Kubernetes. Мониторинг и логирование

План

- Мониторинг Kubernetes
- Логирование в Kubernetes

Мониторинг

Что отслеживать?

- Работоспособность приложений
- Метрики приложений
- Метрики хостов
- Метрики pod’ов и контейнеров
- Метрики и работоспособность самого Kubernetes

Что у k8s есть?

- Probes
- cAdvisor
- 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

Проверяет, что приложение запущено и “живо”

- Readiness

Проверяет, что приложение готово обслуживать запросы

- Startup

Проверяет, что приложение запущено. Блокирует работу liveness и readiness

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
```

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
```

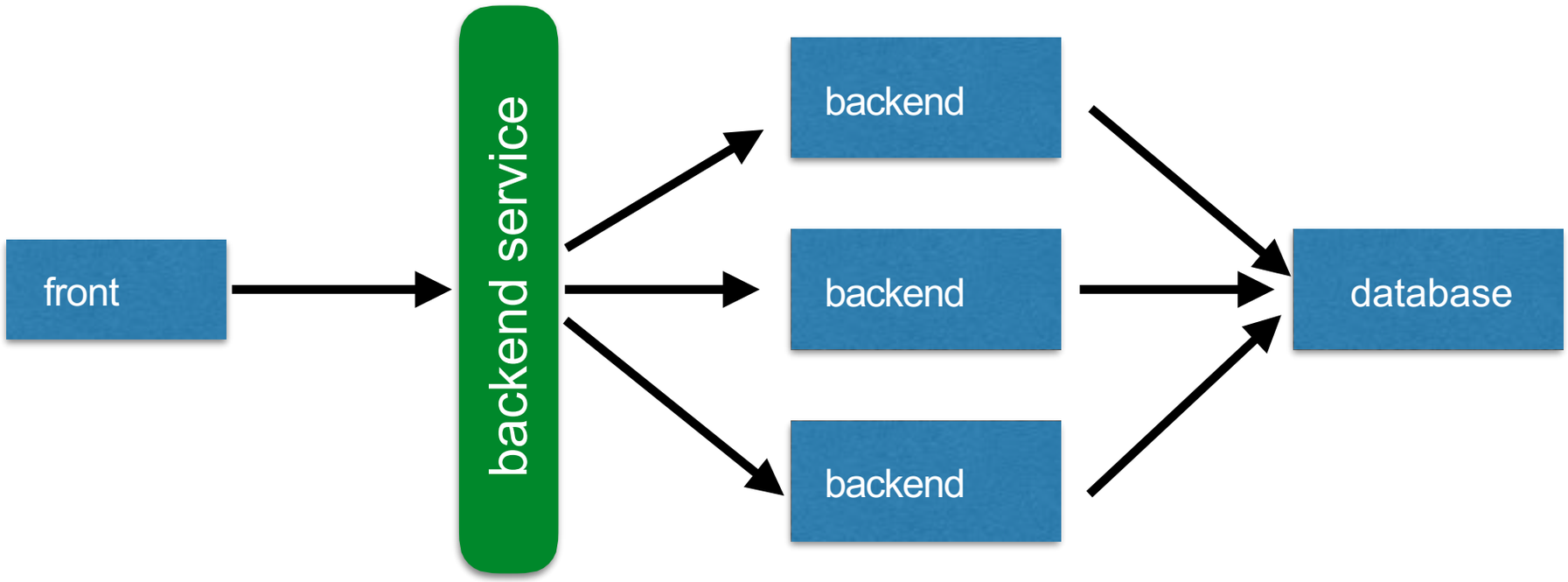
Startup Probe

Проверяет, что приложение готово обслуживать запросы

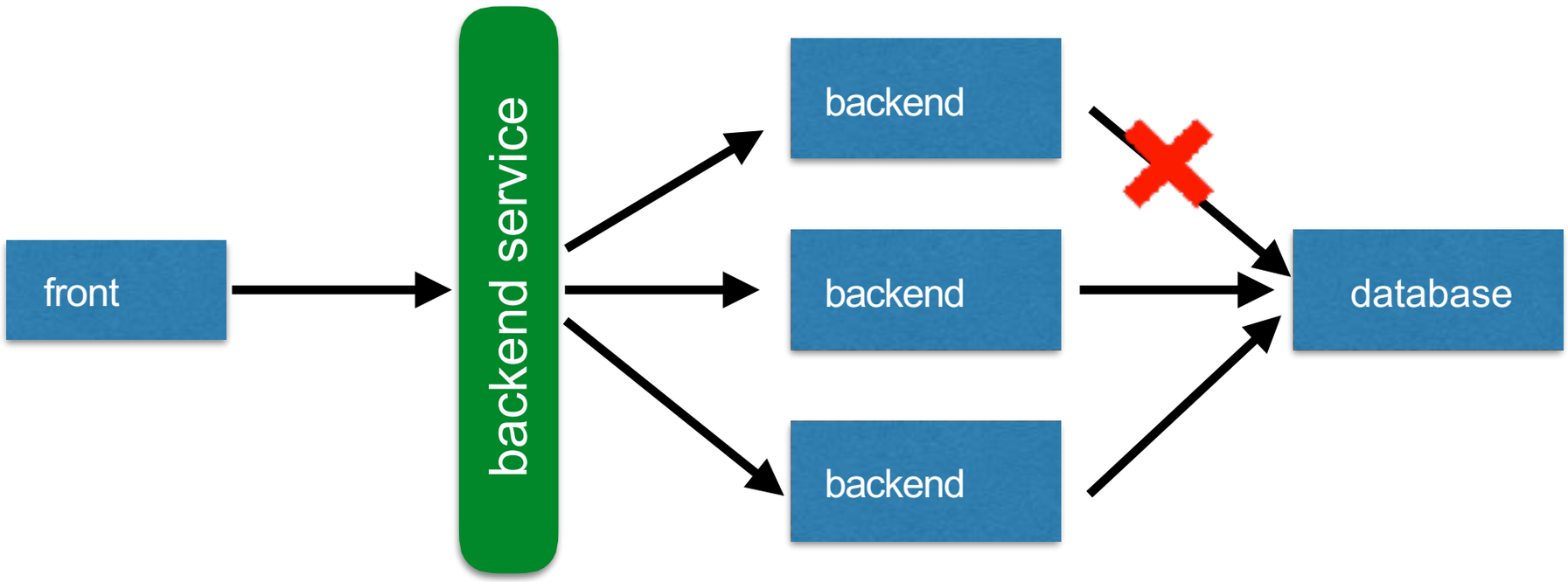
После успешного прохождения пробы – управление передается livenessProbe

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

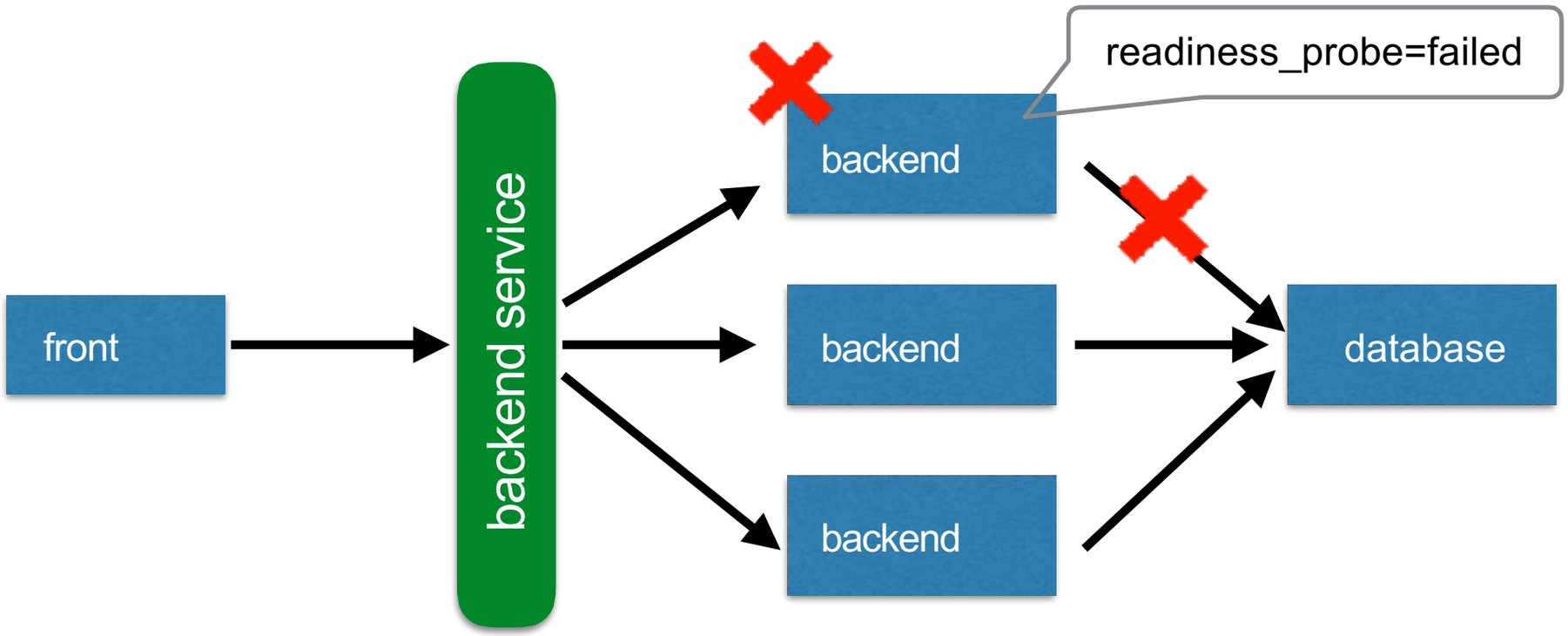
Readiness Probe



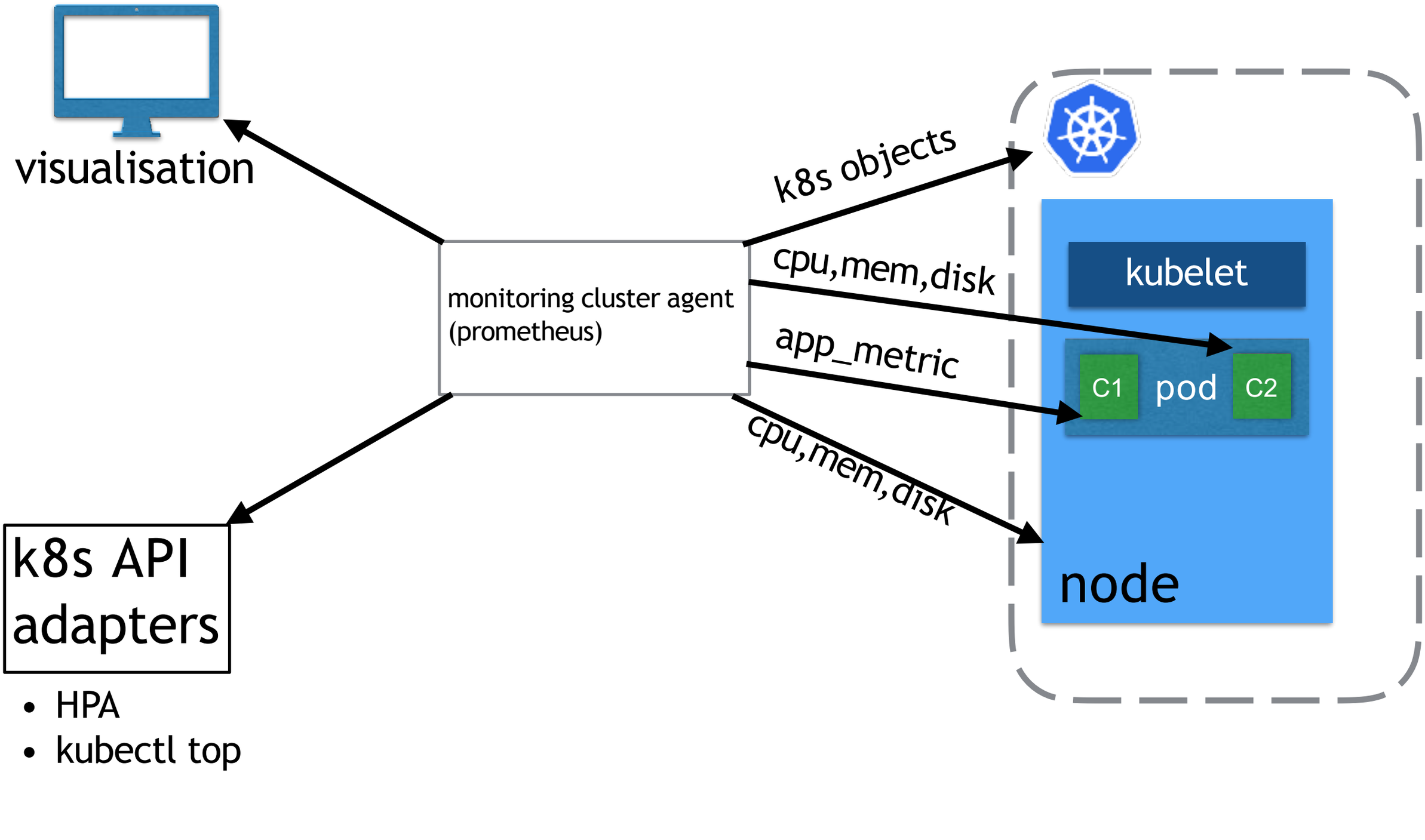
Readiness Probe



Readiness Probe



Monitoring Pipeline

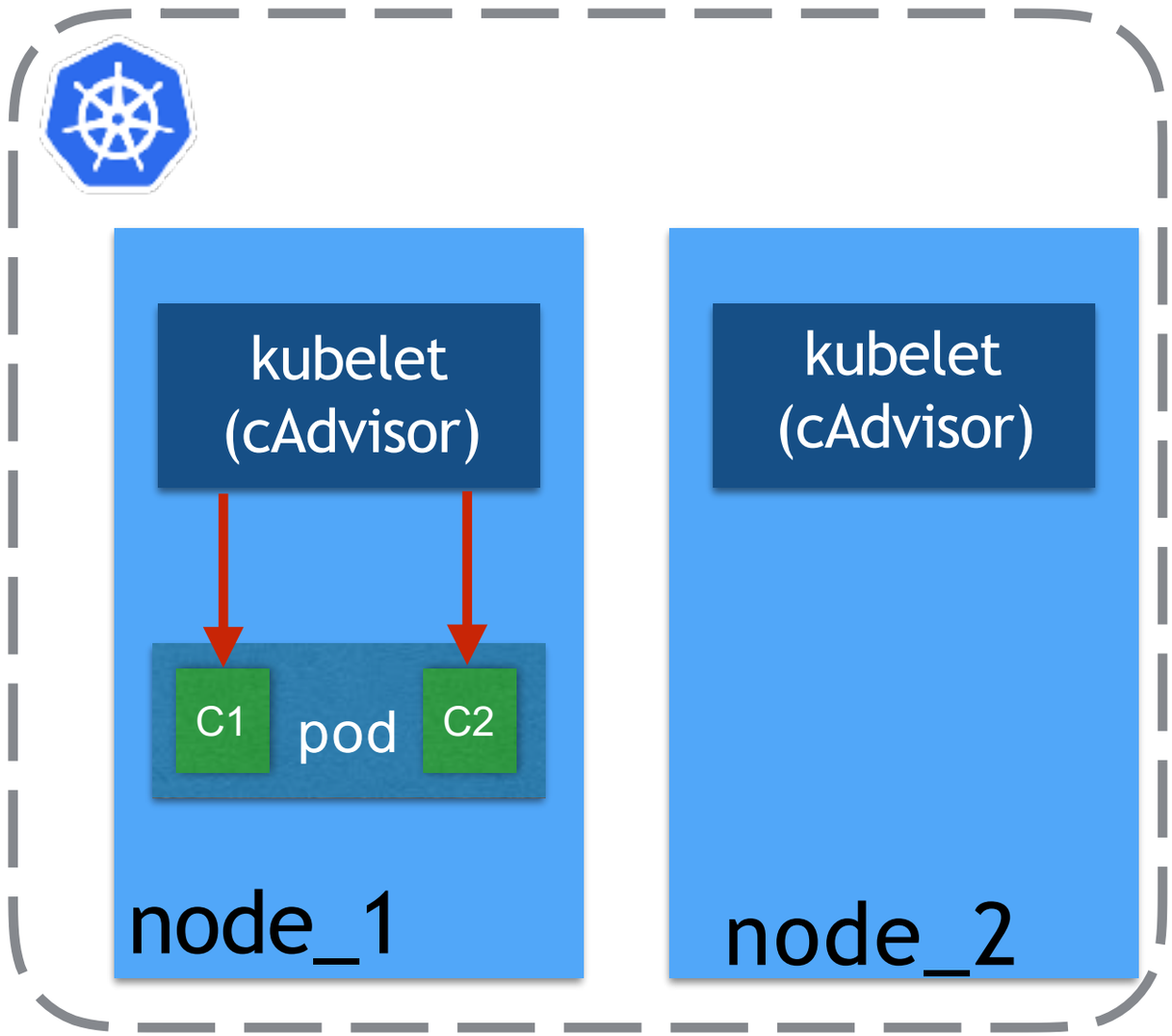


cAdvisor

- Встроен в kubelet
- Получает информацию о контейнерах от CRI (CPU / Memory / Network / I/O)
- Предоставляет только текущее состояние использования ресурсов и метрики производительности
- Работает как Exporter для Prometheus
- Для сбора метрик используется metrics-server

cAdvisor

cAdvisor собирает у Docker метрики контейнеров



kube-state-metrics

Собирает информацию о логических объектах k8s

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

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
- **Service Discovery**

Prometheus

Источники метрик для Prometheus:

- Метрики kube-state-metrics
- Метрики приложений и сервисов
- Метрики хостов, передаваемых через Node-Exporter
- Все что захотите сами ☺

Prometheus

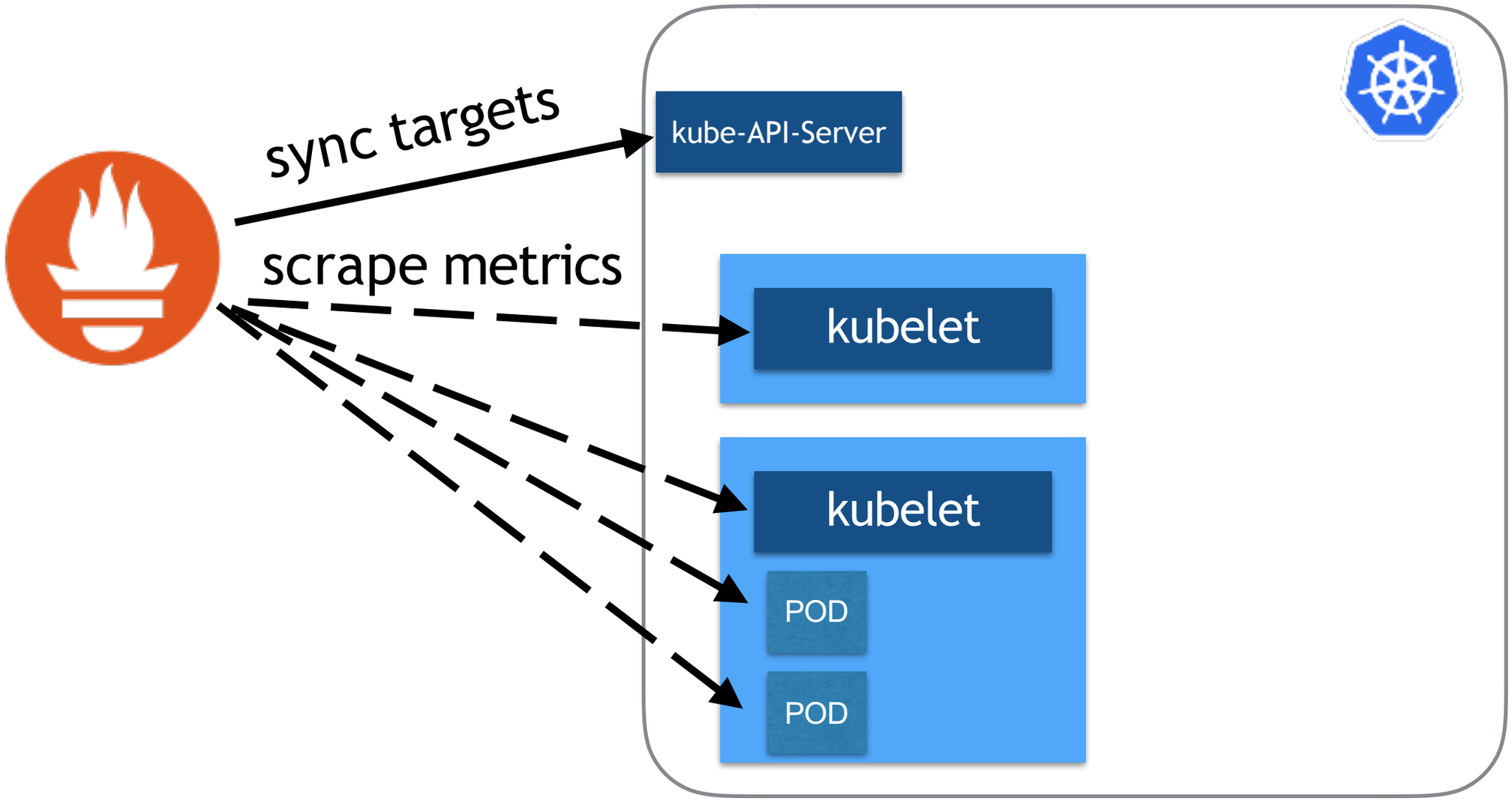
Источники метрик для Prometheus:

- Метрики etcd (**/metrics**)
- Метрики API servers (**/metrics**)
- Метрики cAdvisor (**/api/v1/nodes/node-X/proxy/metrics/cadvisor**)

Термины:

- Targets (endpoint) - источник для сбора метрик
- Jobs: группы источников

Prometheus


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

Prometheus

Процедура сборки метрик:

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

-  job_name: 'kube-apiservices'
   kubernetes_sd_configs:
     …

-  job_name: 'kube-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

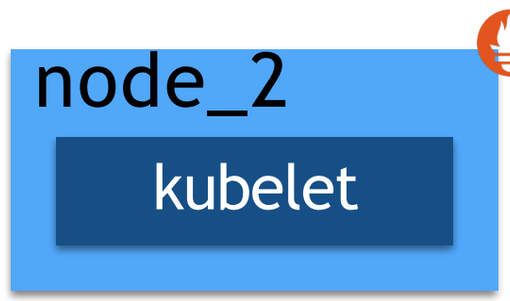
Prometheus

$ cat prometheus.yml

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

Targets:

- node_1
- node_2

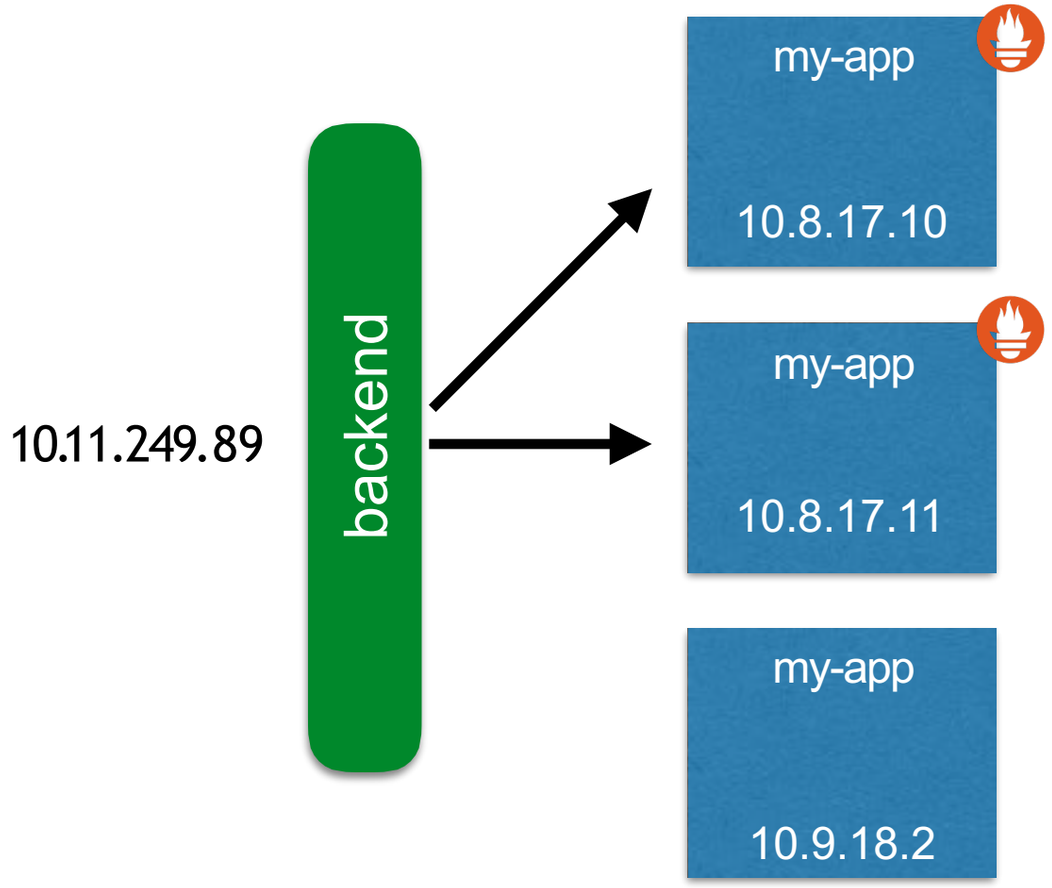


Prometheus

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

Targets:

- 10.8.17.10
- 10.8.17.11

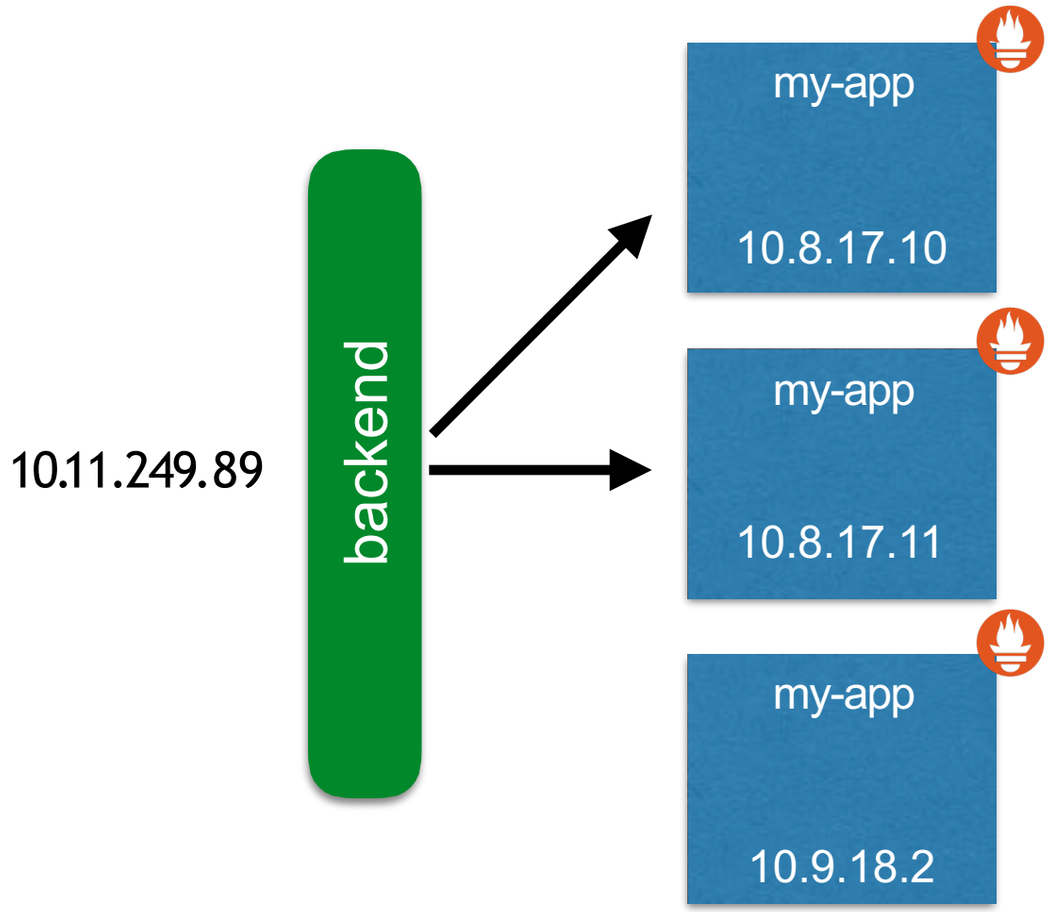


Prometheus

```
$ 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

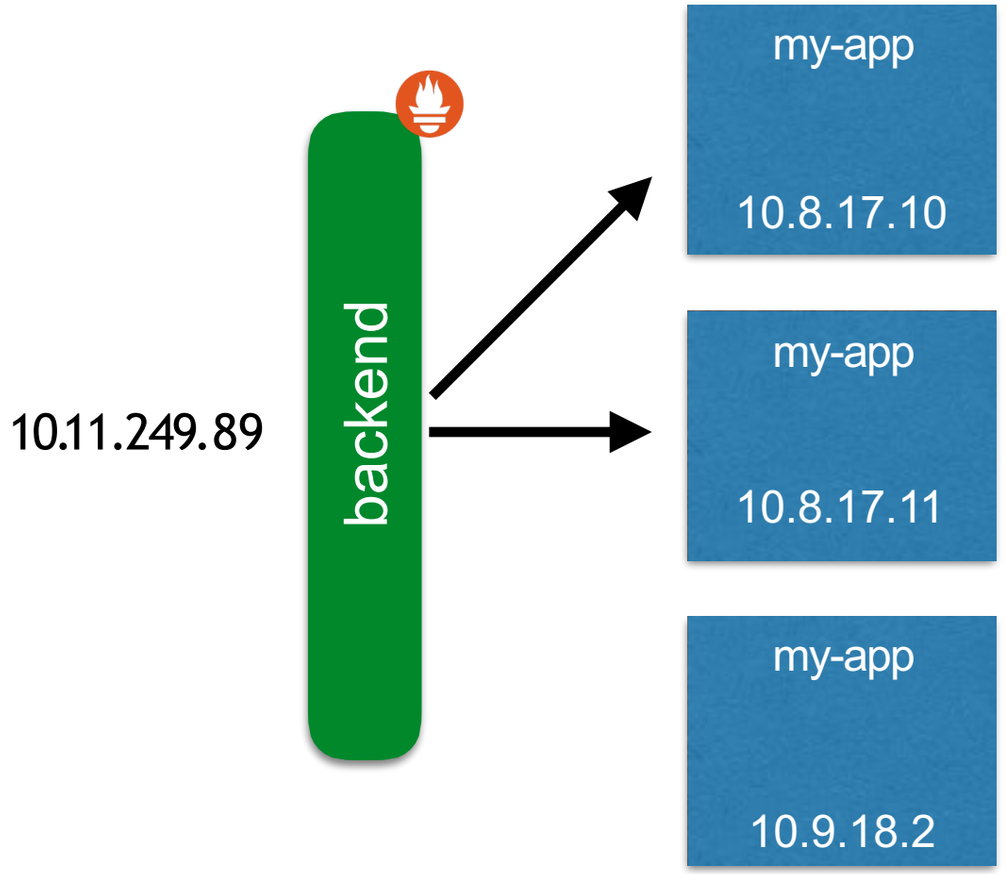


Prometheus

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

Targets:

- 10.11.249.89

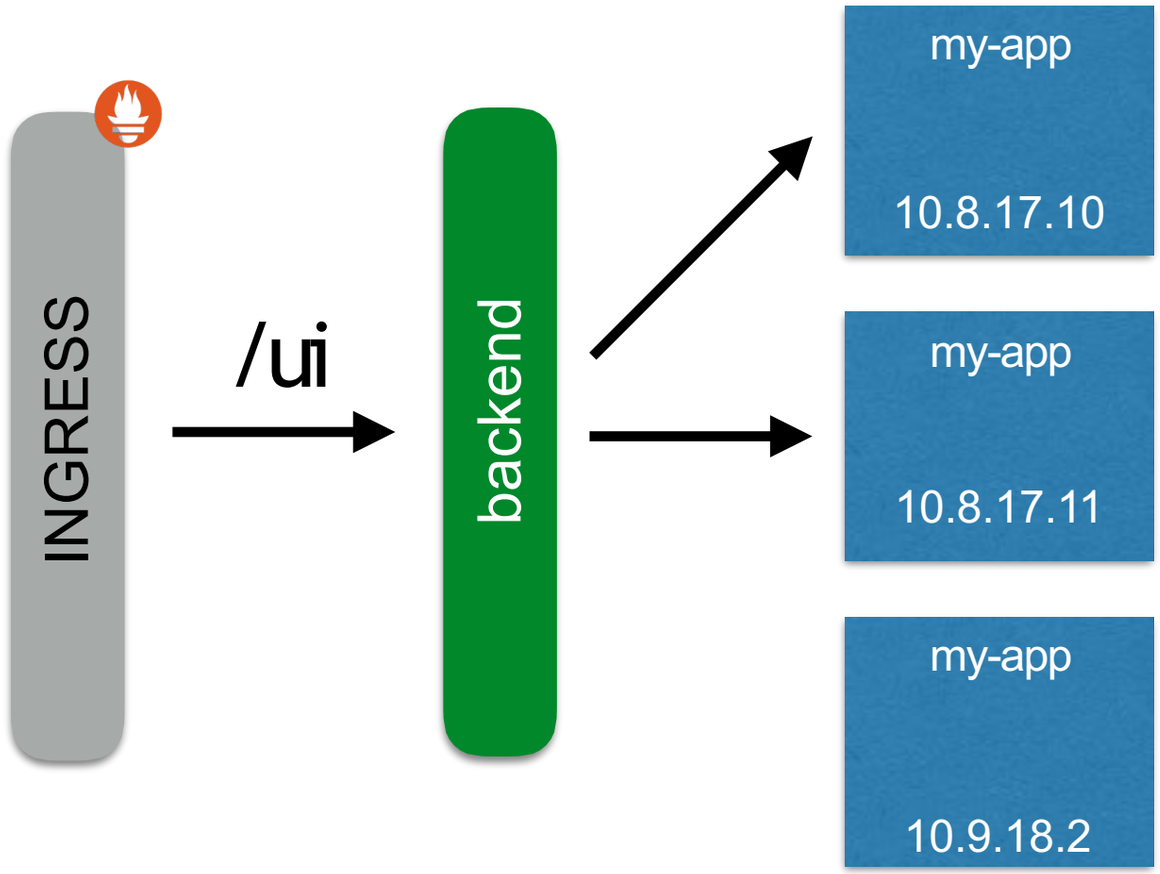


Prometheus

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

Targets:

- ui_service/ui



Prometheus

```
- job_name: serviceMonitor/mon-kube-prometheus-stack-apiserver/0
  honor_timestamps: true
  scrape_interval: 30s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: https
  authorization:
    type: Bearer
    credentials_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    server_name: kubernetes
    insecure_skip_verify: false
follow_redirects: true
enable_http2: true
```

Prometheus

Service Discovery

Filter by labels

- kubernetes-service-endpoints (5 / 36 active targets)
- serviceMonitor/observability/mon-grafana/0 (1 / 1 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-apiserver/0 (1 / 1 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-coredns/0 (1 / 1 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/0 (2 / 2 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/1 (2 / 2 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/2 (2 / 2 active targets)
- serviceMonitor/observability/mon-kube-prometheus-stack-operator/0 (1 / 1 active targets)
- serviceMonitor/observability/mon-kube-prometheus-s
- serviceMonitor/observability/mon-kube-state-metrics/
- serviceMonitor/observability/mon-prometheus-node-e
- undefined (0 / 153 active targets)

Targets

All | Unhealthy | Expand All

Filter by endpoint or labels

kubernetes-service-endpoints (5/5 up) show more

serviceMonitor/observability/mon-grafana/0 (1/1 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-apiserver/0 (1/1 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-coredns/0 (1/1 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/0 (2/2 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/1 (2/2 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-kubelet/2 (2/2 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-operator/0 (1/1 up) show more

serviceMonitor/observability/mon-kube-prometheus-stack-prometheus/0 (1/1 up) show more

serviceMonitor/observability/mon-kube-state-metrics/0 (1/1 up) show more

serviceMonitor/observability/mon-prometheus-node-exporter/0 (1/1 up) show more

Prometheus

Добавляем приложение в мониторинг prometheus

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  labels:
    app: my-app
...
  template:
    annotations:
      prometheus.io/port: '8080’
      prometheus.io/scrape: 'true’
      prometheus.io/path: '/'
```

kubectl top

Текущий статус потребления ресурсов

```
$ kubectl top nodes
```

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

```
$ kubectl top pod
```

NAME	CPU(cores)	MEMORY(bytes)
running-prometheus-server-5fc8847448-24xnw	14m	565Mi
my-app-54b7cbcc69-9md82	4m	41Mi
my-app-54b7cbcc69-mhj5r	4m	42Mi

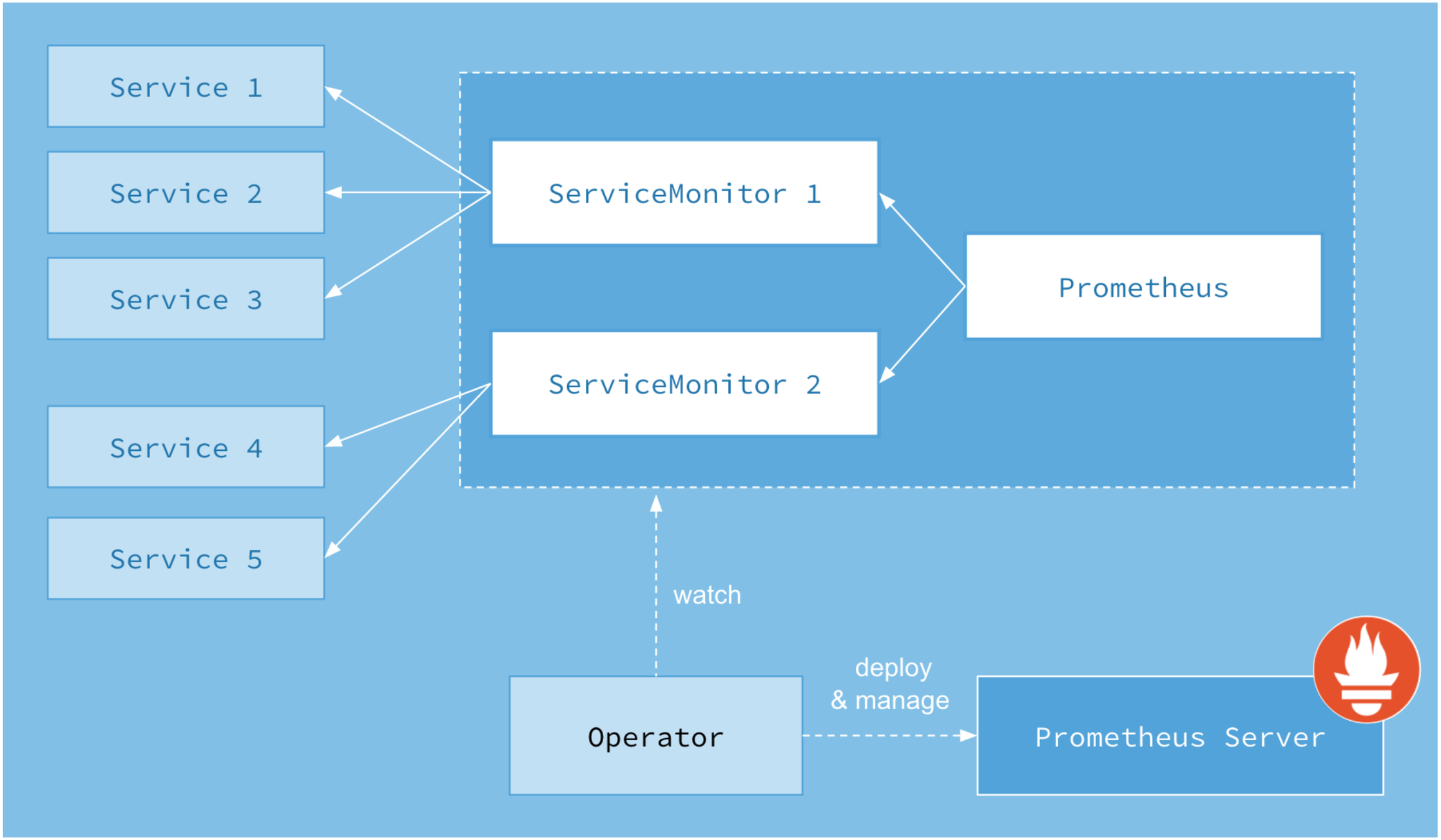
Операторы

An Operator represents human operational knowledge in software, to reliably manage an application.

CoreOS

- etcd Operator
- Rook Operator
- Prometheus Operator
- Vault Operator

Prometheus operator



Prometheus operator

```
kubectl apply -f prometheus-operator.yml
```

```
$ kubectl get crd
NAME                                    AGE
alertmanagers.monitoring.coreos.com     39m
backendconfigs.cloud.google.com         1h
prometheuses.monitoring.coreos.com      39m
prometheusrules.monitoring.coreos.com   39m
servicemonitors.monitoring.coreos.com   39m
```

Prometheus operator

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: my-app
spec:
selector:
  matchLabels:
    app: my-app
endpoints:
- path: /metrics
  port: metrics
  interval: 10s
```

Logging

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

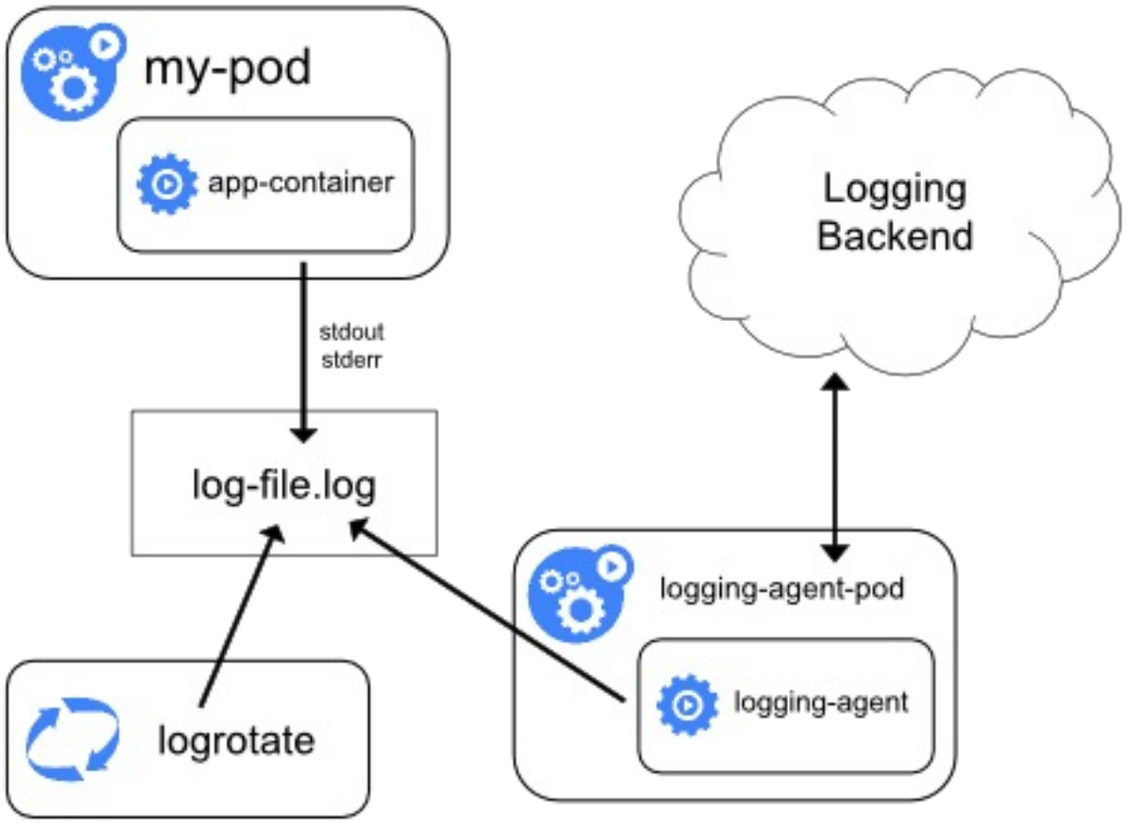
У Kubernetes нет встроенных механизмов для отправки логов (таких как logging drivers в Docker)

Что логировать?

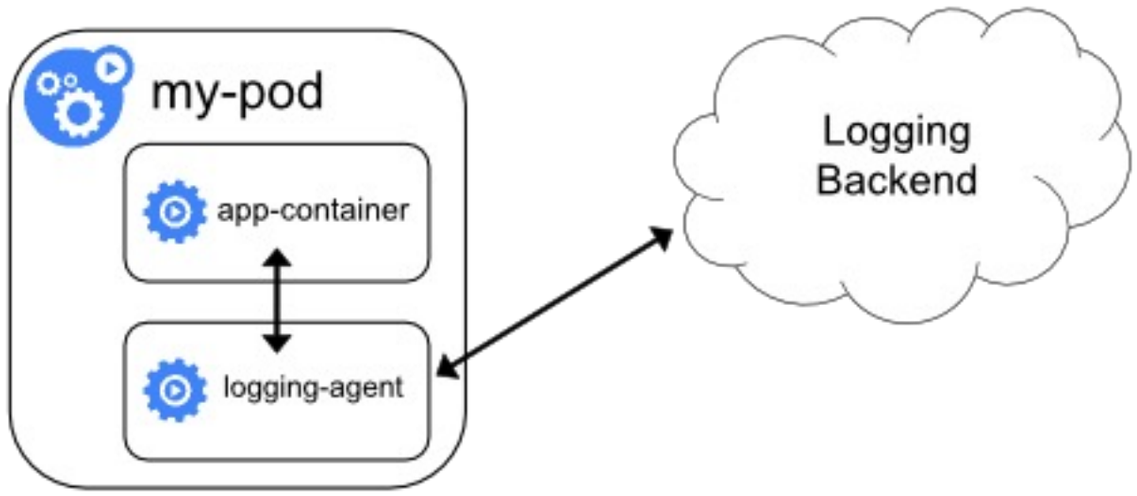
- Логи контейнеров
- Логи хостовых систем
- …

Как логировать?

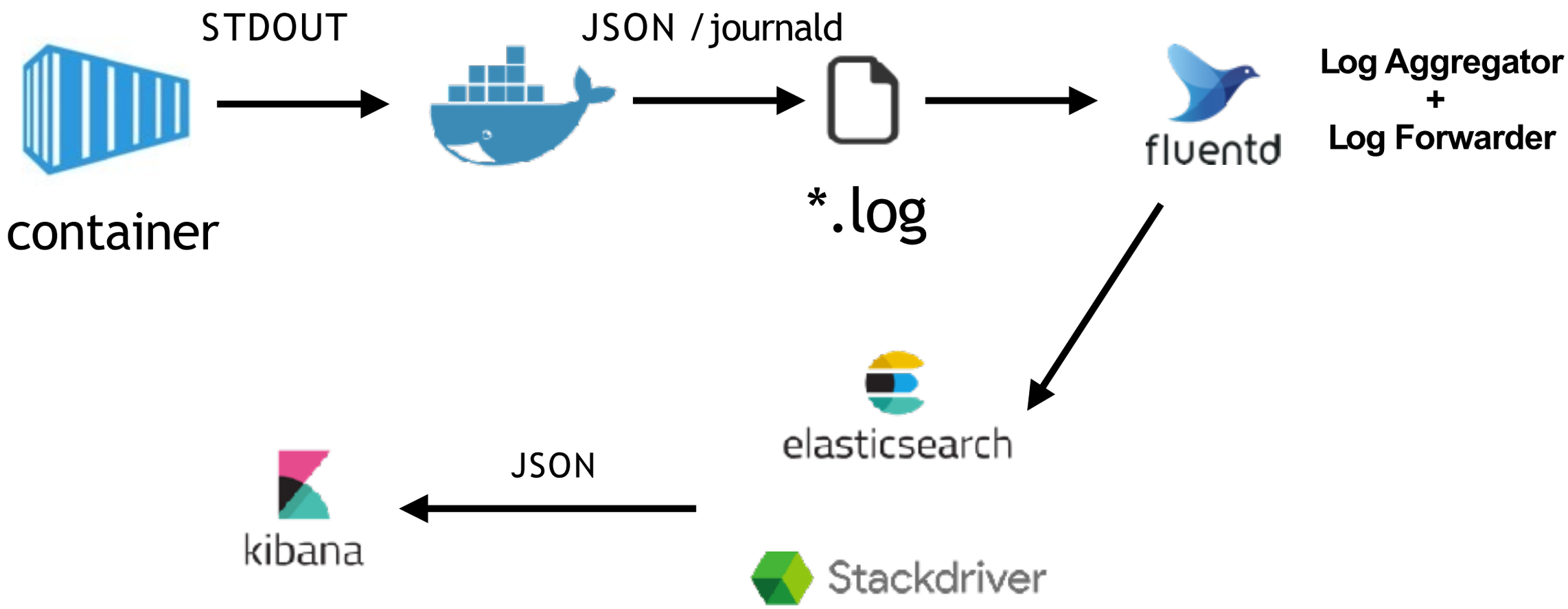
External pod



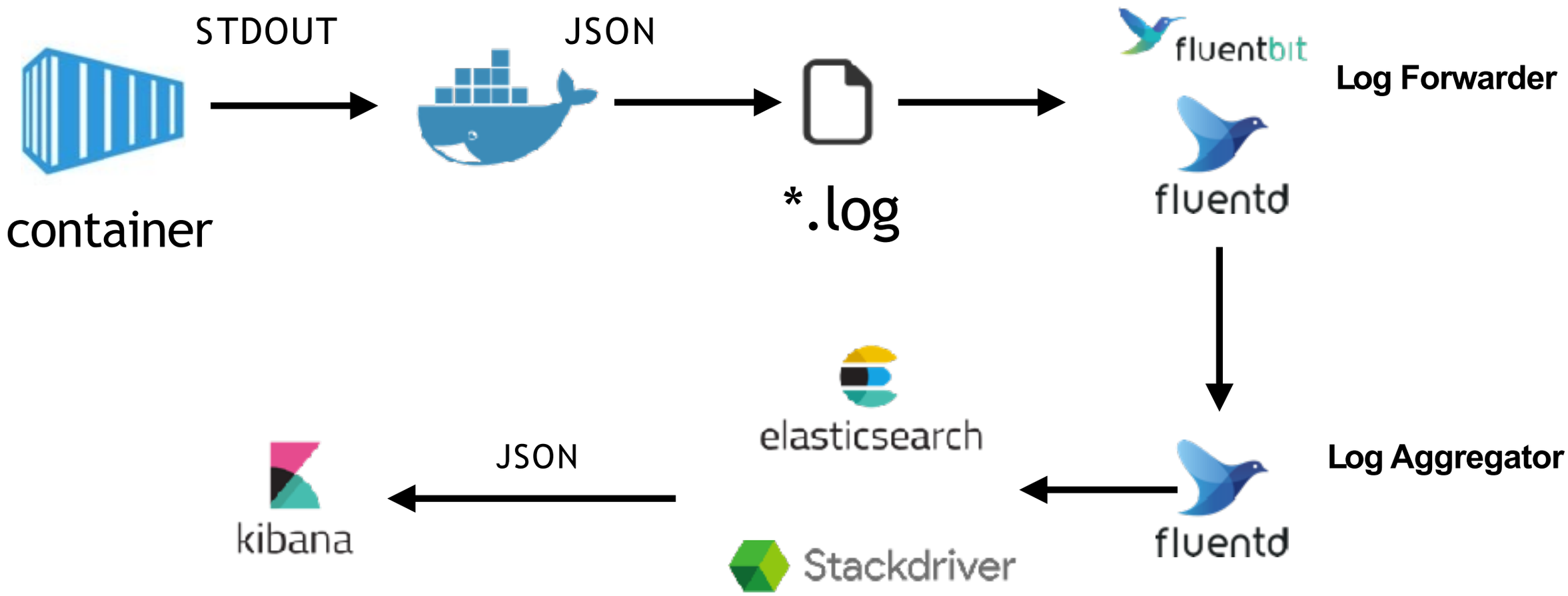
Из приложения / sidecar



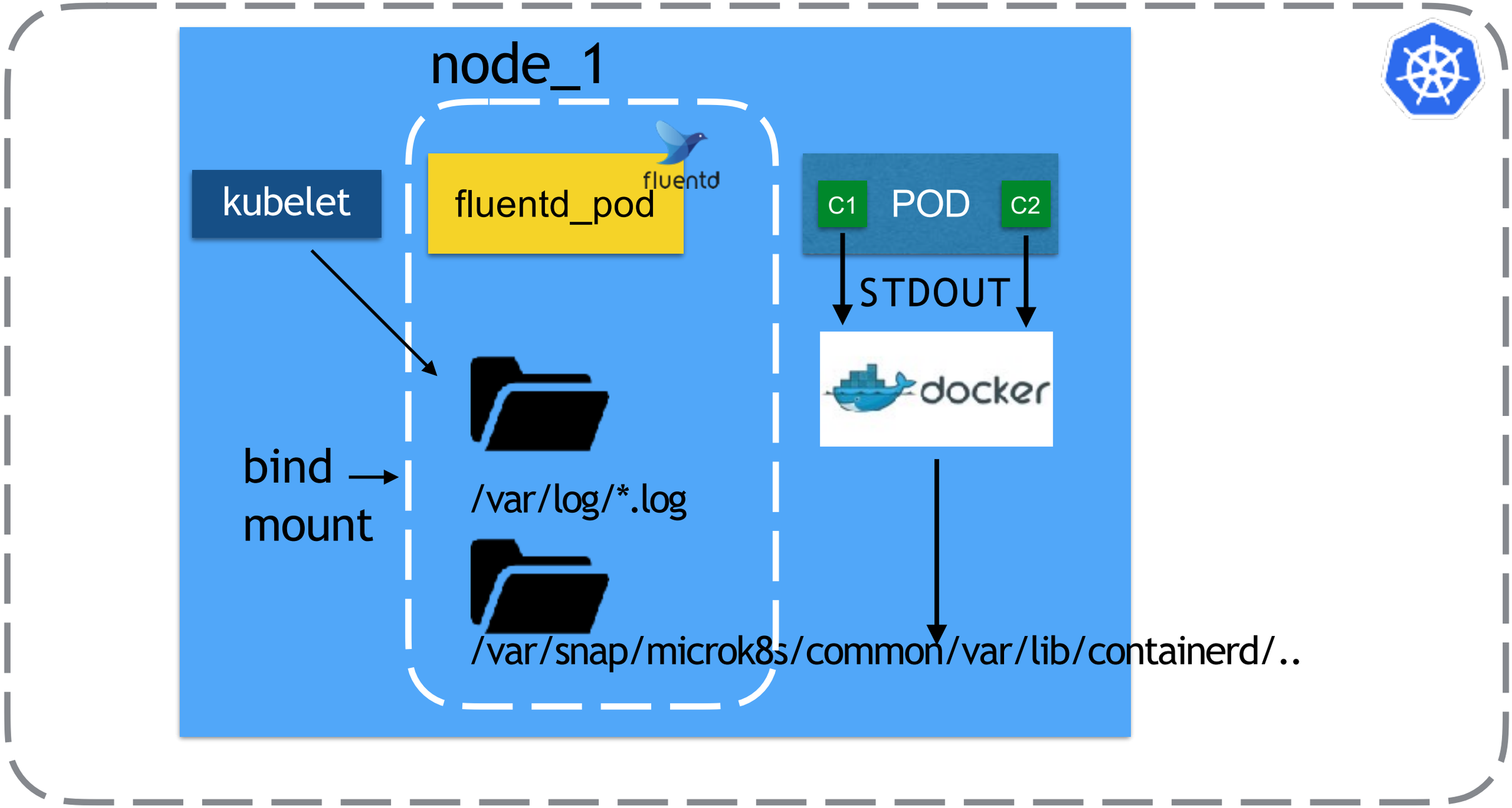
Как логировать?



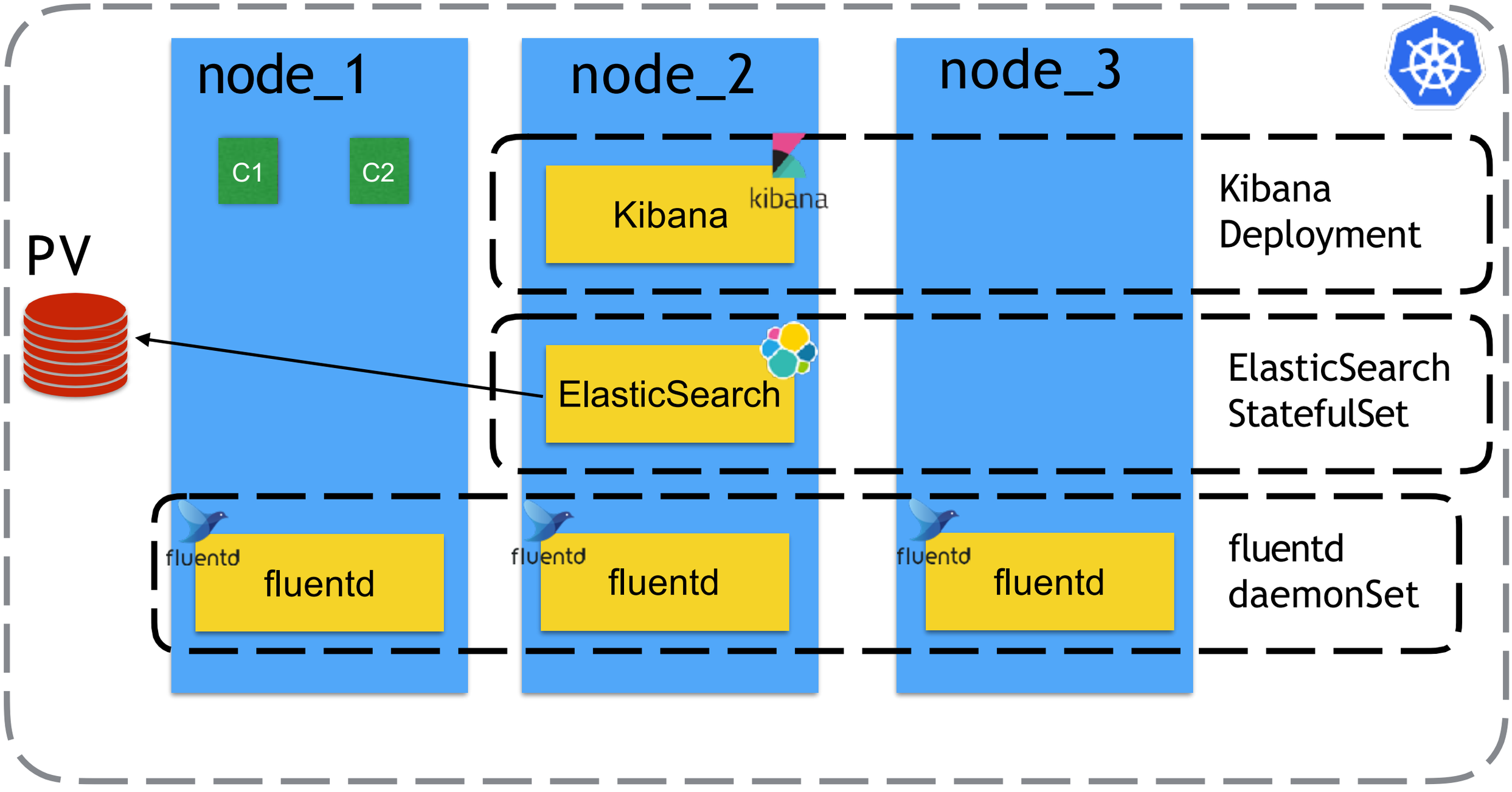
Как логировать?



Как логировать?



Как логировать?



Что почитать

- Alertmanager: https://prometheus.io/docs/alerting/latest/alertmanager/
- Loki: https://grafana.com/oss/loki/
- Push Gateway: https://github.com/prometheus/pushgateway
- Blackbox exporter: https://github.com/prometheus/blackbox_exporter
- Victoria Metrics: https://docs.victoriametrics.com