Установка Prometheus Operator

Цели практики

Целью данной практической работы является демонстрация процесса установки Prometheus Operator в кластер. Установка будет производиться с помощью Helm Chart.

С чем будем работать

- Helm chart **kube-prometheus-stack** - необходим для установки Prometheus Operator в кластер

Где будем выполнять практику

ВНИМАНИЕ: Перед тем, как работать со стендом, его необходимо запустить из Личного кабинета

- Подключаемся по SSH к sbox.slurm.io:

  ```
  ssh s<номер вашего логина>@sbox.slurm.io
  ```

- Подключаемся к своему кластеру Kubernetes:

  ```
  ssh master-1.s<ваш номер логина>
  ```

- Становимся root:

  ```
  sudo -s
  ```

Практика

Подготовительная работа

1. Клонируем манифесты

Для выполнения практической части потребуются манифесты. Их необходимо складировать с gitlab, сделать это можно выполнив команды:

```
cd ~
```

```
git clone git@gitlab.slurm.io:edu/lmk8s.git
```

2. Подключаем репо prometheus-community:

```
helm repo add prometheus-community
https://prometheus-community.github.io/helm-charts
```

```
helm repo update
```

3. Создаем namespace

В рамках данного урока Prometheus Operator будет устанавливаться в отдельный namespace: prometheus-operator. Для его создания необходимо выполнить команду:

```
kubectl create ns prometheus-operator
```

4. Создаем Secret для basic-auth

Из соображений безопасности доступ к Prometheus и Alertmanager необходимо ограничить с помощью basic-auth. Для этого создадим секрет, в котором будут данные для basi auth. Имя пользователя и пароль рекомендовано установить такими же, как и для личного кабинета. Для создания секрета необходимо выполнить команды.

```
htpasswd -c auth  <имя студента, например s0000000>
```

```
kubectl create secret generic admin-basic-auth --from-file=auth -n
prometheus-operator
```

5. Подготовка values

Для установки Prometheus будет использоваться уже подготовленные значения values: `~/lmk8s/6.prometheus_operator/operator_values.yml`

Но в них необходимо заменить адрес Ingress для Prometheus, это можно сделать выполнив команду:

```
sed -i 's/<student number>/Ваш номер студента, например s000000/g' ~/lmk8s/6.prometheus_operator/operator_values.yml
```

Установка Prometheus в кластер

Теперь все готово для установки и достаточно выполнить команду:

```
helm upgrade -i prom-operator prometheus-community/kube-prometheus-stack -n prometheus-operator -f ~/lmk8s/6.prometheus_operator/operator_values.yml
```

В результате будут установлены следующие компоненты:

- Prometheus operator
- Prometheus Server
- Alertmanager
- Grafana
- Kube-state-metrics metrics
- Node exporter
- Servicemonitor для основных компонентов control-plane
- Набор правил alerts для k8s
- Набор dashboards для основных компонентов k8s

Проверка работоспособности

После установки, необходимо проверить, что все работает корректно:

- Открываем в браузере в анонимном режиме `http://prometheus.<номер студента>.edu.slurm.io`

  username: s<номер студента>

  password: Ваш пароль

- Открываем в браузере в анонимном режиме `http://alertmanager.<номер студента>.edu.slurm.io`

username: s<номер студента>

password: Ваш пароль

- Открываем в браузере в анонимном режиме `http://grafana.<номер студента>.edu.slurm.io`

  username: admin

  password: prom-operator

Если у вас в браузере 503 ошибка, проверьте что создался секрет с basic-auth. Для этого необходимо выполнить команду: `kubectl get secrets -n prometheus-operator admin-basic-auth`

Ключевые изменения в values.yml

Обратите внимание, что в values.yml для чарта были сделаны изменения, наиболее ключевые из них:

- **Заданы ресурсы для все объектов** - это общее хорошее правило для всего, что запускается в Kubernetes кластере. Во первых, заданные request и limits позволяют scheduler корректно распределять Pod по node. Во вторых, это важно для избегания ситуации утечки ресурсов, а в случае с Prometheus это особенно актуально, так как он является достаточно "прожорливым".
- Задан **podDisruptionBudget** - данная настройка влияет на то, сколько Pod может быть одновременно выключено (распространяется только на evection api). Она позволяет гарантировать, что при обслуживании кластера Kubernetes не будут выключены все Pod с Prometheus.
- **podAntiAffinity** - данная настройка позволяет гарантировать, что Pod из statefulset не будут запущены на одной node. Без данной настройки возможна ситуация, когда все Pod с Prometheus и Alertmanager будут запущены на одной node, и в случае выхода из строя этой node, кластер останется без мониторинга (Pod для statefulset не перезапускаются автоматически)
- **retentionSize** - эта настройка позволяет ограничить хранимую историю по размеру. Обратите внимание, что мы запрашиваем pvc на 1Gb, а размер

ограничен 700Mb, это связано с тем, что фактический размер pv будет немного меньше 1Gb.

- **serviceMonitorSelectorNilUsesHelmValues** и **podMonitorSelectorNilUsesHelmValues** - это настройка влияет на то, в какие метки должны быть у ServiceMonitor и PodMonitor. При значение true, они должны совпадать с labels, которые выставляет Helm на prometheus operator, как правило, это неудобно.