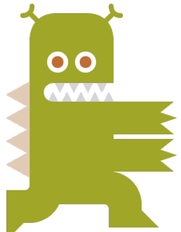
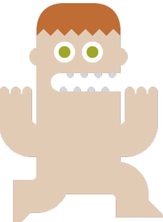
Разбираем автоскейлинг в Kubernetes

https://github.com/dragonsmith/kube-hpa-talk

Кир Кузнецов



EVIL MARTIANS



Что такое автоскейлинг и когда он появился

- Amazon Web Services
  - 2006 - публичный доступ к облаку
  - 2008 - сторонний автоскейлинг
  - *2009 - свой собственный автоскейлинг*
  - **2016 - контейнерный**
- Microsoft Azure
  - 2010 - публичный доступ к облаку
  - *2013 - свой собственный автоскейлинг*
- Google Cloud
  - 2012 - публичный доступ к облаку (Compute Engine)
  - *2014 - свой собственный автоскейлинг*

Почему автоскейлинг все ещё не у каждого?

- Высокая сложность перехода от обычного деплоя к инфраструктуре с автоскейлингом
- Привязка к поставщику/инструменту (Vendor/instrument lock-in)

Инструменты решения проблемы

1. *Публичные облака*
2. Контейнеризация
3. Оркестрация контейнеров

Enter managed Kubernetes

Managed Kubernetes

Cons:

- Да, K8s все еще сложно (и не всегда нужно) внедрять

Pros:

- Стандартизированное API
- Готовое решение, которое само *(но есть нюансы)* заботится о добавлении/удалении/обновлении нод кластера
- Архитектура по-умолчанию подразумевает, что ваше приложение “должно” скейлиться
- Предоставляет гибкие инструменты для автоскейлинга

Чтобы начать автоскейлиться, нужна метрика и “всего лишь YAML файл”

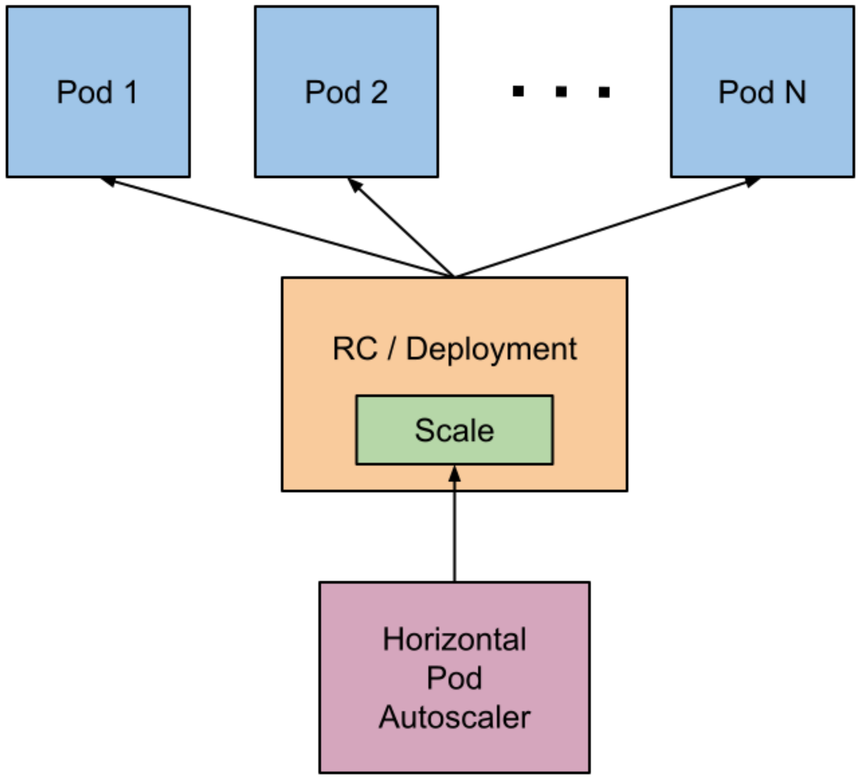
Мы подразумеваем, что у нас уже есть

- **Managed Kubernetes** в публичном облаке
- **metrics-server** для автоскейлинга по cpu/mem
  https://github.com/kubernetes-sigs/metrics-server
- **Prometheus** в качестве системы мониторинга

Kubernets HPA

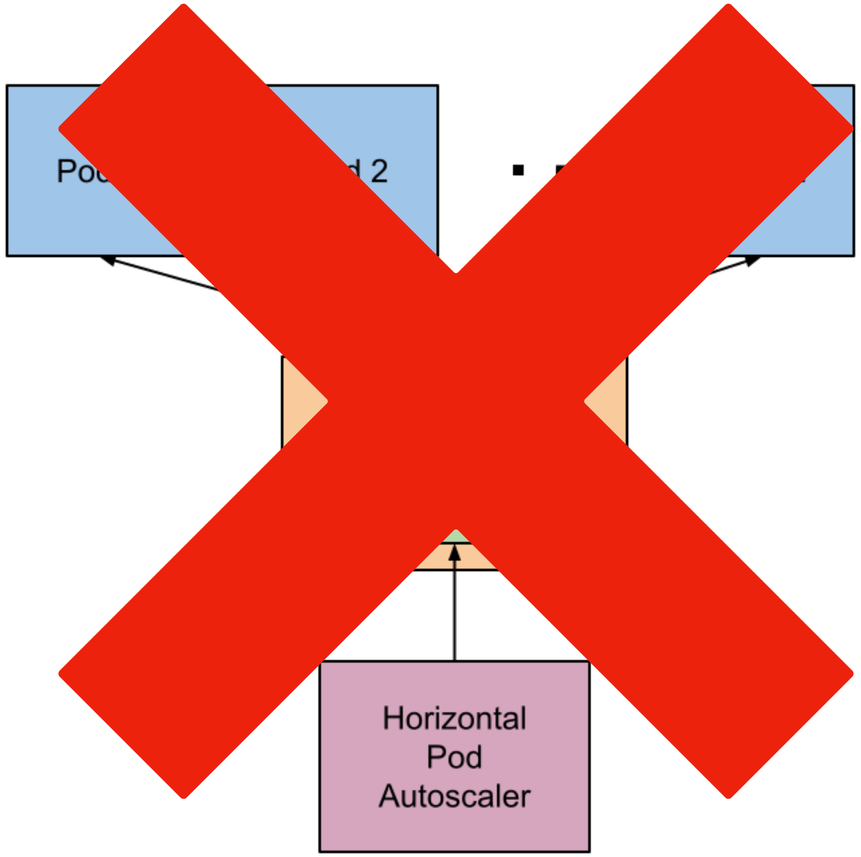
(Horizontal Pod Autoscaler)

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

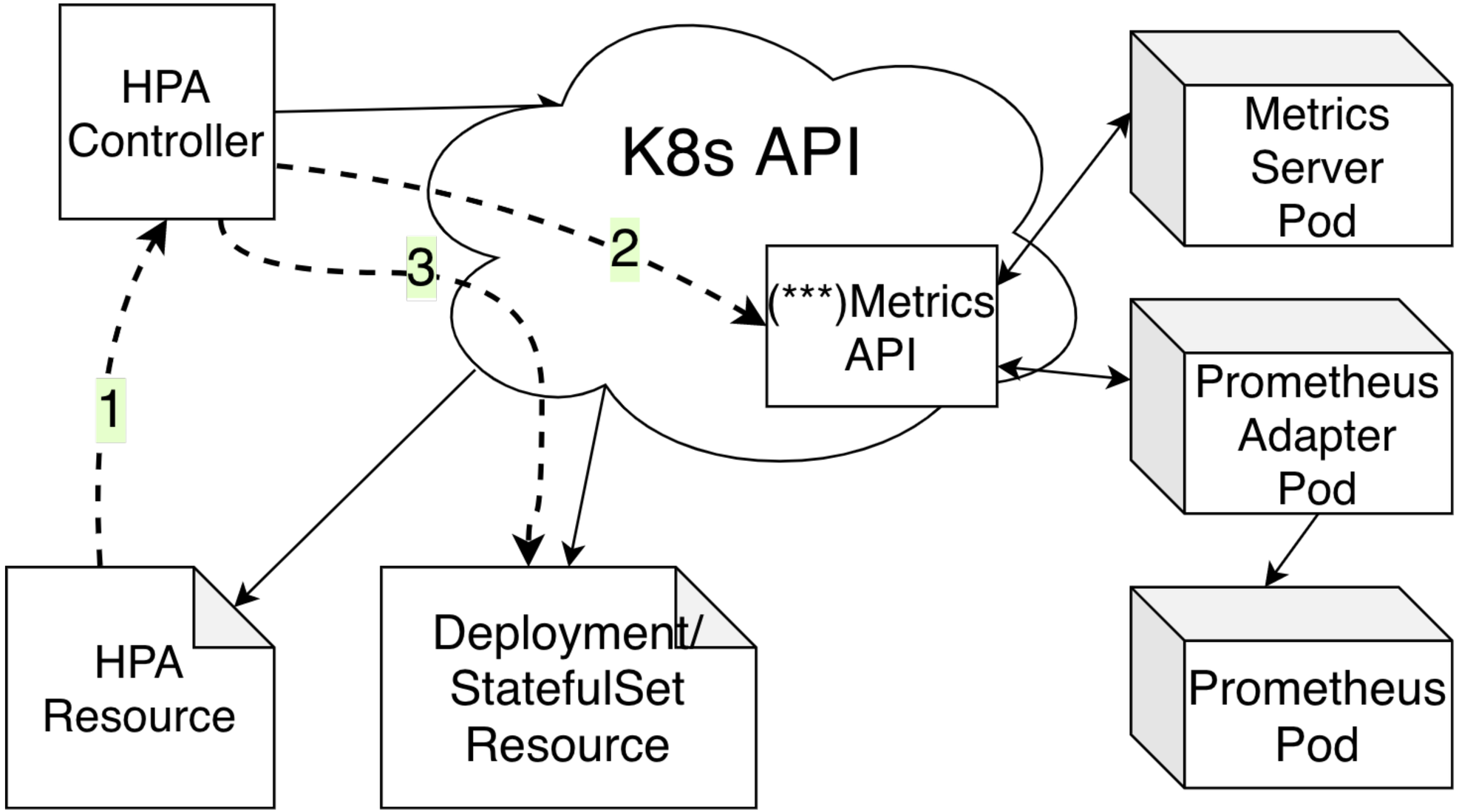


Kubernets HPA

(Horizontal Pod Autoscaler)

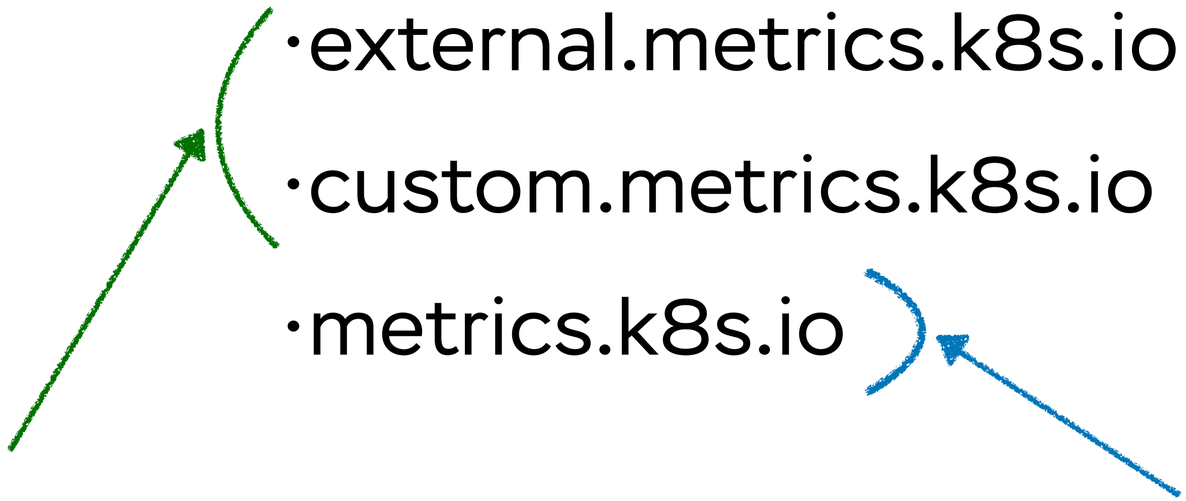


Kubernets HPA



Kubernets Metrics APIs

https://github.com/kubernetes/metrics/tree/master/pkg/apis



Любой сторонний адаптер

(https://github.com/kubernetes-sigs/custom-metrics-apiserver)

может зарегистрировать эти метрики

e.g. Prometheus Adapter

(https://github.com/DirectXMan12/k8s-prometheus-adapter)

Обычно предоставляется metrics-server

https://github.com/kubernetes-sigs/metrics-server

HPA Resource

“Resource”-based metric

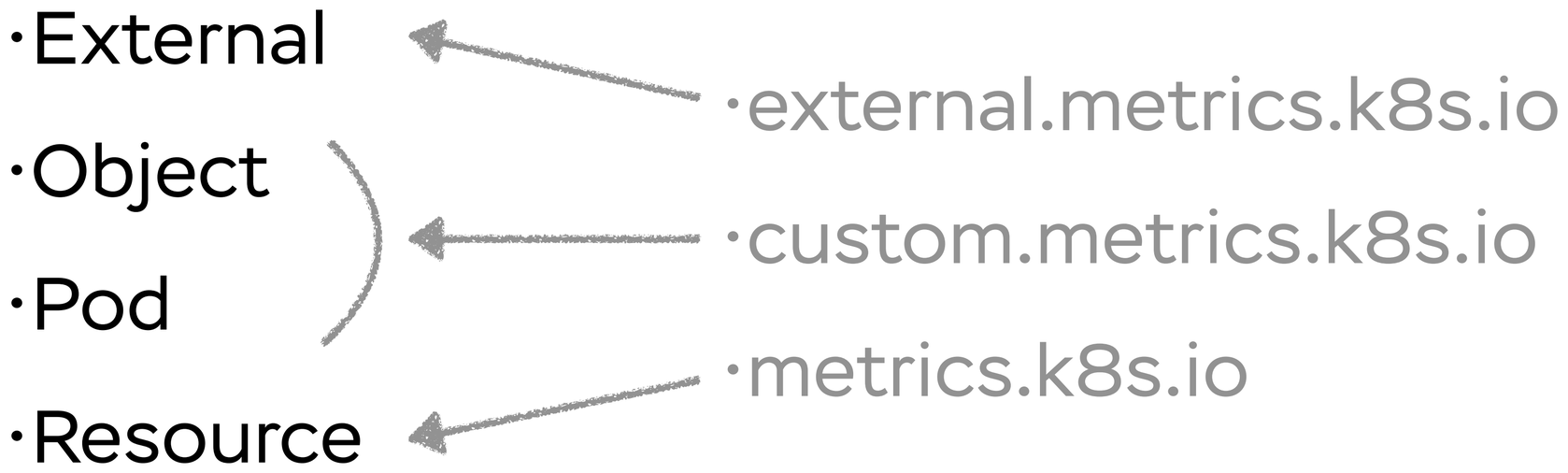
Что именно мы будем автоскейлить

Метрики, на основе которых мы будем это делать

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Виды HPA метрик

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/#metricspec-v2beta2-autoscaling



Примеры метрик

- External — Длина очереди задач
- Object — Количество запросов в секунду
9Xth latency percentile
- Pod — Объем передаваемых сетевых данных
- Resource — Потребление подом CPU/Mem

Пример

Допустим у нас есть простое Web приложение
с названием “php-apache”

(*Apache + PHP + какой-то простой скрипт*):

- Deployment
- Service
- Ingress

Requests per second:

```
sum(
  irate(
    nginx_ingress_controller_requests{ingress=‘php-apache’}[2m]
  )
) by (ingress)
```

Добавляем метрику в API K8s

Нам потребуется Prometheus Adapter

values/prometheus-adapter.yaml

```yaml
logLevel: 6
prometheus:
  url: http://prometheus-operated.monitoring.svc
rules:
  custom:
    - seriesQuery: '{__name__="nginx_ingress_controller_requests"}'
      resources:
        overrides:
          exported_namespace:
            resource: "namespace"
          ingress:
            group: networking.k8s.io
            resource: ingress
      name:
        matches: "^(.*)"
        as: "requests-per-second"
      metricsQuery: 'sum(irate(nginx_ingress_controller_requests{<<.LabelMatchers>>}
        [2m])) by (<<.GroupBy>>)'

```

cli:

```
helm install prometheus-adapter \
  stable/prometheus-adapter \
  -n monitoring \
  -f values/prometheus-adapter.yaml
```

Prometheus Adapter config

https://github.com/DirectXMan12/k8s-prometheus-adapter/blob/master/docs/config.md

Выбираем тип API
(custom или external)

Динамический поиск метрик
(в нашем случае ищем одну конкретную)

Для каких ресурсов K8s эта метрика будет доступна через Custom Metrics API
(в нашем случае только для ingress и namespace)

Меняем имя метрики
(нам так хочется)

Реальный запрос в Prometheus
(получаем значения в том виде, который нам нужен)

6-ой уровень отображает реальные запросы в Prometheus
(потребуется для дебага)

```yaml
logLevel: 6
prometheus:
  url: http://prometheus-operated.monitoring.svc
rules:
  custom:
  - seriesQuery: '{__name__="nginx_ingress_controller_requests"}'
    resources:
      overrides:
        exported_namespace:
          resource: "namespace"
        ingress:
          group: networking.k8s.io
          resource: ingress
    name:
      matches: "^(.*)"
      as: "requests-per-second"
    metricsQuery: 'sum(irate(nginx_ingress_controller_requests{<<.LabelMatchers>>}
      [2m])) by (<<.GroupBy>>)'
```

Явно указывает к какому API относится объект
(API меняется от версии к версии и один и тот же объект может быть доступен через разные куски API)

HPA Resource

“Object”-based metric

То самое имя метрики,
которое мы выбрали выше

Метрику какого объекта,
будем получать
(мы управляем деплойментом,
но на основе метрики от Ingress Controller)

Значение метрики надо усреднить
по количеству запущенных подов

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Object
    object:
      metric:
        name: requests-per-second
      describedObject:
        apiVersion: extensions/v1beta1
        kind: Ingress
        name: php-apache
      target:
        type: AverageValue
        averageValue: 0.1
```

Small debug cheatsheet

```
kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1" | jq
```

```
kubectl get --raw "/apis/external.metrics.k8s.io/v1beta1" | jq
```

```
kubectl get --raw "<FULLMETRICURL>" | jq
```

Конкретный URL можно посмотреть в логах Prometheus Adaptor

```
helm upgrade prometheus-adapter stable/prometheus-adapter -n monitoring --reuse-values \
  --set logLevel=6
```

Чтобы увидеть запросы к Prometheus не забудьте повысить уровень логгирования

Metrics API design docs

Слайд для тех, кто после митапа захочет копнуть глубже

custom-metrics-api.md

external-metrics-api.md

resource-metrics-api.md

Байка из жизни

О том, как мы сломали автоскейлинг из-за большого усердия

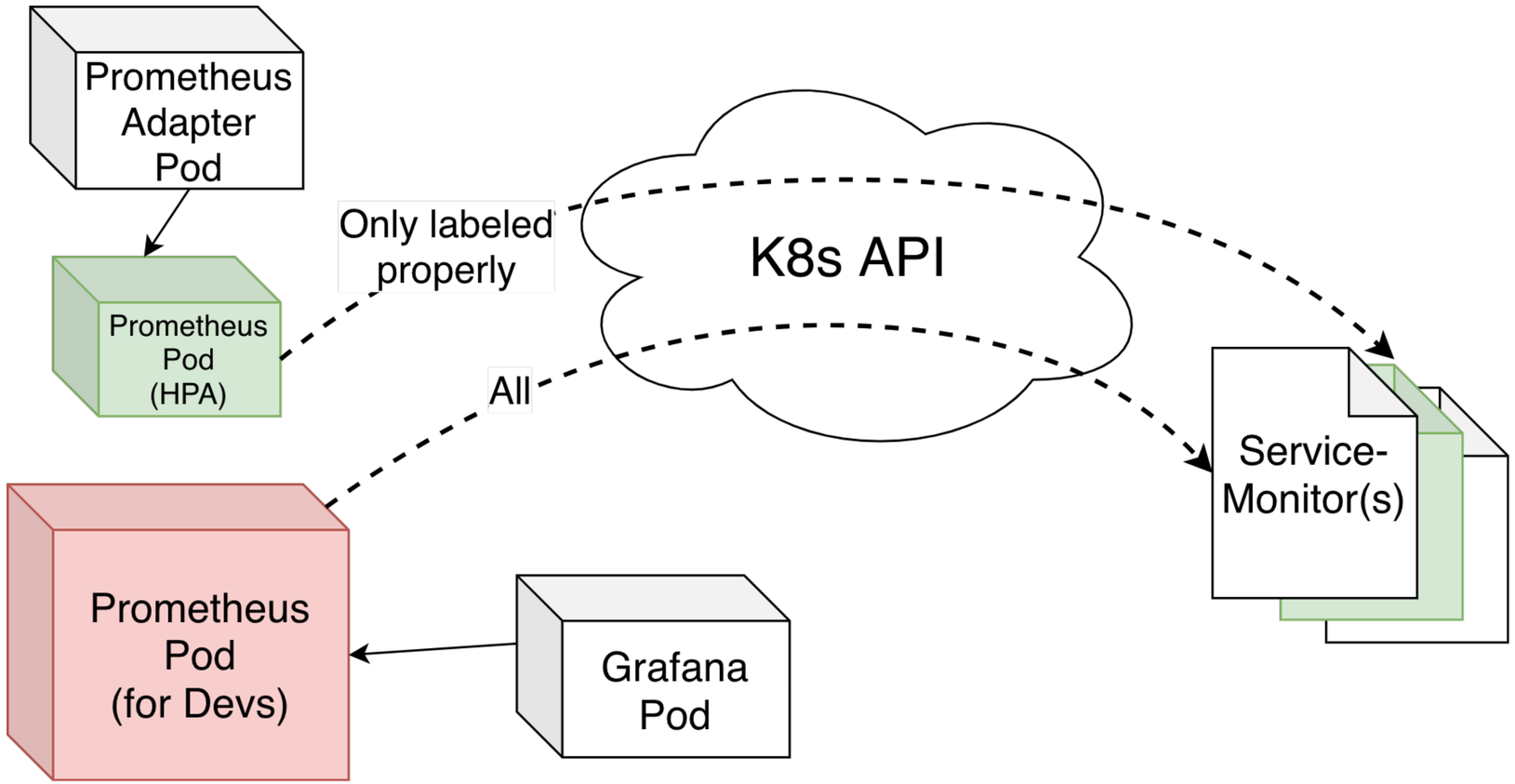
Too...


Instrumentation
Last 6 hours
10s
Number of
received/dropped
notifications received by type
resuests by status
Rails
Throughput
Rails average response time
Throughput / Endpoint
Rails 95% response time by endpoint
Sidekiq
No data

much...


ОЙ, ВСЁ

Делайте отказоустойчивый Prometheus
или
выделяйте отдельный



СПАСИБО!

https://github.com/dragonsmith/kube-hpa-talk

Кир Кузнецов

evilmartians.com/blog
@dragonsmith
@agonsmith
@evilmartians