Мониторинг облачной инфраструктуры

Михаил Прокопчук, Avito, Москва.

Avito

Сайт №1 объявлений в России

35 миллионов пользователей ежедневно

300+ серверов

15000rps к бэкенду

План доклада

Облако в Avito

Архитектура системы мониторинга

Что мониторить в облаке

Немного про SLI/SLO

Облако в Avito

В основе - kubernetes

4 кластера: 2 x production, dev, staging

~50 нод в каждом

~2000 pods в dev

~100 релизов в день в prod (разработчики катят сами)

Orchestration: Kubernetes, Helm

CI/CD: TeamCity

Ресурсы: docker-, helm-registry etc.

Storage: *Ceph (WIP)*

Тулинг: k8s-linter, бот для создания сервисов etc.

Разделение

Сервера/Монолит - System Infrastructure

Облако - Architecture

Сервисы (включая deploy/rollback) - Разработка

Архитектура системы мониторинга

Два стека мониторинга:

- Prometheus

- Graphite/ClickHouse

Два стека мониторинга:

- Prometheus: инфраструктура облака (kubernetes etc.)

- Graphite/ClickHouse: метрики сервисов, инфраструктура монолита, доступность нод

(habr.com/company/avito/blog/343928/)

Prometheus - инструмент оперативного контроля

Преимущества:

- **Поддержка k8s-discovery, интеграция с экосистемой k8s: очень круто**

- Pull-model

- Расширяемость (через exporters)

Недостатки*:

- Нет HA

- Не про долгосрочное хранение данных

Про HA:

github.com/prometheus/prometheus/issues/1500

https://prometheus.io/docs/introduction/overview/#when-does-it-not-fit

This is not a feature we plan to add. Prometheus is intended for your critical monitoring, and having a full-mesh of communication between servers opens risks around cascading failures and all the complexities of having to synchronise metadata across servers.

If you need 100% accuracy, such as for per-request billing, Prometheus is not a good choice as the collected data will likely not be detailed and complete enough. In such a case you would be best off using some other system to collect and analyze the data for billing, and Prometheus for the rest of your monitoring.

Для долгосрочного хранения используйте remote storage adapters

github.com/prometheus/prometheus/tree/master/documentation/examples/remote_storage/remote_storage_adapter

Graphite storage adapter

Передает ВСЕ метрики в graphite

Нет возможности фильтрации множества экспортируемых метрик

Дописали фильтрацию метрик, запустили в тестовом режиме

```
apiVersion: v1
kind: ConfigMap
data:
  filters.yaml: |-
    graphite:
      filter:
        include:
          - metric: "^metric_name_"
```

Не используем prometheus-operator

Prometheus запущен в каждом из kubernetes-кластеров как deployment

Используем:
- cross-monitoring между экземплярами (blackbox-проверки)

- prometheus federation (hierarchical) для некоторых метрик (пока не внедрили Graphite адаптер)

Используем prometheus exporter'ы

Exporter - предоставляет метрики от приложения в формате prometheus

prometheus.io/docs/instrumenting/exporters/

Используем:

BlackBox exporter

Node exporter

Teamcity exporter

Ceph exporter

ElasticSearch exporter

Про эксплуатацию prometheus

"Стоимость"

CPU и RAM

1. CPU

```
sum
  by(pod_name, namespace)
(rate(container_cpu_usage_seconds_total{job="kubelet"}[1m]))
```

Query time (dev-кластер): ~5s

и именно cpu-time

Решение: recording rules

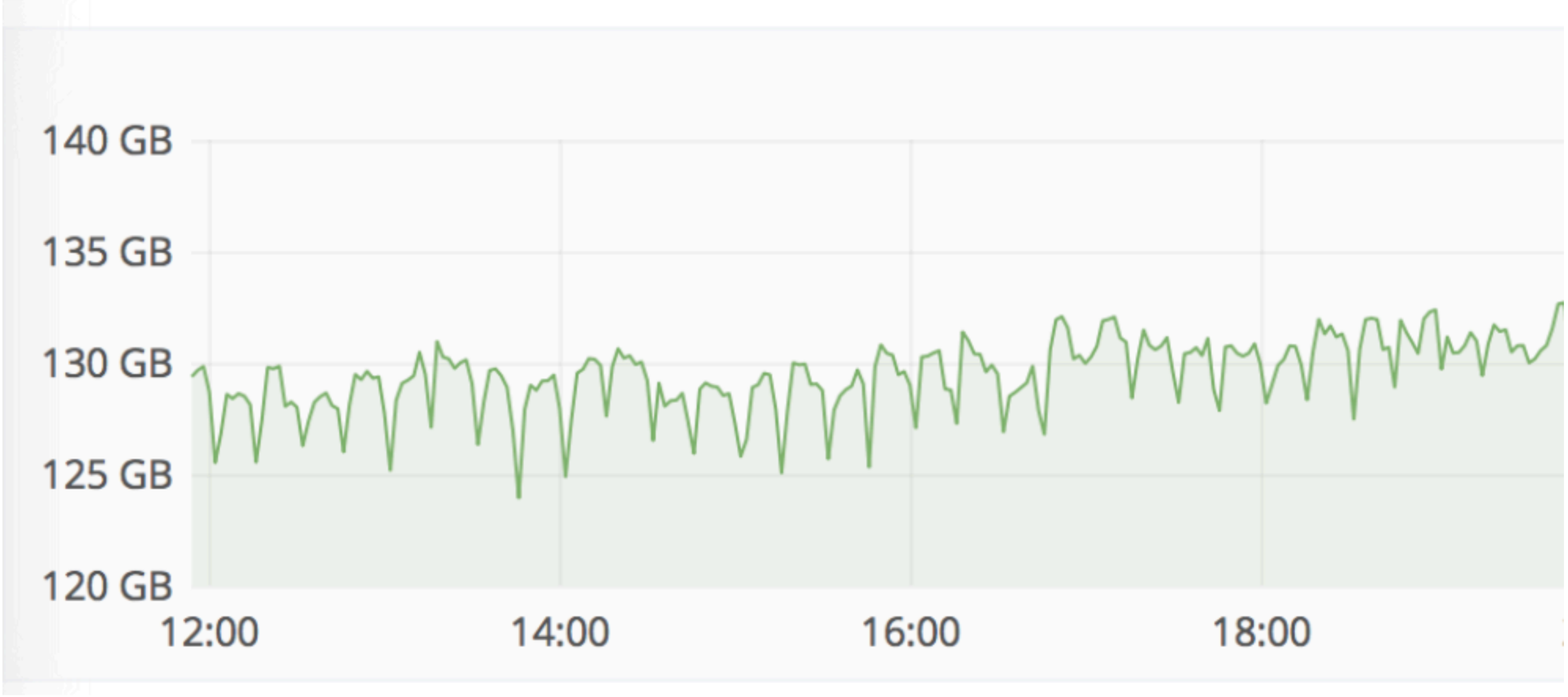
Предкомпилированные запросы, по которым создаются новые метрики

Нужно делать для всех (ну почти) метрик для графиков в Grafana'e

2. RAM: RSS

Dev-кластер: 70 nodes, ~2000 pod'ов

Data retention 24h



RSS memory

Почему так много?

Большое число pod'ов + частые перевыкатки (QA-automation etc.)

По каждому pod'у собирается ряд метрик (RAM/CPU usage etc.)

Для dev-кластера выставили retention 2h

130Gb (было) -> 9Gb

UPD (2018-05-08): и всё-таки memory leak
в prometheus kubernetes-discovery

github.com/prometheus/prometheus/issues/3685

github.com/prometheus/prometheus/issues/4095

github.com/prometheus/prometheus/pull/4117

UPD (2018-05-08): выкатили 2.2.1-release

Работает стабильно

Решения:

- Снижать retention и/или увеличивать scrape-interval и использовать remote storage adapters

- Снижать число метрик с помощью фильтрации (оочень аккуратно: *нужно точно знать, какие не потребуются!*)

3. RAM: VSZ

Prometheus database - TSDB

Написана с нуля специально для профиля нагрузки Prometheus

fabxc.org/tsdb/

mmap-based

mmap-based

Для высокой производительности prometheus необходимо иметь на ноде достаточно свободной памяти под страницы pagecache

Prometheus в dev-кластере: немного цифр

Число метрик: ~ 1 200 000

Размер датасета: ~5 Gb

Data retention: 2h

Prometheus в dev-кластере: немного цифр

CPU usage: ~ 300%

Memory usage (RSS): ~ 9Gb

Memory usage (VSZ): ~ 12Gb

Prometheus - это здорово, но не "бесплатно"

Оценивайте ресурсы

Алертинг

Kubernetes/облако: Grafana (slack)

Бизнес-метрики/монолит/доступность: Moira (slack/sms)

Не используем prometheus alertmanager...

Внедряем интеграцию с имеющимся стеком:

prometheus ->
graphite storage adatpter (+metrics filtering) ->
moira ->
slack/**sms**

Что мониторить в облаке

Ключевые метрики k8s-инфраструктуры:

- Ёмкость кластера

- Утилизация ресурсов кластера (Network/RAM/CPU)

Ключевые метрики k8s-инфраструктуры:

- Nodes ready

- Kube-apiserver rps
- Kube-spiserver latency

- Pods per node
- Pods not in running state

Ключевые метрики pods/containers:

- Requests/Limits (k8s)

- Usage

Blackbox-проверки:

- HTTP code

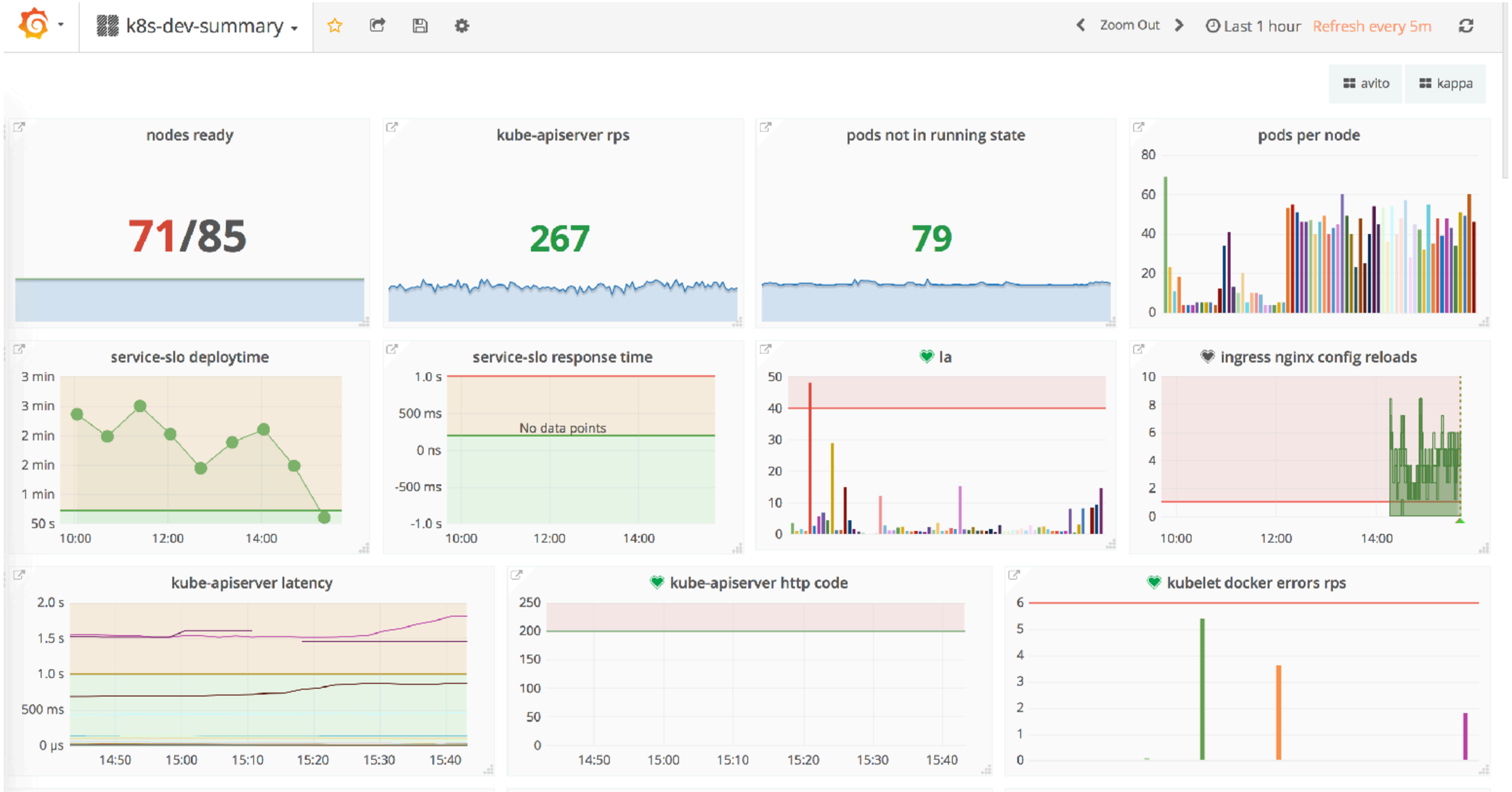
- Latency

TODO: комплексные проверки ресурсов (docker pull, go get etc.)

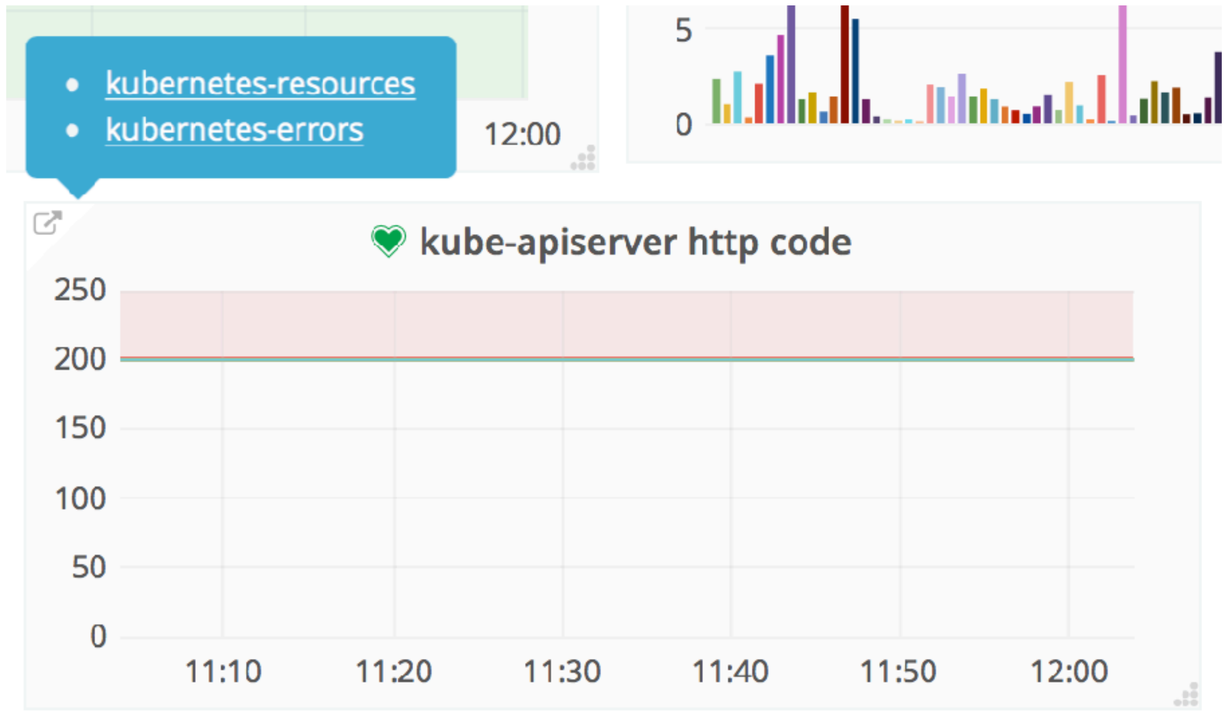
Все эти ключевые метрики - на summary-dashboards

Из них - ссылки на связанные dashboards

Как выглядит:


k8s-dev-summary
Zoom Out
Last 1 hour
Refresh every 5m
avito
kappa
nodes ready
71/85
kube-apiserver rps
267
pods not in running state
79
pods per node
80
60
40
20
0
service-slo deploytime
3 min
3 min
2 min
2 min
1 min
50 s
10:00
12:00
14:00
service-slo response time
1.0 s
500 ms
No data points
0 ns
-500 ms
-1.0 s
10:00
12:00
14:00
la
50
40
30
20
10
0
ingress nginx config reloads
10
8
6
4
2
0
10:00
12:00
14:00
kube-apiserver latency
2.0 s
1.5 s
1.0 s
500 ms
0 µs
14:50
15:00
15:10
15:20
15:30
15:40
kube-apiserver http code
250
200
150
100
50
0
14:50
15:00
15:10
15:20
15:30
15:40
kubelet docker errors rps
6
5
4
3
2
1
0

Связанные dashboards:



Немного про SLI/SLO

Что это такое?

Что это такое?

SLI - Service Level Indicator

SLO - Service Level Objective

(SLA - Service-Level Agreement)

cloudplatform.googleblog.com/2017/01/availability-part-deux--CRE-life-lessons.html

У нас пока WIP

SLI - критерий: 95p latency <= 100ms

SLO - частота: SLI выполняется 99.9%
(те самые девятки)

Также позволяет определять "бюджет":

99% ~ 14m/day

99.9% ~ 90s/day

99.99% ~ 4min/month

Лучше больше метрик чем меньше

Определяйте ключевые/высокоуровневые метрики

Стройте summary-dashboards

Определяйте и замеряйте SLI/SLO

Полезные ссылки

Prometheus HA:
github.com/prometheus/prometheus/issues/1500

Alerting:
My Philosophy of Alerting

SLI/SLO:
landing.google.com/sre/book/chapters/service-level-objectives.html

Спасибо за внимание

Вопросы?