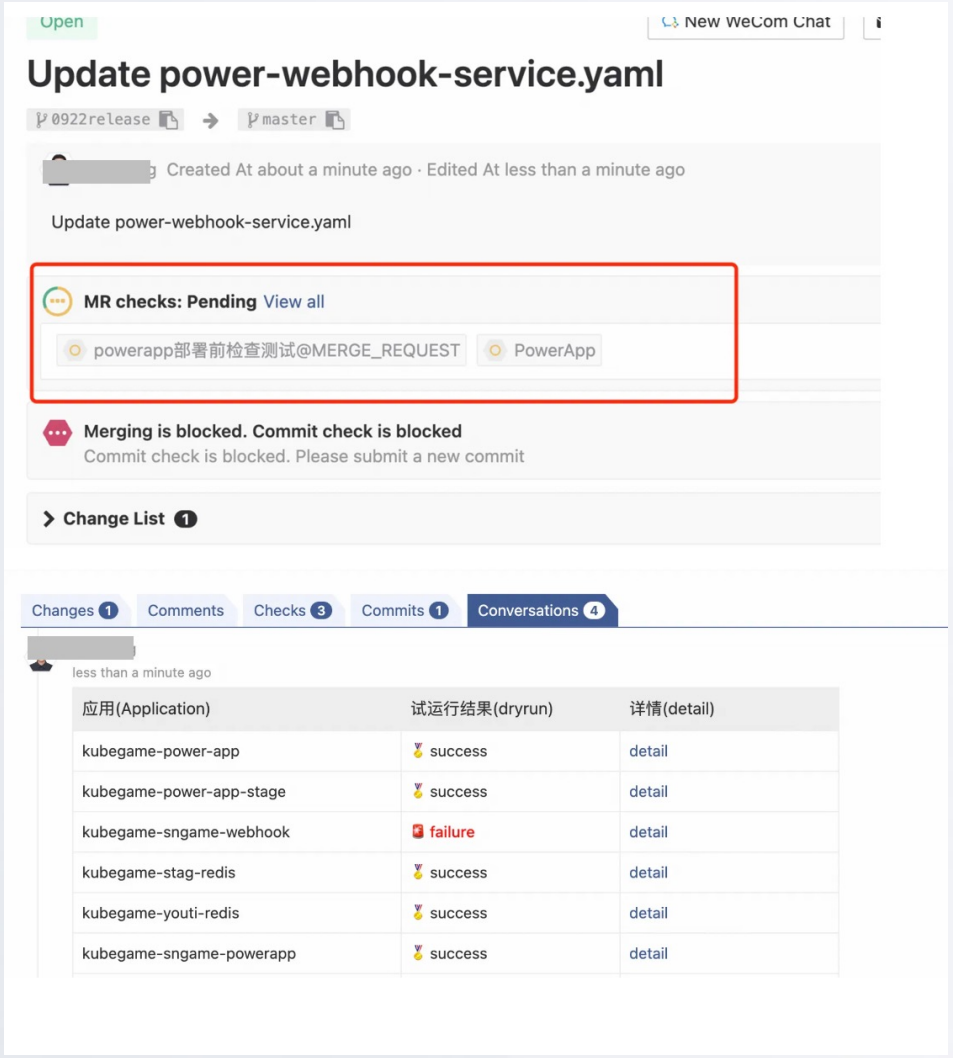
GitOps持续交付：部署前检查

SREelite

“是否可以在PR/MR过程中查看实际的变更差异以及模拟运行结果？”



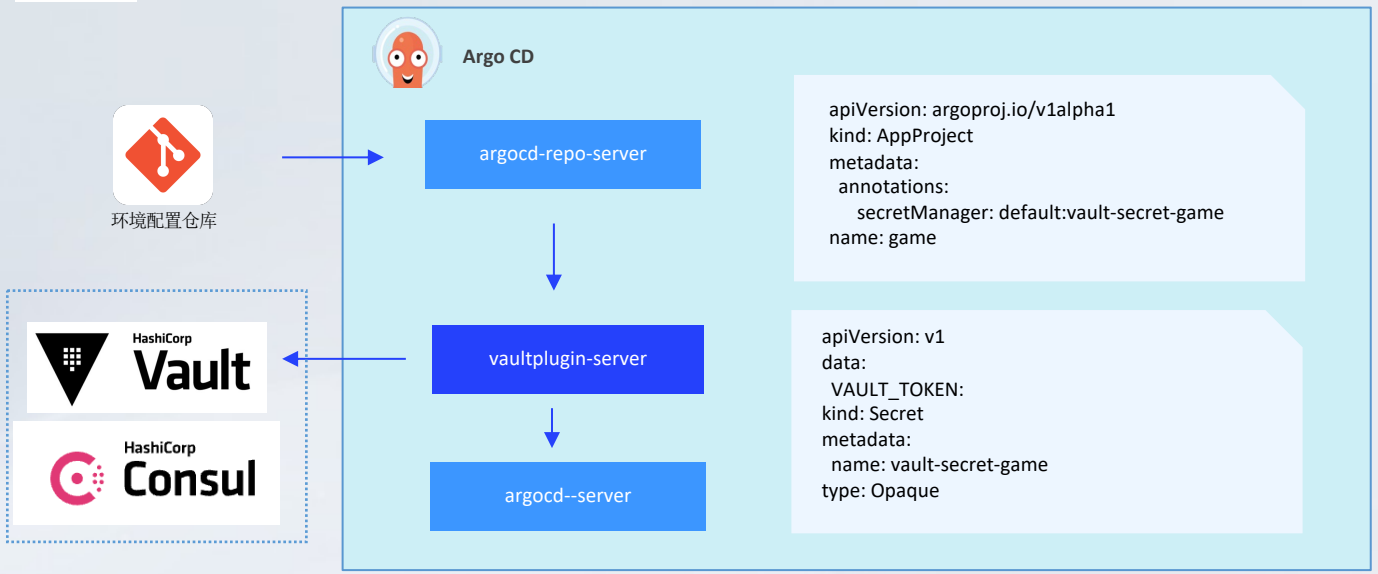
1. PR自动触发部署前检查，PR被锁定，任何人不能进行合并
2. 在PR的会话中回写检查结果，包含helm diff 、模拟运行的结果、是否合规
3. 确认异常原因并修复
4. 重新提交PR
5. Review配置变更

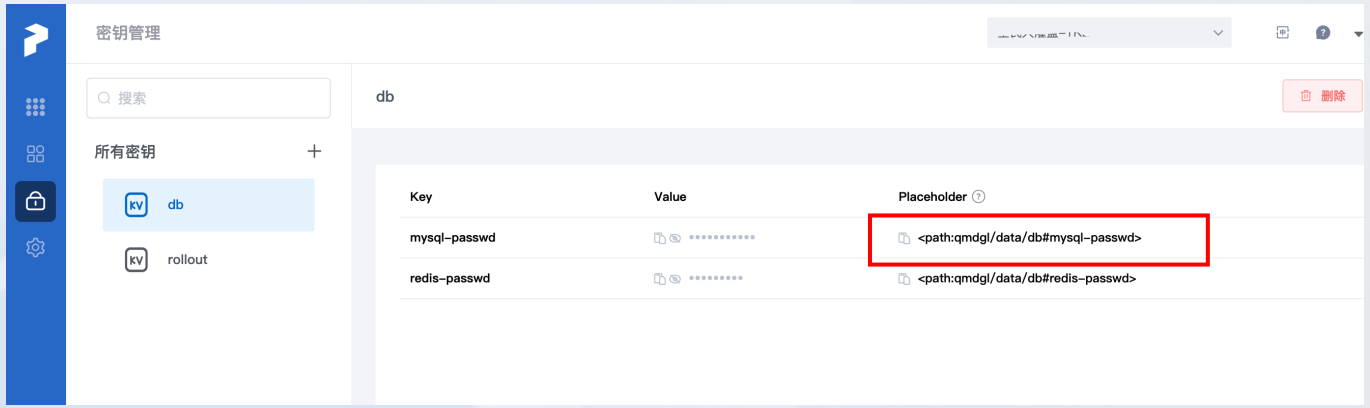


GitOps持续交付：密钥管理

SREelite

“敏感信息不能上传到Git中”





Yaml配置中只需要填写对应的placeholder

qa911-values.yaml

qmdgl > apps > gamesvr > bcs-values > qa911-values.yaml > { } global > { } mysql

```
66    preSecAcceptMax: 0
67    preSecRecvMax: 0
68
69  raceProxyServer:
70    preSecAcceptMax: 0
71    preSecRecvMax: 0
72
73  watchProxyServer:
74    preSecAcceptMax: 0
75    preSecRecvMax: 0
76  #偏移时间
77  serverDate: 0
78  idipGateProtocol: http
79  idipPubEnable: 0
80
81  clb:
82    serviceInnerID:
83    serviceInnerIP:
84    servicePubID:
85    servicePubIDdianxin:
86    servicePubIDliantong:
87    servicePubIDyidong:
88    servicePubIDcap:
89    servicePubIP:
90
91  ZooKeeper:
92    host:                                    .svc.cluster.local:2181
93  mysql:
94    user: qmdgl
95    passwd: <path:qmdgl/data/db#mysql-passwd>
96    host:
97    port: 25000
98  redis:
99    passwd: <path:qmdgl/data/db#redis-passwd>
100   host:
101   port: 40911
102
103 resourceImage:                              /resource
104 resourceImageTag: 20230809-170843-reload
105 DirectAccess: true
```

04

总结

SREelite

https://sre-elite.com

总结

SREelite

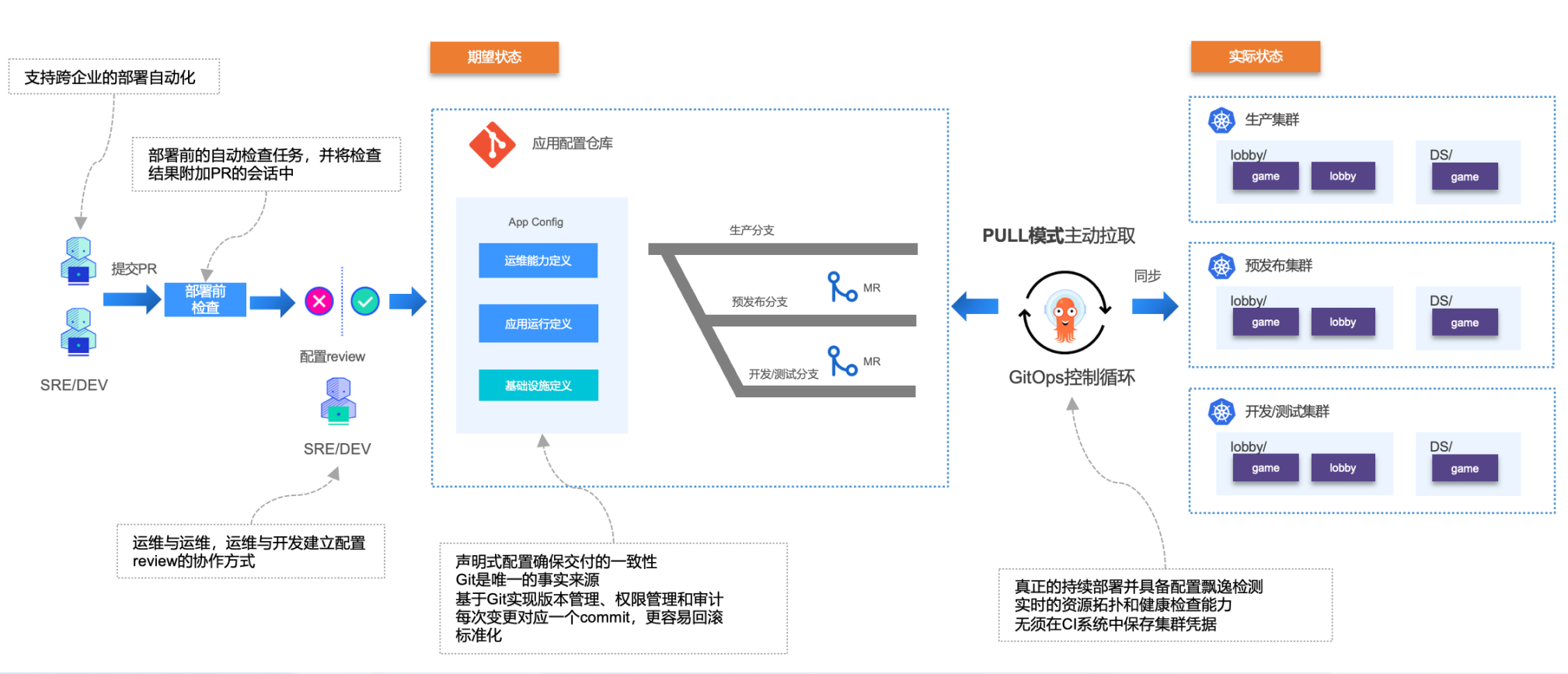
GitOps 是一种云原生模式下的应用交付模式，以**声明式**的方式描述整个系统，并通过Git进行**版本控制**，同时具有确保部署环境与Git中**期望状态相匹配**的自动化流程。

总结

SREelite

GitOps提升交付质量和交付效率：声明式应用定义结合GitOps，确保交付的一致性，带来应用交付以及运维协作的变革

云原生SRE服务场景扩展：围绕游戏生态构建一系列声明式的自助式的服务场景，向平台工程靠拢，做到可衡量、可管理、可复制，并且持续优化和创新



Q&A

https://sre-elite.com

附录

供Meetup主持人和分享嘉宾参考

SREelite

https://sre-elite.com

“SRE精英联盟”概述

SREelite

2003年
Google 成立 SRE 团队

2016年
孙宇聪翻译出版首部 SRE 著作

2020年
赵成牵头成立 SRE研讨社群

SREelite
2023年11 月
SRE 精英联盟正式成立

《SRE 实践白皮书》首次正式发布

2024年2月

联盟首次直播

2024年2月29 日

联盟首次线下 Meetup

2024年4月13 日

https://sre-elite.com

《SRE实践白皮书》

SREelite

- 经历数年，20 多位一线专家协作编写。
- 扫码下载 v1.0.1。版本持续更新迭代中。
- 在官网 https://sre-elite.com/notice/下载最新版。

关注我们

SREelite

公众号

视频号

B 站

YouTube