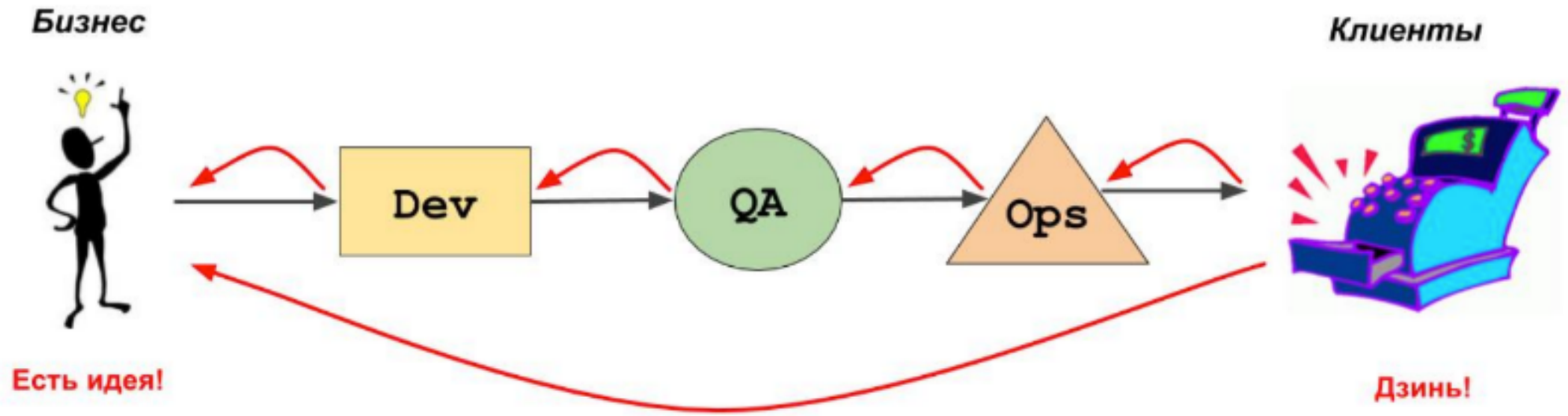
Kubernetes. Мониторинг и логирование

План

- Мониторинг Kubernetes
- Логирование в Kubernetes

Второй путь DevOps



Мониторинг

Что отслеживать?

Что отслеживать?

- Работоспособность приложений

Что отслеживать?

- Работоспособность приложений
- Метрики host-систем

Что отслеживать?

- Работоспособность приложений
- Метрики host-систем
- Метрики POD’ов и Container’ов

Что отслеживать?

- Работоспособность приложений
- Метрики host-систем
- Метрики POD’ов и Container’ов
- Метрики приложений

Что отслеживать?

- Работоспособность приложений
- Метрики host-систем
- Метрики POD’ов и Container’ов
- Метрики приложений
- Метрики и работоспособность самого Kubernetes

Что у k8s есть?

- Probes
- cAdvisor
- Heapster
- Kubernetes Dashboard
- kube-state-metrics

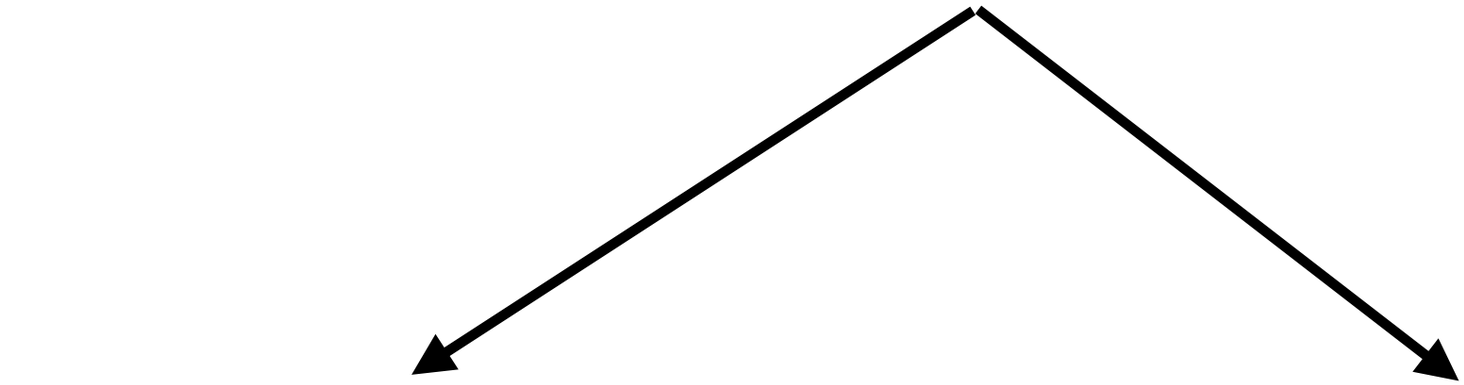
Probes

Probes - периодические проверки pod’а на жизнеспособность

Как узнать, что сервис “жив” и готов к работе?

- **ExecAction** - выполнить команду и ждать exit code 0
- **TCPSocketAction** - проверить, что TCP-порт открыт
- **HTTPGetAction** - отправить HTTP GET-запрос

Probes



Liveness Probes

Проверяет, что приложение запущено и “живо”

Readiness Probes

Проверяет, что приложение готово обслуживать запросы

Не совместимы с Docker HEALTHCHECK

Liveness Probe

Проверяет, что приложение запущено и “**живо**”

Если это не так, то POD будет перезапущен.

Liveness Probe

Проверяет, что приложение запущено и “**живо**”

Если это не так, то POD будет перезапущен.

```
containers:
- name: container
  livenessProbe:
      httpGet:
        path: /health
        port: 8005
      initialDelaySeconds: 180
      timeoutSeconds: 15
```

Liveness Probe

Проверяет, что приложение запущено и “**живо**”

Если это не так, то POD будет перезапущен.

```
containers:
- name: container
  livenessProbe:
    httpGet:
      path: /health
      port: 8005
    initialDelaySeconds: 180
    timeoutSeconds: 15
```

ИЛИ

```
  livenessProbe:
    tcpSocket:
      port: 8005
    initialDelaySeconds: 15
    periodSeconds: 20
```

Liveness Probe

Проверяет, что приложение запущено и “**живо**”

Если это не так, то POD будет перезапущен.

```
containers:
- name: container
  livenessProbe:
      httpGet:
        path: /health
        port: 8005
      initialDelaySeconds: 180
      timeoutSeconds: 15
```

ИЛИ

```
  livenessProbe:
      tcpSocket:
        port: 8005
      initialDelaySeconds: 15
      periodSeconds: 20
```

curl нам больше не нужен =)

Readiness Probe

Проверяет, что приложение готово обслуживать запросы

Если это не так, то POD **удален из всех Service'ов** как Endpoint

Readiness Probe

Проверяет, что приложение готово обслуживать запросы

Если это не так, то POD **удален из всех Service'ов** как Endpoint

```
containers:
- name: container
  readinessProbe:
    httpGet:
      path: /health
      port: 8005
    initialDelaySeconds: 10
    timeoutSeconds: 15
```

Readiness Probe

Проверяет, что приложение готово обслуживать запросы

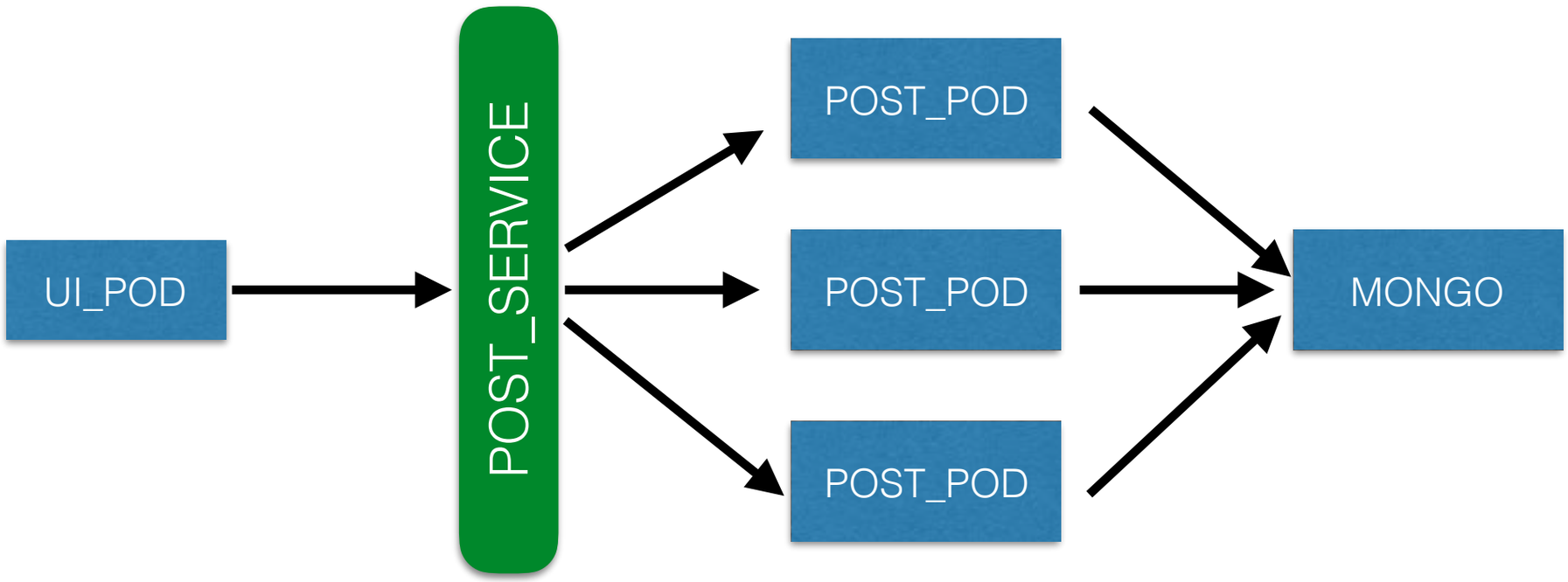
Если это не так, то POD **удален из всех Service'ов** как Endpoint

```
containers:
- name: container
  readinessProbe:
    httpGet:
      path: /health
      port: 8005
    initialDelaySeconds: 10
    timeoutSeconds: 15
```

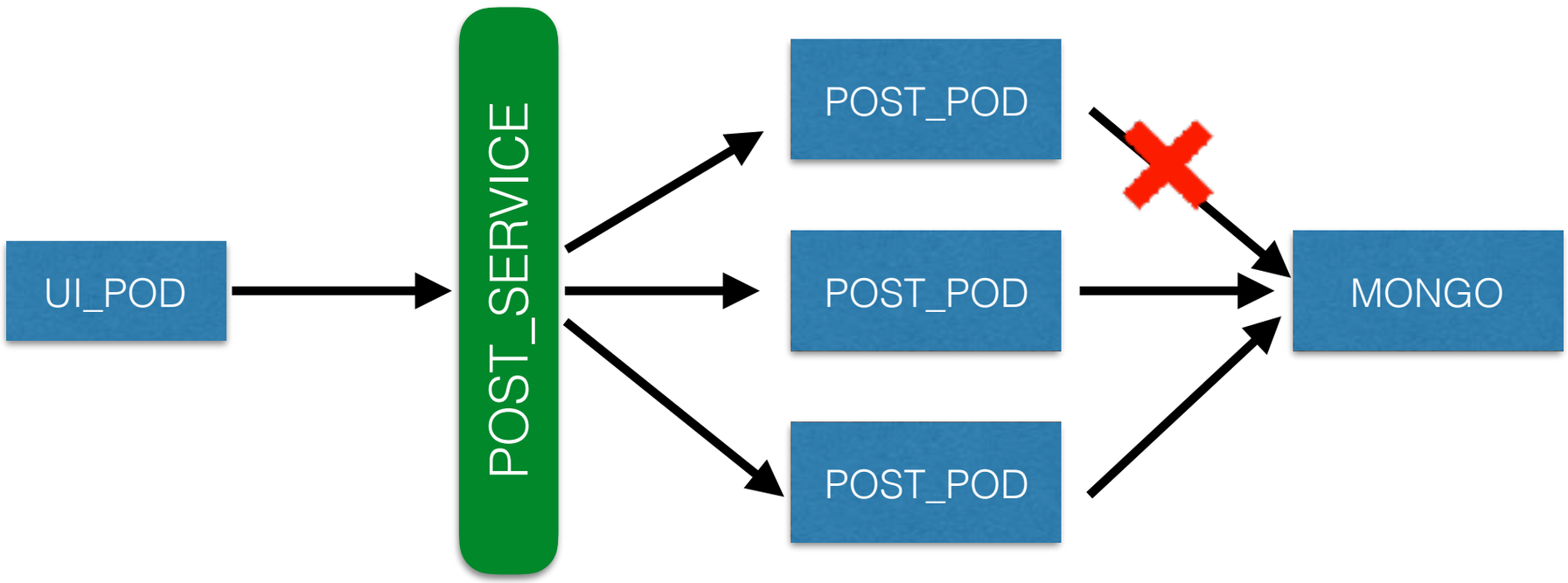
ИЛИ

```
  readinessProbe:
    tcpSocket:
      port: 8005
    initialDelaySeconds: 10
    periodSeconds: 15
```

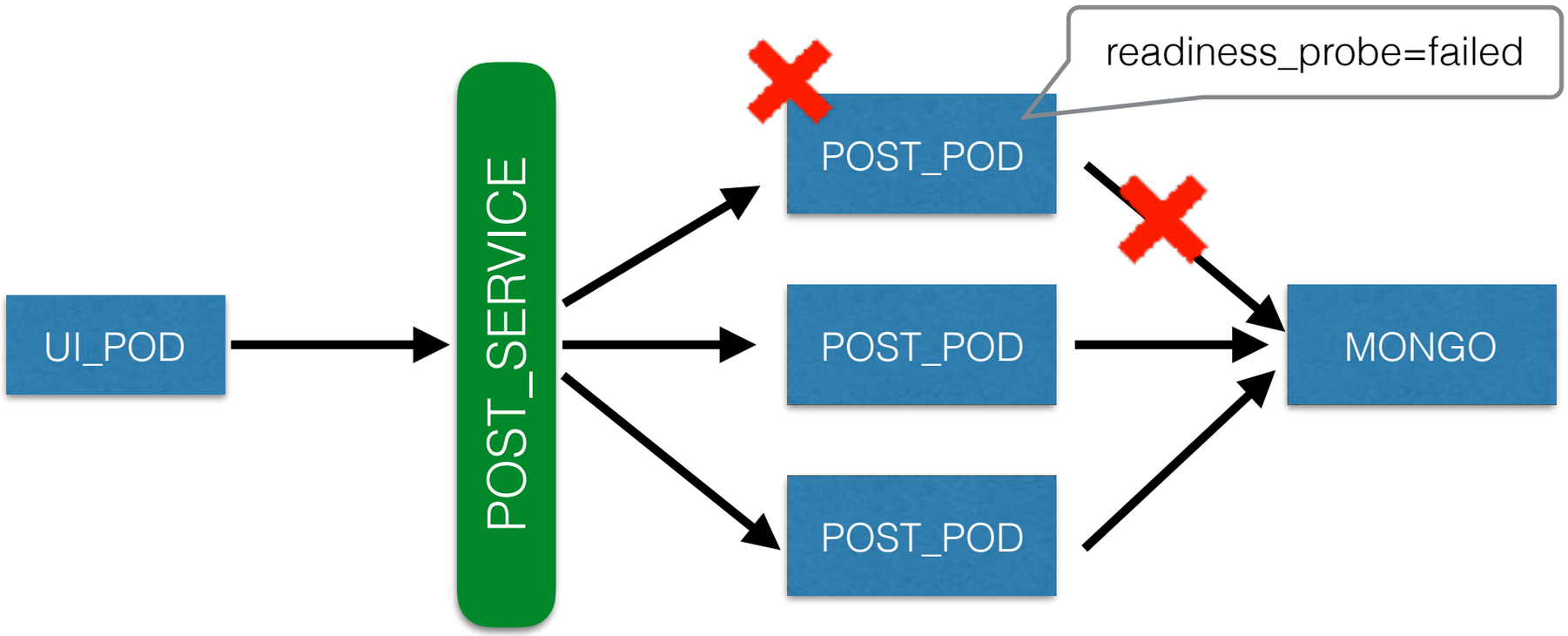
Readiness Probe



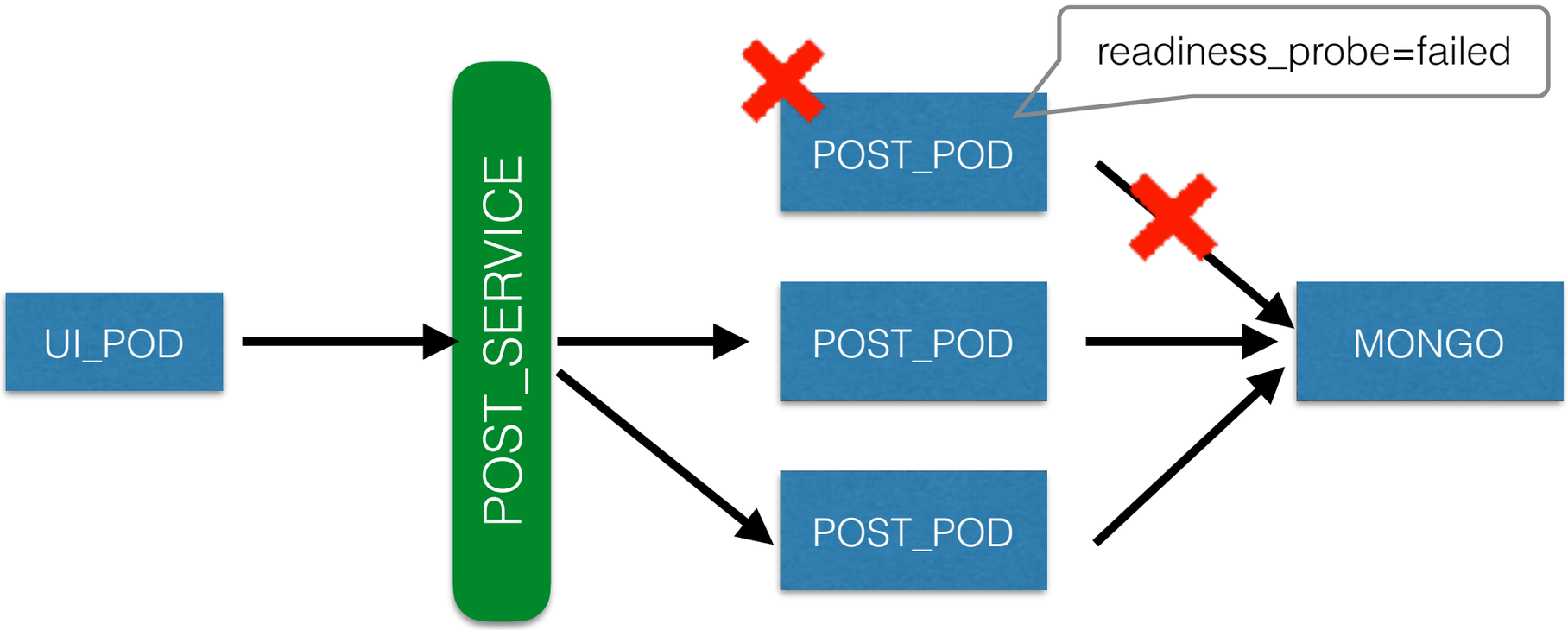
Readiness Probe



Readiness Probe



Readiness Probe



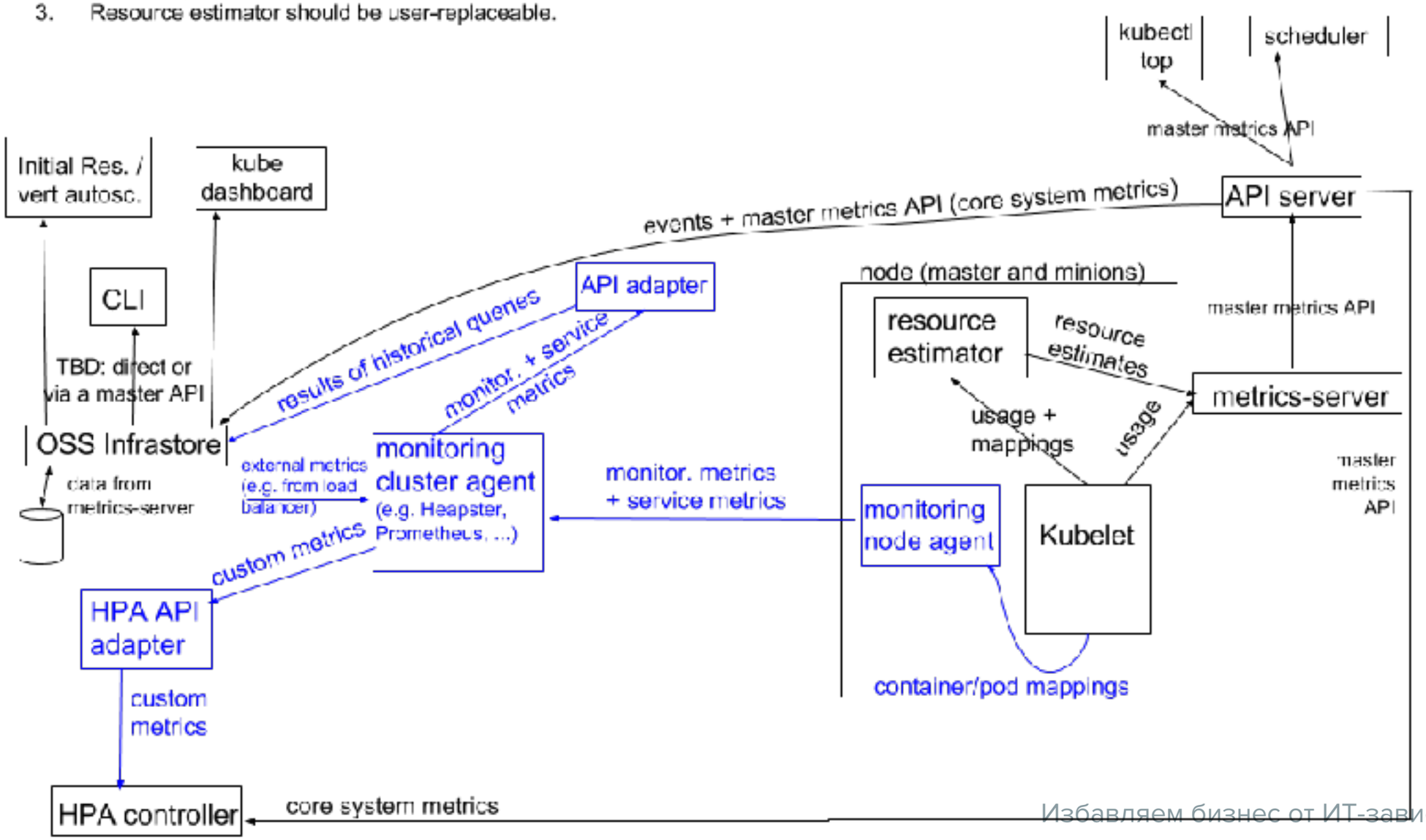
Monitoring Pipeline

Monitoring architecture proposal: OSS

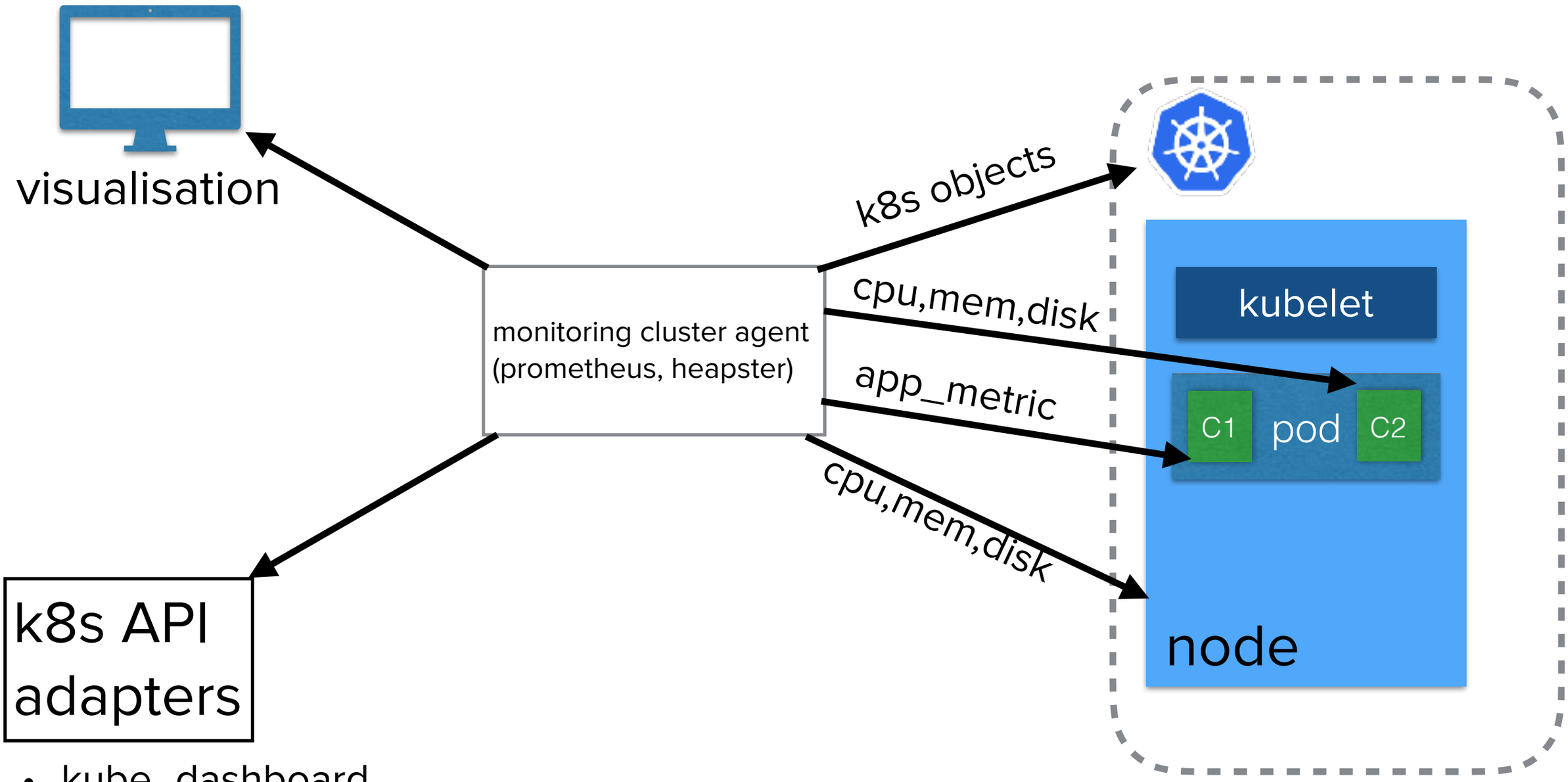
(arrows show direction of metrics flow)

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.



Monitoring Pipeline



k8s API adapters

- kube_dashboard
- Horizontal Pod Autoscaling

Monitoring Pipeline

Различные комбинации

- cAdvisor + collectd + Heapster
- cAdvisor + Prometheus
- cAdvisor + Heapster + InfluxDB
- ...

Metrics

- Метрики хостовых систем
- Метрики контейнеров
- Метрики состояния объектов k8s
- Метрики приложений и сервисов
- Метрики инфраструктуры k8s

Cadvisor

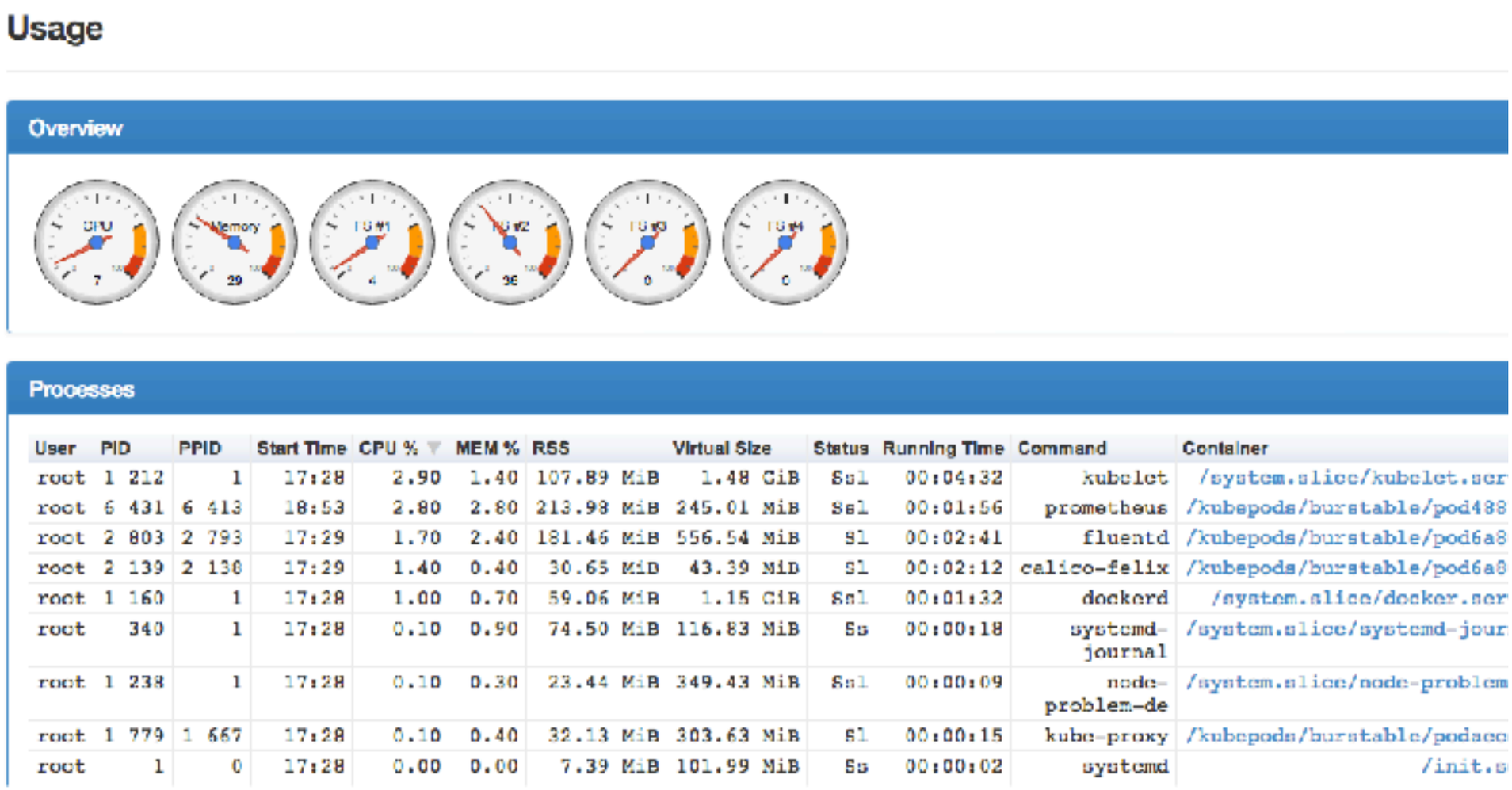
- CPU
- Memory
- Network
- Block I/O
- + Docker daemon

Cadvisor

- Встроен в kubelet
- Получает информацию о контейнерах от Docker
- Предоставляет только текущее состояние использования ресурсов и метрики производительности
- Работает как Exporter для Prometheus

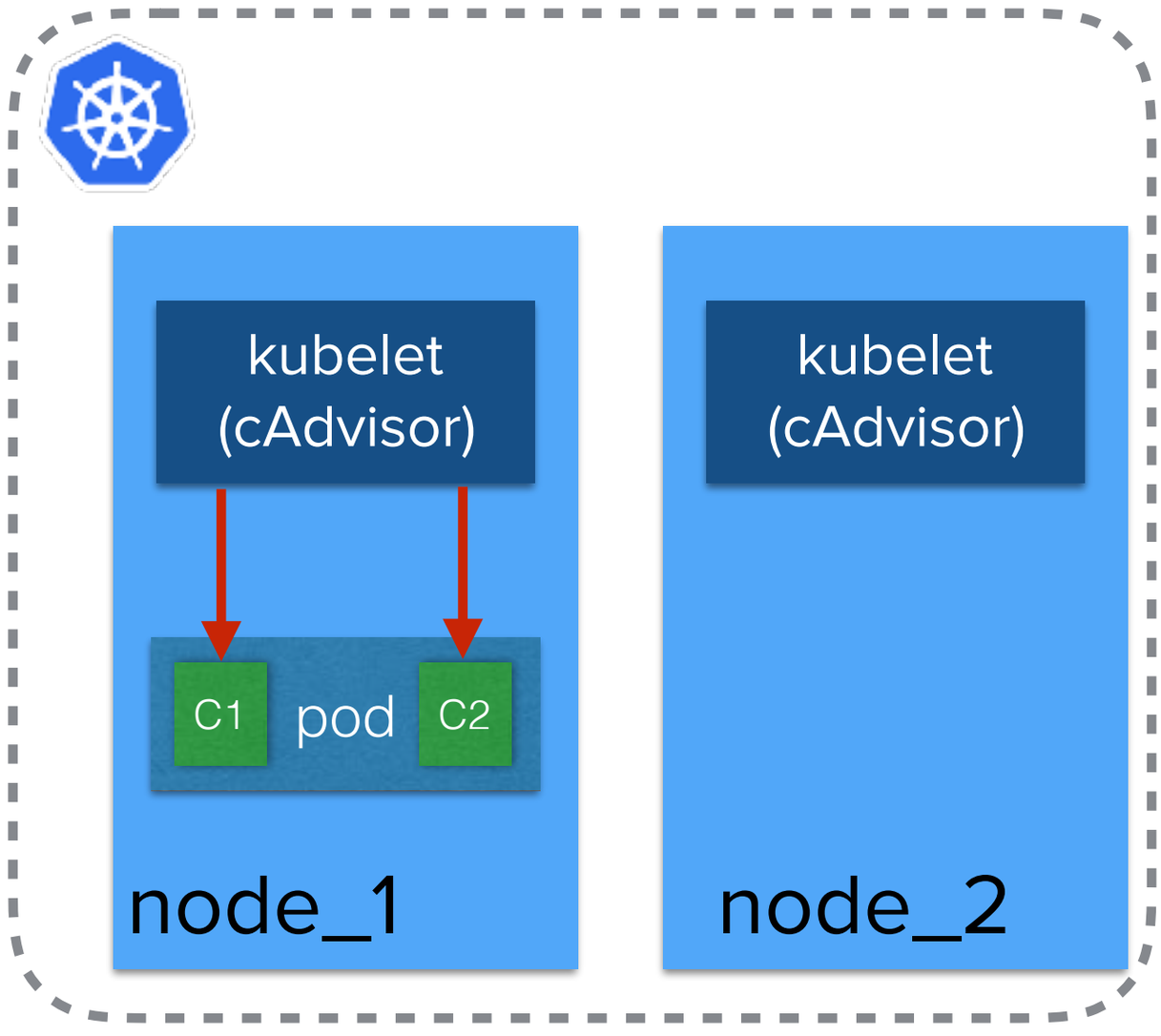
Cadvisor

http://node-ip:4194



cAdvisor

cAdvisor собирает у Docker метрики контейнеров



Heapster

Heapster

- Аддон для Kubernetes

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)

Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)
- Используется в качестве API для других частей k8s:

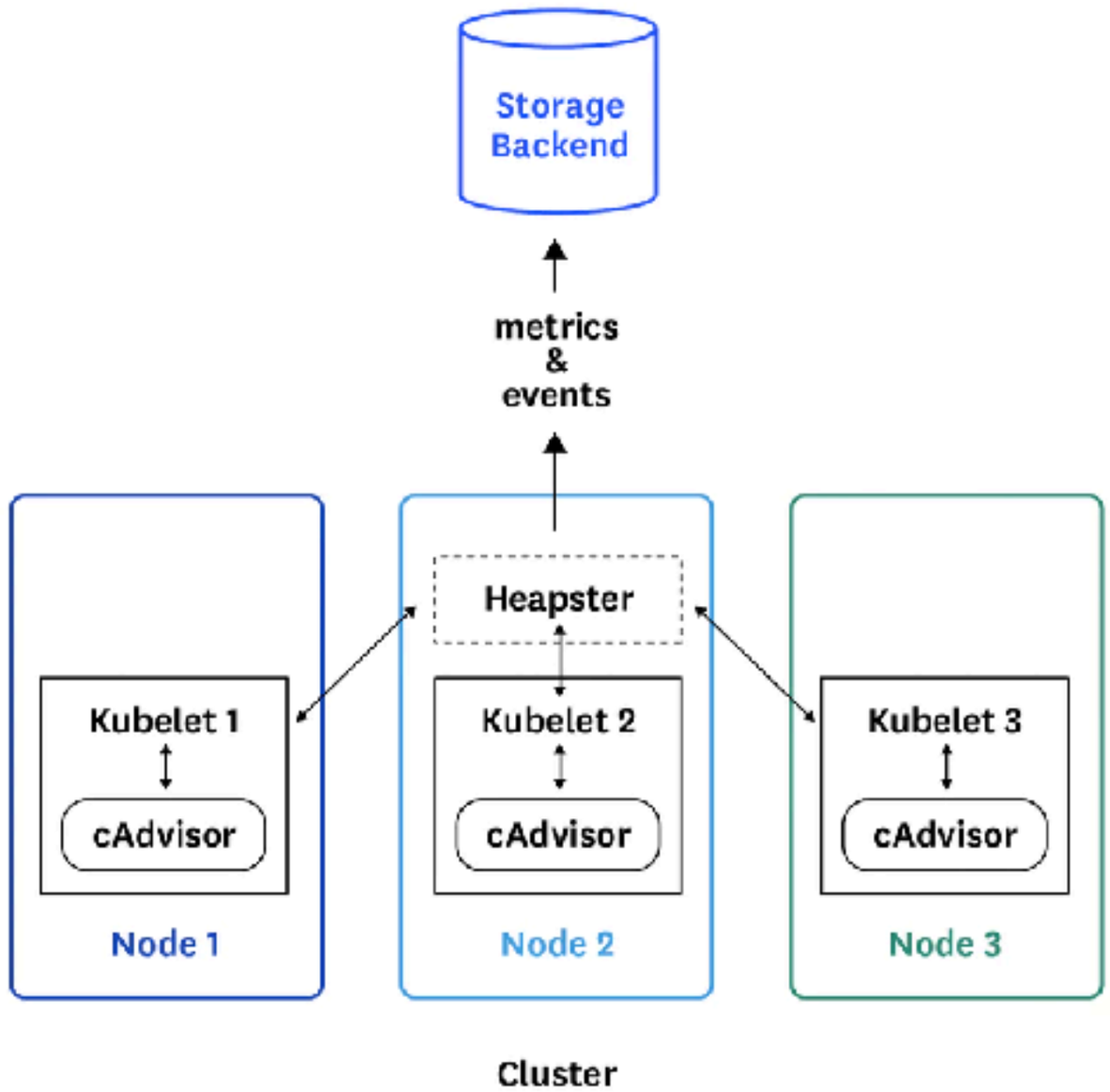
Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)
- Используется в качестве API для других частей k8s:
  - kubernetes dashboard

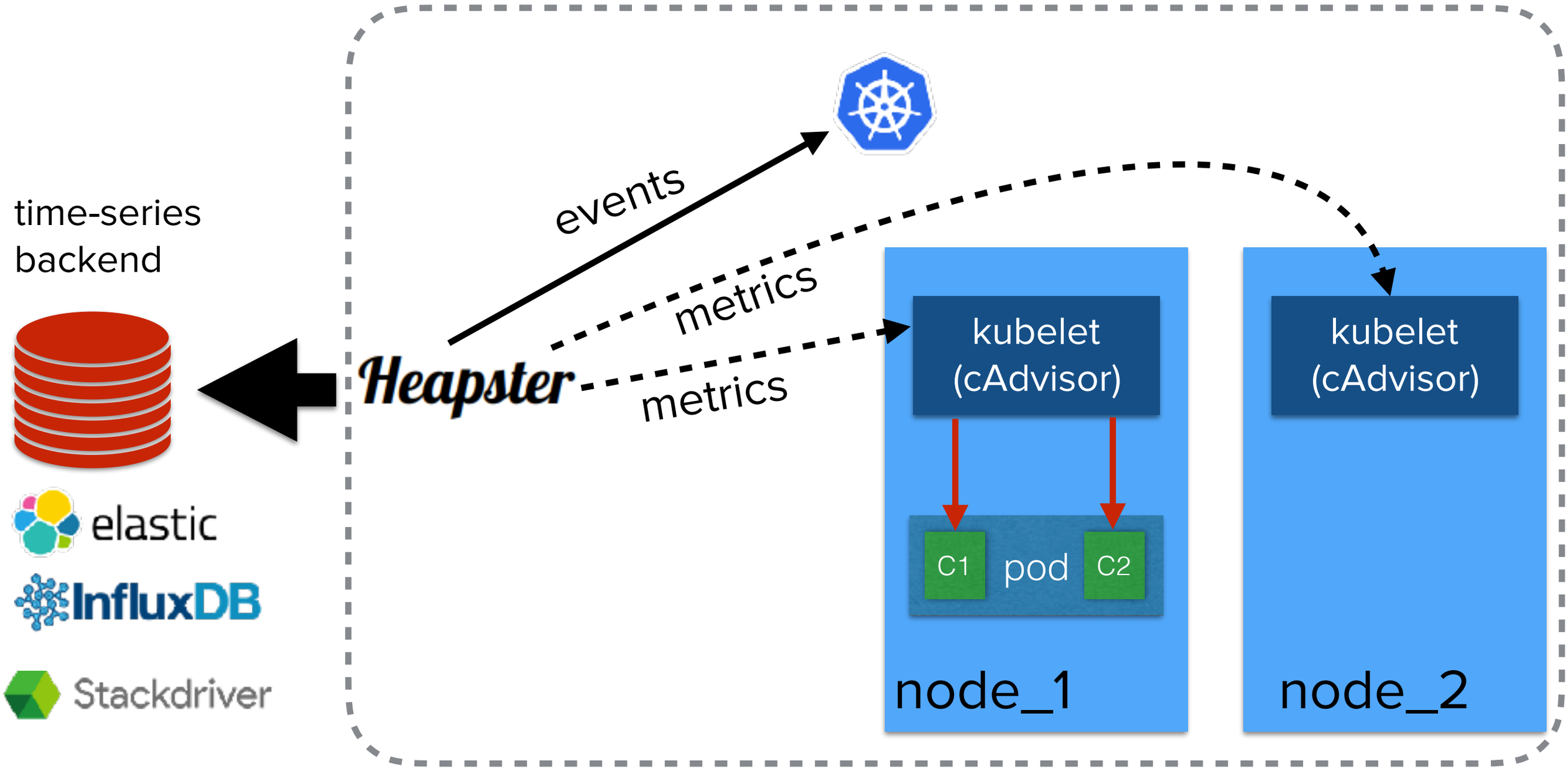
Heapster

- Аддон для Kubernetes
- Собирает метрики с Cadvisor’ов
- Собирает Event’ы из kubernetes API
- Складывает их в time-series бекенды (influxdb, elasticsearch,...)
- Используется в качестве API для других частей k8s:
  - kubernetes dashboard
  - horizontal pod autoscaler

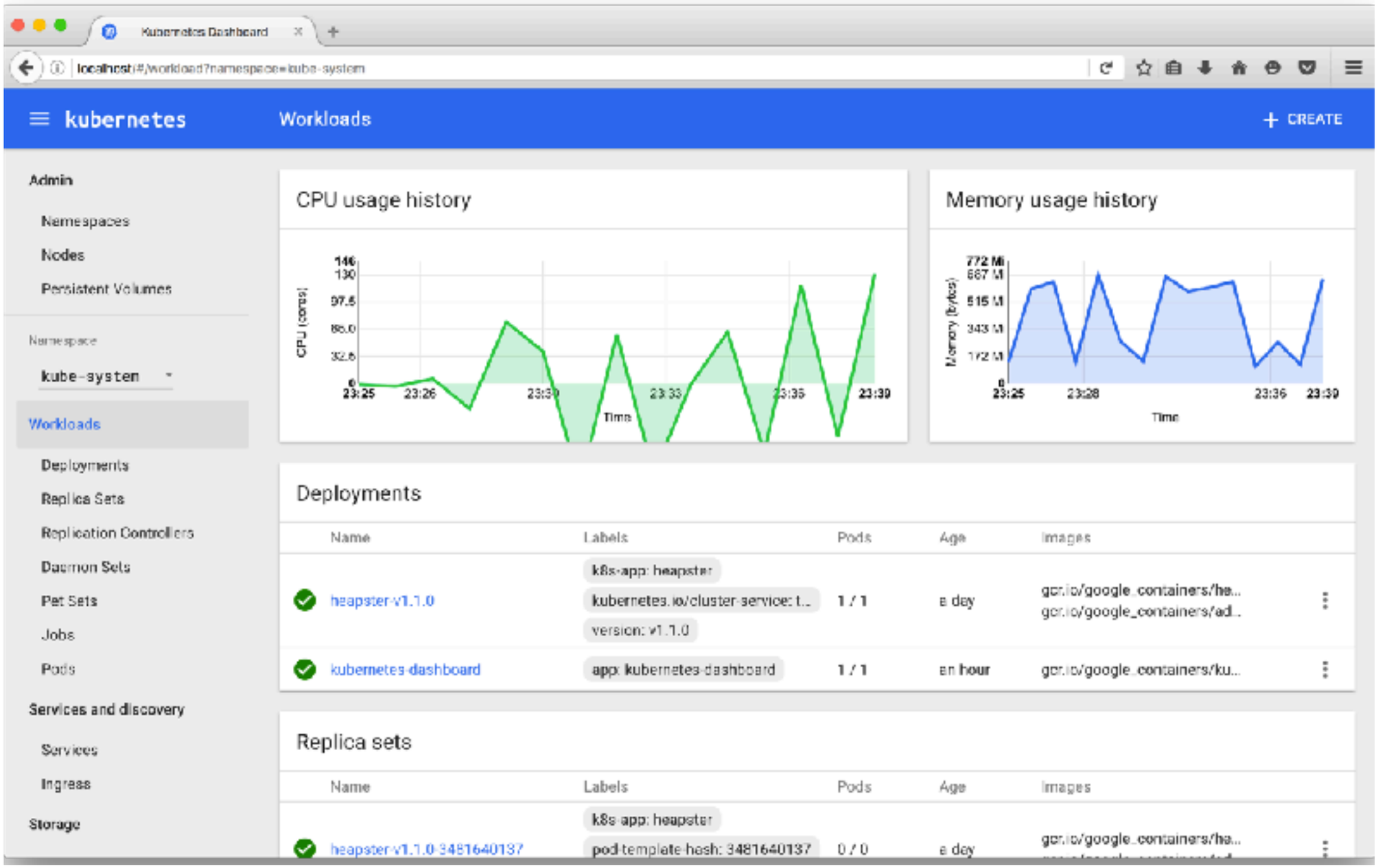
Heapster


Storage Backend
metrics & events
Heapster
Kubelet 1
cAdvisor
Node 1
Kubelet 2
cAdvisor
Node 2
Kubelet 3
cAdvisor
Node 3
Cluster

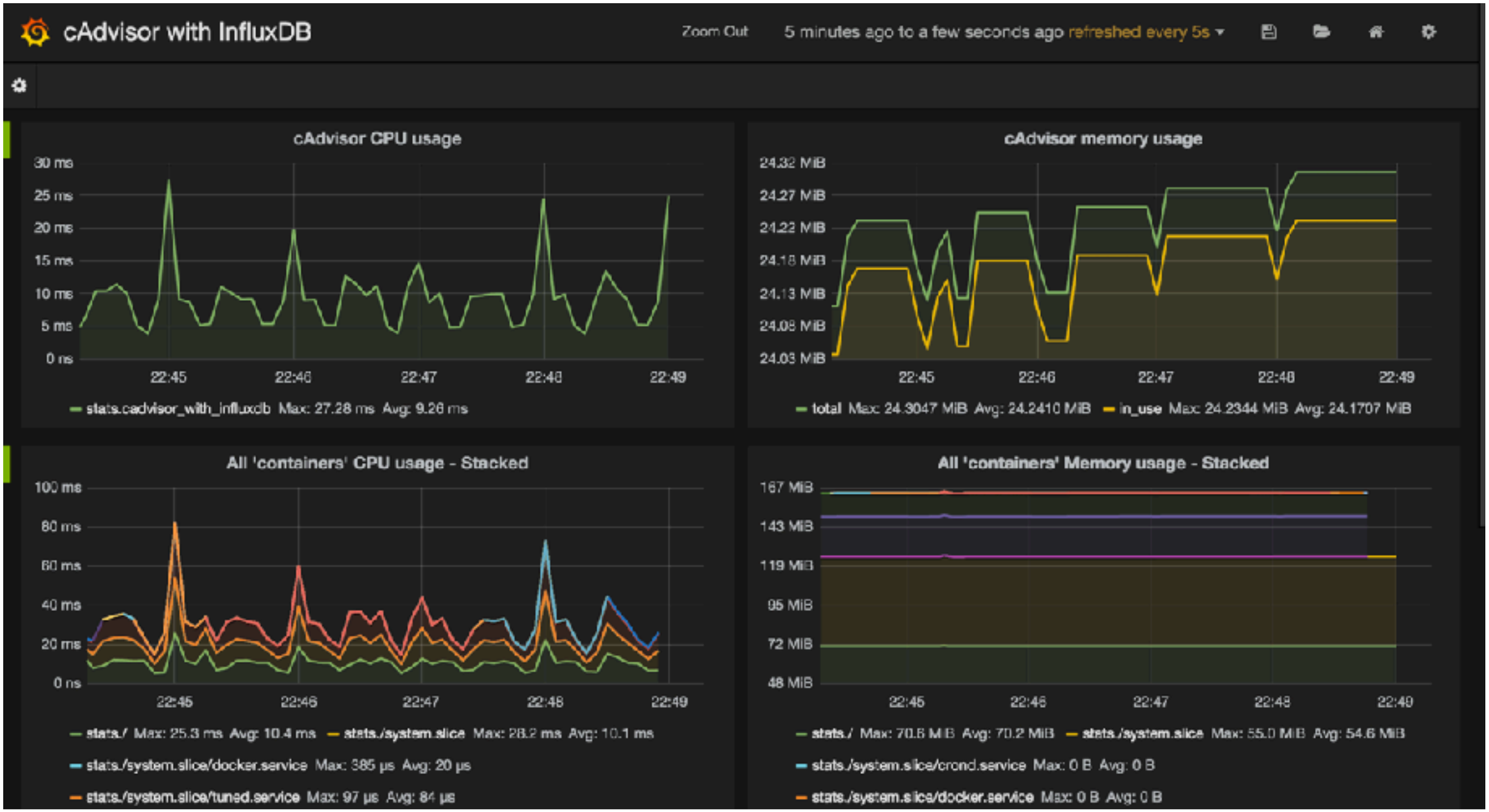
Monitoring Pipeline



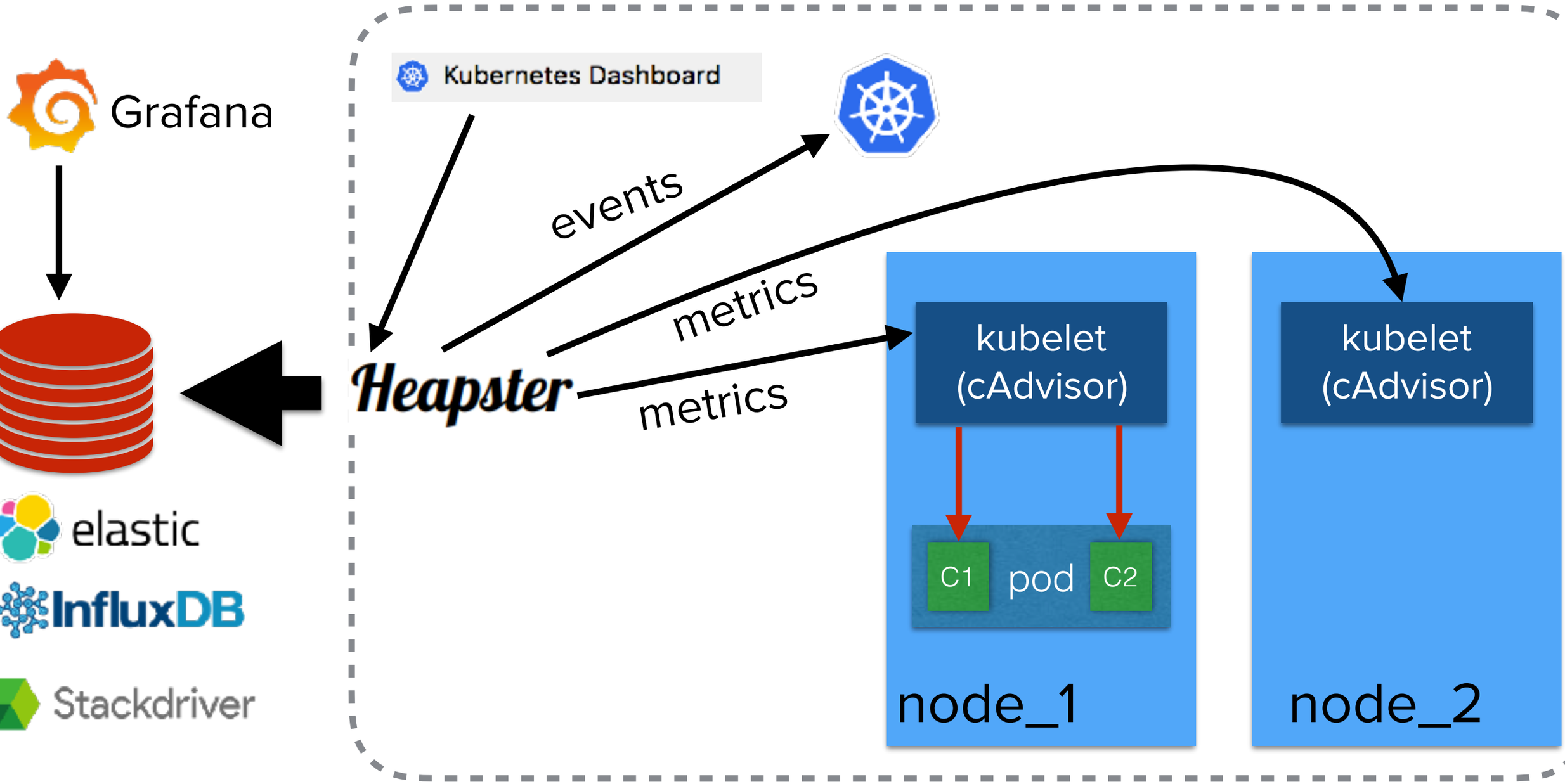
Dashboard


Kubernetes Dashboard
localhost:8/#/workload?namespace=kube-system
kubernetes
Workloads
+ CREATE
Admin
Namespaces
Nodes
Persistent Volumes
Namespace
kube-system
Workloads
Deployments
Replica Sets
Replication Controllers
Daemon Sets
Pet Sets
Jobs
Pods
Services and discovery
Services
Ingress
Storage
CPU usage history
CPU (cores)
146
130
97.6
65.0
32.5
0
23:25
23:26
23:30
23:33
23:35
23:39
Time
Memory usage history
Memory (bytes)
772 Mi
687 Mi
515 Mi
343 Mi
172 Mi
0
23:25
23:28
23:36
23:39
Time
Deployments
Name
Labels
Pods
Age
Images
heapster-v1.1.0
k8s-app: heapster
kubernetes.io/cluster-service: t...
version: v1.1.0
1 / 1
a day
gcr.io/google_containers/he...
gcr.io/google_containers/ad...
kubernetes-dashboard
app: kubernetes-dashboard
1 / 1
an hour
gcr.io/google_containers/ku...
Replica sets
Name
Labels
Pods
Age
Images
heapster-v1.1.0-3481640137
k8s-app: heapster
pod-template-hash: 3481640137
0 / 0
a day
gcr.io/google_containers/he...

Grafana


cAdvisor with InfluxDB
Zoom Out
5 minutes ago to a few seconds ago refreshed every 5s
cAdvisor CPU usage
30 ms
25 ms
20 ms
15 ms
10 ms
5 ms
0 ns
22:45
22:46
22:47
22:48
22:49
stats.cadvisor_with_influxdb Max: 27.28 ms Avg: 9.26 ms
cAdvisor memory usage
24.32 MiB
24.27 MiB
24.22 MiB
24.18 MiB
24.13 MiB
24.08 MiB
24.03 MiB
total Max: 24.3047 MiB Avg: 24.2410 MiB
in_use Max: 24.2344 MiB Avg: 24.1707 MiB
All 'containers' CPU usage - Stacked
100 ms
80 ms
60 ms
40 ms
20 ms
0 ns
stats./ Max: 25.3 ms Avg: 10.4 ms
stats./system.slice Max: 28.2 ms Avg: 10.1 ms
stats./system.slice/docker.service Max: 385 µs Avg: 20 µs
stats./system.slice/tuned.service Max: 97 µs Avg: 84 µs
All 'containers' Memory usage - Stacked
167 MiB
143 MiB
119 MiB
95 MiB
72 MiB
48 MiB
stats./ Max: 70.6 MiB Avg: 70.2 MiB
stats./system.slice Max: 55.0 MiB Avg: 54.6 MiB
stats./system.slice/crond.service Max: 0 B Avg: 0 B
stats./system.slice/docker.service Max: 0 B Avg: 0 B

Monitoring Pipeline



Heapster + cAdvisor

Недостатки решения:

- работает только с k8s
- получаем информацию только о контейнерах
- умеет только перекладывать метрики
- требует time-series backend’a

Monitoring Pipeline

- Prometheus + kube-state-metrics + cAdvisor + …

kube-state-metrics

Собирает информацию о логических объектах k8s

- использование ресурсов на nod’ах
- статусы replicaset-ов
- информацию о pod’ах
- статусы deployment-ов

kube-state-metrics

В отличие от **Heapster**:

- собирает метрики Kubernetes (а не контейнеров)
- не собирает метрики производительности
- держит в себе текущий снимок состояния
- не перенаправляет метрики куда-либо дальше
- работает как Exporter для prometheus

Prometheus

- Развитие проекта началось в 2012
- Создан на основе Borgmon, бывшими работниками Google
- Open source
- Написан на Go
- Whitebox, Pull система
- Умеет **Service Discovery**

Prometheus

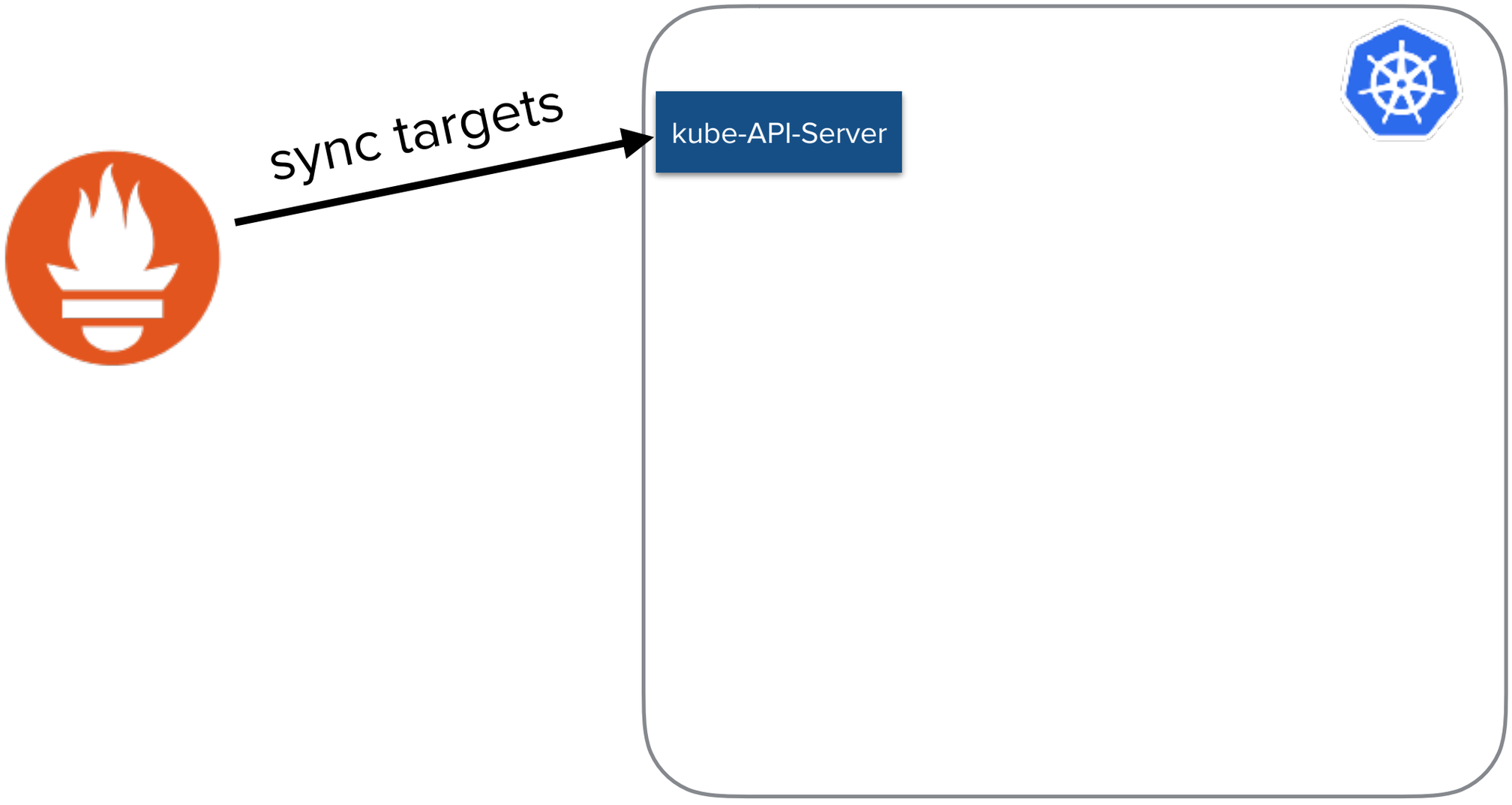
Источники метрик для Prometheus:

- Метрики cAdvisor
- Метрики kube-state-metrics
- Метрики приложений и сервисов
- Метрики хостов, передаваемых через Node-Exporter

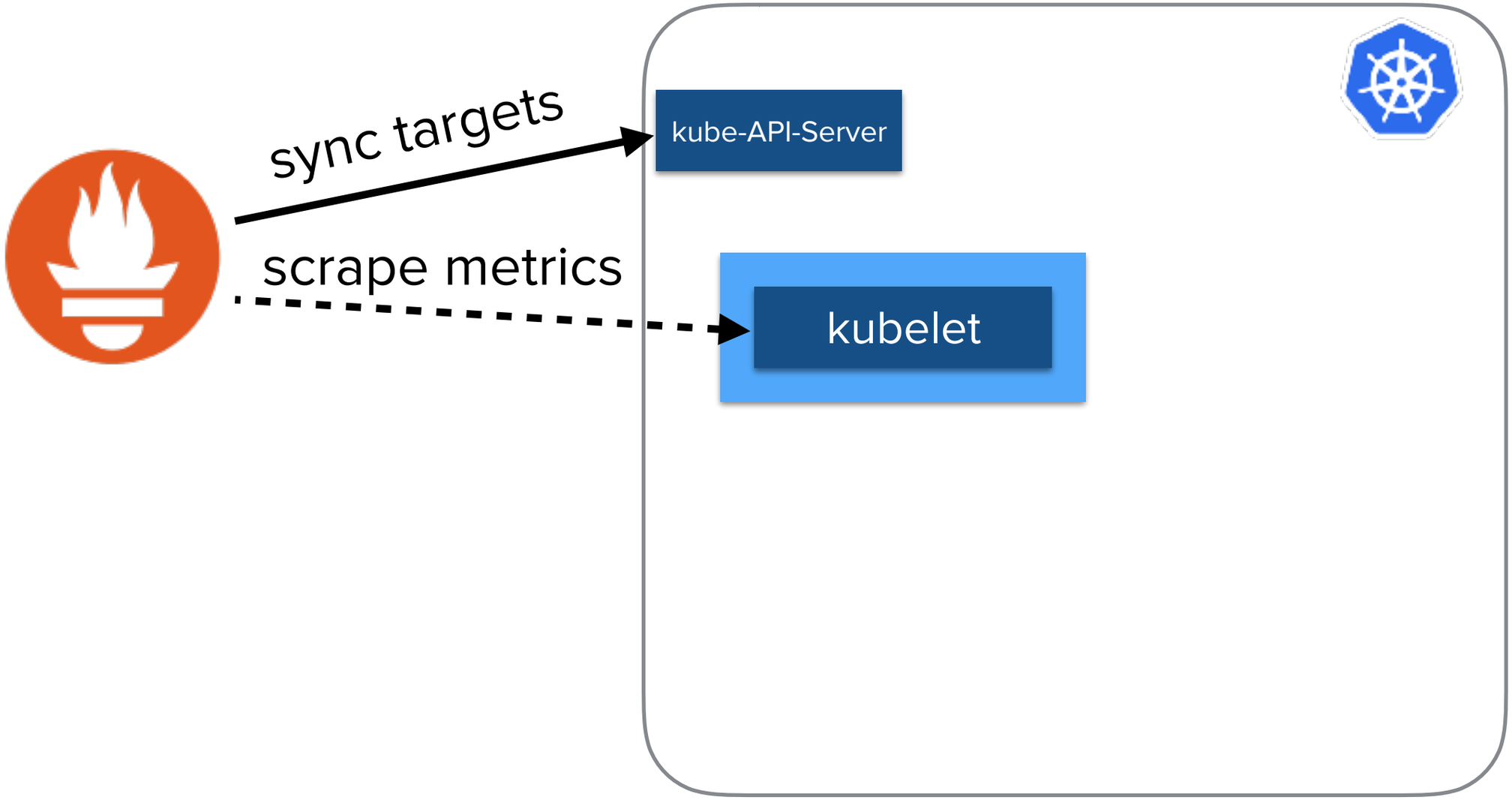
Prometheus

- Targets (endpoint) - источник для сбора метрик
- Группы источников объединяют в jobs

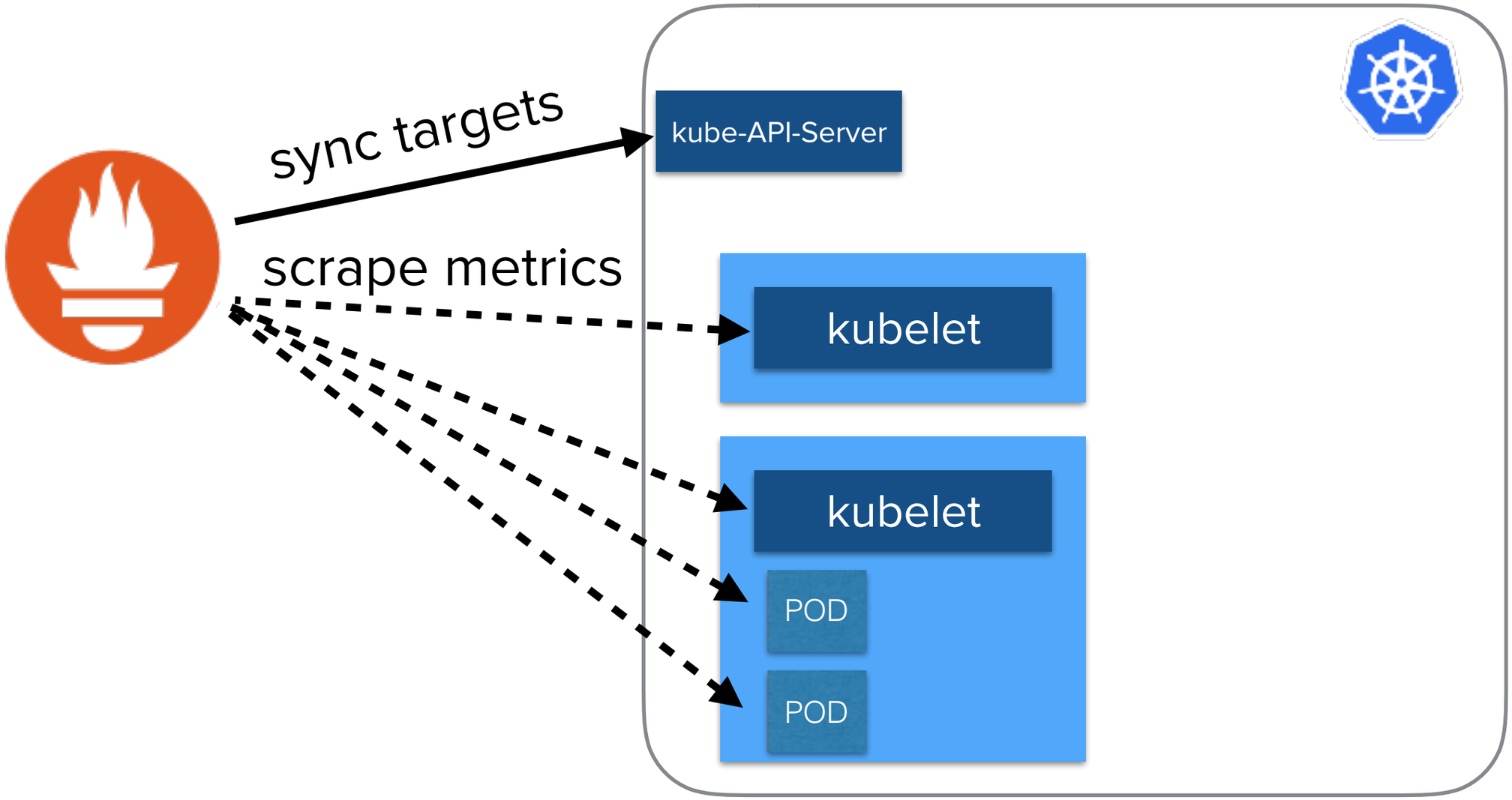
Prometheus



Prometheus



Prometheus


sync targets
scrape metrics
kube-API-Server
kubelet
kubelet
POD
POD

Prometheus

Описываем процедуру сборки метрик

```
scrape_configs:
   - job_name: 'post-endpoints'
     kubernetes_sd_configs:
       - role: endpoints
       …

- job_name: 'kubernetes-apiservices'
   kubernetes_sd_configs:
     …

- job_name: 'kubernetes-nodes'
   kubernetes_sd_configs:
     - role: nodes
   …
```

Prometheus

Находим цели

```
$ cat prometheus.yml

 …
 scrape_configs:
     - job_name: 'post-endpoints'
       kubernetes_sd_configs:
           - role: endpoints
```

Role

объект, который нужно найти:

- node
- endpoints
- pod
- service
- ingress

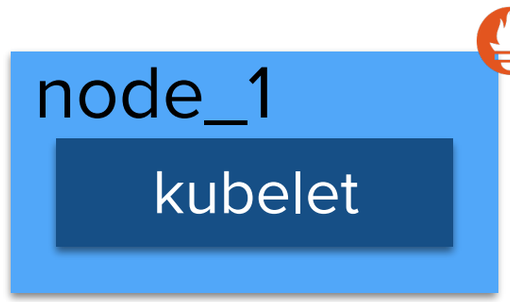
Prometheus

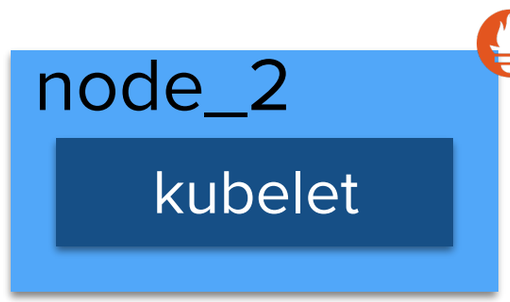
```
$ cat prometheus.yml

    …
    scrape_configs:
        - job_name: 'post-endpoints'
          kubernetes_sd_configs:
              - role: node
```

Targets:

- node_1
- node_2



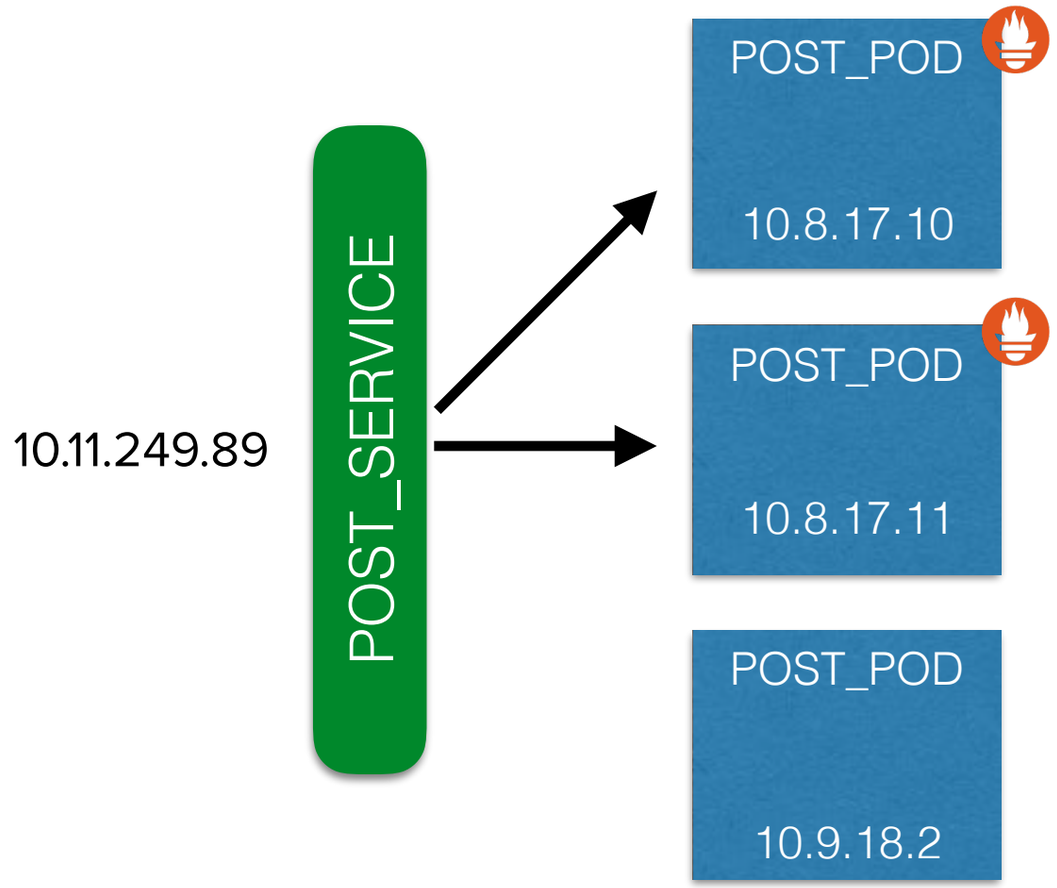


Prometheus

```
scrape_configs:
   - job_name: 'post-endpoints'
     kubernetes_sd_configs:
        - role: endpoints
```

Targets:

- 10.8.17.10
- 10.8.17.11

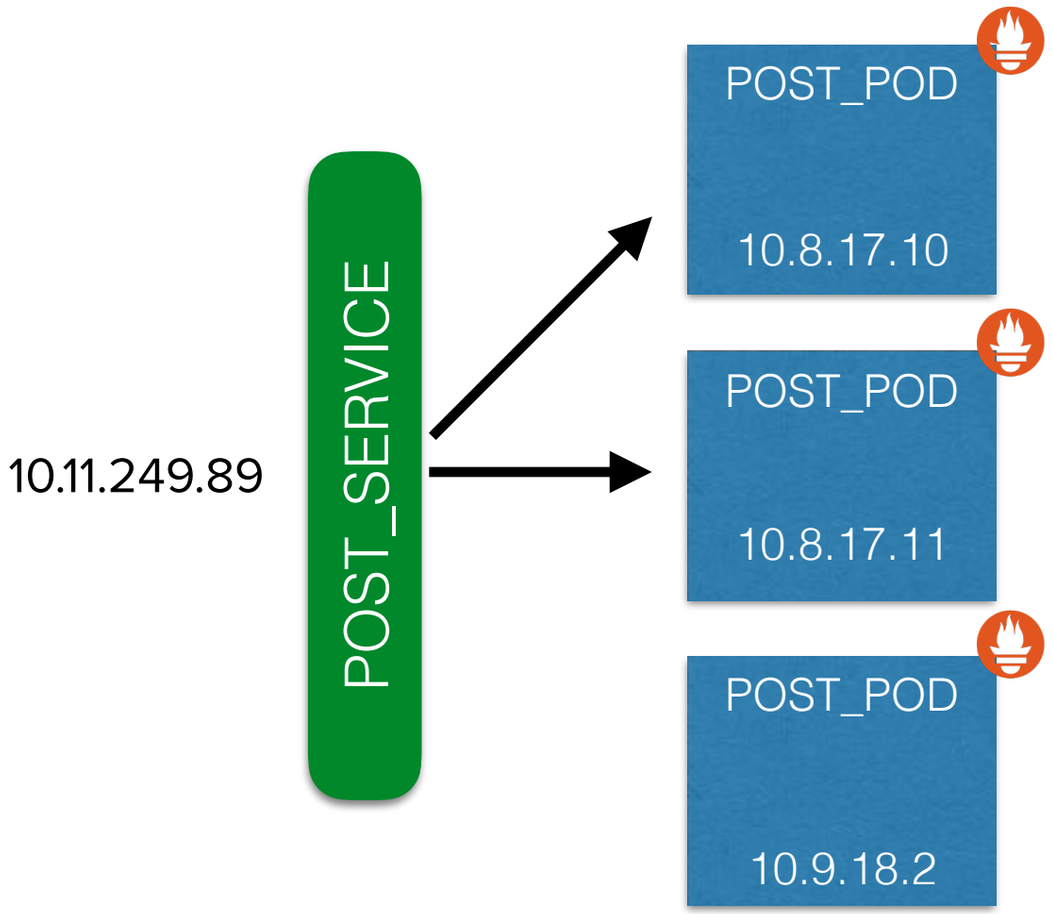


Prometheus

```
$ cat prometheus.yml
    …
    scrape_configs:
       - job_name: 'post-endpoints'
         kubernetes_sd_configs:
            - role: pod
```

Targets:

- 10.8.17.10
- 10.8.17.11
- 10.9.18.2

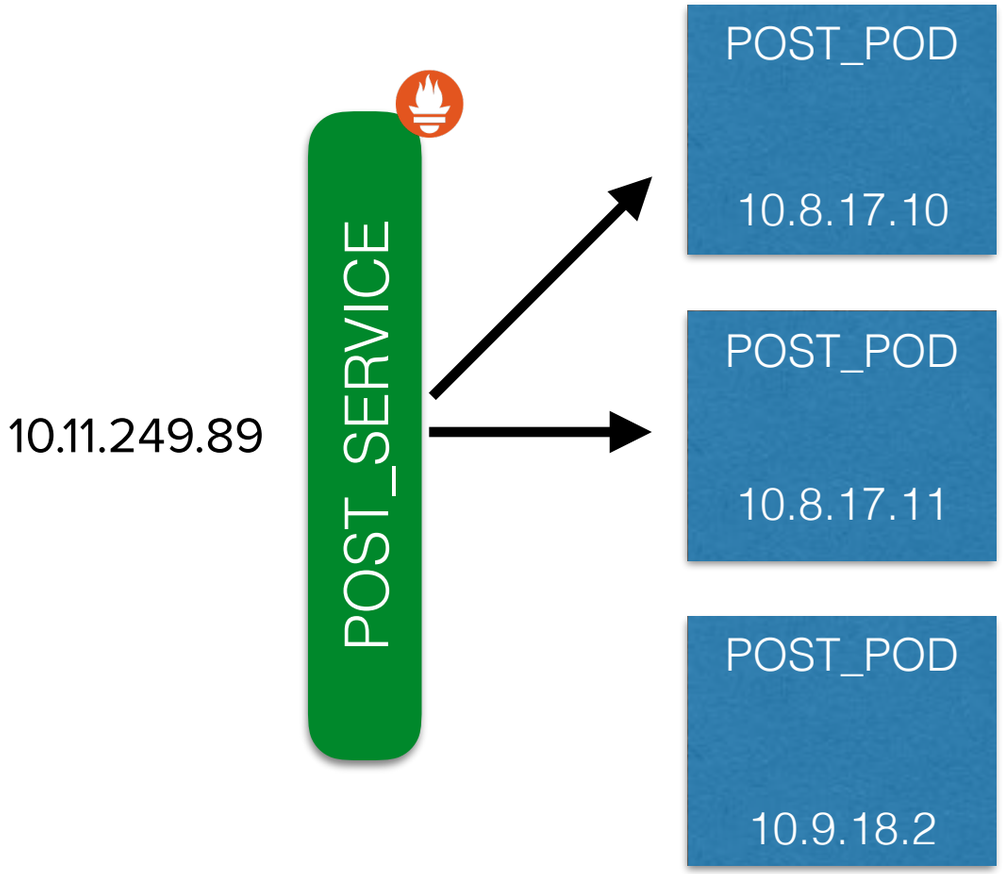


Prometheus

```
$ cat prometheus.yml
    …
    scrape_configs:
       - job_name: 'post-endpoints'
         kubernetes_sd_configs:
             - role: service
```

Targets:

- 10.11.249.89

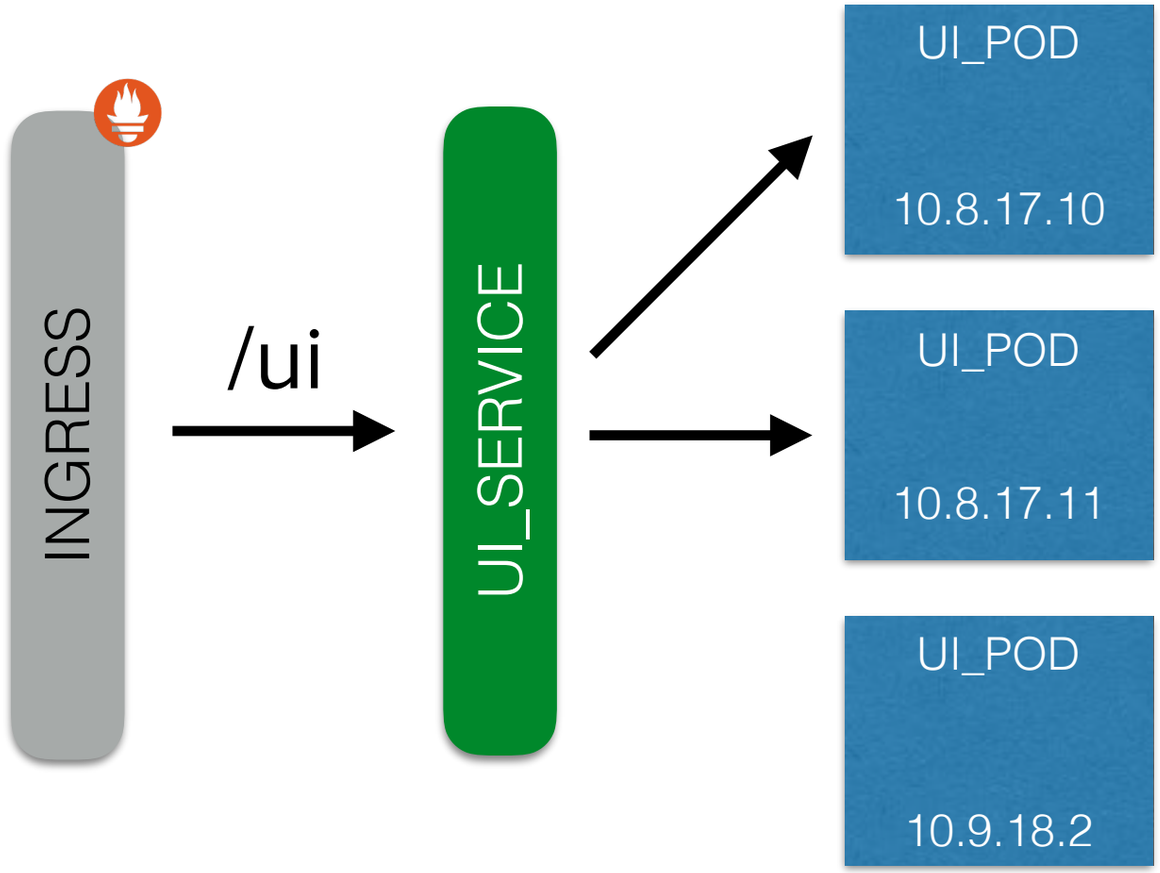


Prometheus

```
$ cat prometheus.yml
   …
   scrape_configs:
      - job_name: 'post-endpoints'
        kubernetes_sd_configs:
          - role: ingress
```

Targets:

- ui_service/ui



Prometheus

```
scrape_configs:
   - job_name: 'post-endpoints'
     kubernetes_sd_configs:
       - role: endpoints
         api_server: … # адрес k8s API
         tls_config: … #CA-сертификат k8s API
         bearer_token: … # токен serviceAccount’а
         namespaces: … # в каких namespace’ах искать
       names:
         - default
         - dev
```

по-умолчанию уже заданы

Prometheus

Targets

kubernetes-apiservers (1/1 up)
kubernetes-nodes (3/3 up)
kubernetes-service-endpoints (4/4 up)
post-endpoints (3/3 up)

Prometheus

Добавляем приложение в мониторинг prometheus

```
---
apiVersion: v1
kind: Service
metadata:
  name: post
  labels:
    app: reddit
    component: post
  annotations:
    prometheus.io/scrape: "true"
spec:
```

Prometheus

post-endpoints (3/3 up)

Endpoint	State	Labels
http://10.8.17.10:5000/metrics	UP	app="reddit" component="post" instance="10.8.17.10:5000" kubernetes_name="post-test-post"
http://10.8.17.11:5000/metrics	UP	app="reddit" component="post" instance="10.8.17.11:5000" kubernetes_name="post-test-post"
http://10.8.17.12:5000/metrics	UP	app="reddit" component="post" instance="10.8.17.12:5000" kubernetes_name="post-test-post"

post

post_count

post_read_db_seconds_bucket

post_read_db_seconds_count

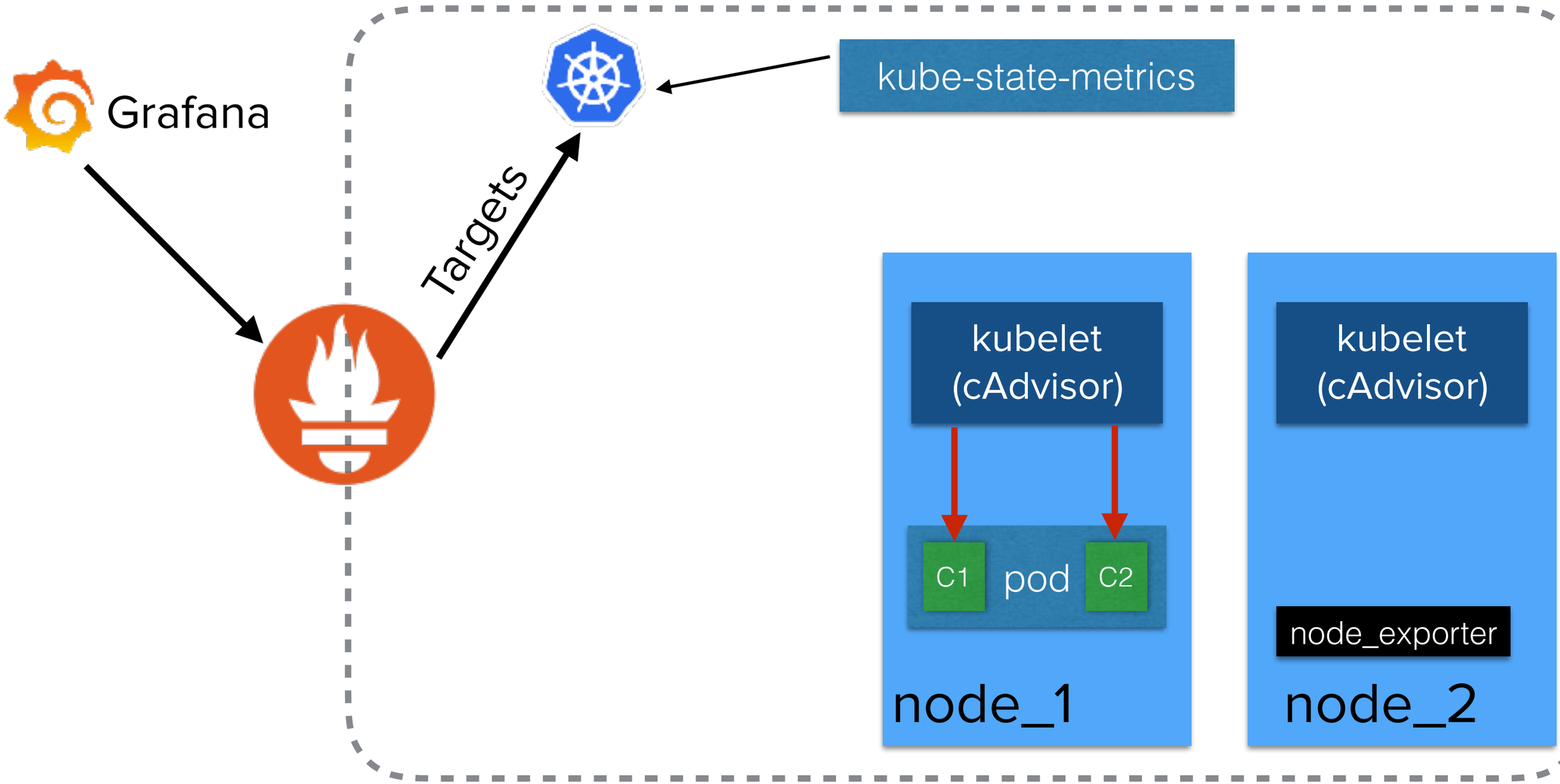
post_read_db_seconds_sum

Prometheus

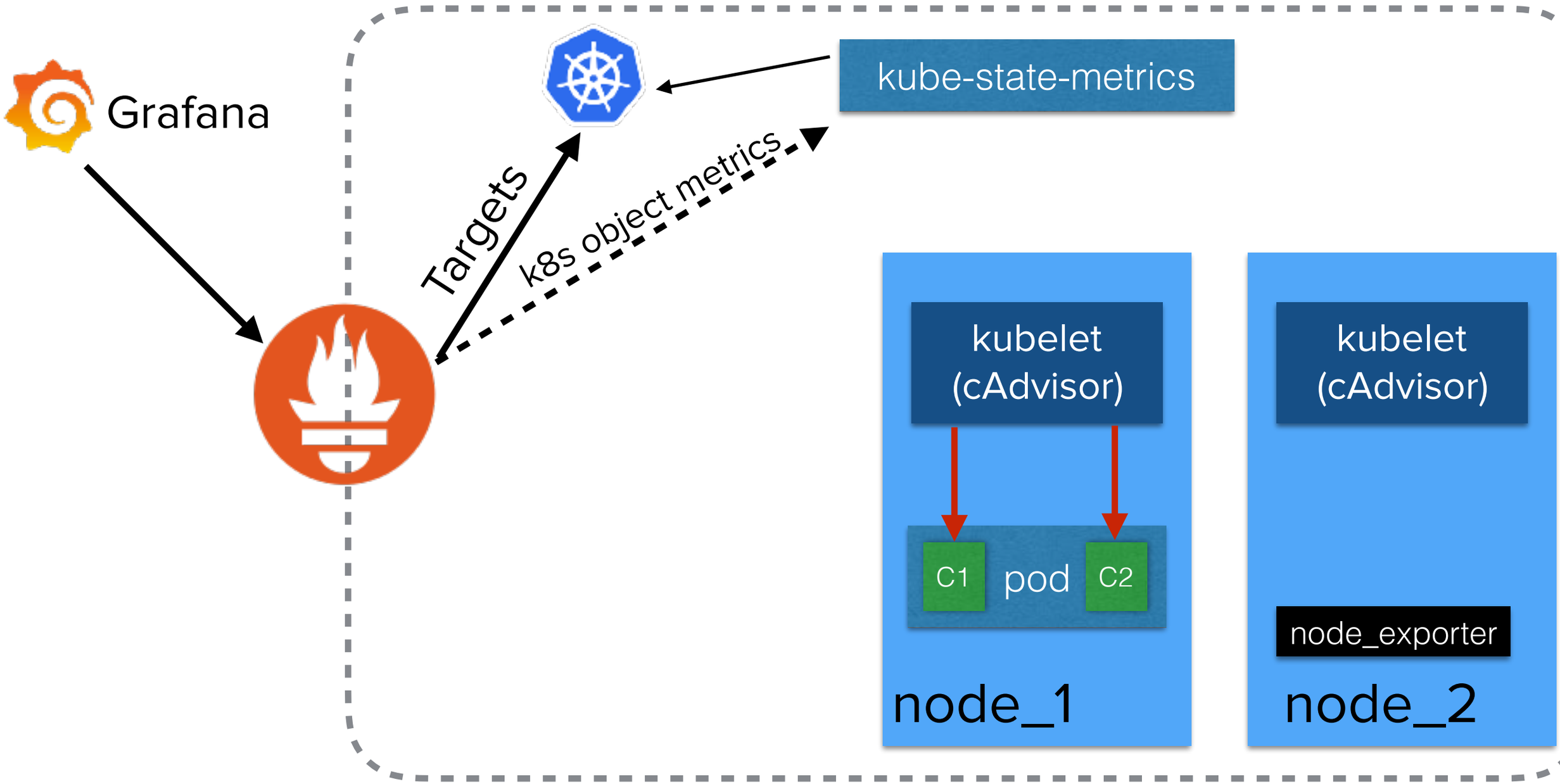
Можем поменять конфиг

```
---
apiVersion: v1
kind: Service
metadata:
  name: post
  labels:
    app: reddit
    component: post
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/path: "/healthz"
    prometheus.io/port: "5000"
spec:
```

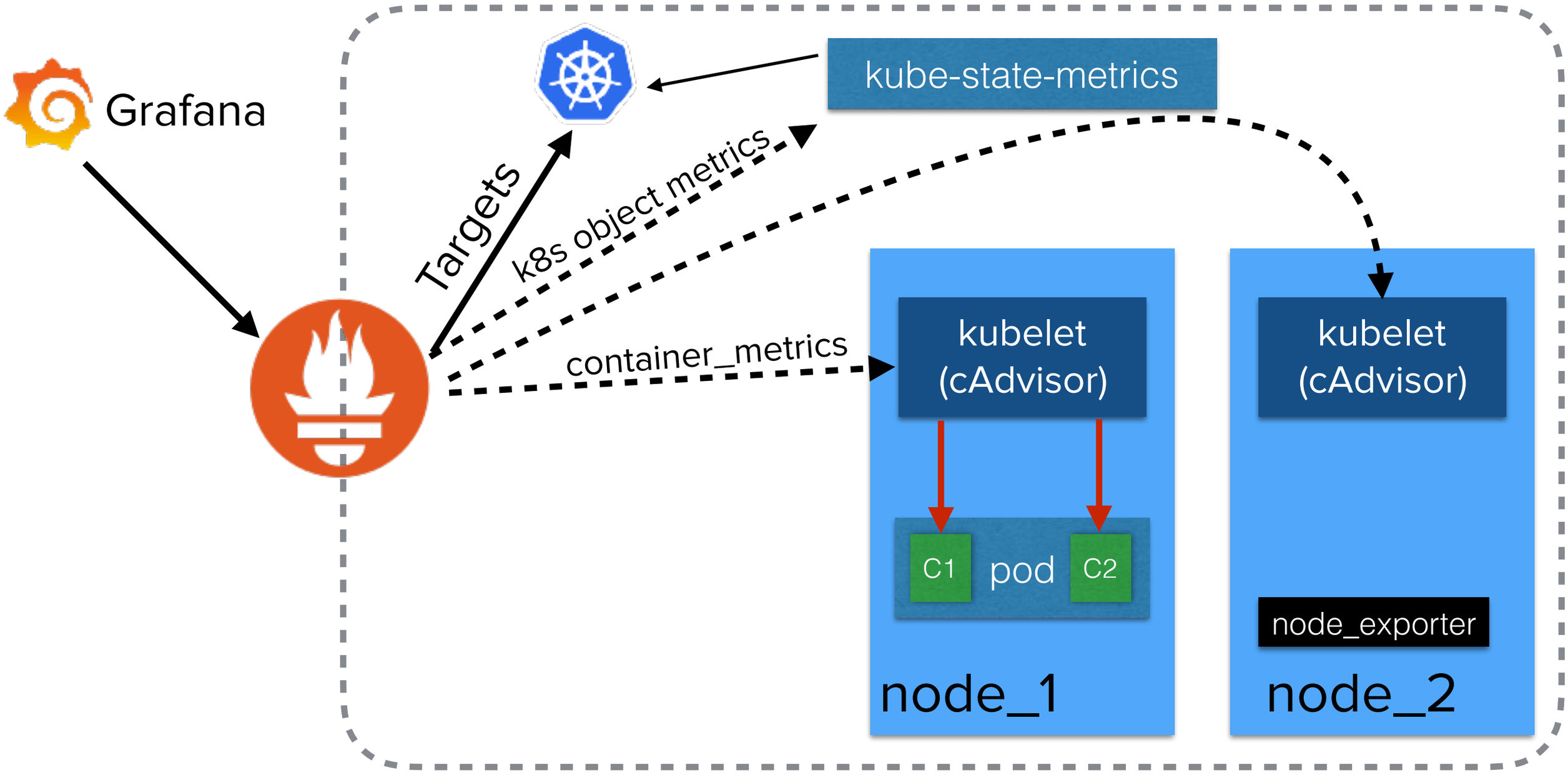
Monitoring Pipeline



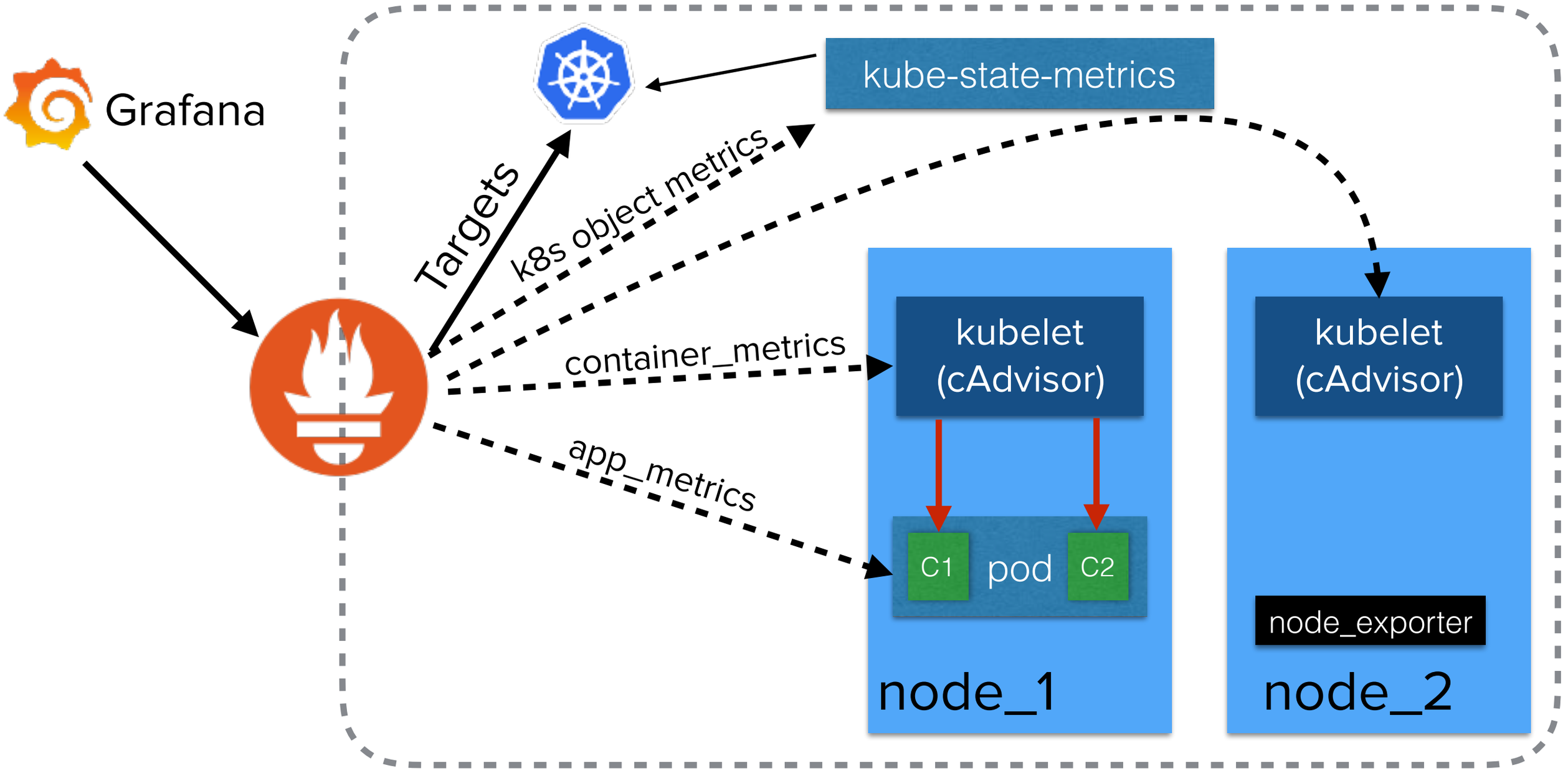
Monitoring Pipeline



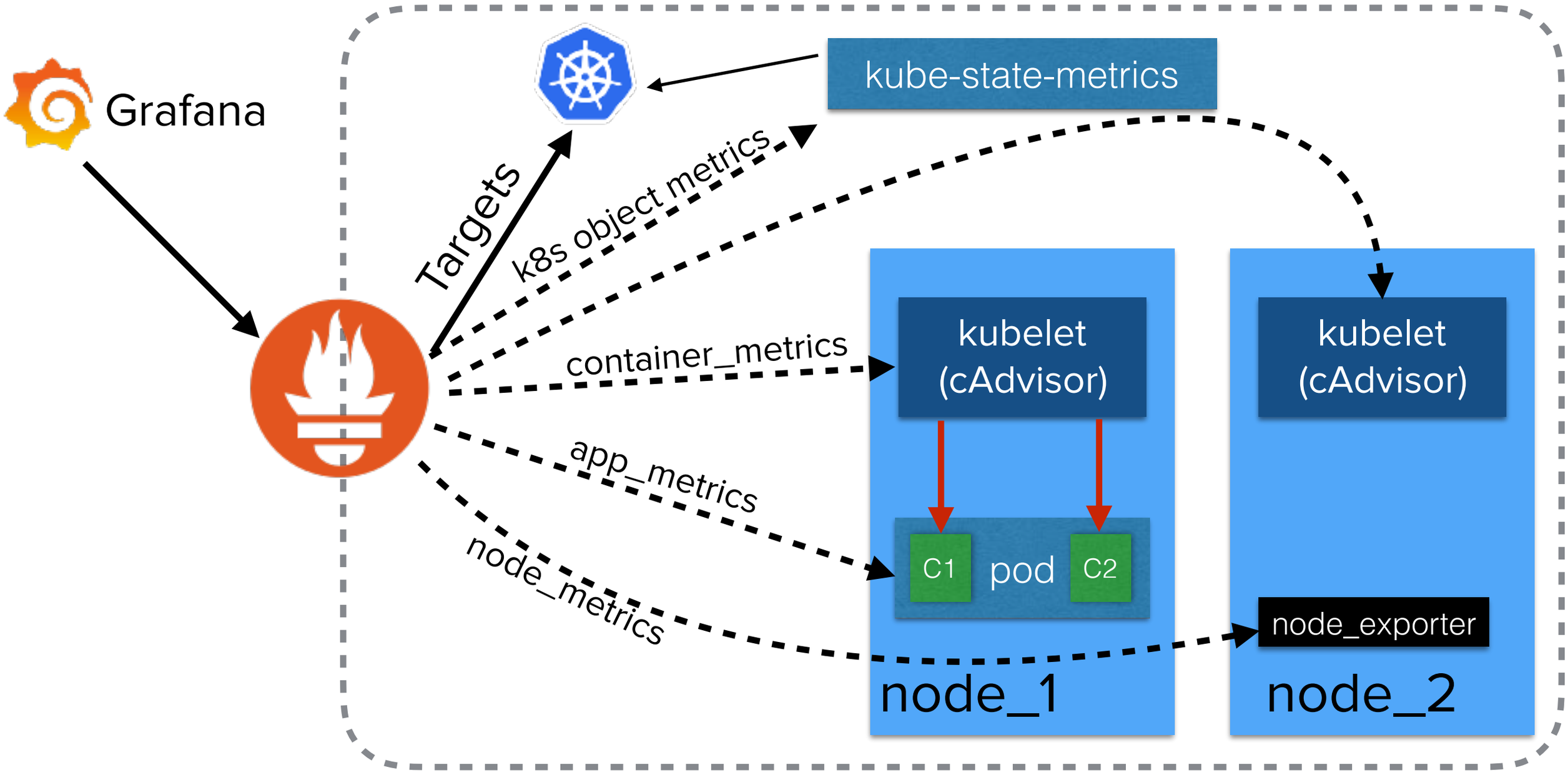
Monitoring Pipeline



Monitoring Pipeline



Monitoring Pipeline



Grafana


Kubernetes cluster monitoring (via Prometheus)
Zoom Out
Last 1 hour
Refresh every 10s
Cluster memory usage
36%
Cluster CPU usage
16.93%
Cluster Filesystem usage
N/A
Pod CPU usage
0.55%
0.50%
0.45%
0.40%
0.35%
0.30%
11:30
11:35
11:40
11:45
11:50
11:55
12:00
12:05
12:10
12:15
12:20
12:25
avg
current
io_kubernetes_container_name
0.379%
0.389%
Pod memory usage
572 MiB
477 MiB
381 MiB
286 MiB
191 MiB
95 MiB
0 B
avg
current
io_kubernetes_container_name 489.32 MiB 504.26 MiB
io_kubernetes_container_name 172.33 MiB 171.40 MiB
io_kubernetes_container_name 142.61 MiB 144.90 MiB
io_kubernetes_container_name 132.66 MiB 133.15 MiB
io_kubernetes_container_name 42.09 MiB 43.50 MiB
io_kubernetes_container_name 35.78 MiB 35.77 MiB
io_kubernetes_container_name 28.64 MiB 28.47 MiB
io_kubernetes_container_name 27.94 MiB 27.95 MiB
io_kubernetes_container_name 25.89 MiB 26.72 MiB

kubectl top

Текущий статус потребления ресурсов

```
$ kubectl top nodes
```

NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
gke-cluster-1-default-pool-f9c66281-rgld	116m	12%	1030Mi	88%
gke-cluster-1-default-pool-f9c66281-dbb2	72m	7%	845Mi	72%
gke-cluster-1-big-pool-b4209075-19l5	118m	6%	1231Mi	21%

```
$ kubectl top pod
```

NAME	CPU(cores)	MEMORY(bytes)
post-test-post-54b7cbcc69-pnbf7	4m	42Mi
running-gorilla-prometheus-server-5fc8847448-24xnw	14m	565Mi
post-test-post-54b7cbcc69-9md82	4m	41Mi
post-test-post-54b7cbcc69-mhj5r	4m	42Mi
running-gorilla-prometheus-kube-state-metrics-695f989d6-btzqq	1m	16Mi

Logging

kubectl logs

Посмотреть логи контейнеров

```
$ kubectl logs post-test-post-54b7cbcc69-9md82

  10.8.19.5 - - [04/Dec/2017 15:04:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:05:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:06:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:07:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:08:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:09:16] "GET /metrics HTTP/1.1" 200 -
```

kubectl logs

Посмотреть логи контейнеров

```
$ kubectl logs post-test-post-54b7cbcc69-9md82  cont_1_name

  10.8.19.5 - - [04/Dec/2017 15:04:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:05:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:06:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:07:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:08:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:09:16] "GET /metrics HTTP/1.1" 200 -
```

имя контейнера
в POD’е

kubectl logs

Посмотреть логи контейнеров

```
$ kubectl logs post-test-post-54b7cbcc69-9md82  cont_1_name  --tail 10

  10.8.19.5 - - [04/Dec/2017 15:04:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:05:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:06:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:07:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:08:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:09:16] "GET /metrics HTTP/1.1" 200 -
```

имя контейнера в POD’е

последние 10 записей

kubectl logs

Посмотреть логи контейнеров

```
$ kubectl logs post-test-post-54b7cbcc69-9md82  cont_1_name  --tail 10  -f

  10.8.19.5 - - [04/Dec/2017 15:04:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:05:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:06:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:07:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:08:16] "GET /metrics HTTP/1.1" 200 -
  10.8.19.5 - - [04/Dec/2017 15:09:16] "GET /metrics HTTP/1.1" 200 -
```

имя контейнера в POD’е

последние 10 записей

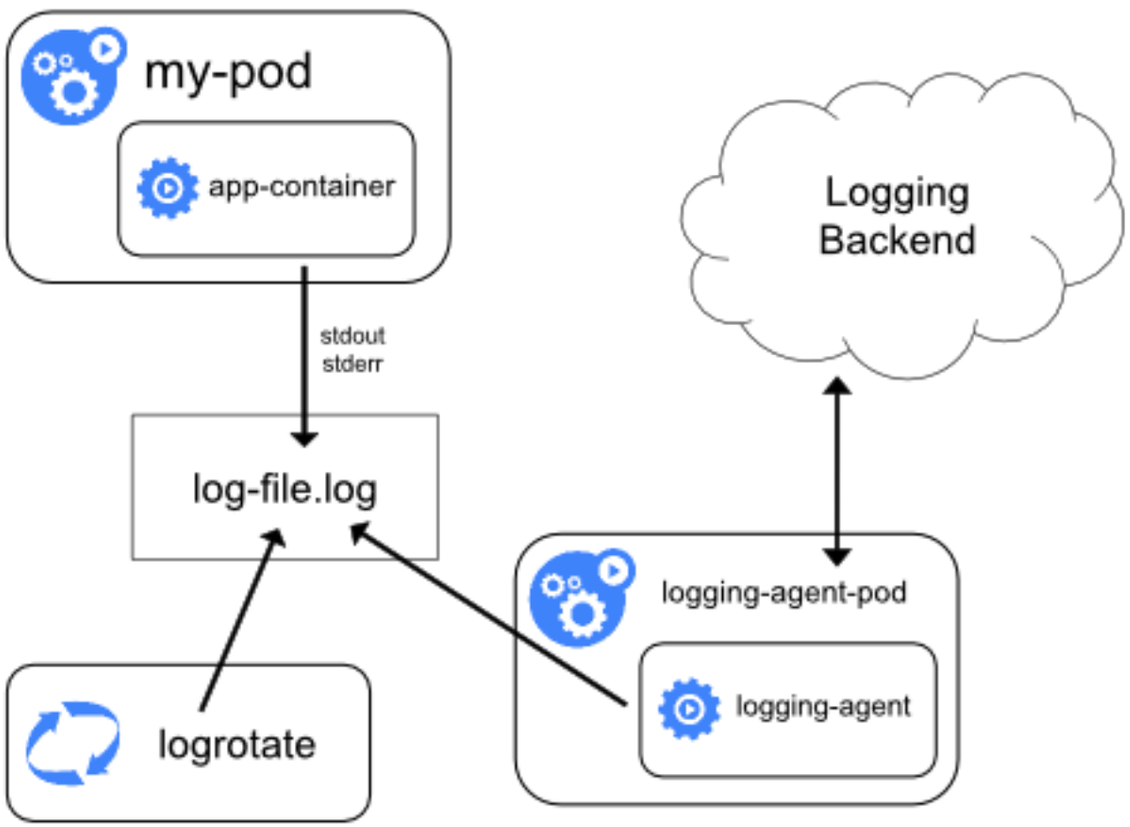
продолжать следить

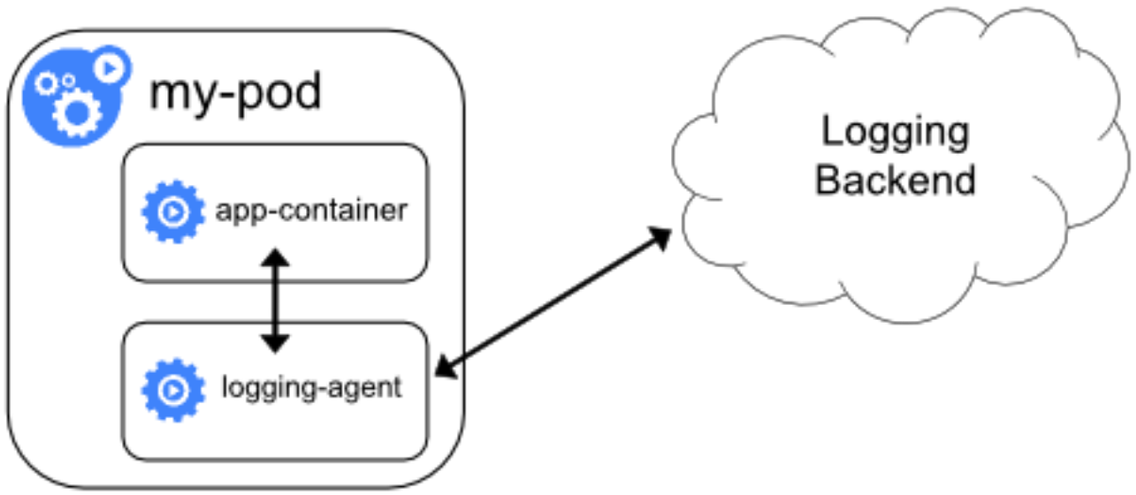
Что логировать?

- Логи контейнеров
- Логи хостовых систем
- Логи Docker
- Логи k8s для аудита
- ...

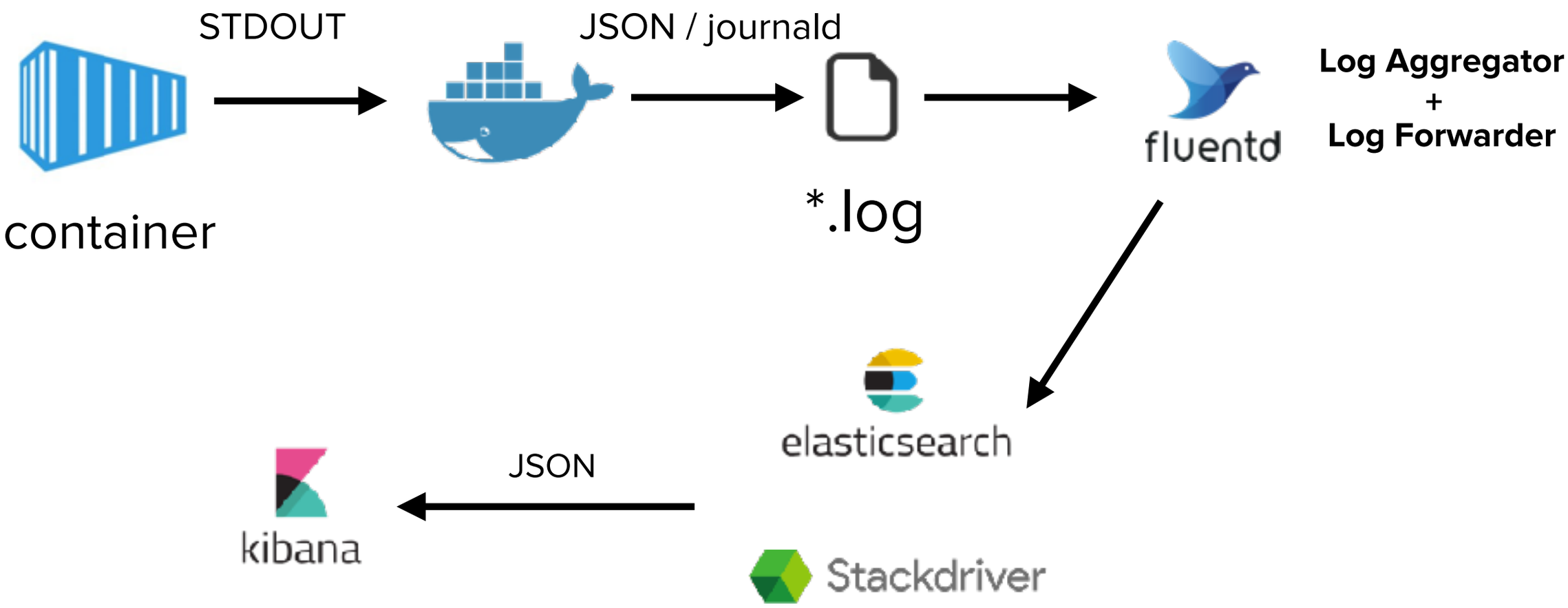
Как логировать?

У k8s нет встроенных механизмов для отправки логов (как logging drivers в docker)

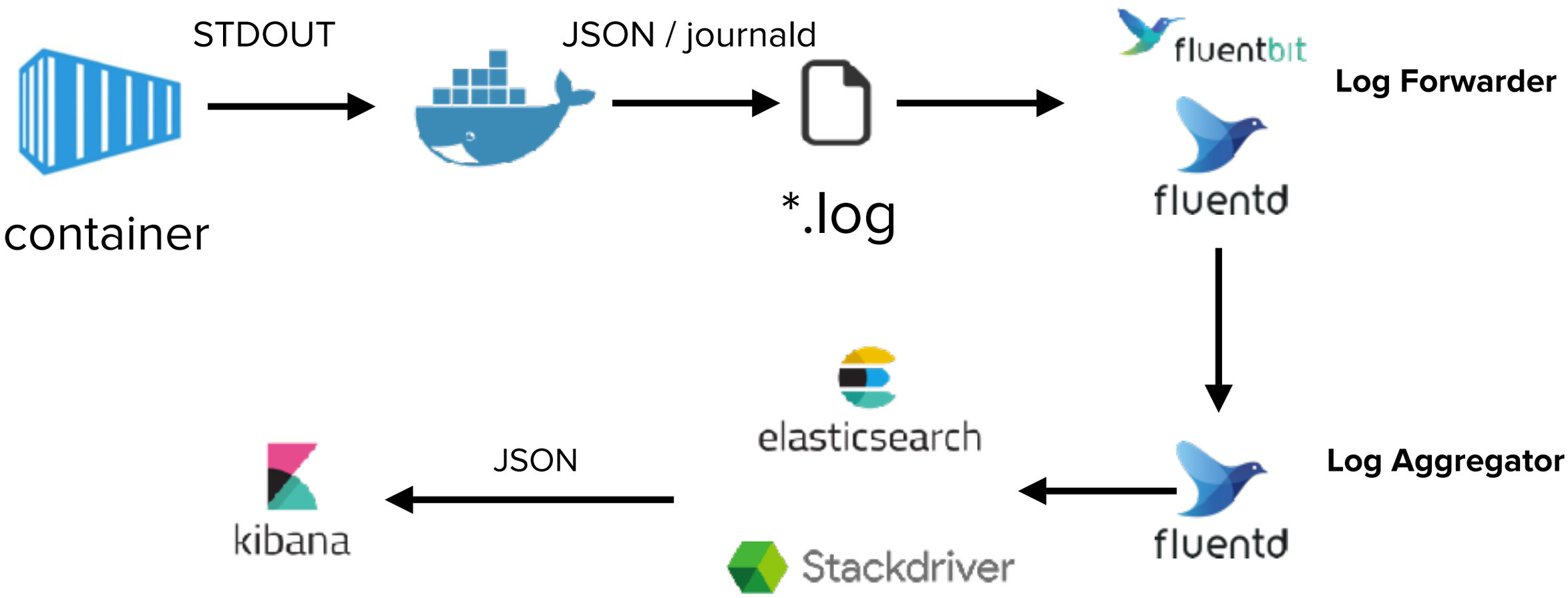




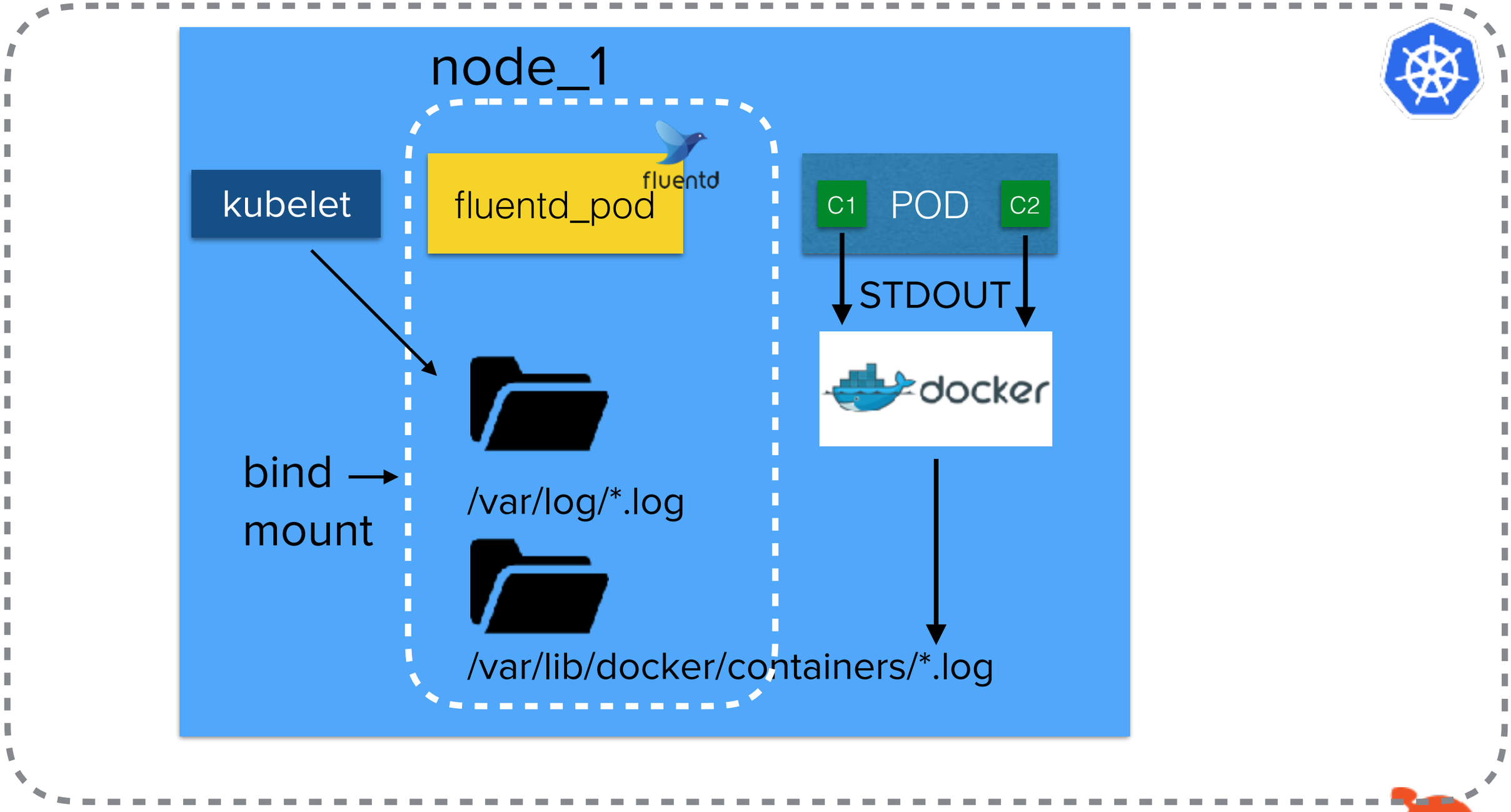
Как логировать?



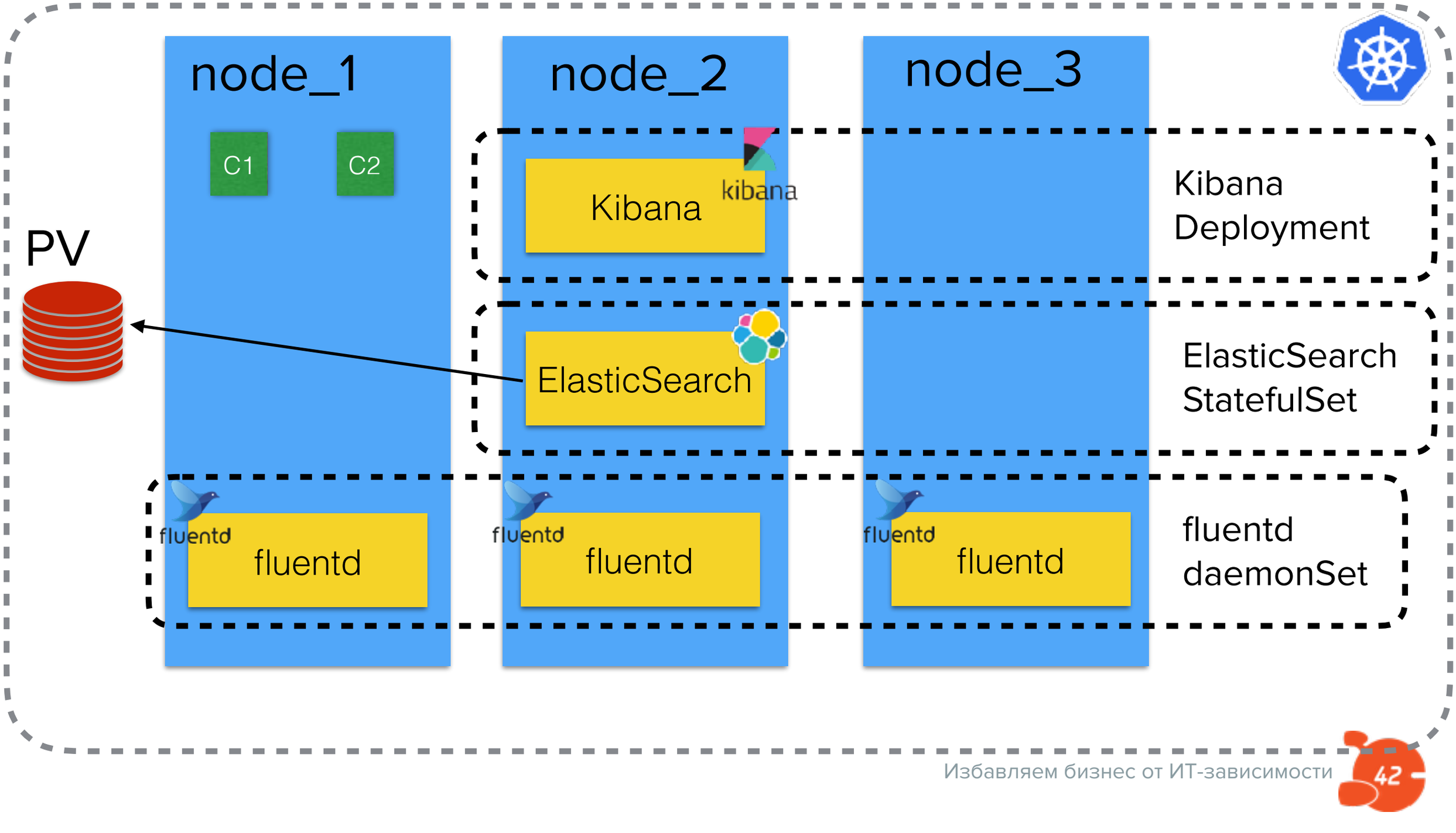
Как логировать?



Как логировать?



Как логировать?



Из приложения