GIAC | 全球互联网架构大会
GLOBAL INTERNET ARCHITECTURE CONFERENCE

如何利用 Kubernetes 建设 AI 时代的 DevOps 平台

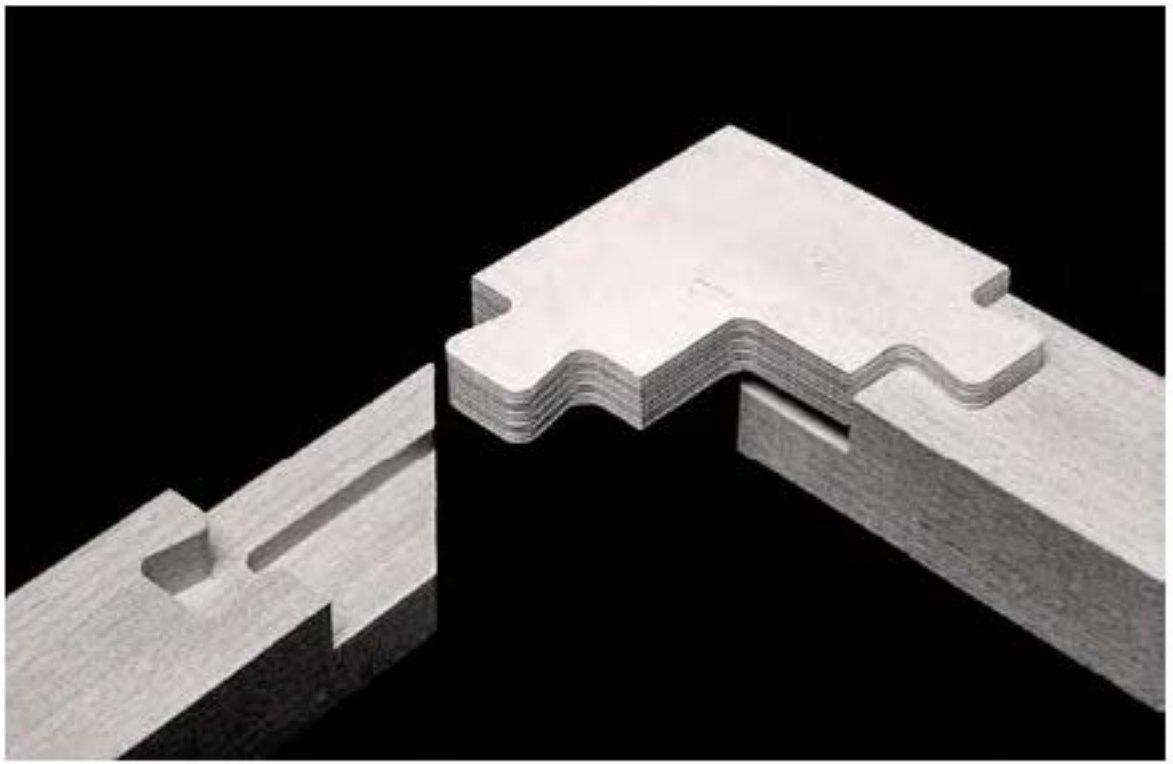
王渊命 @jolestar
Kubernetes/FaaS 布道师

个人介绍

- 王渊命 @jolestar 前青云容器平台负责人
- 多栖程序员 java/go/python 开发/运维/测试
- 微博/通讯工具/协作工具/容器/云
- 技术写作者，https://jolestar.com
- Kubernetes "布道师"，《Kubernetes 完全教程》作者

什么是 AI DevOps?

AI + DevOps ? How ?



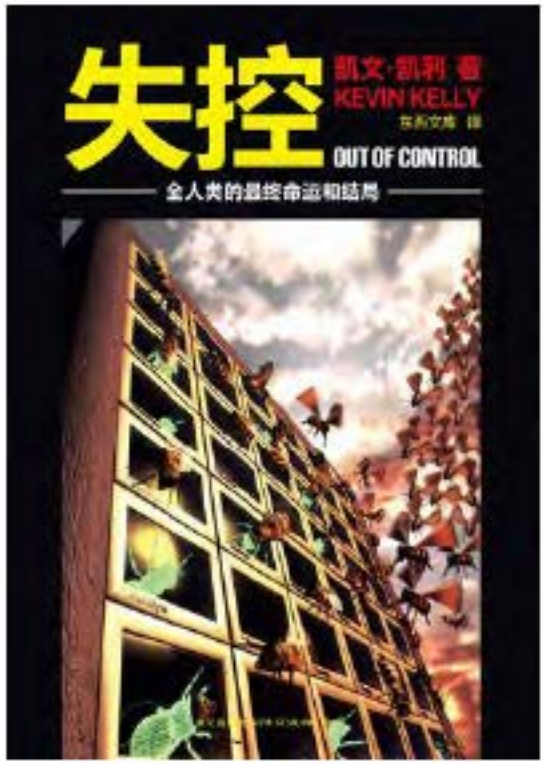
www.flexiblestream.org/project/50-digital-wood-joints

从两个角度思考（工程师）

- 编程以及架构范式
- AI 和 DevOps 的结合点

什么样的编程以及架构范式更适应 AI

- 放弃精细化控制
- 提供声明式接口



放弃精细化控制

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

声明式(Declarative) VS 命令式(Imperative)

```
SELECT * FROM members WHERE group='archnotes'
and gender='female';
```

```
//members = [{'group':"archnotes",gender:"male","name":"jo

var femaleMembers = []
var member
for (var i=0; i< members.length; i++){
        member = members[i]
        if ( member.group == "archnotes"
                && member.gender == "female" ){
        femaleMembers.push(member)
    }
}
```

声明式(Declarative) VS 命令式(Imperative)

```
<div id='#draw-shapes'>
  <svg width="280" height="200">
    <circle cx="72" cy="100" r="50" fill="#FF8000"
            stroke="orangered" stroke-width="5" />
  </svg>
</div>
```

```
// Make an instance of two and place it on the page.
var elem = document.getElementById('draw-shapes').children
var params = { width: 280, height: 200 };
var two = new Two(params).appendTo(elem);
var circle = two.makeCircle(72, 100, 50);
circle.fill = '#FF8000';
circle.stroke = 'orangered'; // Accepts all valid css colo
circle.linewidth = 5;
// Don't forget to tell two to render everything
// to the screen
two.update();
```

@jole

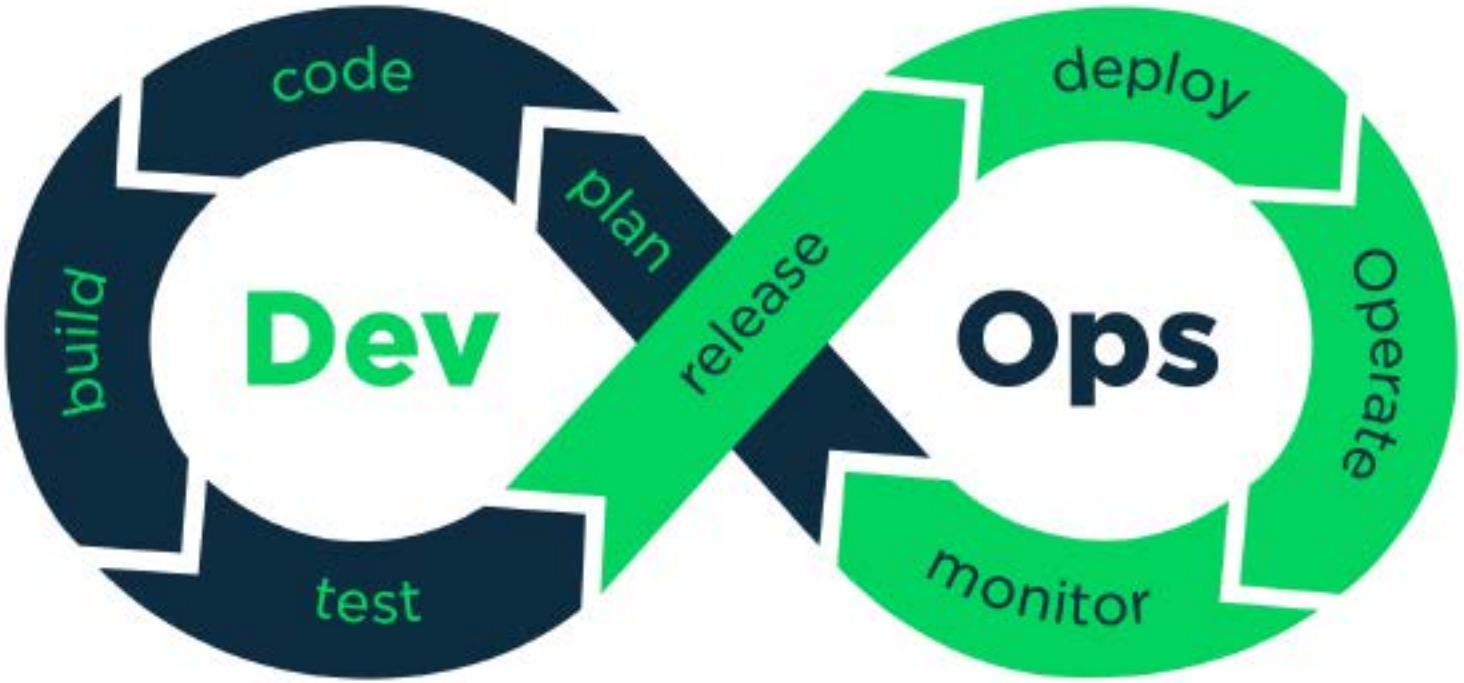
思考

- 那种方式更容易将实现细节交由 AI 实现?
- 那种方式更容易通过 AI 生成?

声明式的优势

- 关注于目标而不是途径
- 更符合人类的心智模式，而不是机器的运作模式
- 更容易屏蔽实现细节
- 更容易通过基础组件组装出高级组件（如：高阶函数）

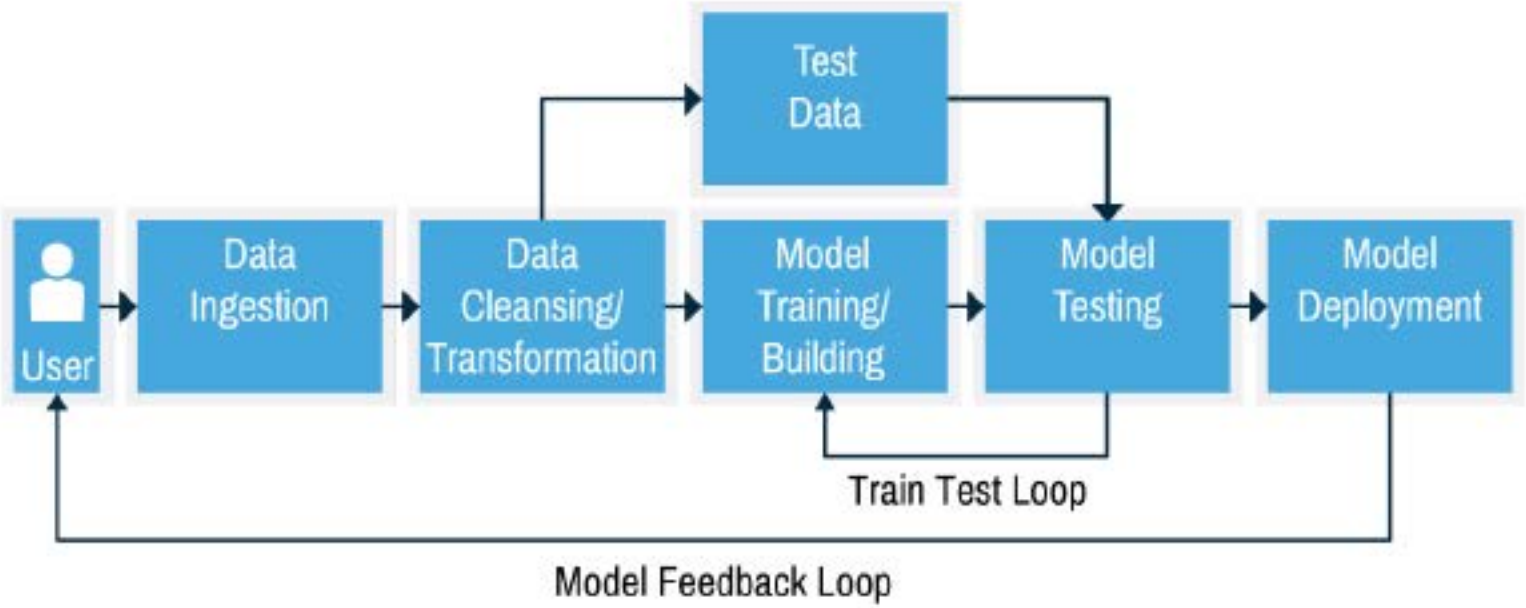
AI 和 DevOps 平台的结合点



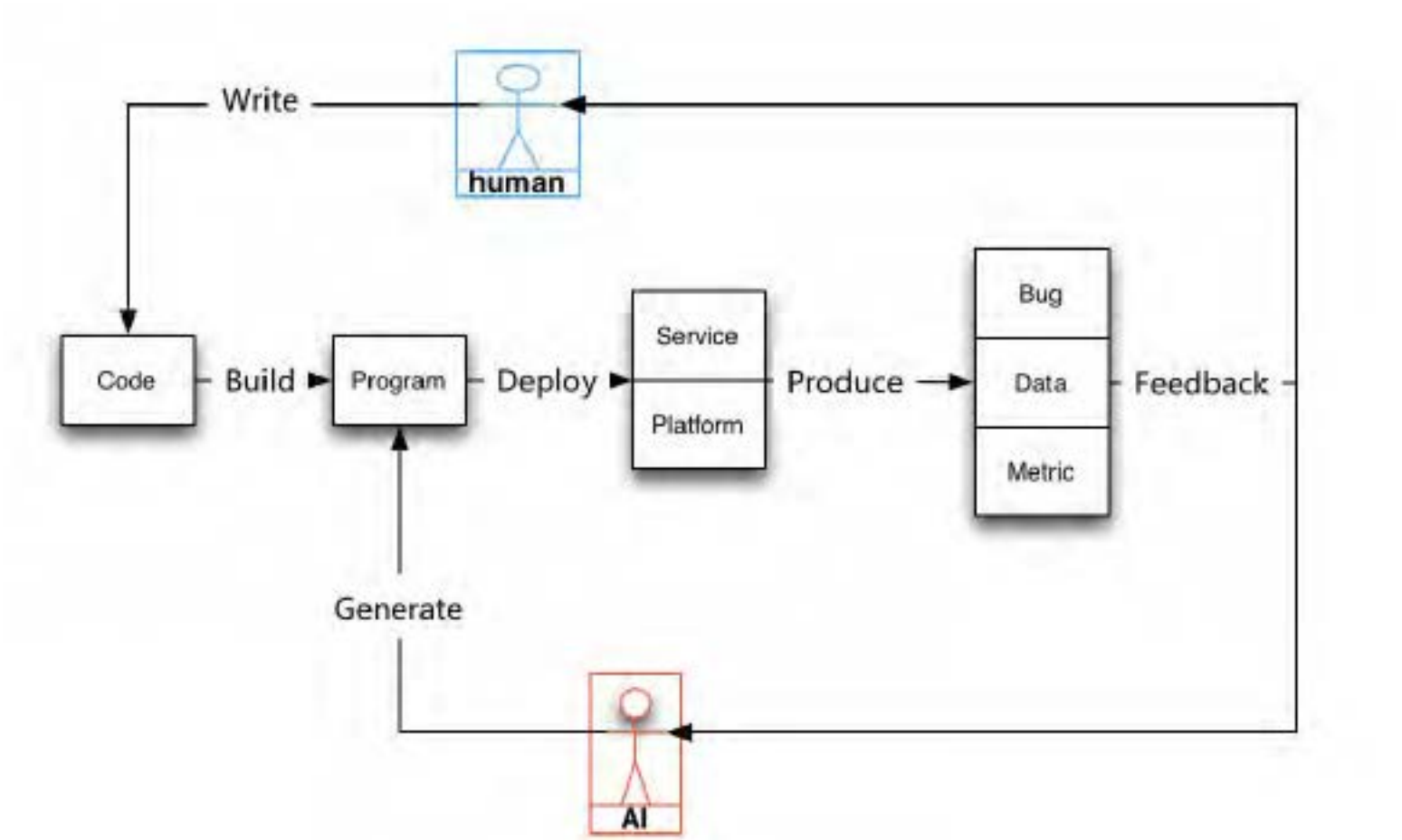
AI 和 DevOps 平台的结合点

- AI 通过 DevOps 输出自己的能力
- DevOps 利用 AI 的能力智能化

AI Workflow


Test Data
User
Data Ingestion
Data Cleansing/ Transformation
Model Training/ Building
Model Testing
Model Deployment
Train Test Loop
Model Feedback Loop

AI DevOps 的本质 - 平台需要接纳另一种软件演进方式



AI 对 DevOps 平台的要求

- 更全面的数据收集
- 更自动的对照测试（A/B 测试）
- 更快速的交付循环

当前 DevOps 平台现状

- 各工具运行在各自的环境中，各自实现分布式
- 工具链通过各自的配置文件，脚本，触发器，人工操作粘合在一起

AI 时代的 DevOps

- 统一的运行环境
- 统一的控制平面
- 工具之间互相感知，自动化配置
- 全自动化交付

如何利用 Kubernetes 建设 AI 时代的 DevOps 平台

@jolestar

为什么是 Kubernetes？

Kubernetes

> Kubernetes is not a mere 'orchestration system'; it eliminates the need for orchestration. The technical definition of 'orchestration' is execution of a defined workflow: do A, then B, then C. In contrast, **Kubernetes is comprised of a set of independent, composable control processes that continuously drive current state towards the provided desired state.** It shouldn't matter how you get from A to C: make it so.

Kubernetes 的特点

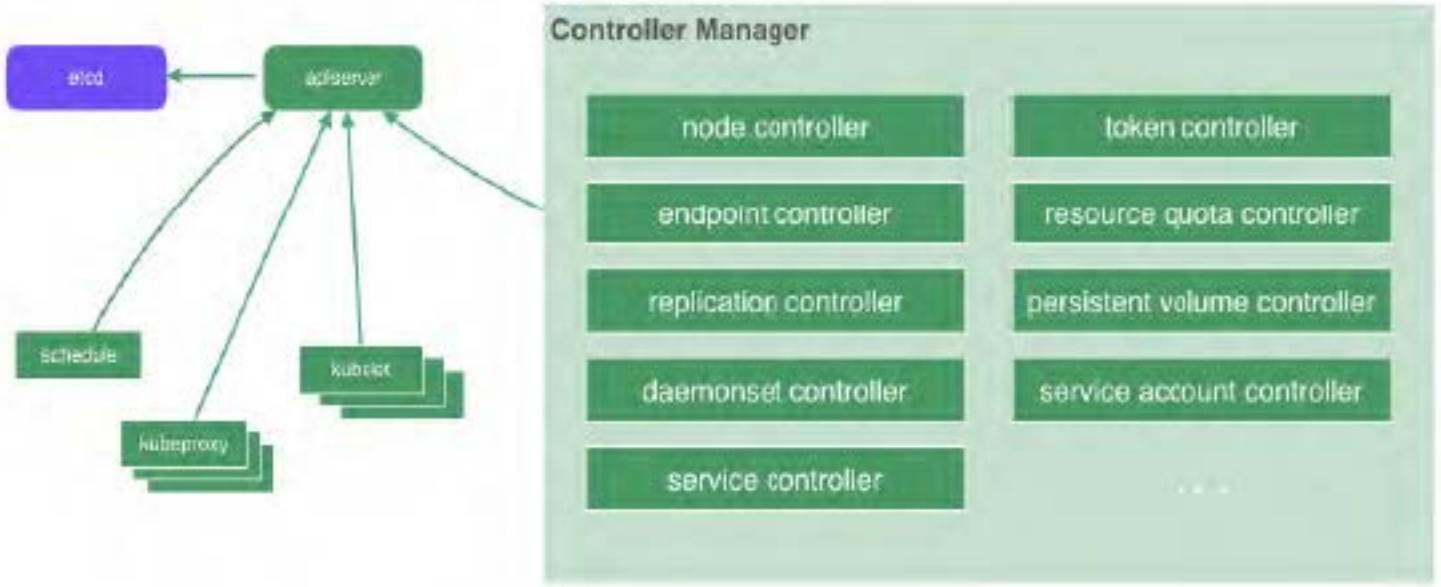
- 声明式
- 共享状态
- 独立控制器
- 可组装
- 易扩展

声明式接口规范

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
  namespace: default
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

共享状态独立控制器架构

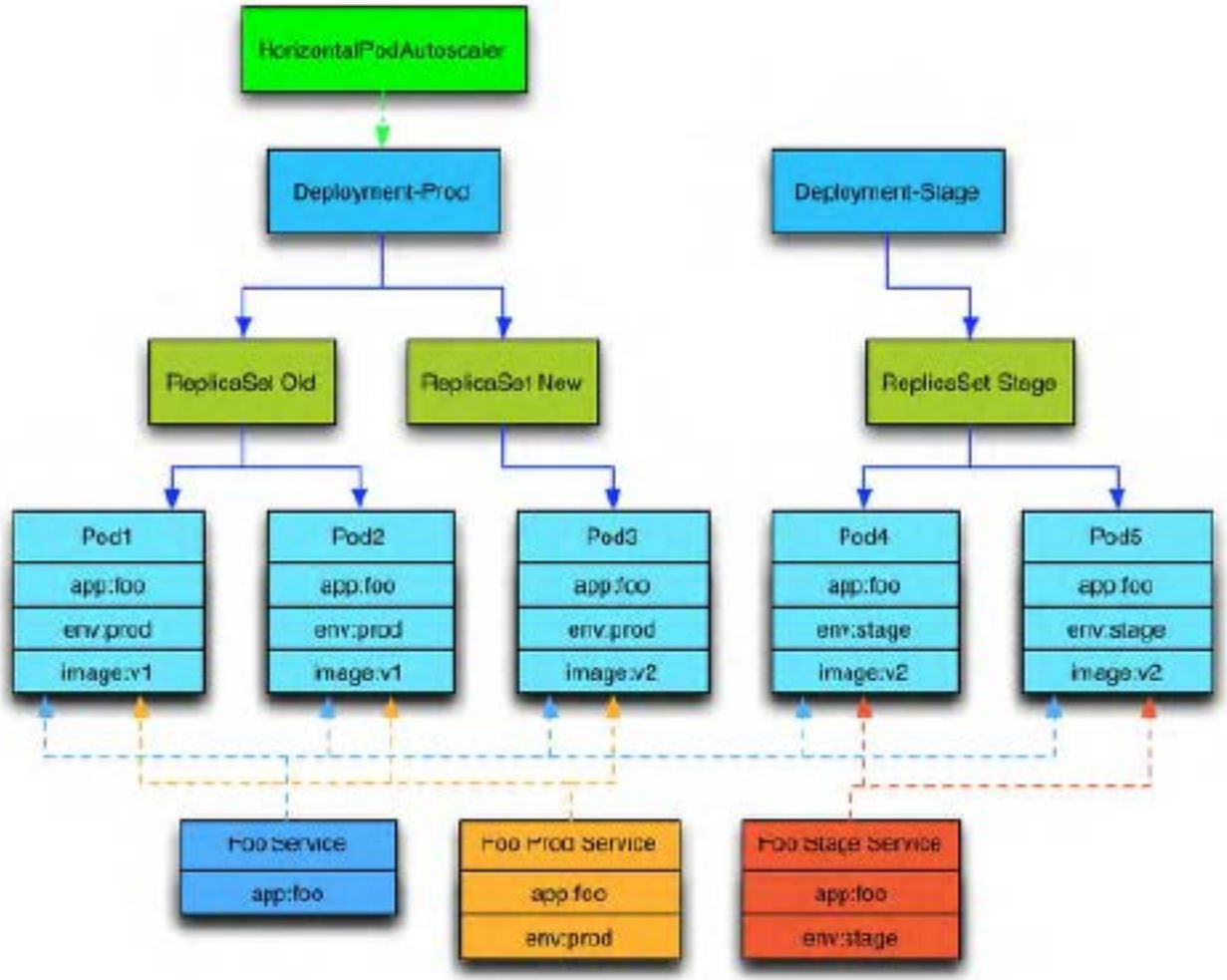
一个状态存储，多个控制器



Kubernetes 控制器逻辑

```
for {
  desired := getDesiredState()
  current := getCurrentState()
  makeChanges(desired, current)
}
```

可组装的独立组件


HorizontalPodAutoscaler
Deployment-Prod
Deployment-Stage
ReplicaSet Old
ReplicaSet New
ReplicaSet Stage
Pod1
app:foo
env:prod
image:v1
Pod2
app:foo
env:prod
image:v1
Pod3
app:foo
env:prod
image:v2
Pod4
app:foo
env:stage
image:v2
Pod5
app:foo
env:stage
image:v2
Foo Service
app:foo
Foo Prod Service
app:foo
env:prod
Foo Stage Service
app:foo
env:stage

Kubernetes 的扩展机制

- Annotation + 自定义控制器
- CustomResourceDefinitions
  - 自定义资源对象
  - 自定义控制器
  - 自定义 API

Kubernetes 在 DevOps 平台中的角色

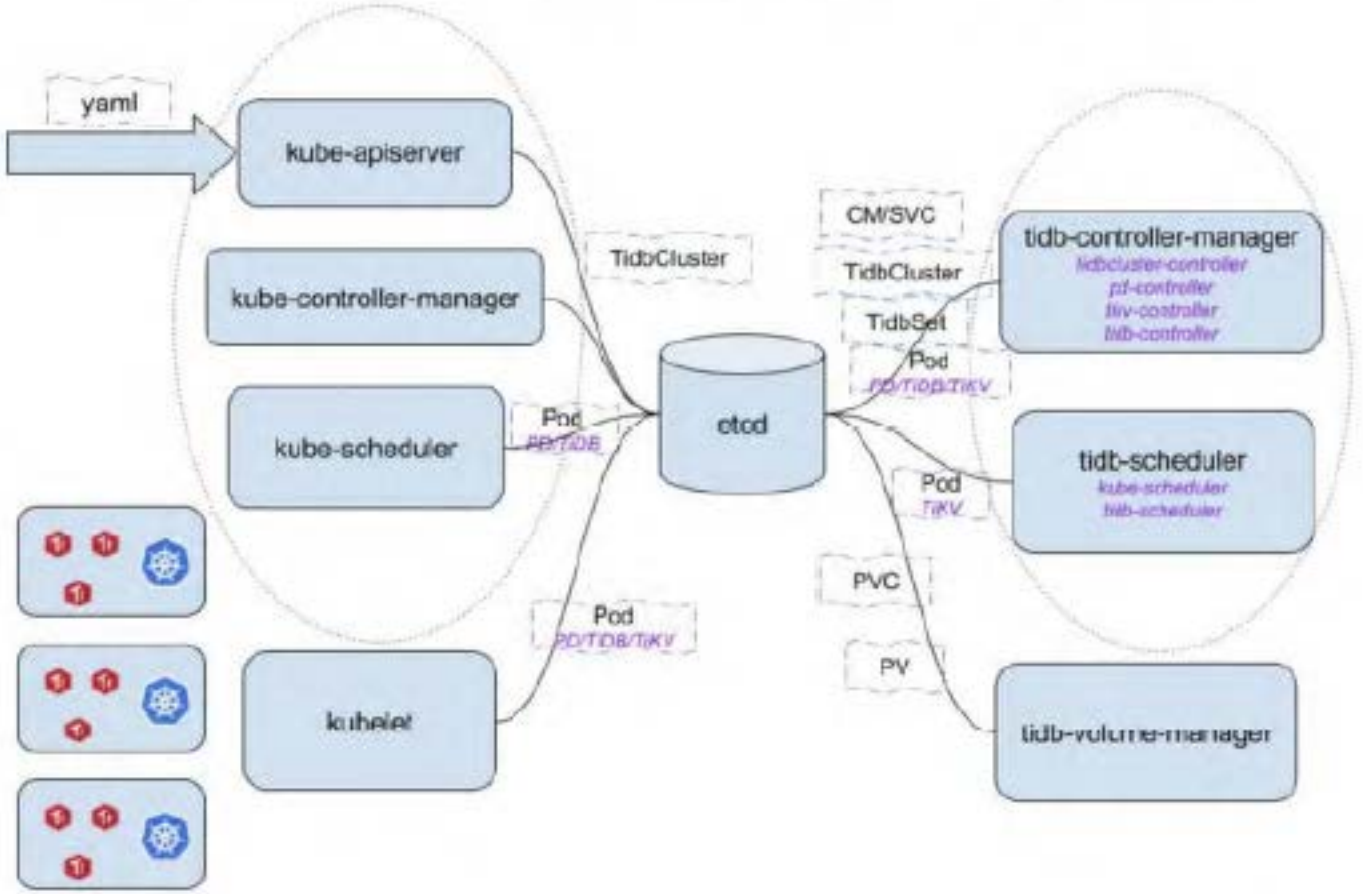
- The Linux of the Cloud
- DevOps 工具链的『大脑』

案例：基于 Kubernetes 的 RDS -- TiDB

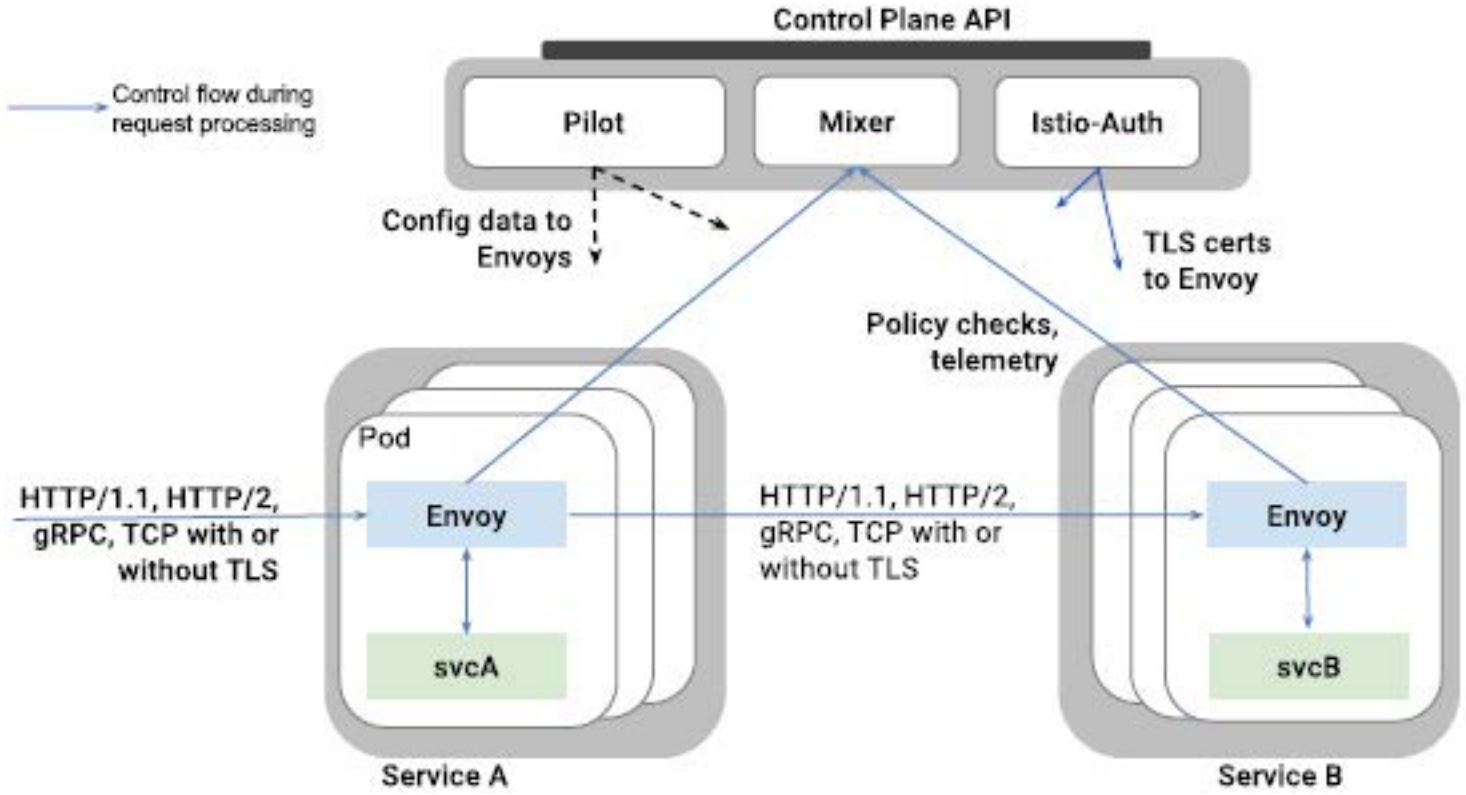
- TiDB-Operator

```
apiVersion: pingcap.com/v1
kind: TidbCluster
metadata:
  name: demo-cluster
spec:
  pd:
    size: 1
    image: pingcap/pd:latest
  tidb:
    size: 2
    image: pingcap/tidb:latest
  tikv:
    size: 3
    image: pingcap/tikv:latest
  service: NodePort
```

TiDB-Operator 架构



案例：Kubernetes 上的 ServiceMesh -- Istio



Istio RouteRule

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: ratings-test-delay
spec:
  destination:
    name: ratings
  httpFault:
    delay:
      fixedDelay: 7.000s
      percent: 100
  match:
    request:
      headers:
        cookie:
          regex: ^(.*?;)?(user=jason)(;.*)?$
  precedence: 2
```

@jole

Istio CRD

- metric
- rule
- logentry

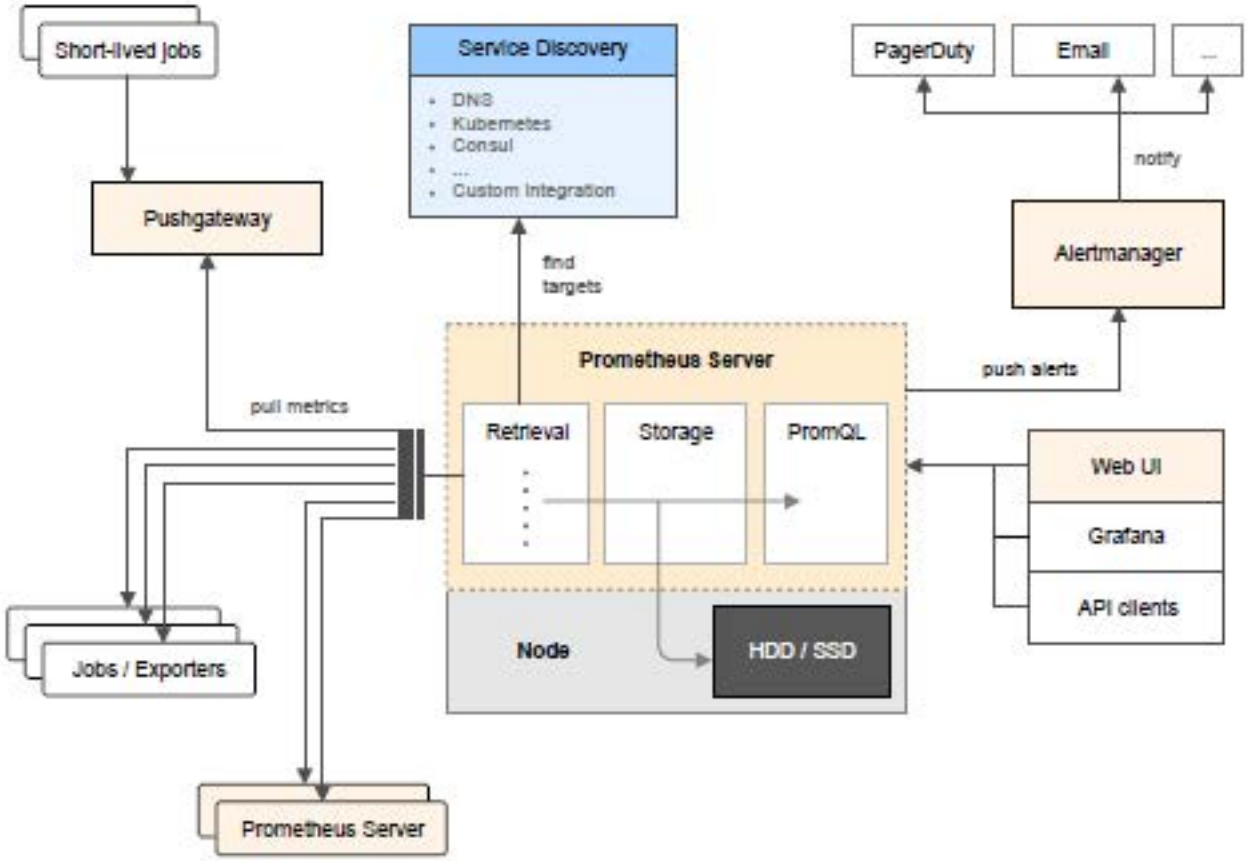
```yaml
apiVersion: "config.istio.io/v1alpha2"
kind: metric
metadata:
  name: doublerequestcount
  namespace: istio-system
spec:
  value: "2" # count each request twice
  dimensions:
    source: source.service | "unknown"
    destination: destination.service | "unknown"
    message: '"twice the fun!"'
  monitored_resource_type: '"UNSPECIFIED"'
```

@jolecian

案例：基于 Kubernetes 的自动化的监控系统

- Prometheus Operator
  - Prometheus
  - ServiceMonitor
  - Alertmanager
- Kubernetes prometheus adapter
- Metadata integration

Prometheus 架构



Prometheus Operator:Prometheus

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: k8s
spec:
  replicas: 2
  version: v2.0.0
  serviceMonitorSelector:
    matchExpressions:
    - {key: k8s-app, operator: Exists}
  ruleSelector:
    matchLabels:
      role: prometheus-rulefiles
      prometheus: k8s
  alerting:
    alertmanagers:
    - namespace: monitoring
      name: alertmanager-main
      port: web
```

@jolestar

Prometheus Operator:ServiceMonitor

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus
  labels:
    k8s-app: prometheus
spec:
  selector:
    matchLabels:
      prometheus: k8s
  namespaceSelector:
    matchNames:
    - monitoring
  endpoints:
  - port: web
    interval: 30s
```

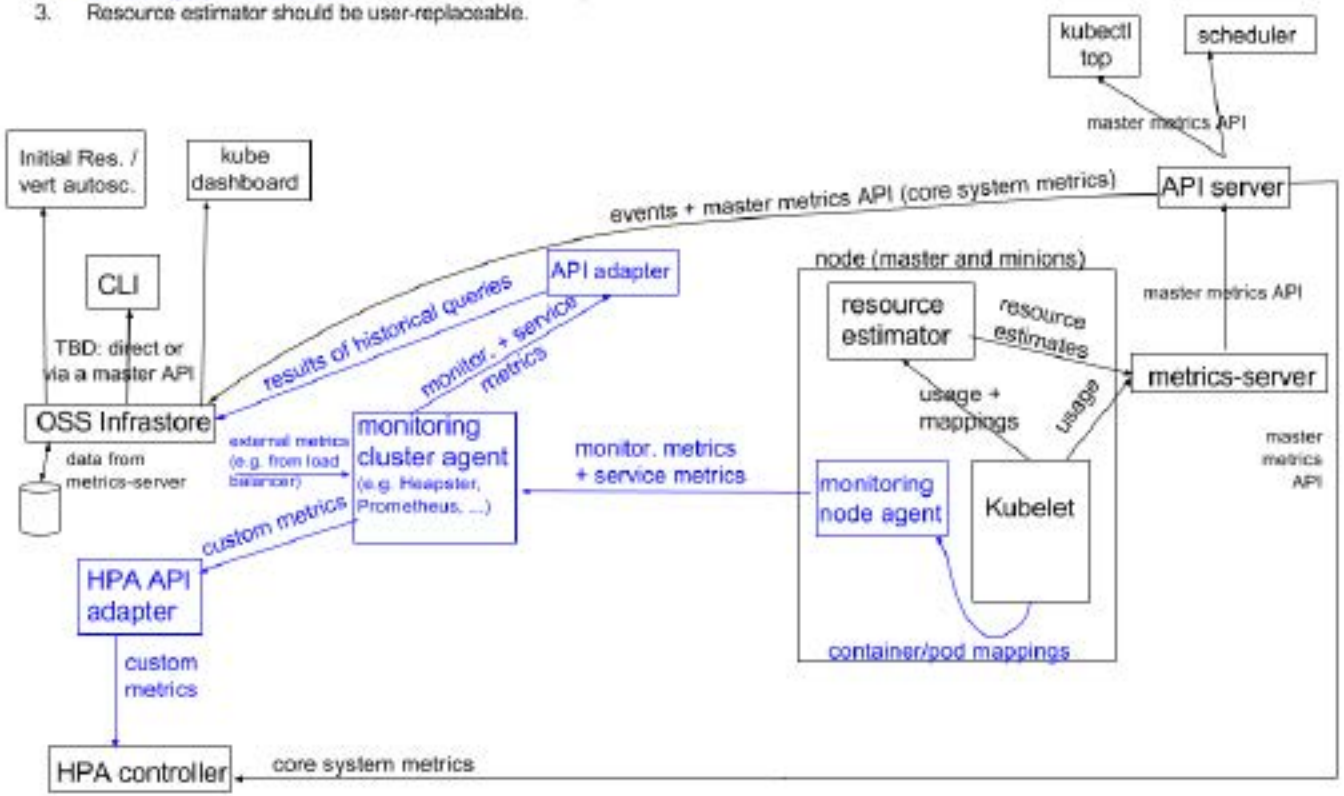
Prometheus Operator:Alertmanager

```
apiVersion: monitoring.coreos.com/v1
kind: Alertmanager
metadata:
  name: main
  labels:
    alertmanager: main
spec:
  replicas: 3
  version: v0.9.1
```

Kubernetes 监控架构

Monitoring architecture proposal: OSS
(arrows show direction of metrics flow)

Notes
1. Arrows show direction of metrics flow.
2. Monitoring pipeline is in blue. It is user-supplied and optional.
3. Resource estimator should be user-replaceable.



AI DevOps 构想：AIAutoscaler

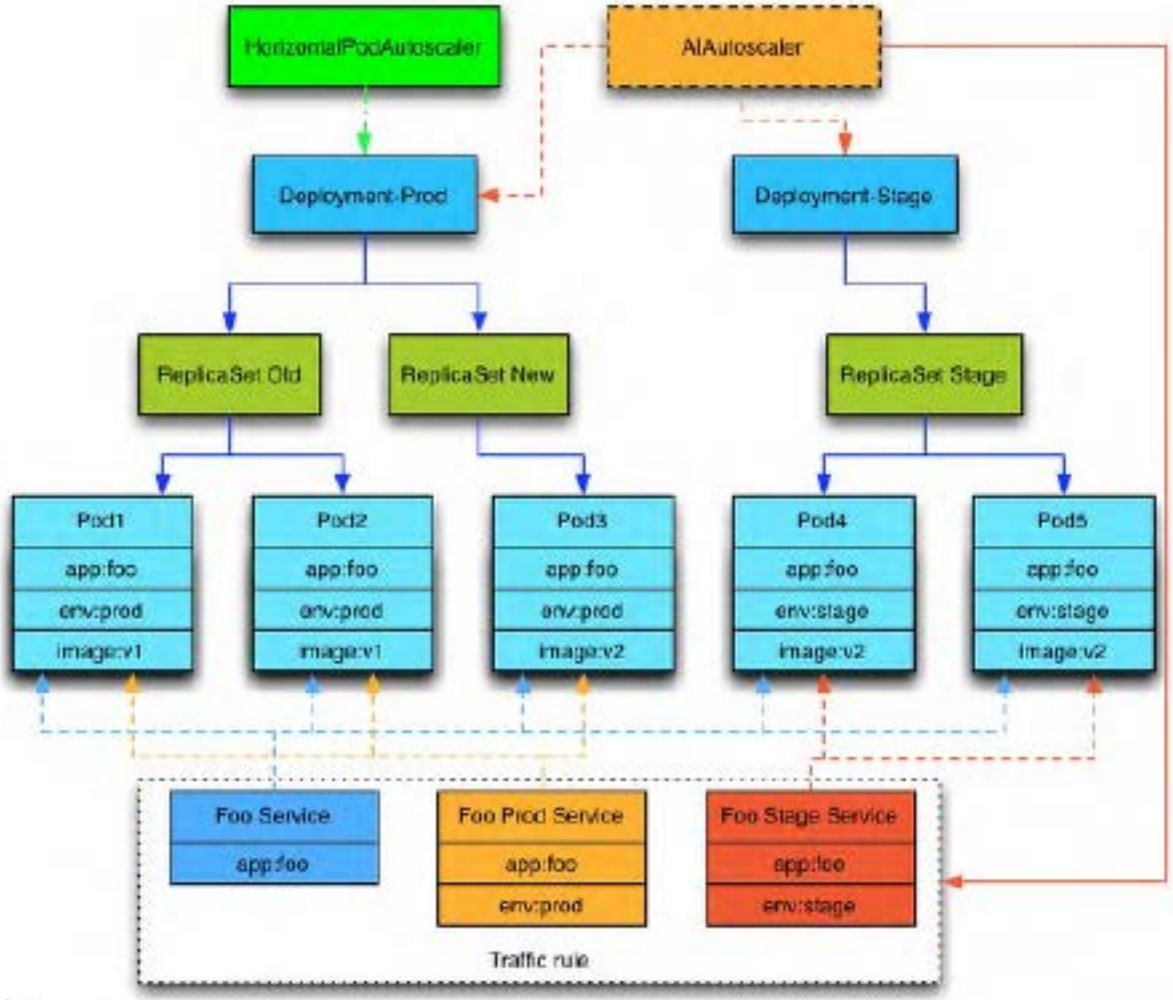
自动伸缩的困境

- 很难确定唯一的伸缩指标
- 多层服务需要联动伸缩
- 有的资源瓶颈无法靠伸缩解决

AI DevOps 构想： AIAutoscaler

```
apiVersion: archnotes.com/v1
kind: AIAutoscaler
metadata:
  name: autoscaler
spec:
  success: 999
  averageLatency: 200ms
  cost: x$/h
  selector:
    app: myapp
```

AI DevOps 构想： AIAutoscaler



基于 Kubernetes 构建 DevOps 平台的建议原则

- 将运行环境以及分布式功能委托给 Kubernetes
- 充分利用 Kubernetes 共享信息，自动化配置
- 自动填充 Kubernetes 元数据到 DevOps 工具输出结果中
- 通过声明式规范抽象 DevOps 中的组件以及配置，和具体的工具解耦
- 平台和服务都需要实现自反馈，自演进机制

总结

1. AI DevOps 的定义
2. AI 和 DevOps 的结合点
3. Kubernetes 的架构特点
4. Kubernetes DevOps 工具整合案例
5. 基于 Kubernetes 构建 DevOps 平台的建议原则