Kubernetes. Мониторинг и логирование

План

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

Мониторинг

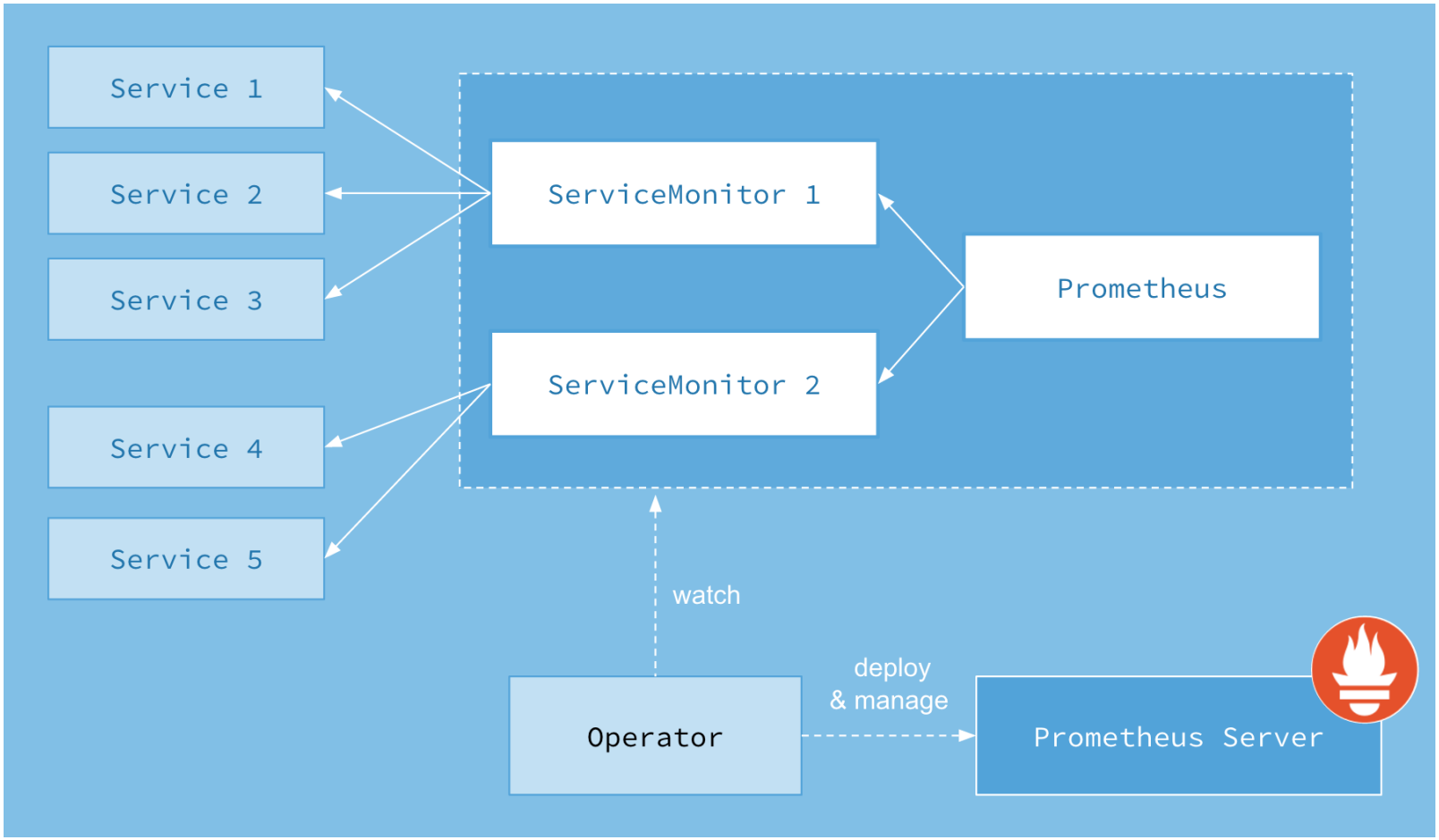
Стек

В задании будем использовать следующие инструменты:

- Prometheus - сервер сбора и хранения метрик
- Grafana - сервер визуализации метрик
- Различные экспортеры для метрик Prometheus

Prometheus отлично подходит для работы с контейнерами и средами с динамичным размещением сервисов

Схема


Service 1
Service 2
Service 3
Service 4
Service 5
ServiceMonitor 1
ServiceMonitor 2
Prometheus
watch
Operator
deploy
& manage
Prometheus Server

Установим Prometheus

С установкой могут возникнуть определенные нюансы

Пройдите по ссылке в официальный репозиторий И установите руками CRD описанные в документации:

```
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_alertmanagers.yaml
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_podmonitors.yaml
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_prometheuses.yaml
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_prometheusrules.yaml
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_servicemonitors.yaml
kubectl apply -f https://raw.githubusercontent.com/coreos/prometheus-operator/release-
0.38/example/prometheus-operator-crd/monitoring.coreos.com_thanosrulers.yaml
```

Или воспользуйтесь этим скриптом

Установим Prometheus

Скопируйте содержимое gist по ссылке в файл `prometheus-operator.values.yaml` и запустите развертывание (обратите внимание на **CHANGE_ME**):

```
kubectl create ns monitoring
helm upgrade --install prometheus-operator stable/prometheus-operator --set prometheusOperator.createCustomResource=false -f prometheus-operator.values.yaml -n monitoring --version 8.11.0
```

> `CHANGE_ME` берем отсюда:

```
$ kubectl get svc -n nginx-ingress
NAME                       TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)     AGE
nginx-ingress-controller   LoadBalancer   10.0.104.13   104.155.22.61   ...        18h
...
```

Prometheus Operator | Установка

Проверьте, что Prometheus и Grafana доступны по следующим ссылкам:

- http://prometheus.CHANGE_ME.xip.io
- http://grafana.CHANGE_ME.xip.io

Targets

У нас уже присутствует ряд endpoint’ов для сбора метрик:

- Метрики kube-apiserver
- Метрики cAdvisor
- ...

Targets

Цели для сбора метрик были найдены с помощью `Service Discovery (SD)`, настроенного в конфиге prometheus.

```
- job_name: monitoring/prometheus-prometheus-oper-apiserver/0
  kubernetes_sd_configs:
  - role: endpoints
    namespaces:
      names:
      - default
  bearer_token_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
  relabel_configs:
  - source_labels: [__meta_kubernetes_service_label_component]
    separator: ;
    regex: apiserver
    replacement: $1
    action: keep
```

Targets

```
# Настройки service discovery
  kubernetes_sd_configs:
  - role: endpoints # node, endpoints, pod, service, ingress
```

```
# Настройки подключения
  scheme: https
  bearer_token_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
```

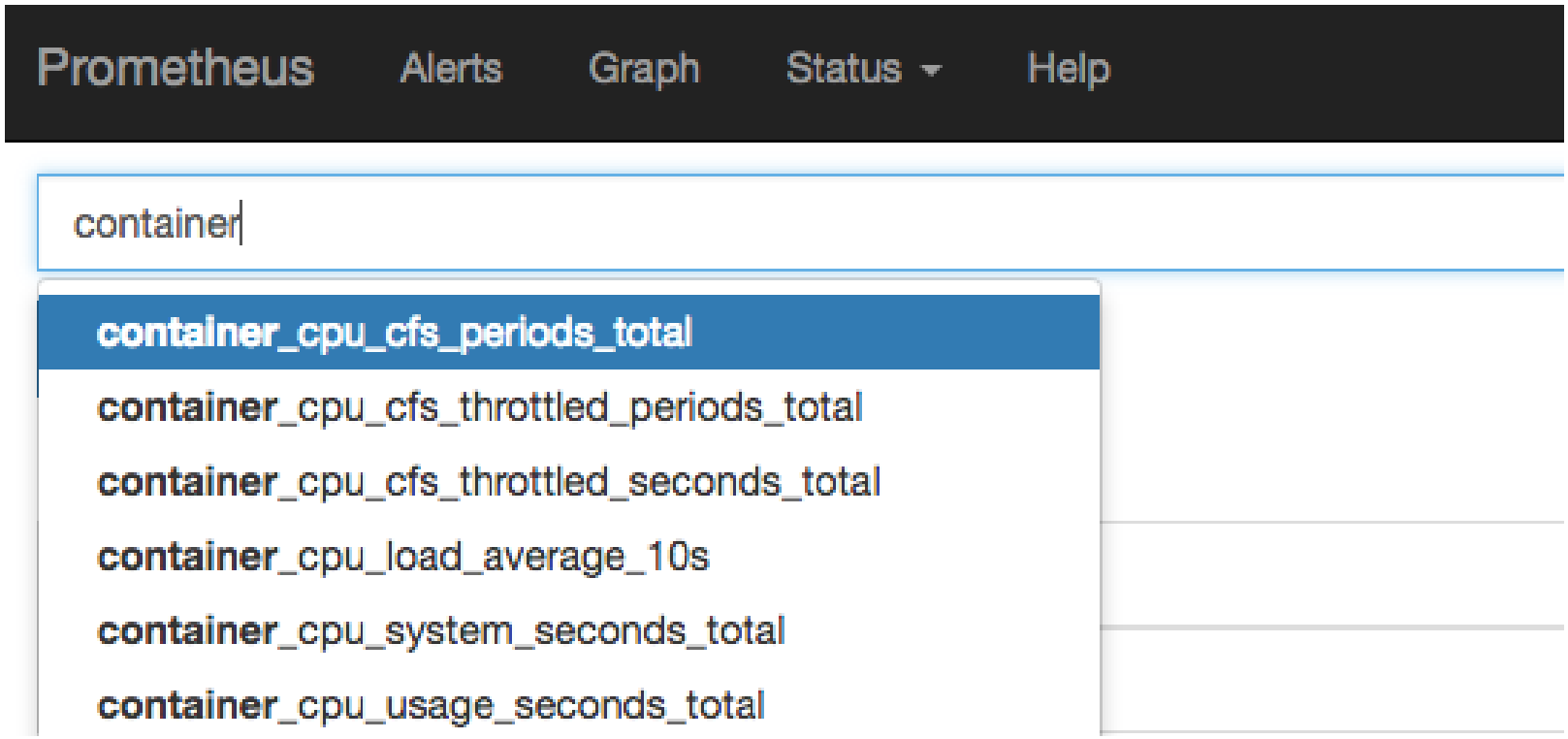
```
# Настройка меток, фильтрация целей и их изменение
  relabel_configs:
```

Подробнее про relabel

```
relabel_configs:
  # Преобразовать все k8s лейблы таргета в лейблы prometheus
  - action: labelmap
    regex: __meta_kubernetes_node_label_(.+)
  # Поменять лейбл для адреса сбора метрик
  - target_label: __address__
    replacement: kubernetes.default.svc:443
  # Поменять лейбл для пути сбора метрик
  - source_labels: [__meta_kubernetes_node_name]
    regex: (.+)
    target_label: __metrics_path__
    replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor
```

Metrics

Все найденные метрики сразу же отобразятся в списке (вкладка Graph). Метрики cAdvisor начинаются с префикса `container_`.



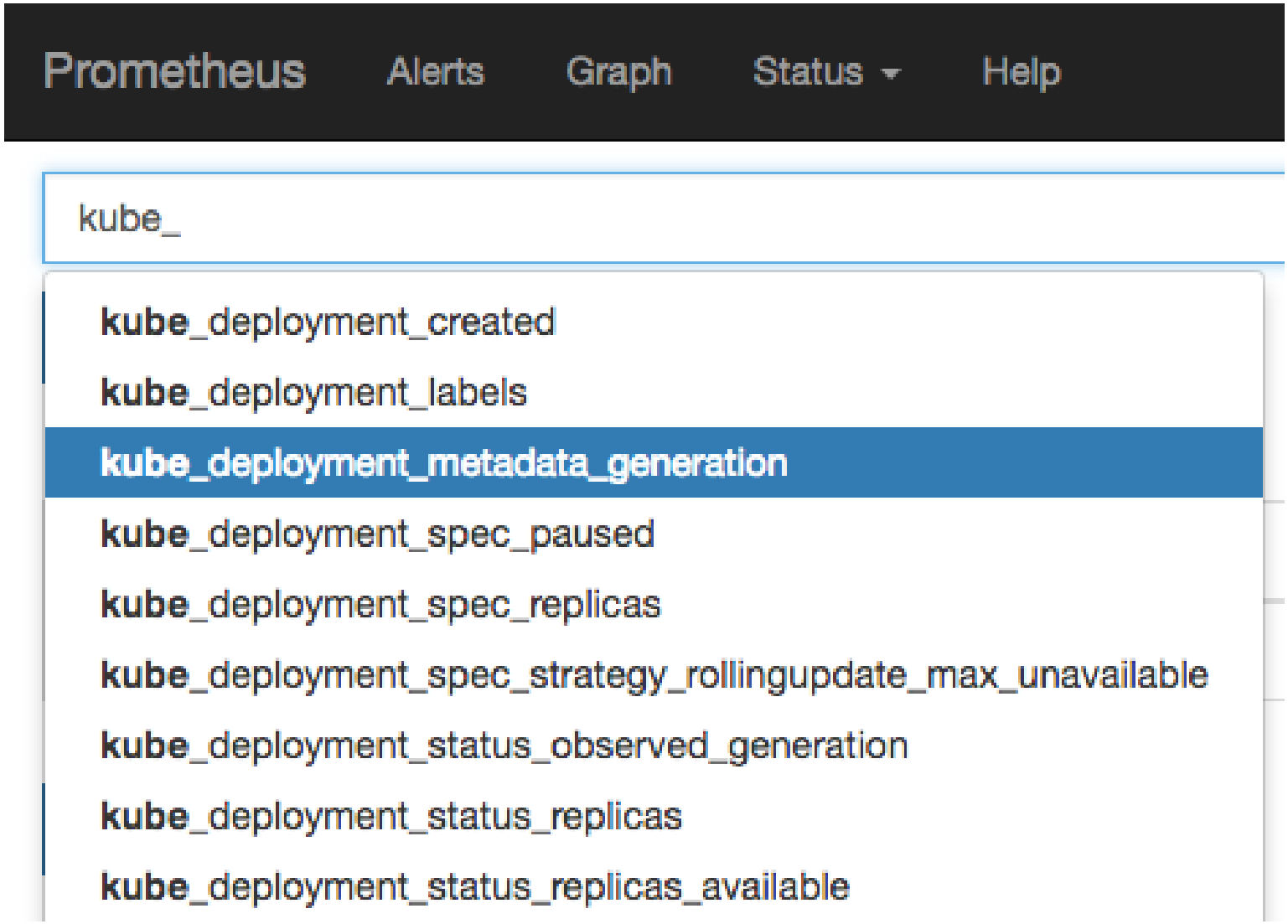
kube-state-metrics

cAdvisor собирает лишь информацию о потреблении ресурсов и производительности отдельных docker-контейнеров. При этом он ничего не знает о сущностях k8s (deployment, replicaset, etc.).

Для сбора этой информации будем использовать сервис `kube-state-metrics`. Он уже включен.

kube-state-metrics

В Graph:



Node exporter

Node exporter также включен.

Проверьте наличие Target `monitoring/prometheus-prometheus-oper-node-exporter/0`

И что у вас собираются метрики `node_*`

Метрики приложений

Теперь давайте подключим приложение. Метрики с него будут сниматься точно также, как и с инфраструктурных компонент.

Обновим Helm chart’ы

В чарты сервисов `ui`, `post` и `comment` добавьте блоки `metadata.labels` и `spec.ports.name`

```
1   metadata:
2     labels:
3       {{- range $key, $value := .Values.labels }}
4         {{ $key }}: {{ $value }}
5       {{- end }}
6   spec:
7     ports:
8     - name: http-metrics
```

Закоммитьте результат в репозитории

Итоговый вариант Helm Chart’a

```yaml
kind: Service
apiVersion: v1
metadata:
  name: {{ .Chart.Name }}
  labels:
    {{- range $key, $value := .Values.labels }}
      {{ $key }}: {{ $value }}
    {{- end }}
spec:
  type: {{ .Values.service.type }}
  selector:
    {{- range $key, $value := .Values.labels }}
      {{ $key }}: {{ $value }}
    {{- end }}
  ports:
  - protocol: TCP
    name: http-metrics
    port: {{ .Values.service.port }}
    targetPort: {{ .Values.service.targetPort }}
```

Подготовим Service Monitor

Создайте файл `reddit-servicemonitor.yml` следующего содержания

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: reddit-servicemonitor
  labels:
    release: prometheus
spec:
  namespaceSelector:
    any: true
  endpoints:
  - port: http-metrics
  jobLabel: reddit
  selector:
    matchLabels:
      app: reddit
```

ServiceMonitor

Применим манифест с ServiceMonitor:

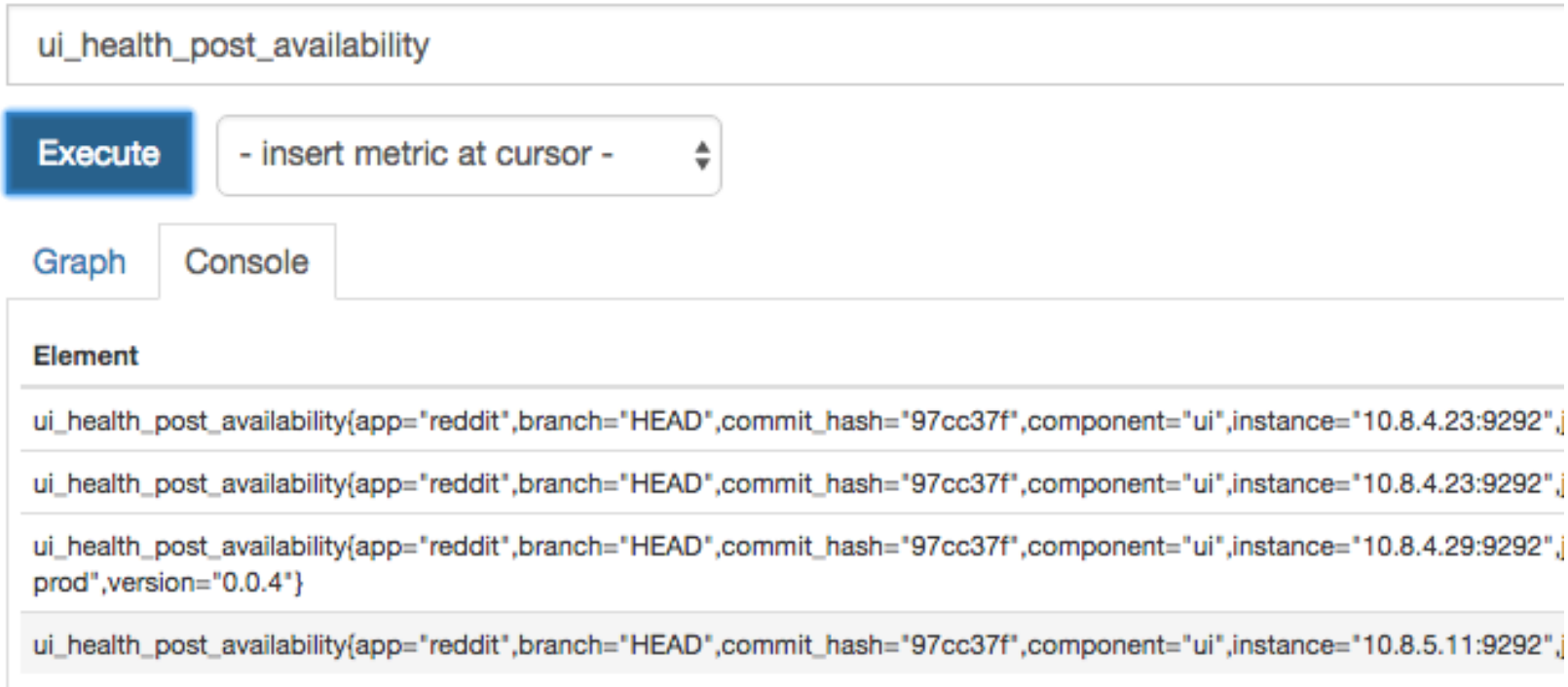
```
kubectl apply -f reddit-servicemonitor.yml
```

Конфигурация Prometheus обновляется каждые три минуты. Применилось ли обновление можно посмотреть командой:

```
kubectl logs -n monitoring prometheus-prometheus-operator-prometheus-0 prometheus-config-reloader
```

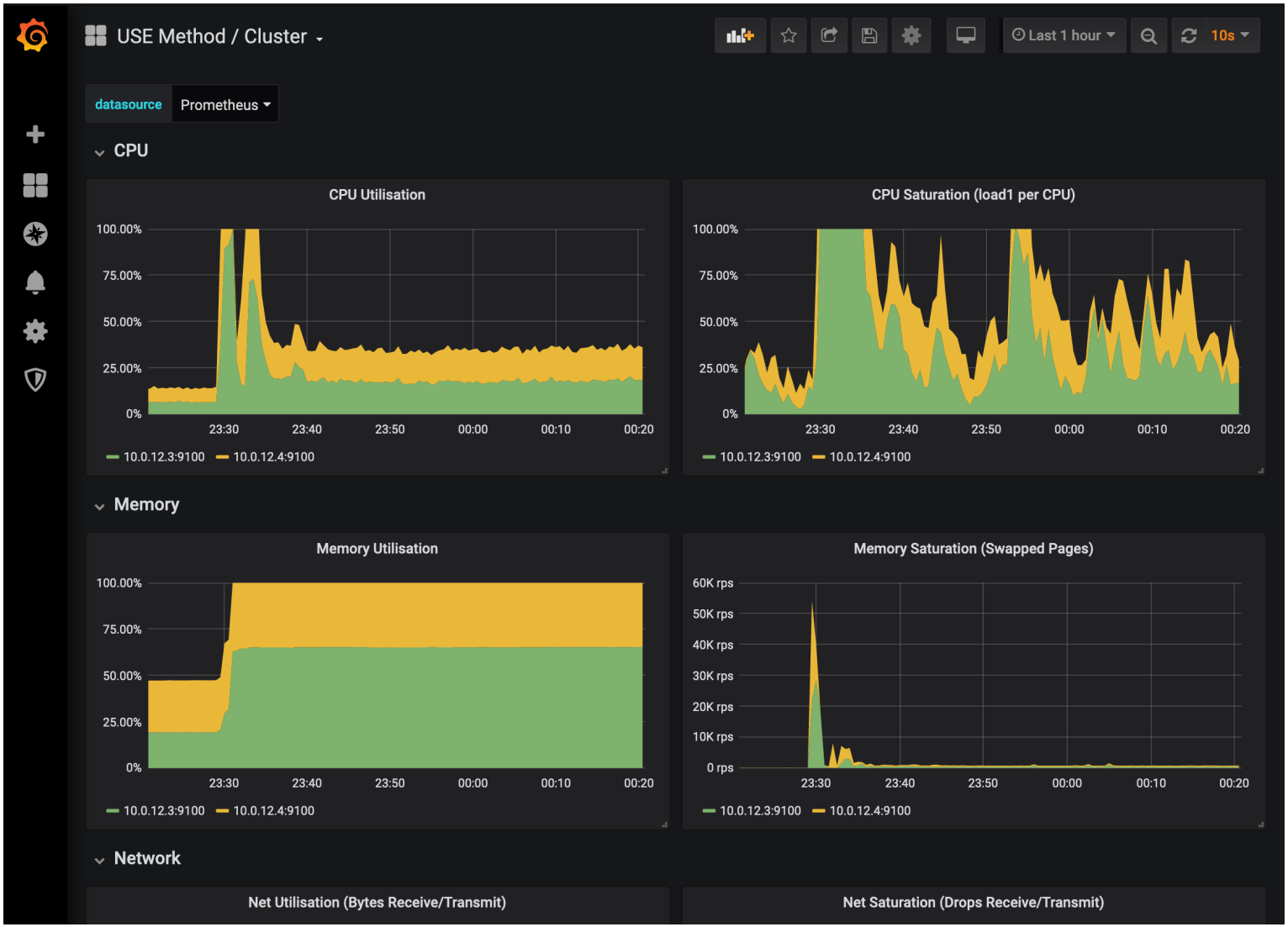
Метрики приложений

Метрики будут отображаться для всех инстансов приложений.



Визуализация

У нас уже установлена Grafana:



Визуализация

Скопируйте содержимое gist по ссылке в файл `dashboard-configmap.yaml` и примените данный манифест:

```
1  kubectl apply -f dashboard-configmap.yaml
```

После этого в Grafana появится dashboard `Reddit Monitoring`, на котором будут отражены некоторые метрики, отдаваемые нашими микросервисами.

Визуализация

Повысим надежность

Добавьте блок `prometheus.prometheusSpec.storageSpec` в `prometheus-operator.values.yaml`

```yaml
prometheus:
  prometheusSpec:
    storageSpec:
      volumeClaimTemplate:
        spec:
          accessModes:
            - "ReadWriteOnce"
          resources:
            requests:
                storage: 3Gi
```

Gist доступен данной ссылке

Проверка

Примените изменения

```
helm upgrade --install prometheus-operator stable/prometheus-operator --set prometheusOperator.createCustomResource=false -f prometheus-operator.values.yaml -n monitoring --version 8.11.0
```

И проверьте:

```
helm delete prometheus-operator -n monitoring
kubectl get all -n monitoring

helm upgrade --install prometheus-operator stable/prometheus-operator --set prometheusOperator.createCustomResource=false -f prometheus-operator.values.yaml -n monitoring --version 8.11.0
```

Prometheus Operator

https://github.com/helm/charts/tree/master/stable/prometheus-operator

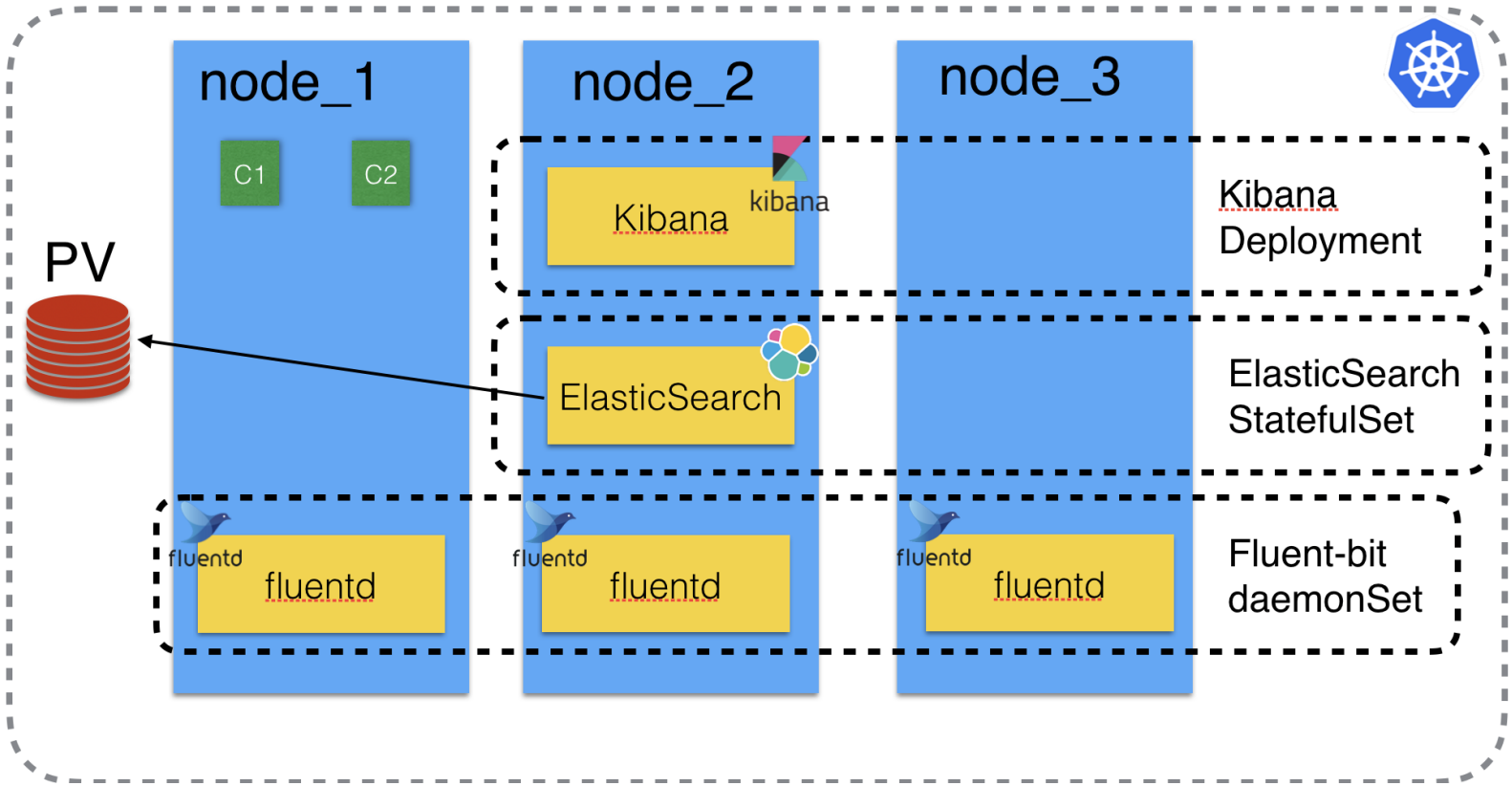
Логирование

Стек

Логирование в k8s будем выстраивать с помощью стека EFK:

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

Стек


node_1
node_2
node_3
C1
C2
Kibana
kibana
Kibana
Deployment
PV
ElasticSearch
ElasticSearch
StatefulSet
fluentd
fluentd
fluentd
fluentd
fluentd
fluentd
Fluent-bit
daemonSet

EFK

Добавим репозиторий helm elastic:

```
1   helm repo add elastic https://helm.elastic.co
2   helm repo update
```

Создадим namespace для логирования:

```
1  kubectl create namespace logging
```

EFK

Запустите стек в вашем Kubernetes кластере. Для начала скачайте gist `elasticsearch.values.yaml` по данной ссылке

```
helm upgrade --install elasticsearch elastic/elasticsearch --namespace logging --version
7.7.1 -f elasticsearch.values.yaml
```

```
helm upgrade --install kibana elastic/kibana --namespace logging --version 7.7.1
```

```
 helm upgrade --install fluent-bit stable/fluent-bit --namespace logging \
 --set "backend.es.host=elasticsearch-master" --set "backend.type=es" \
 --set "filter.mergeJSONLog=false" --version 2.8.16
```

> **Самостоятельно:** создайте `values.yaml` файл для Kibana, в котором будет описано добавление ingress

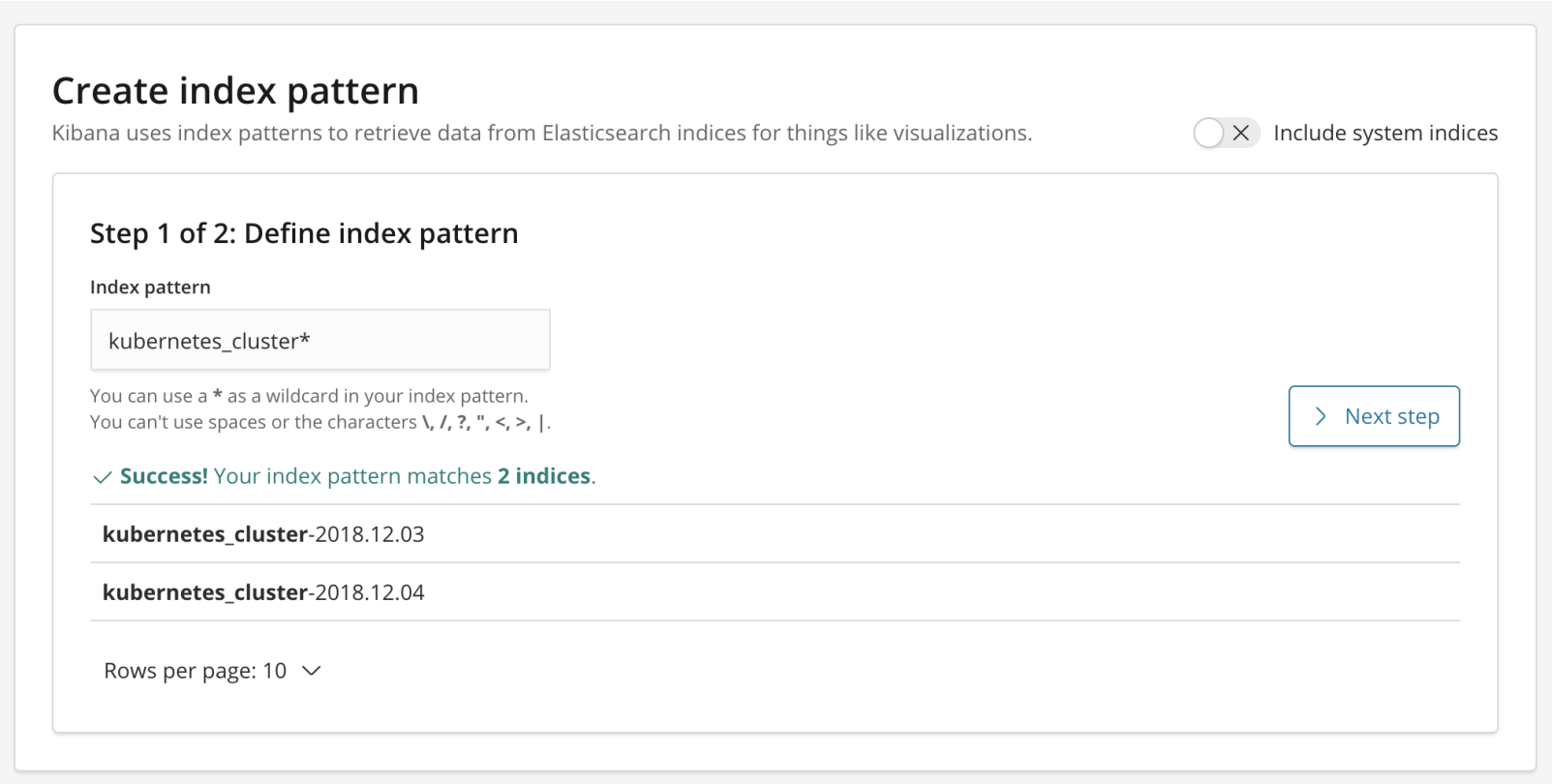
EFK

Если с `values.yaml` для kibana возникли проблемы, используйте gist по данной ссылке

```
helm upgrade --install kibana elastic/kibana --namespace logging --version 7.7.1 -f values.yaml
```

Kibana

Создайте index pattern, который будет указывать на индексы, создаваемые Fluent Bit:



Kibana

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
```

Kibana

Откройте любое событие из результата поиска - в нем видно множество полей с данными о Kubernetes.

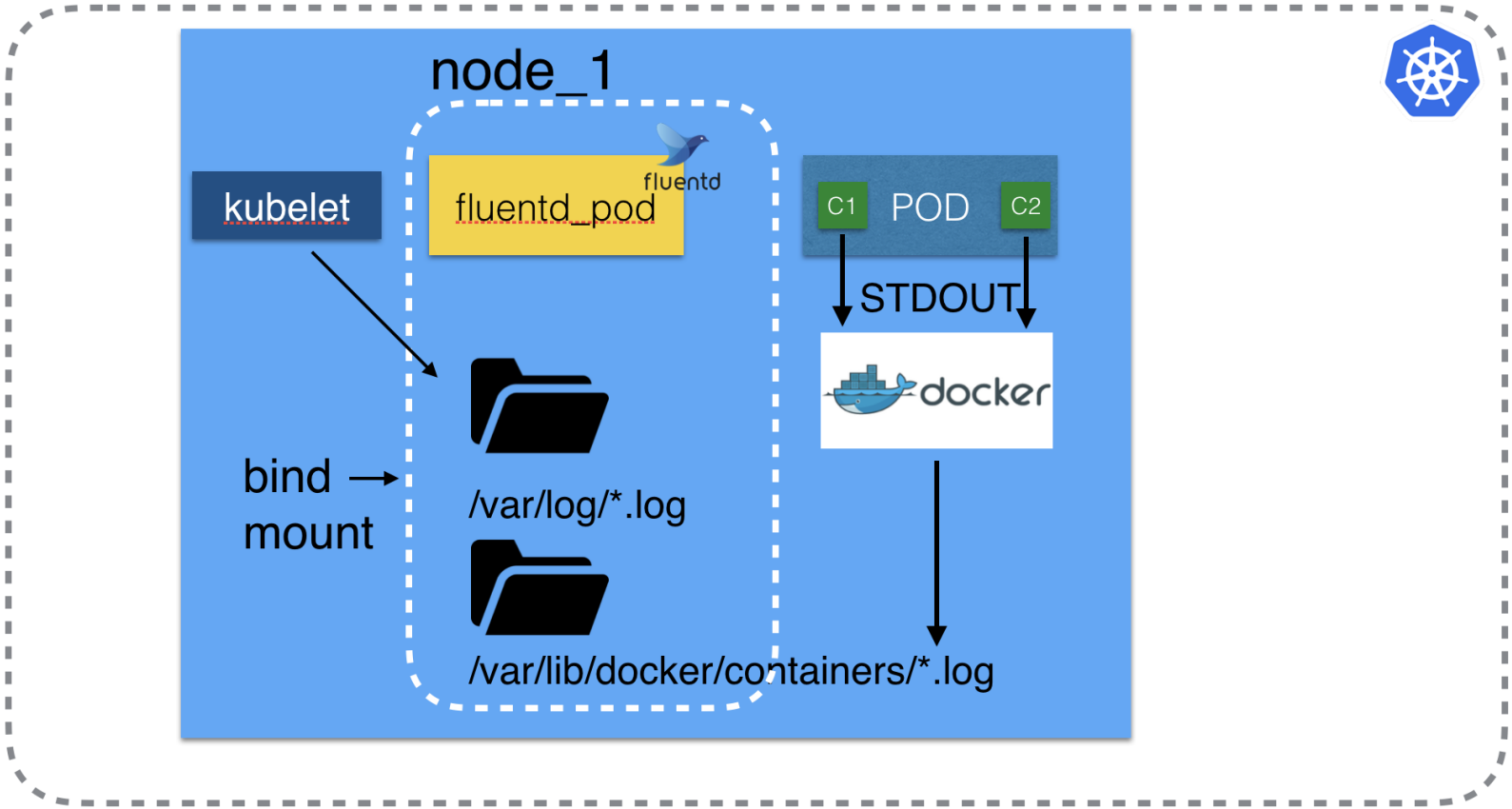
Table | JSON | View surrounding documents | View single document

@timestamp	December 4th 2018, 17:14:35.299
t _id	zp6SeWcBPW0vTSJYFAfo
t _index	kubernetes_cluster-2018.12.04
# _score	-
t _type	flb_type
t kubernetes.container_name	ui
t kubernetes.docker_id	4d677f5676a0582122d985d1b9a89b323a1d970f2bd3000fb852050c6e2a6f2c
t kubernetes.host	gke-user34-default-pool-d7cb4c75-q6x0
t kubernetes.labels.app	reddit
t kubernetes.labels.component	ui
t kubernetes.labels.pod-template-hash	3897109324
t kubernetes.labels.release	staging
t kubernetes.namespace_name	staging
t kubernetes.pod_id	82a4ce42-f6f9-11e8-8f7d-42010af00186
t kubernetes.pod_name	staging-ui-7dfc54f768-jm5km
t log	E, [2018-12-04T14:14:35.298680 #1] ERROR -- : Error while connecting to http://zipkin:9411/api/v1/spans: Faraday::ConnectionFailed with message 'Failed to open TCP connection to zipkin:9411 (getaddrinfo: Name does not resolve)'. Please make sure the URL / port are properly specified for the Zipkin server.
t stream	stdout
time	December 4th 2018, 17:14:35.299

EFK

1. Особенность работы `Fluent Bit` 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 `Fluent Bit` - видно в его configmap.yaml и daemonset.yaml
3. Общая схема работы `Fluent Bit` представлена на следующем слайде

Fluent Bit


node_1
kubelet
fluentd_pod
fluentd
C1
POD
C2
STDOUT
docker
bind
mount
/var/log/*.log
/var/lib/docker/containers/*.log