Kubernetes. Мониторинг и логирование

Подготовка

Создайте новую ветку в репозитории **Microservices** для выполнения данного ДЗ. Т.к. это пятое задание по Kubernetes, то назовите ее **kubernetes-5**

Проверка данного ДЗ будет производиться через Pull Request (PR назовите **kubernetes-5)**

Поле с Reviewers следует оставить пустым

После того, как один из преподавателей сделает approve пул реквеста, ветку с ДЗ можно смерджить.

Подготовка

У вас должен быть развернуть кластер k8s:

- минимум 2 ноды g1-small (1,5 ГБ)
- минимум 1 нода n1-standard-2 (7,5 ГБ)

В настройках:

- Stackdriver Logging - Отключен
- Stackdriver Monitoring - Отключен
- Устаревшие права доступа - Включено

Подготовка

Из Helm-чарта установим ingress-контроллер nginx

```
$ helm install stable/nginx-ingress --name nginx
```

Найдите IP-адрес, выданный nginx'у

```
$ kubectl get svc
NAME                                    TYPE           CLUSTER-IP      EXTERNAL-IP
kubernetes                              ClusterIP      10.11.240.1     <none>
nginx-nginx-ingress-controller          LoadBalancer   10.11.243.76    104.154.96.52
```

Добавьте в /etc/hosts

```
104.154.96.52  reddit reddit-prometheus reddit-grafana reddit-non-prod production reddit-kibana staging prod
```

План

- Развертывание Prometheus в k8s
- Настройка Prometheus и Grafana для сбора метрик
- Настройка EFK для сбора логов

Мониторинг

Стек

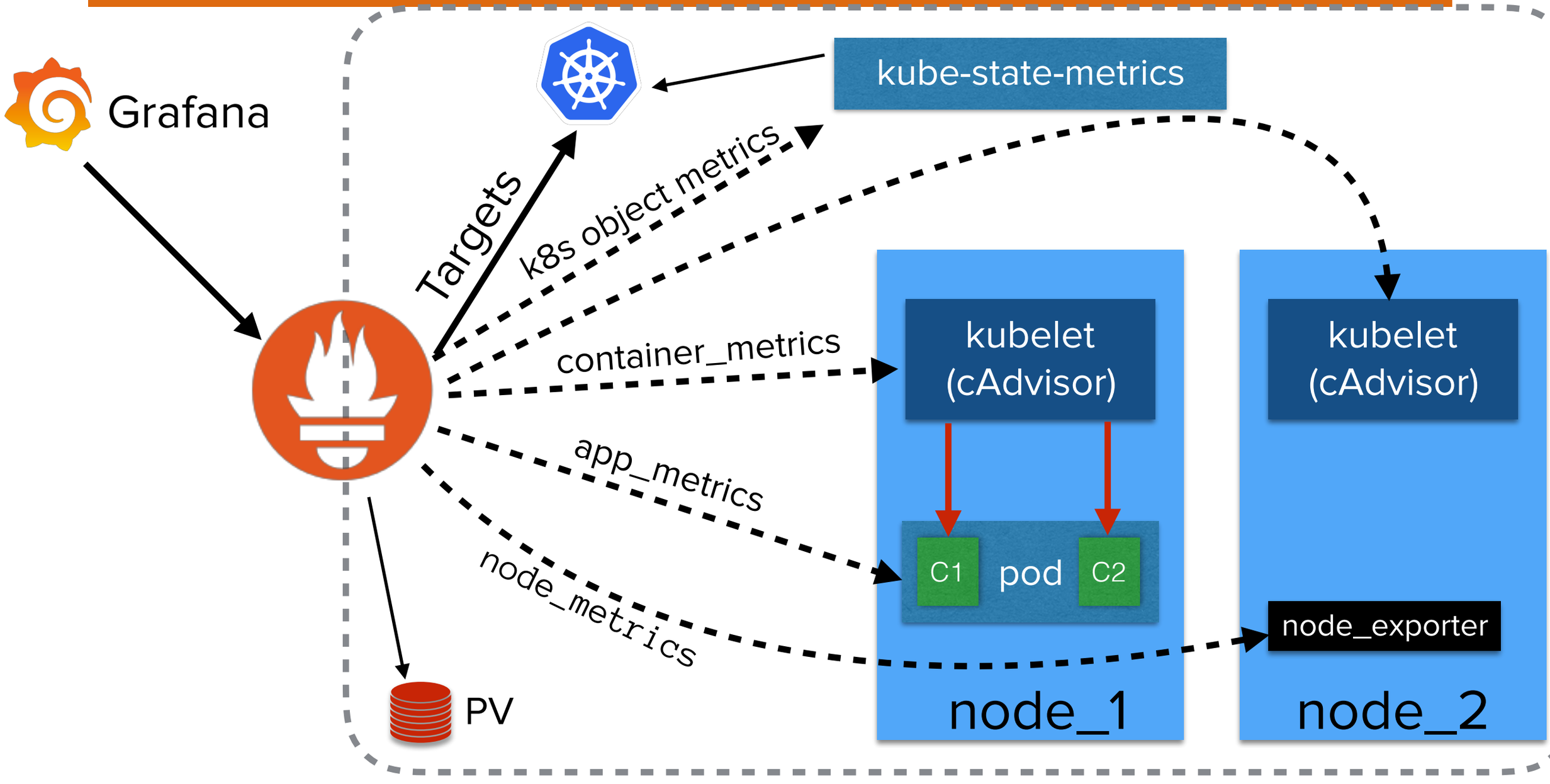
В задании будем использовать уже знакомые нам инструменты:

- prometheus - сервер сбора и
- grafana - сервер визуализации метрик
- alertmanager - компонент prometheus для алертинга
- различные экспортеры для метрик prometheus

Prometheus отлично подходит для работы с контейнерами и динамичным размещением сервисов

Общая схема работы представлена на следующем слайде

Monitoring Pipeline



Установим Prometheus

Prometheus будем ставить с помощью Helm чарта

Загрузим prometheus локально в Charts каталог

```
$ cd kubernetes/charts && helm fetch --untar stable/prometheus
```

Установим Prometheus

Создайте внутри директории чарта файл **custom_values.yml**.
Поместите в него содержимое (ссылка на gist)

Основные отличия от **values.yml**:

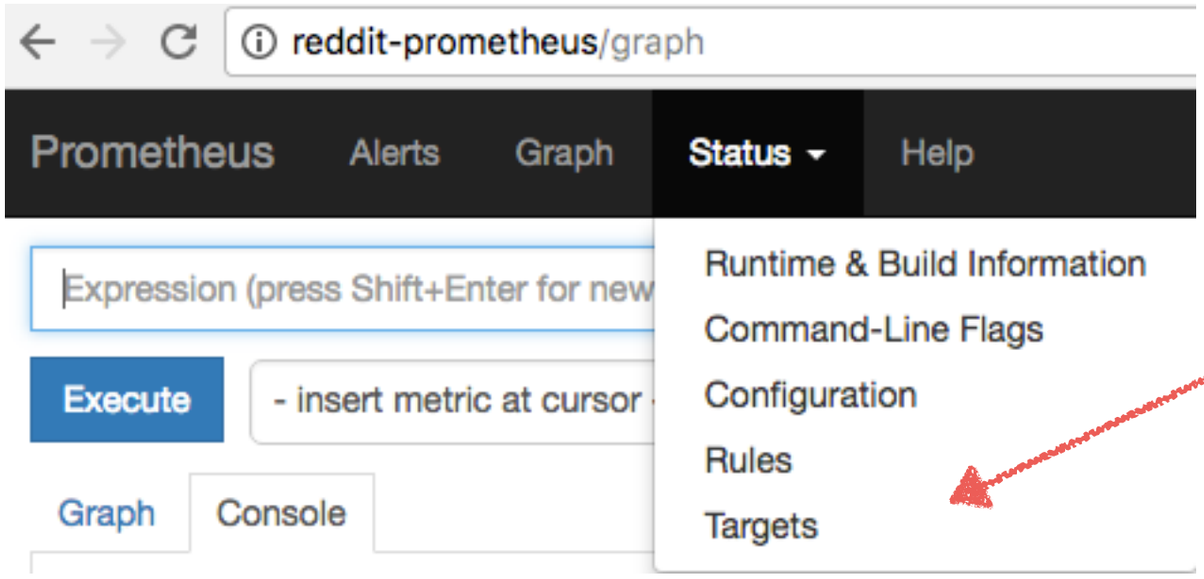
- отключена часть устанавливаемых сервисов (pushgateway, alertmanager, kube-state-metrics)
- включено создание Ingress'а для подключения через nginx
- поправлен endpoint для сбора метрик cadvisor
- уменьшен интервал сбора метрик (с 1 минуты до 30 секунд)

Установим Prometheus

Запустите Prometheus в k8s из charsts/prometheus

```
$ helm upgrade prom . -f custom_values.yml --install
```

http://reddit-prometheus



Сюда

Targets

У нас уже присутствует ряд endpoint’ов для сбора метрик

- Метрики API-сервера
- метрики нод с cadvisor’ов
- сам prometheus

Targets

kubernetes-apiservers (1/1 up)

Endpoint	State
https://35.202.145.198:443/metrics	UP

kubernetes-nodes (3/3 up)

Endpoint	State	Labels
https://kubernetes.default.svc:443/api/v1/nodes/gke-cluster-1-big-pool-b4209075-jlnq/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" failure_domain_beta_kubernetes_io kubernetes_io_hostname="gke-clus
https://kubernetes.default.svc:443/api/v1/nodes/gke-cluster-1-default-pool-f9c66281-8gkc/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" failure_domain_beta_kubernetes_io kubernetes_io_hostname="gke-clus

Targets

Отметим, что можно собирать метрики cadvisor'а (который уже является частью kubelet) через проксирующий запрос в kube-api-server

```
https://kubernetes.default.svc:443/api/v1/nodes/gke-
cluster-1-big-pool-b4209075-
jlnq/proxy/metrics/cadvisor
```

Если зайти по ssh на любую из машин кластера и запросить `$ curl http://localhost:4194/metrics` то получим те же метрики у kubelet напрямую

Но вариант с kube-api предпочтительней, т.к. этот трафик шифруется TLS и требует аутентификации.

Targets

Таргеты для сбора метрик найдены с помощью **service discovery** (SD), настроенного в конфиге prometheus (лежит в **custom-values.yml**)

```
prometheus.yml:
      …
      - job_name: 'kubernetes-apiservers'
      …
      - job_name: 'kubernetes-nodes'
        kubernetes_sd_configs:
          - role: node

        scheme: https
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
          insecure_skip_verify: true
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token

        relabel_configs:
…
```

kubernetes-apiservers (1/1 up)

kubernetes-nodes (3/3 up)

Настройки Service Discovery (для поиска target'ов)

Настройки подключения к target'ам (для сбора метрик)

Настройки различных меток, фильтрация найденных таргетов, их изменение

Targets

Использование **SD** в kubernetes позволяет нам динамично менять кластер (как сами хосты, так и сервисы и приложения)
Цели для мониторинга находим с помощью запросов к k8s API:

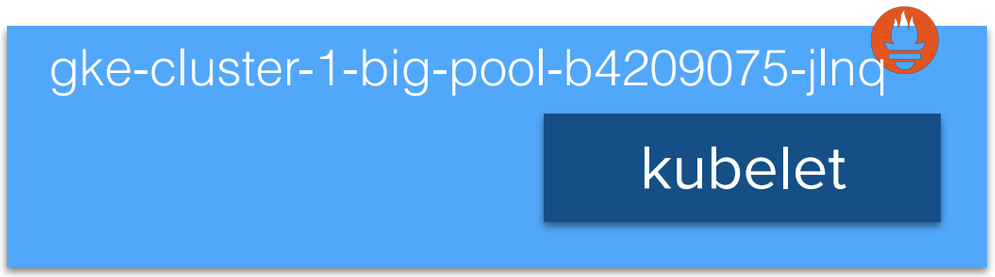
```
prometheus.yml:
  …
    scrape_configs:
    - job_name: 'kubernetes-nodes'
        kubernetes_sd_configs:
          - role: node
```

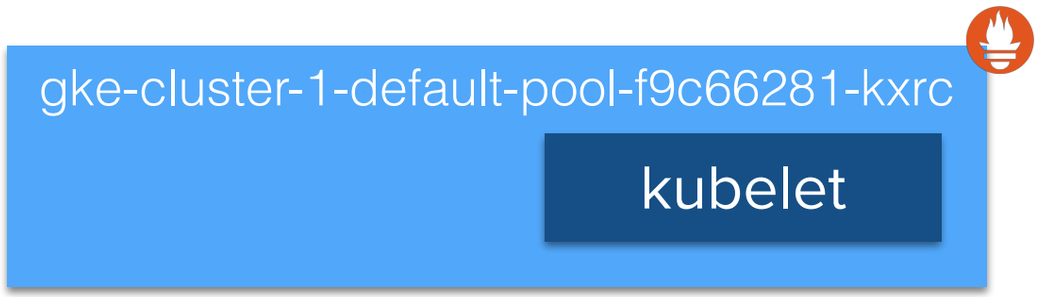
Role объект, который нужно найти:

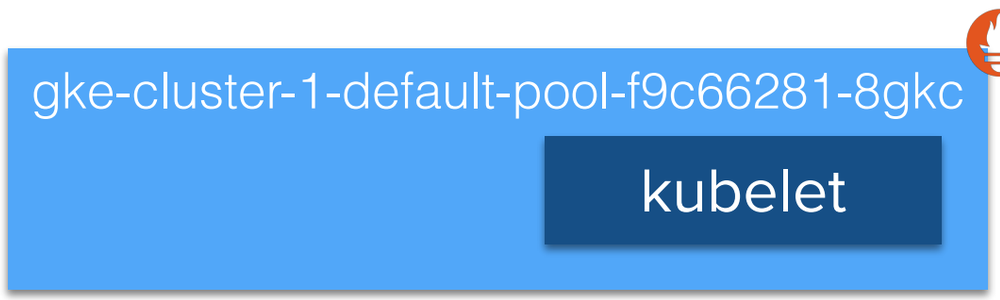
- node
- endpoints
- pod
- service
- ingress

Targets

```
prometheus.yml
…
scrape_configs:
   - job_name: 'kubernetes-nodes'
     kubernetes_sd_configs:
       - role: node
```







Targets

Т.к. сбор метрик prometheus осуществляется поверх стандартного HTTP-протокола, то могут понадобится доп. настройки для безопасного доступа к метрикам.
Ниже приведены настройки для сбора метрик из k8s API

```
scheme: https
tls_config:
  ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
  insecure_skip_verify: true
bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

1. Схема подключения - http (default) или https
2. Конфиг TLS - коревой сертификат сервера для проверки достоверности сервера
3. Токен для аутентификации на сервере

Targets:

- gke-cluster-1-default-pool-f9c66281-kxrc
- gke-cluster-1-default-pool-f9c66281-8gkc
- gke-cluster-1-big-pool-b4209075-jlnq

Подробнее о том, как работает relabel_config

1) преобразовать все k8s лейблы таргета в лейблы prometheus

2) Поменять лейбл для адреса сбора метрик

3) Поменять лейбл для пути сбора метрик

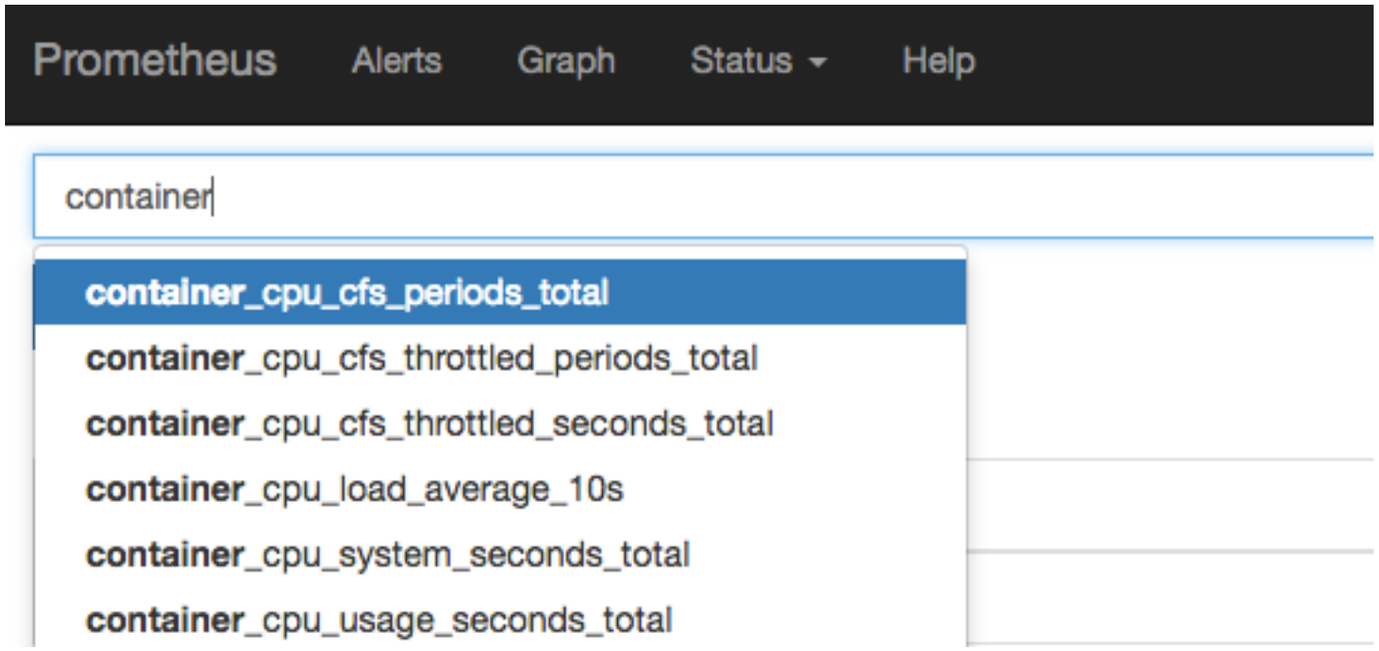
```
#Kubernetes nodes
relabel_configs:
    - action: labelmap
      regex: __meta_kubernetes_node_label_(.+)
    - target_label: __address__
      replacement: kubernetes.default.svc:443
    - source_labels: [__meta_kubernetes_node_name]
      regex: (.+)
      target_label: __metrics_path__
      replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor
```

kubernetes-nodes (3/3 up)

Endpoint	State	Labels	Last Scrape	Error
https://kubernetes.default.svc:443/api/v1/nodes/gke-cluster-1-big-pool-b4209075-jlnq/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_fluentd_ds_ready="true" beta_kubernetes_io_instance_type="n1-standard-2" beta_kubernetes_io_os="linux" cloud_google_com_gke_nodepool="big-pool" failure_domain_beta_kubernetes_io_region="us-central1" failure_domain_beta_kubernetes_io_zone="us-central1-a" instance="gke-cluster-1-big-pool-b4209075-jlnq" kubernetes_io_hostname="gke-cluster-1-big-pool-b4209075-jlnq"	6.88s ago	
https://kubernetes.default.svc:443/api/v1/nodes/gke-cluster-1-default-pool-f9c66281-8gkc/proxy/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_fluentd_ds_ready="true" beta_kubernetes_io_instance_type="g1-small" beta_kubernetes_io_os="linux" cloud_google_com_gke_nodepool="default-pool" failure_domain_beta_kubernetes_io_region="us-central1" failure_domain_beta_kubernetes_io_zone="us-central1-a" instance="gke-cluster-1-default-pool-f9c66281-8gkc" kubernetes_io_hostname="gke-cluster-1-default-pool-f9c66281-8gkc"	11.777s ago	

Metrics

Все найденные на эндпоинтах метрики сразу же отобразятся в списке (вкладка **Graph**). Метрики Cadvisor начинаются с **container_.**



Metrics

Cadvisor собирает лишь информацию о потреблении ресурсов и производительности отдельных docker-контейнеров. При этом он ничего не знает о сущностях k8s (деплойменты, репликасеты, ...).

Для сбора этой информации будем использовать сервис **kube-state-metrics.** Он входит в чарт Prometheus. Включим его. prometheus/**custom_values.yml**

```
…
kubeStateMetrics:
  ## If false, kube-state-metrics will not be installed
  ##
  enabled: true
```

Обновим релиз

```
$ helm upgrade prom . -f custom_values.yml --install
```

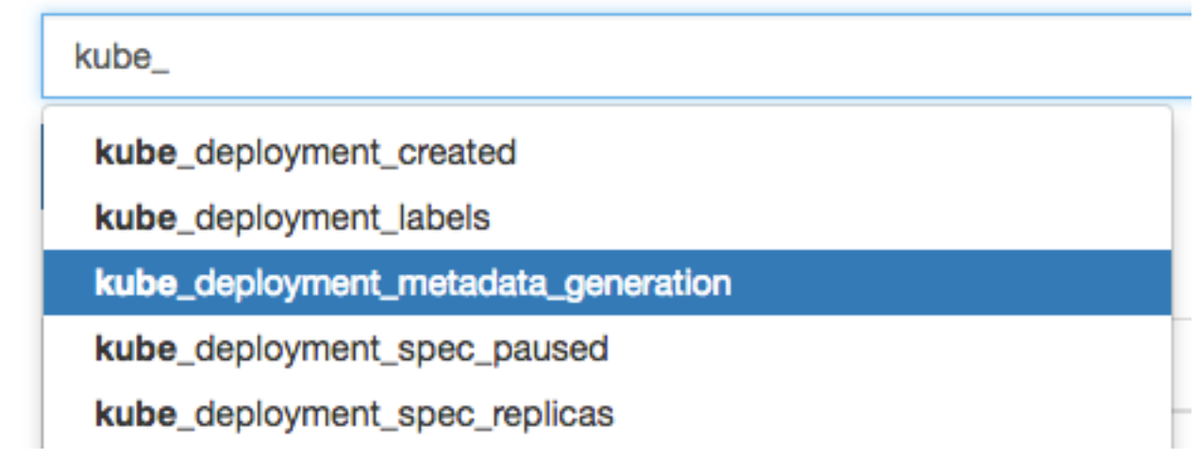
Metrics

в Targets

kubernetes-service-endpoints (1/1 up)

Endpoint	State	Labels
http://10.8.2.9:8080/metrics	UP	app="prometheus" chart="prometheus-5.0.0" component="kube-state-metrics" heritage="Tiller" instance="10.8.2.9:8080" kubernetes_name="prom-prometheus-kube-state-metrics" kubernetes_namespace="default" release="prom"

в Graph



Задание

По аналогии с kube_state_metrics включите (enabled: true) поды node-exporter в **custom_values.yml**.

Проверьте, что метрики начали собираться с них.

Метрики приложений

Запустите приложение из helm чарта reddit

```
$ helm upgrade reddit-test ./reddit –install
$ helm upgrade production --namespace production  ./reddit --install
$ helm upgrade staging --namespace staging  ./reddit –install
```

Метрики приложений

Раньше мы “хардкодили” адреса/dns-имена наших приложений для сбора метрик с них.

prometheus.yml

```
- job_name: 'ui'
  static_configs:
    - targets:
      - 'ui:9292'

- job_name: 'comment'
  static_configs:
    - targets:
      - 'comment:9292'
```

Теперь мы можем использовать механизм ServiceDiscovery для обнаружения приложений, запущенных в k8s.

Метрики приложений

Приложения будем искать так же, как и служебные сервисы k8s.

Модернизируем конфиг prometheus

`custom_values.yml`

```
- job_name: 'reddit-endpoints'
   kubernetes_sd_configs:
     - role: endpoints
   relabel_configs:
     - source_labels: [__meta_kubernetes_service_label_app]
       action: keep
       regex: reddit
```

Используем действие keep, чтобы оставить только эндпоинты сервисов с метками “app=reddit”

Обновите релиз prometheus

```
$ helm upgrade prom . -f custom_values.yml --install
```

Метрики приложений

Мы получили эндпоинты, но что это за поды мы не знаем.
Добавим метки k8s
Все лейблы и аннотации k8s изначально отображаются в prometheus в формате:
__meta_kubernetes_service_label_**labelname**
__meta_kubernetes_service_annotation_**annotationname**

Отобразить все совпадения групп из regex в label'ы Prometheus

custom_values.yml

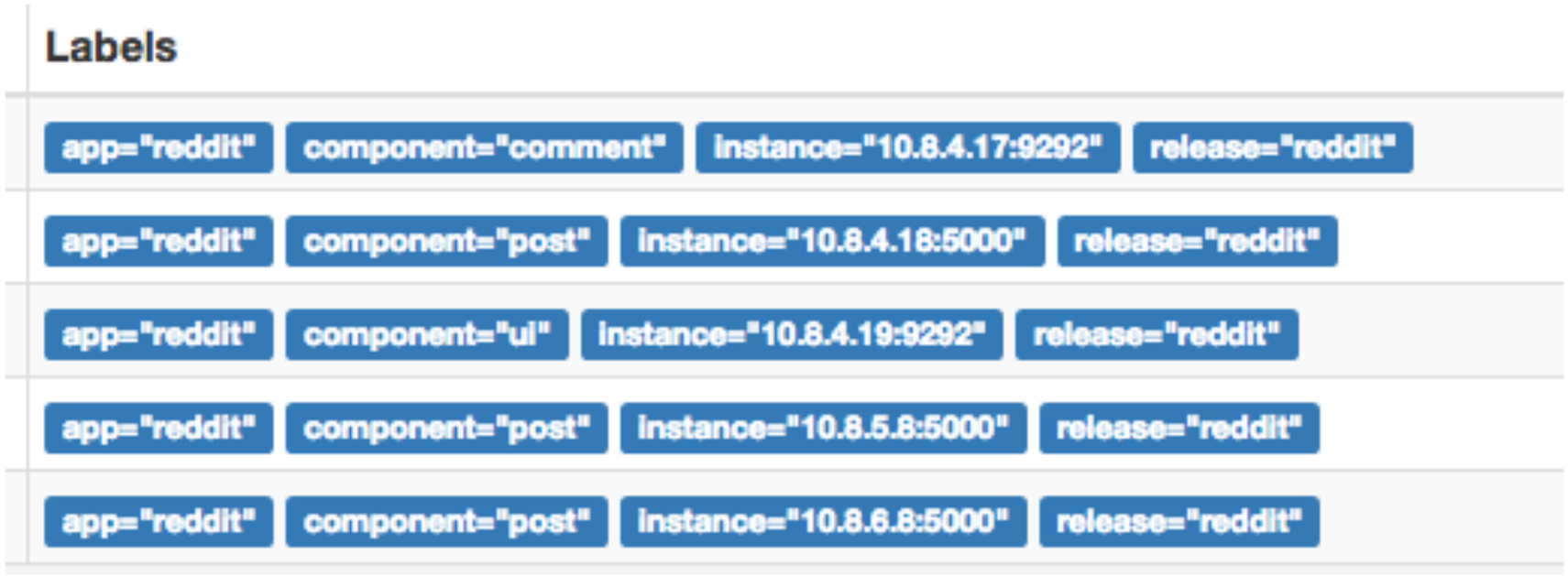
```
relabel_configs:
  - action: labelmap
    regex: __meta_kubernetes_service_label_(.+)
```

Обновите релиз prometheus

```
$ helm upgrade prom . -f custom_values.yml --install
```

Метрики приложений

Теперь мы видим лейблы k8s, присвоенные POD'ам



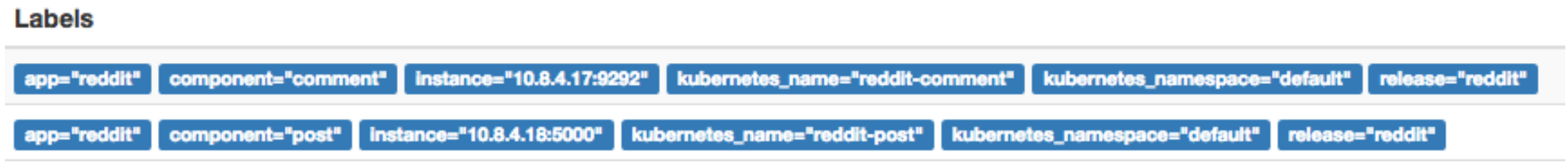
Метрики приложений

Добавим еще label'ы для prometheus и обновим helm-релиз

Т.к. метки вида __meta_* не публикуются, то нужно создать свои, перенеся в них информацию

```
- source_labels: [__meta_kubernetes_namespace]
  target_label: kubernetes_namespace
- source_labels: [__meta_kubernetes_service_name]
  target_label: kubernetes_name
```

Обновите релиз prometheus и …



Метрики приложений

Сейчас мы собираем метрики со всех сервисов reddit’a в 1 группе target-ов.
Мы можем отделить target-ы компонент друг от друга (по окружениям, по самим компонентам), а также выключать и включать опцию мониторинга для них с помощью все тех же label-ов. Например, добавим в конфиг еще 1 job (ссылка на gist):

```
- job_name: 'reddit-production'
    kubernetes_sd_configs:
      - role: endpoints
    relabel_configs:
      - action: labelmap
        regex: __meta_kubernetes_service_label_(.+)
      - source_labels: [__meta_kubernetes_service_label_app, __meta_kubernetes_namespace]
        action: keep
        regex: reddit;(production|staging)+
      - source_labels: [__meta_kubernetes_namespace]
        target_label: kubernetes_namespace
      - source_labels: [__meta_kubernetes_service_name]
        target_label: kubernetes_name
```

Для разных лейблов разные регекспы

Метрики приложений

Обновим релиз prometheus и посмотрим

reddit-production (5/5 up)

Endpoint	State	Labels
http://10.8.4.22:9292/metrics	UP	app="reddit" component="comment" instance="10.8.4.22:9292" kubernetes_name="reddit-prod-comment" ku
http://10.8.4.23:9292/metrics	UP	app="reddit" component="ui" instance="10.8.4.23:9292" kubernetes_name="reddit-prod-ui" kubernetes_name
http://10.8.4.24:5000/metrics	UP	app="reddit" component="post" instance="10.8.4.24:5000" kubernetes_name="reddit-prod-post" kubernetes_
http://10.8.5.13:5000/metrics	UP	app="reddit" component="post" instance="10.8.5.13:5000" kubernetes_name="reddit-prod-post" kubernetes_
http://10.8.6.15:5000/metrics	UP	app="reddit" component="post" instance="10.8.6.15:5000" kubernetes_name="reddit-prod-post" kubernetes_

Метрики приложений

Метрики будут отображаться для всех инстансов приложений

ui_health_post_availability

Execute - insert metric at cursor -

Graph Console

Element

ui_health_post_availability{app="reddit",branch="HEAD",commit_hash="97cc37f",component="ui",instance="10.8.4.23:9292",j

ui_health_post_availability{app="reddit",branch="HEAD",commit_hash="97cc37f",component="ui",instance="10.8.4.23:9292",j

ui_health_post_availability{app="reddit",branch="HEAD",commit_hash="97cc37f",component="ui",instance="10.8.4.29:9292",j
prod",version="0.0.4"}

ui_health_post_availability{app="reddit",branch="HEAD",commit_hash="97cc37f",component="ui",instance="10.8.5.11:9292",j

Задание

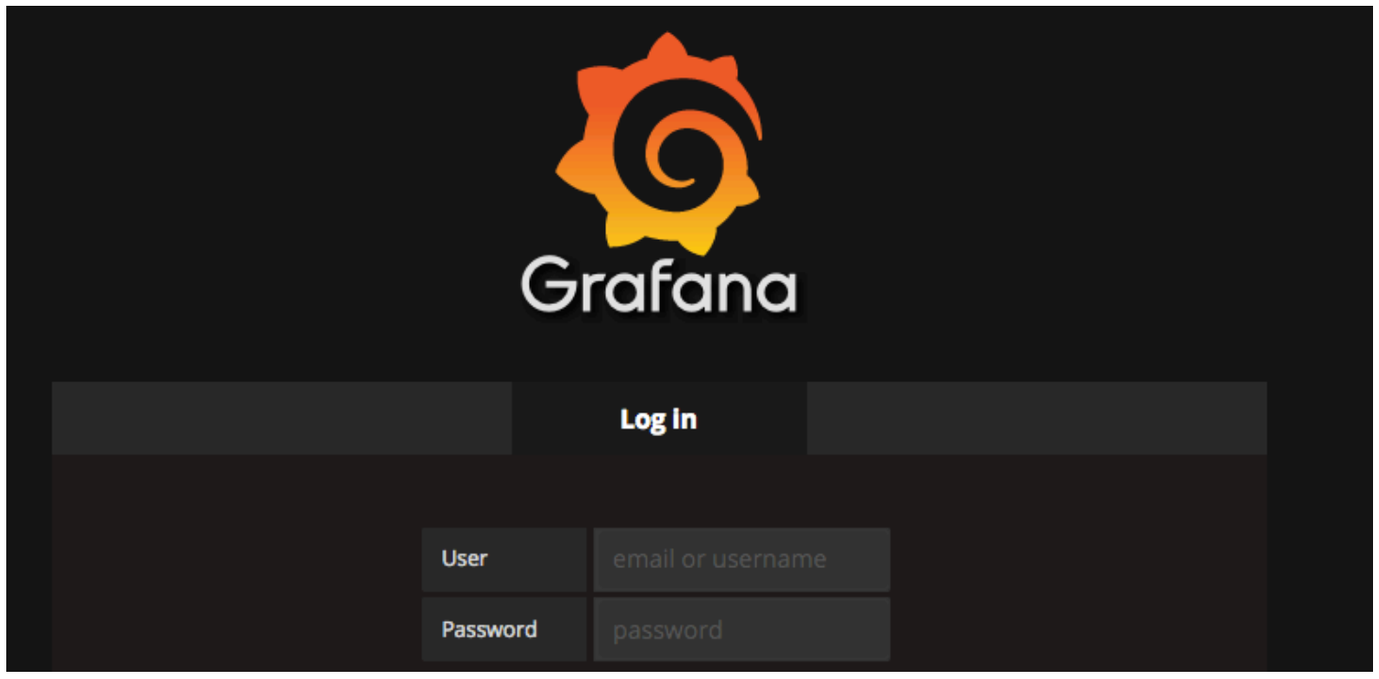
Разбейте конфигурацию job’а `**reddit-endpoints**` (слайд 24) так, чтобы было 3 job’а для каждой из компонент приложений (post-endpoints, comment-endpoints, ui-endpoints), а reddit-endpoints уберите.

Визуализация

Поставим также grafana с помощью helm (ссылка на gist)

```
$ helm upgrade --install grafana stable/grafana --set "adminPassword=admin" \
--set "service.type=NodePort" \
--set "ingress.enabled=true" \
--set "ingress.hosts={reddit-grafana}"
```

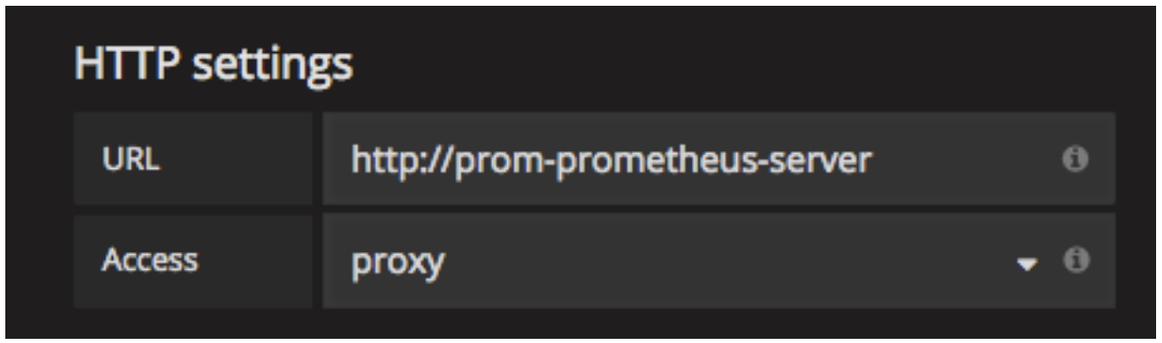
http://reddit-grafana



user: admin
pass: admin

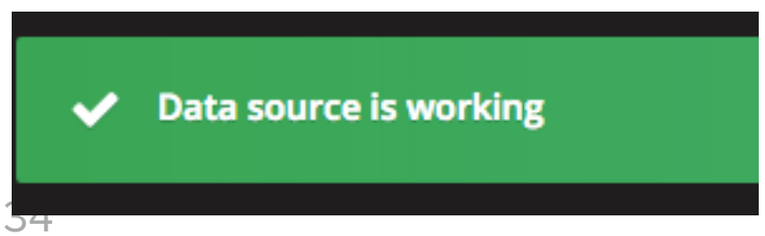
Визуализация

Добавьте prometheus data-source



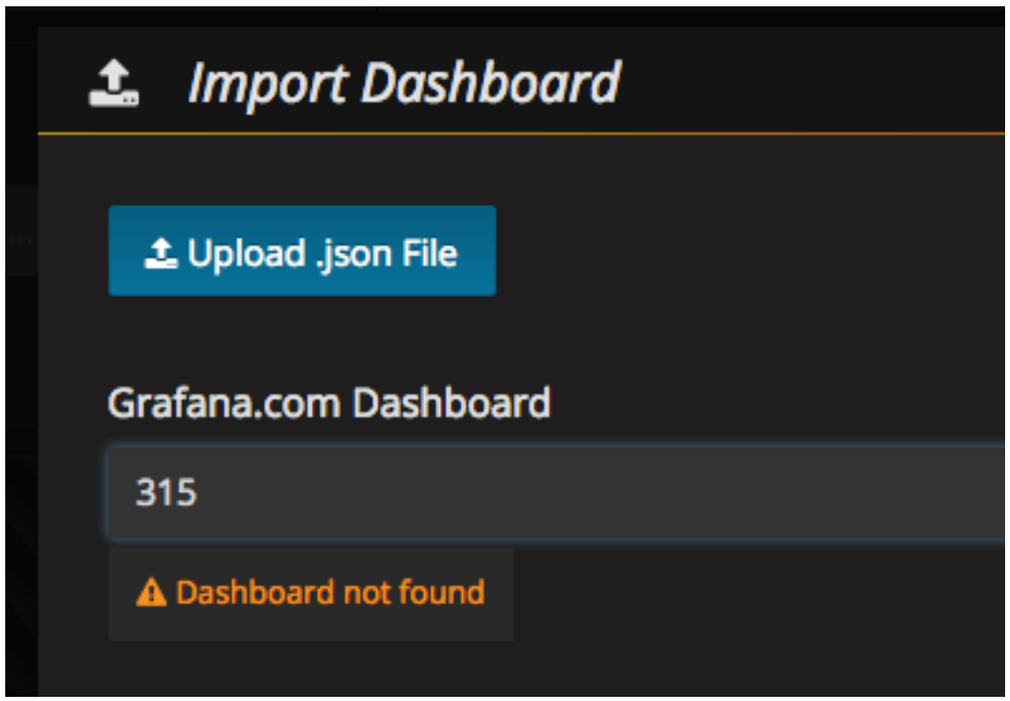
Адрес найдите из имени сервиса prometheus сервера

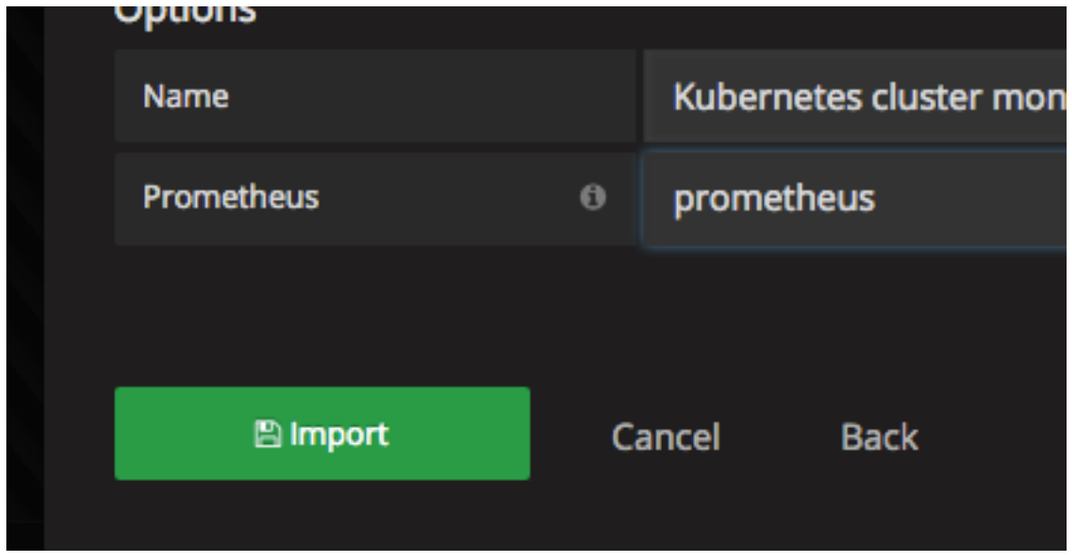
```
$ kubectl get svc
NAME                                  TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
grafana-grafana                       NodePort       10.11.252.216   <none>          80:31886/TCP                 22m
kubernetes                            ClusterIP      10.11.240.1     <none>          443/TCP                      22d
nginx-nginx-ingress-controller        LoadBalancer   10.11.243.76    104.154.94.52   80:32293/TCP,443:30193/TCP   7h
nginx-nginx-ingress-default-backend   ClusterIP      10.11.248.132   <none>          80/TCP                       7h
prom-prometheus-server                LoadBalancer   10.11.247.75    35.224.121.85   80:30282/TCP                 4d
```



Визуализация

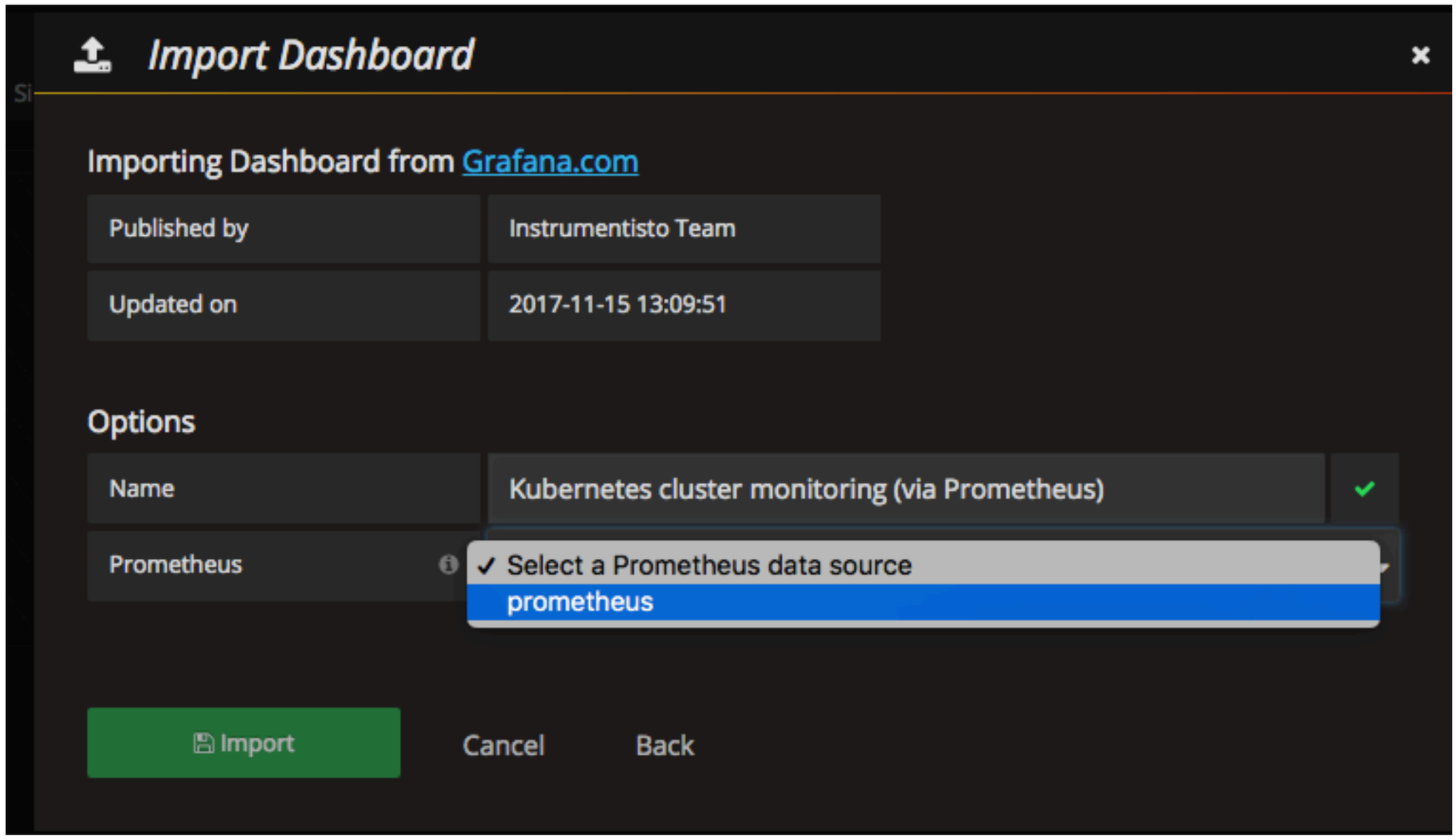
Добавим самый распространенный dashboard для отслеживания состояния ресурсов k8s



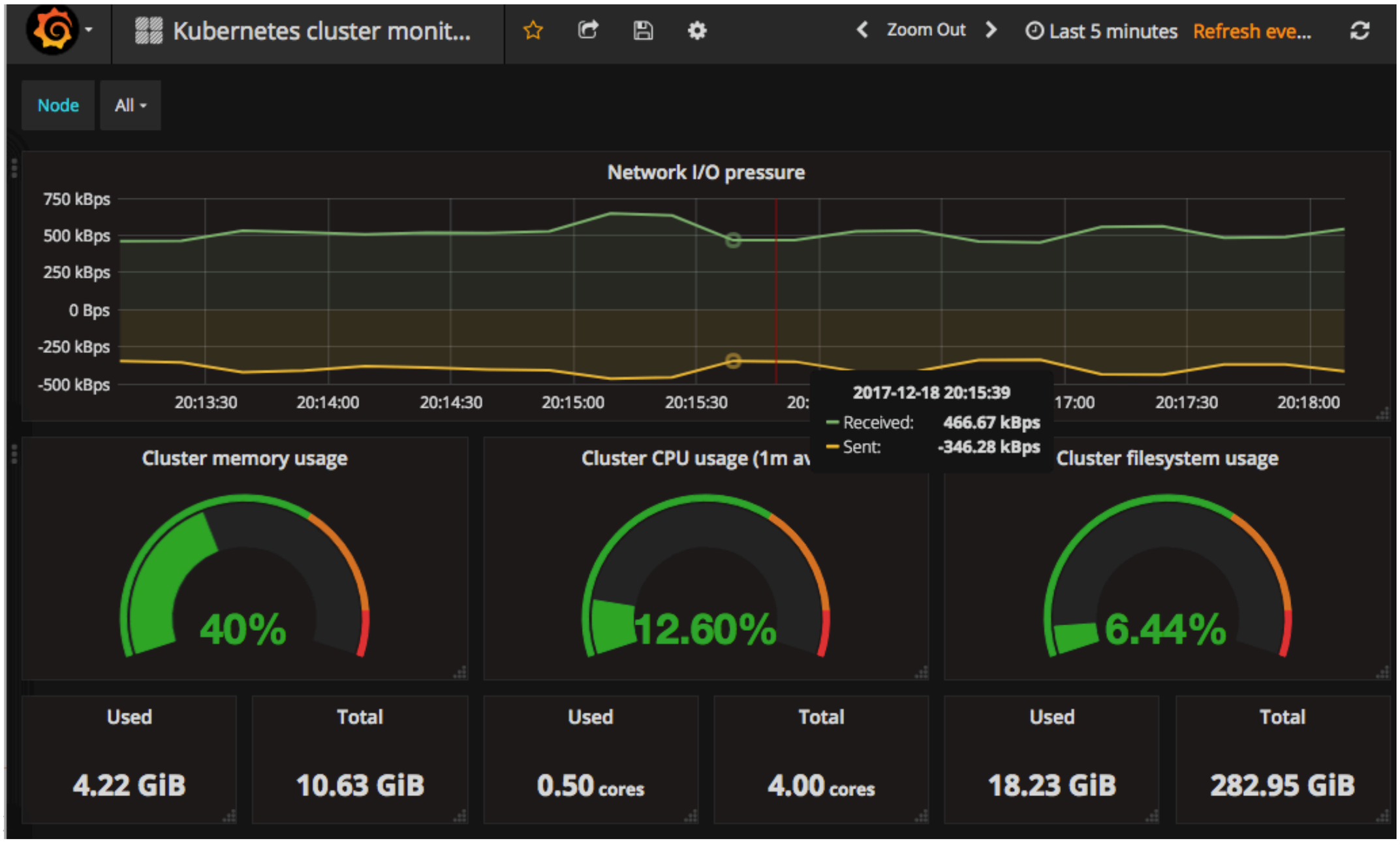


Визуализация

Выберем datasource

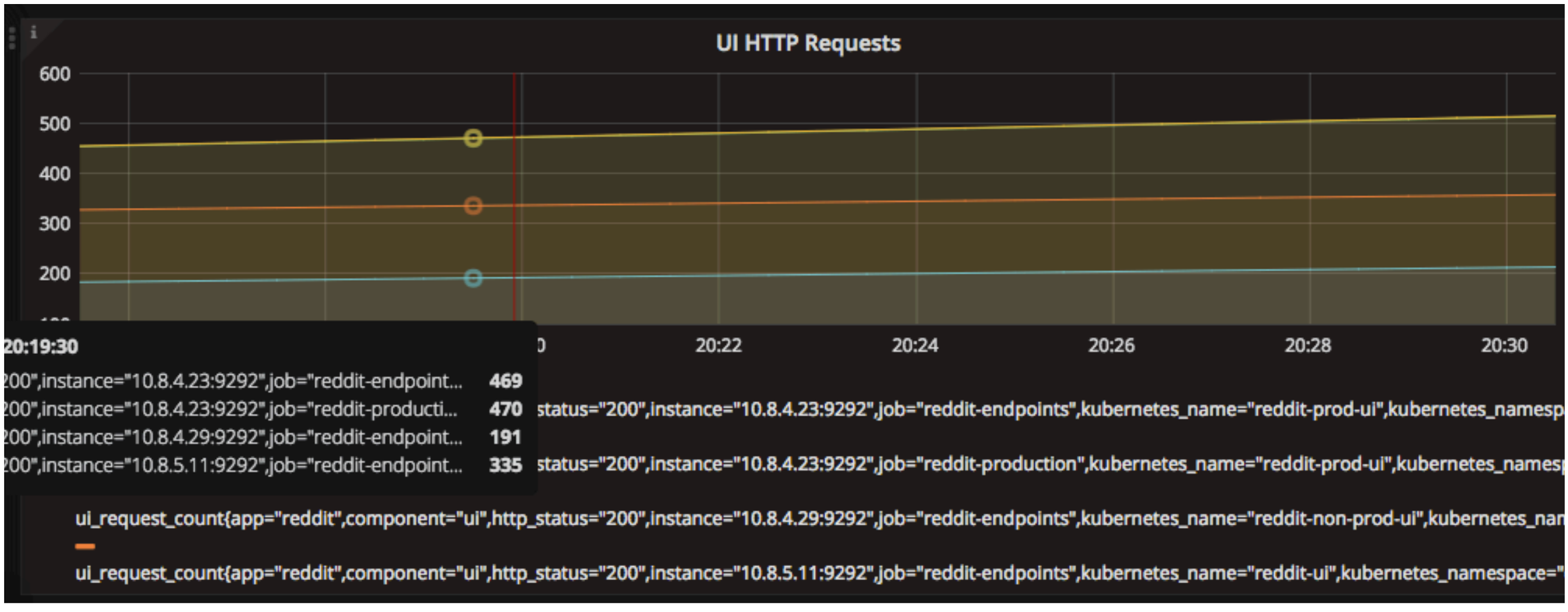


Визуализация



Задание

Добавьте собственные дашборды, созданные ранее (в ДЗ по мониторингу). Они должны также успешно отобразить данные. Должно быть что-то вроде такого:

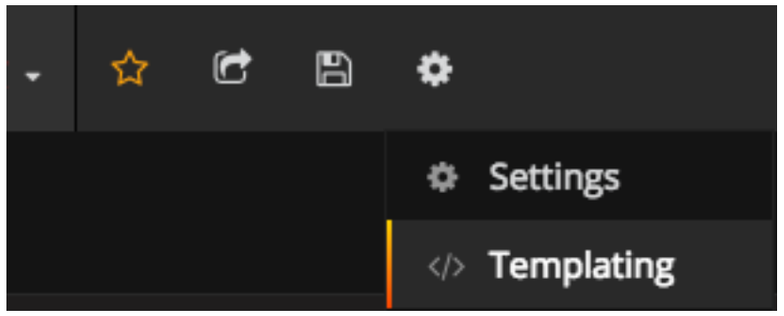


Templating

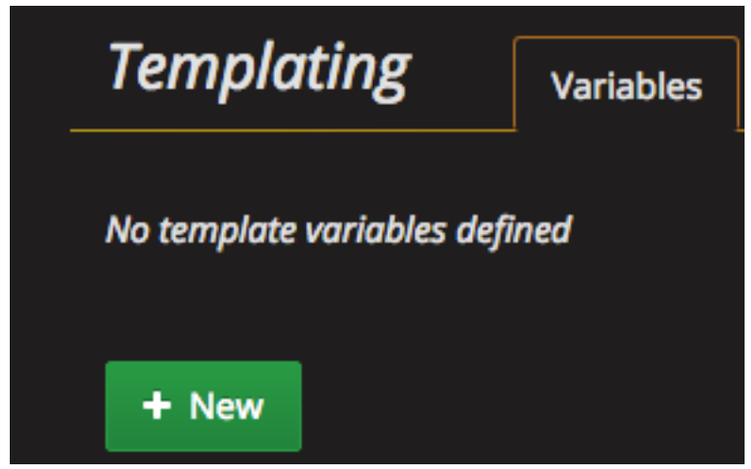
В текущий момент на графиках, относящихся к приложению, одновременно отображены значения метрик со всех источников сразу. При большом количестве сред и при их динамичном изменении имеет смысл сделать динамичной и удобно настройку наших дашбордов в Grafana.

Сделать это можно в нашем случае с помощью механизма **templating’a**

Templating



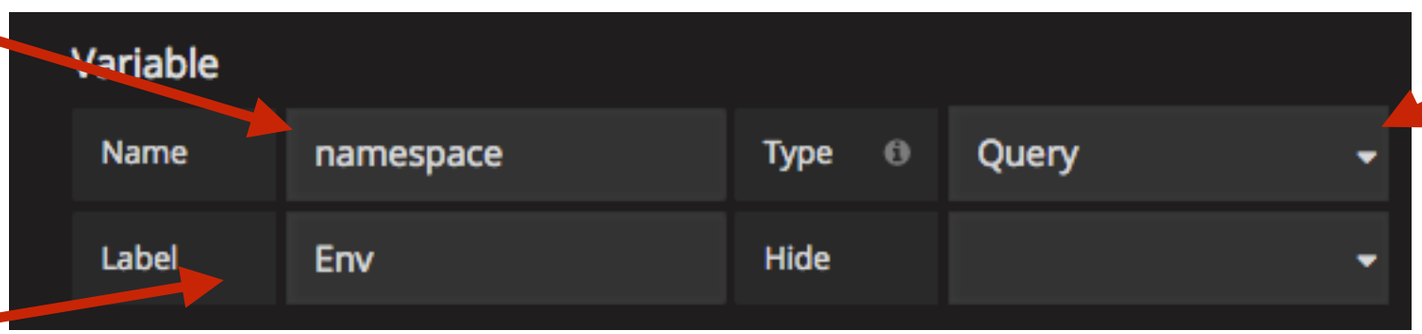
Создадим новую переменную



Имя переменной

Как ее будем видеть мы

Тип (запрос в prometheus)



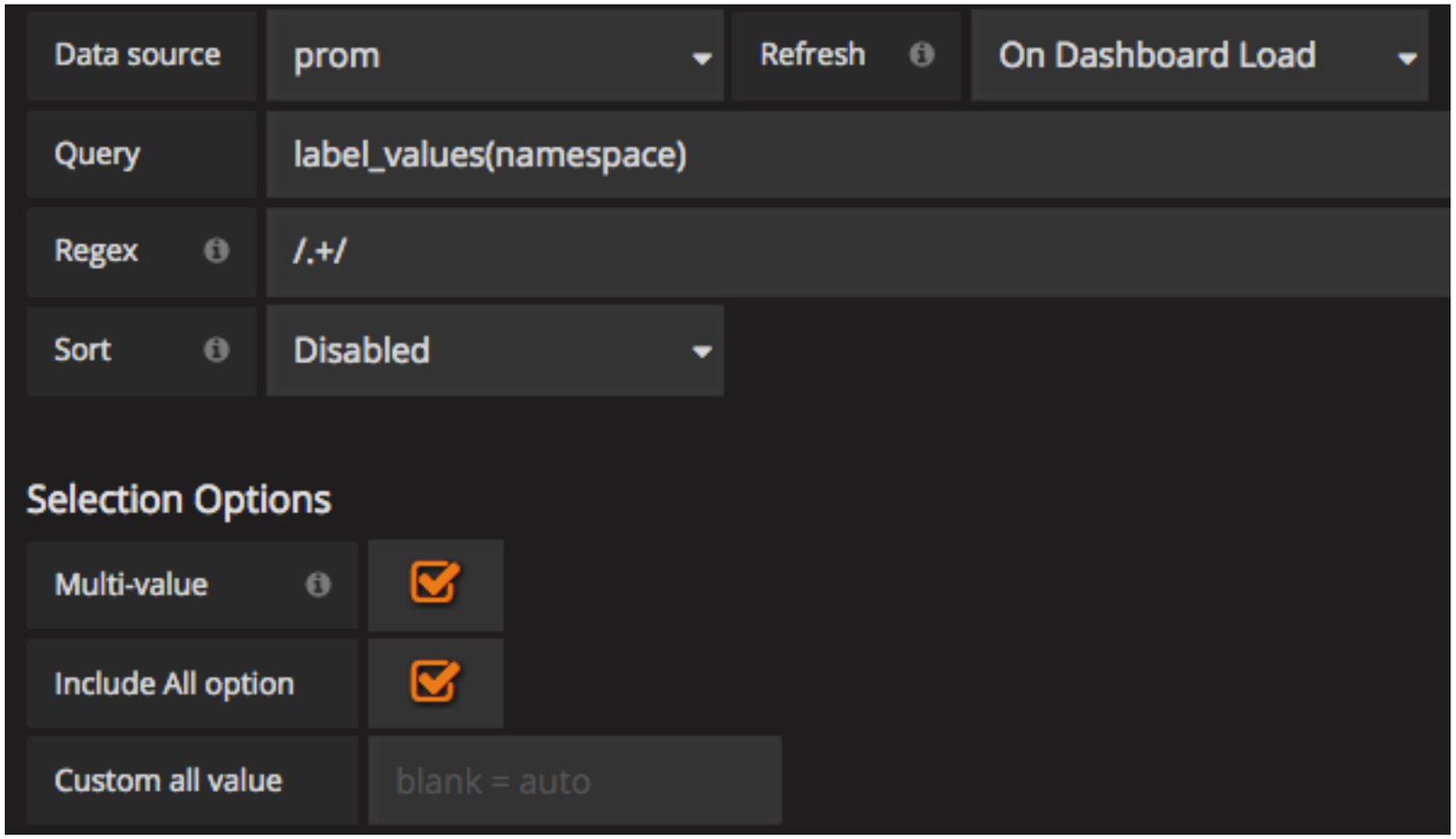
Templating

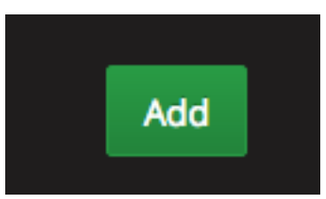
Получить значения всех label-ов kubernetes_namespace

отфильтруем (уберем пустой namespace)

Возможность выбирать несколько значений

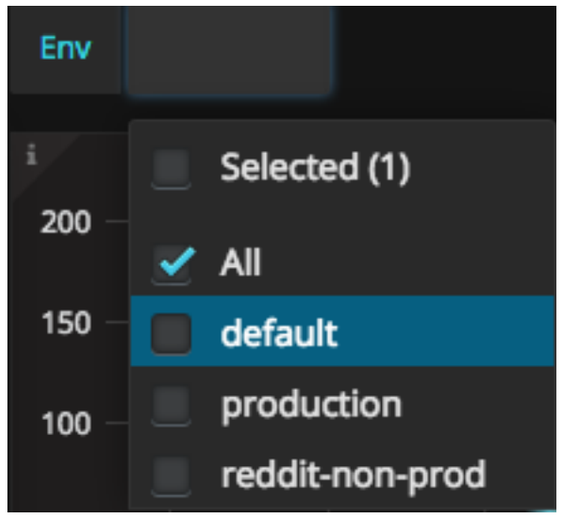
Возможность выбирать все значения одной кнопкой





Templating

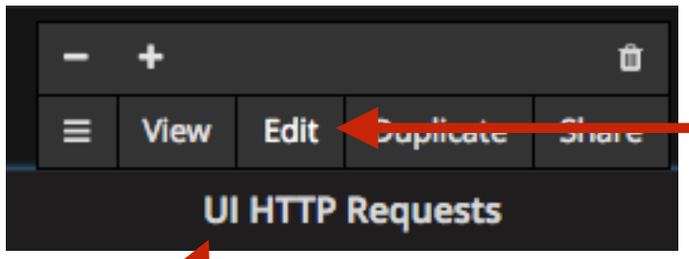
У нас появился список со значениями переменной.



Пока что они бесполезны

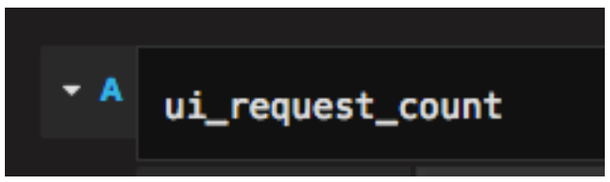
Чтобы их использование имело эффект нужно шаблонизировать запросы к Prometheus

Templating



Затем Edit

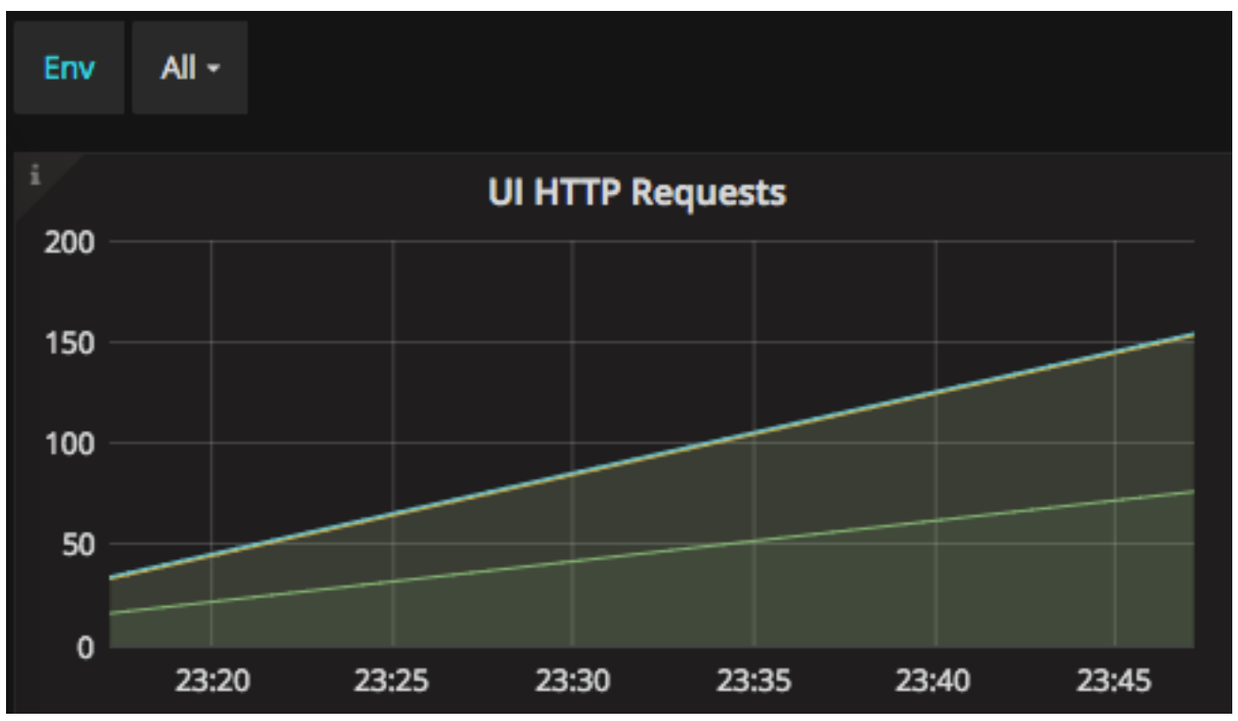
Сначала на имя графика

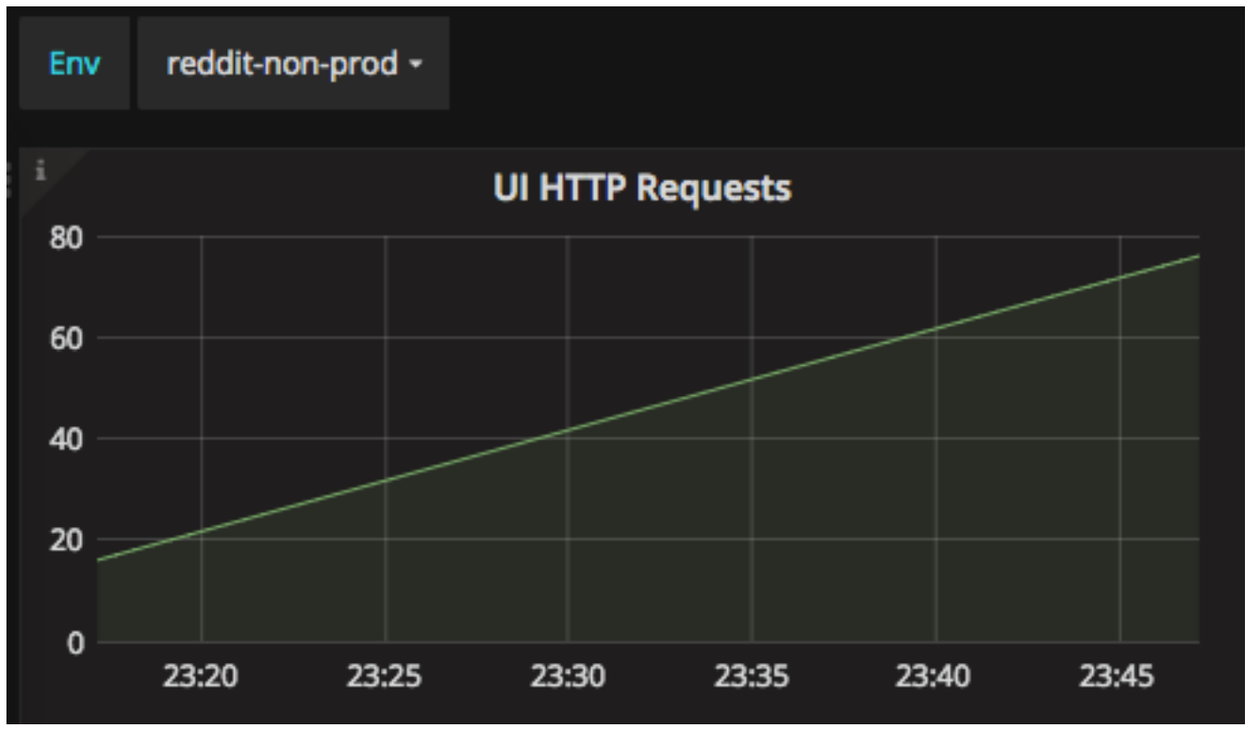


меняем запрос

Templating

Теперь мы можем настраивать общие шаблоны графиков и с помощью переменных менять в них нужные нам поля (в нашем случае это namespace)





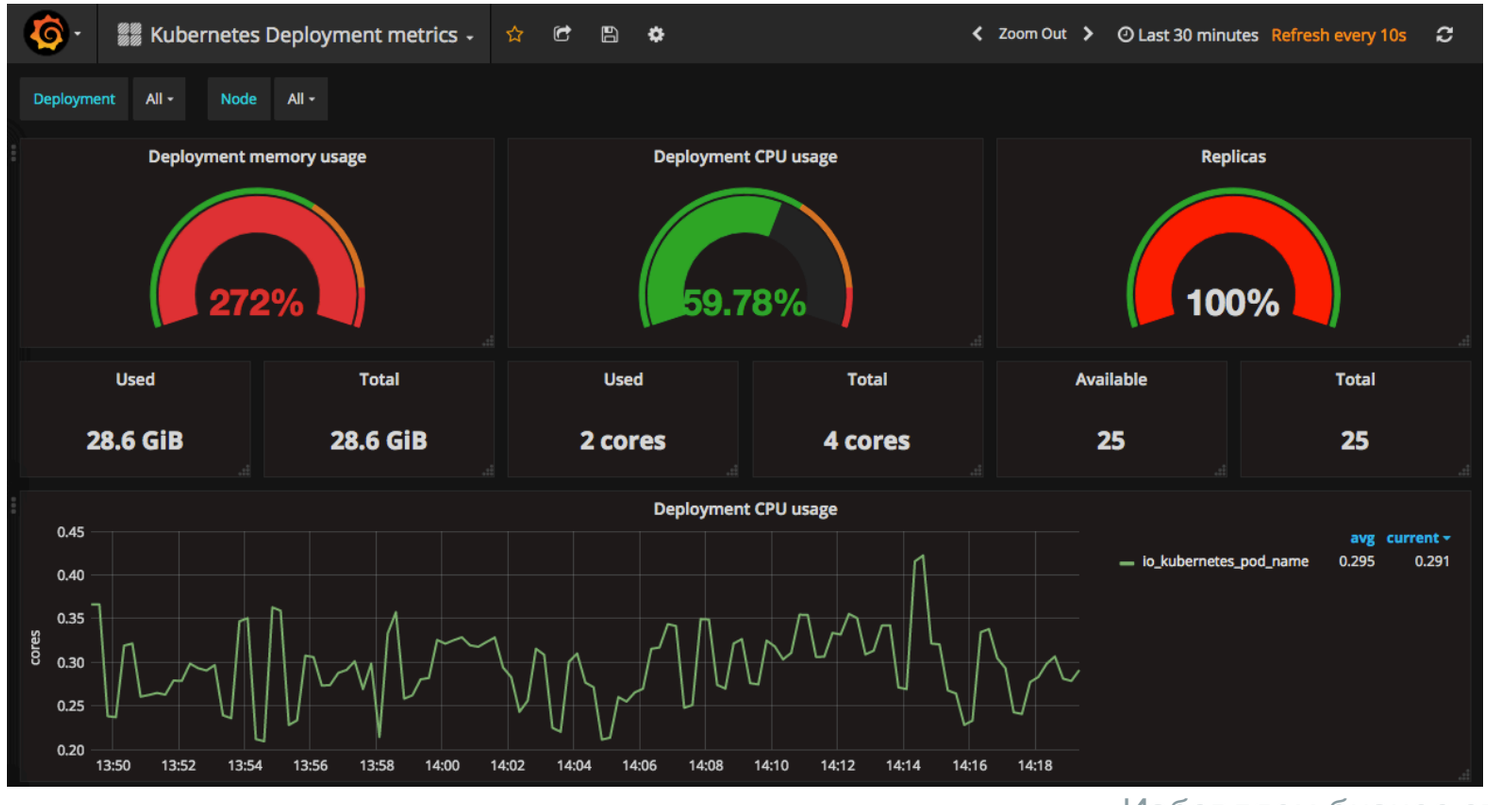
Задание

Параметризуйте все Dashboard’ы, отражающие параметры работы приложения (созданные вами в предыдущих ДЗ) reddit для работы с несколькими окружениями (неймспейсами).

Получившиеся дашборды не забудьте сохранить в репозиторий.

Смешанные графики

Импортируйте следующий график :https://grafana.com/dashboards/741
На этом графике одновременно используются метрики и шаблоны из **cAdvisor**, и из **kube-state-metrics** для отображения сводной информации по деплойментам



Задание со*

В целом, принципы работы с инструментами не поменялись. Добавились лишь особенности, поэтому мы можем использовать старые наработки и для k8s.

Задание:
запустить **alertmanager** в k8s и настроить правила для контроля за доступностью api-сервера и хостов k8s

P.S. не забудьте, что формат описания правил в версии 2.0 изменился на yaml

Задание со *

- Установите в кластер Prometheus Operator (можно воспользоваться простой пошаговой инструкцией или helm чартом)
- Настройте мониторинг post endpoints
- Приложите используемый манифест serviceMonitor

Логирование

Подготовка

1. Данную часть ДЗ рекомендуется выполянять уже после выполения ДЗ № 25 (по логированию), наработки из которого используюстя в данной работе

3. Выполнение данной части ДЗ не является обязательным
4. Добавьте label самой мощной ноде в кластере

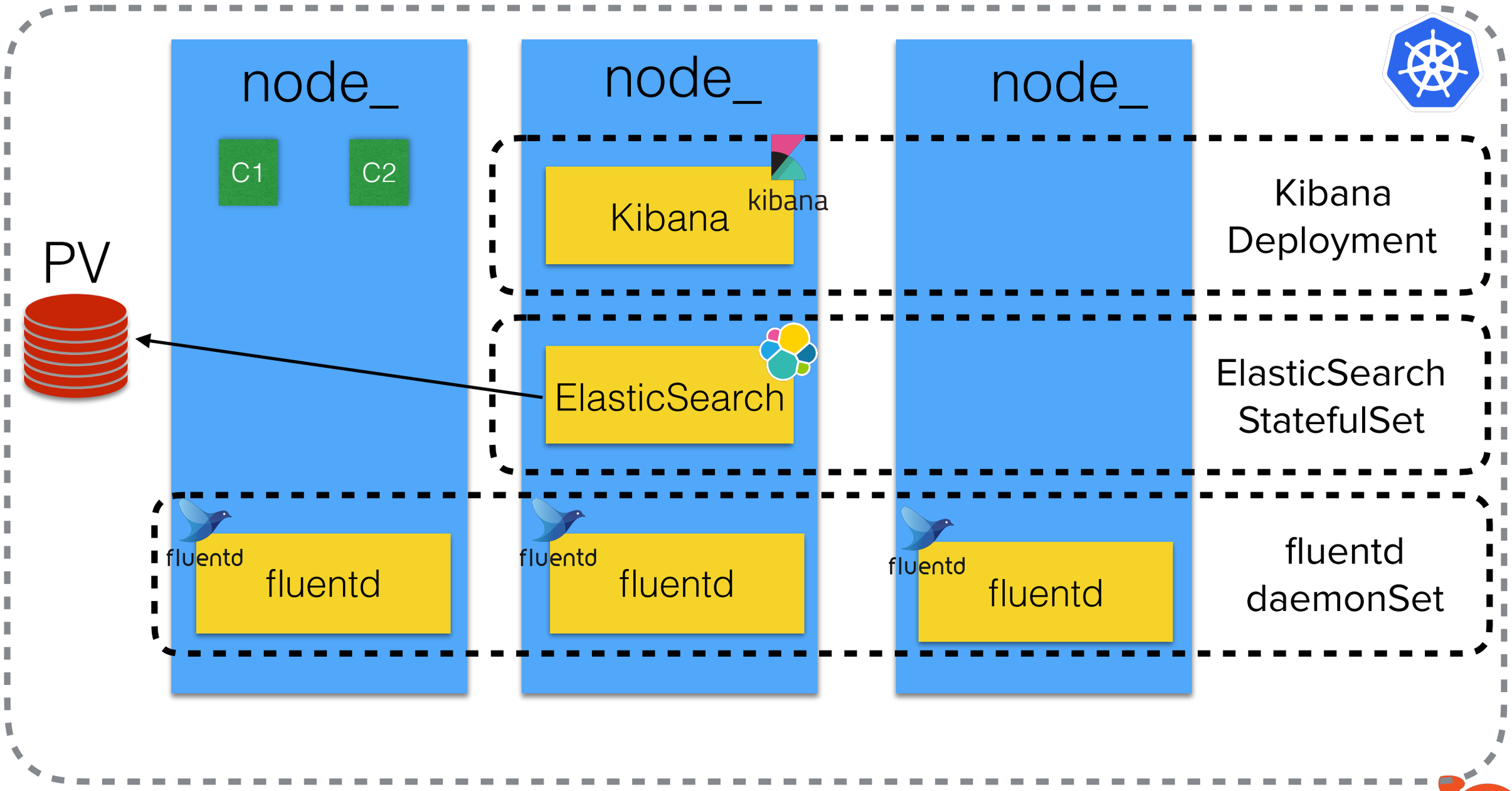
```
$ kubectl label node gke-cluster-1-big-pool-b4209075-tvn3 elastichost=true
```

Стек

Логирование в k8s будем выстраивать с помощью уже известного стека EFK:

- ElasticSearch - база данных + поисковый движок
- Fluentd - шипер (отправитель) и агрегатор логов
- Kibana - веб-интерфейс для запросов в хранилище и отображения их результатов

Как логировать?



Создайте файлы в новой папке **kubernetes/efk**/

fluentd-ds.yaml (ссылка на gist)

fluentd-configmap.yaml (ссылка на gist)

es-service.yaml (ссылка на gist)

es-statefulSet.yaml (ссылка на gist)

es-pvc.yaml (ссылка на gist)

Запустите стек в вашем k8s

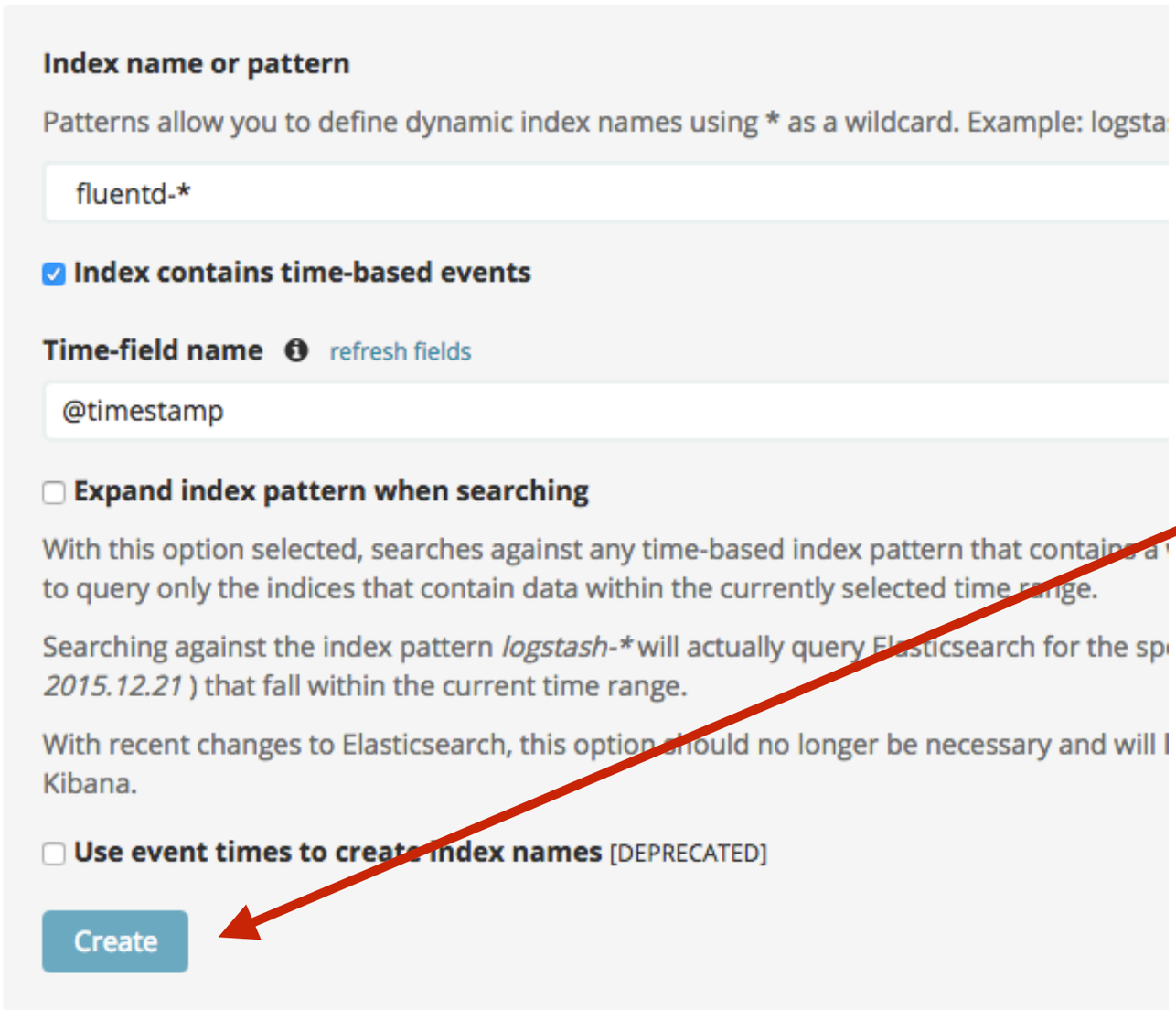
```
$ kubectl apply  -f ./efk
```

Kibana поставим из helm чарта (ссылка на gist)

```
$ helm upgrade --install kibana stable/kibana \
--set "ingress.enabled=true" \
--set "ingress.hosts={reddit-kibana}" \
--set "env.ELASTICSEARCH_URL=http://elasticsearch-logging:9200" \
--version 0.1.1
```

EFK

http://reddit-kibana/



Создайте шаблон индекса

Откройте вкладку **Discover** в Kibana и введите в строку поиска выражение

kubernetes.labels.component:post OR kubernetes.labels.component:comment OR kubernetes.labels.component:ui

Откройте любой из рез-тов поиска - в нем видно множество инфы о k8s

t	kubernetes.container_name	comment
t	kubernetes.host	gke-cluster-1-big-pool-b4209075-tvn3
t	kubernetes.labels.app	reddit
t	kubernetes.labels.component	comment
t	kubernetes.labels.pod-template-hash	1952347765
t	kubernetes.labels.release	reddit-non-prod
t	kubernetes.master_url	https://10.11.240.1:443/api
t	kubernetes.namespace_name	reddit-non-prod
t	kubernetes.pod_id	354bc737-e439-11e7-b9f9-42010a80001b
t	kubernetes.pod_name	reddit-non-prod-comment-5f9678ccb9-ch4nf

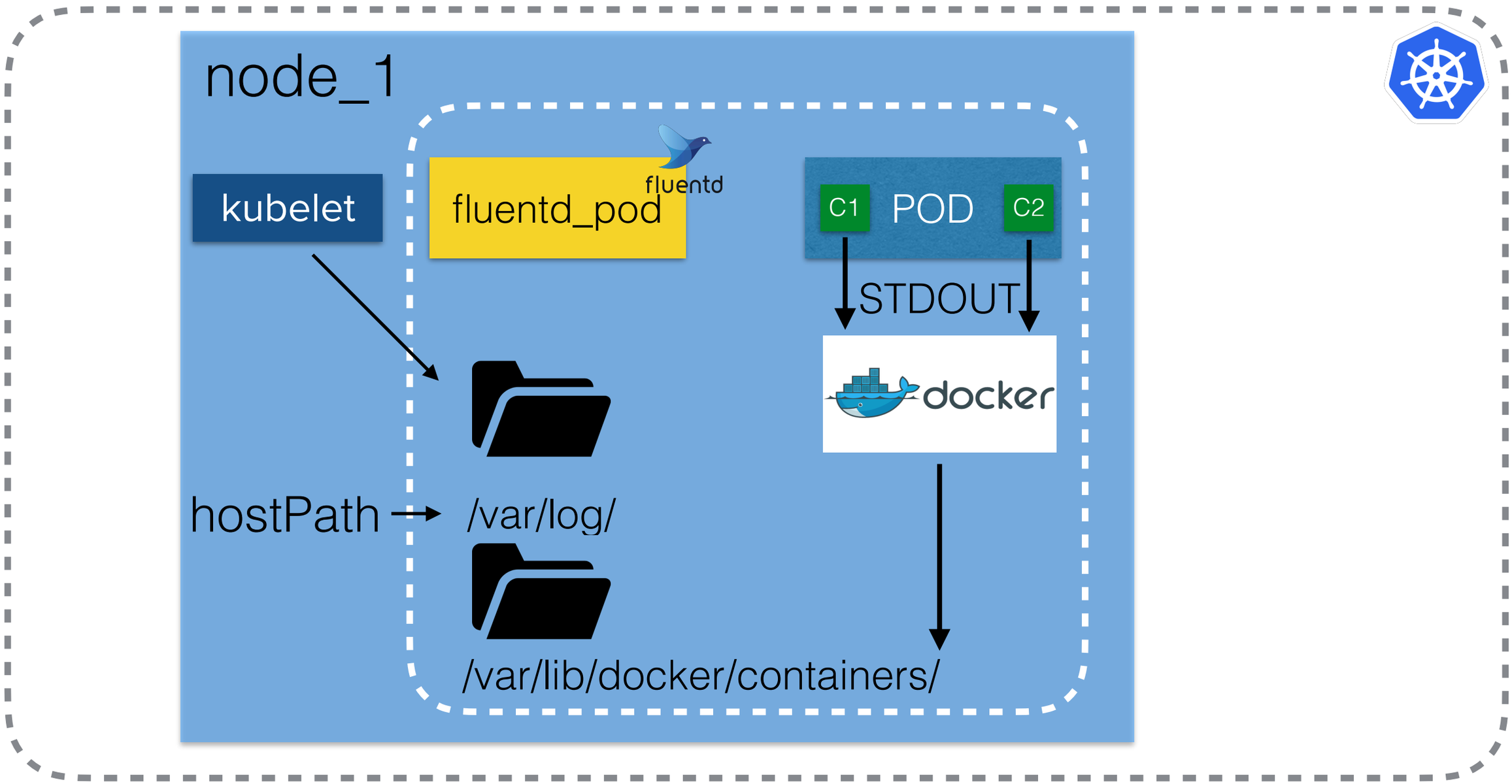
EFK

1. Особенность работы fluentd в k8s состоит в том, что его задача помимо сбора самих логов приложений, сервисов и хостов, также распознать дополнительные метаданные (как правило это дополнительные поля с лейблами)

2. Откуда и какие логи собирает fluentd - видно в его **fluentd-configmap.yaml** и в **fluentd-ds.yaml**

3. Общая схема работы fluentd представлена на след. слайде

Fluentd


node_1
kubelet
fluentd_pod
fluentd
C1
POD
C2
STDOUT
docker
hostPath
/var/log/
/var/lib/docker/containers/

Задание со*

Создайте Helm-чарт для установки стека EFK и поместите в директорию charts