Урок 7. Продвинутые абстракции Kubernetes

Практическое задание

Студент Виталий Зайцев, DE\_622

Contents

[Домашнее задание 2](#_Toc84082220)

[Разверните в кластере сервер системy мониторинга Prometheus 2](#_Toc84082221)

[Создайте в кластере ConfigMap со следующим содержимым 2](#_Toc84082222)

[Создайте объекты для авторизации Prometheus сервера в Kubernetes-API. 2](#_Toc84082223)

[Создайте StatefulSet для Prometheus сервера из образа prom/prometheus:v2.19.2 с одной репликой 3](#_Toc84082224)

[Реализация 4](#_Toc84082225)

[configmap.yml 4](#_Toc84082226)

[Oбъекты для авторизации Prometheus сервера в Kubernetes-API: serviceaccount.yml 5](#_Toc84082227)

[serviceaccount.yml 5](#_Toc84082228)

[StatefulSet для Prometheus сервера из образа prom/prometheus:v2.19.2 с одной репликой: statefulset.yml 6](#_Toc84082229)

[statefulset.yml 6](#_Toc84082230)

[Service для стейтфулсета: service.yml 7](#_Toc84082231)

[service.yml 7](#_Toc84082232)

[Ingress для стейтфулсета: ingress.yml 8](#_Toc84082233)

[ingress.yml 8](#_Toc84082234)

[НЕУСПЕХ! Развёртывание 9](#_Toc84082235)

[НЕУСПЕХ! Проверьте что при обращении из браузера на белый IP вы видите открывшееся приложение Prometheus 13](#_Toc84082236)

[Создание DaemonSet node-exporter: daemonset.yml 14](#_Toc84082237)

[daemonset.yml 14](#_Toc84082238)

[Развёртывание daemonset.yml 14](#_Toc84082239)

[НЕУСПЕХ! Откройте в браузере интерфейс Prometheus 14](#_Toc84082240)

# Домашнее задание

**! Задание нужно выполнять в нэймспэйсе default**

## Разверните в кластере сервер системy мониторинга Prometheus

### Создайте в кластере ConfigMap со следующим содержимым

prometheus.yml: |  
global:  
scrape\_interval: 30s

scrape\_configs:  
- job\_name: 'prometheus'  
static\_configs:  
- targets: ['localhost:9090']

- job\_name: 'kubernetes-nodes'

kubernetes\_sd\_configs:

- role: node

relabel\_configs:

- source\_labels: [\_\_address\_\_]

regex: (.+):(.+)

target\_label: \_\_address\_\_

replacement: ${1}:9101

### Создайте объекты для авторизации Prometheus сервера в Kubernetes-API.

---

apiVersion: v1  
kind: ServiceAccount  
metadata:  
 name: prometheus

namespace: default

---

apiVersion: rbac.authorization.k8s.io/v1beta1  
kind: ClusterRole  
metadata:  
 name: prometheus  
rules:  
- apiGroups: [""]  
 resources:  
 - nodes

verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1beta1  
kind: ClusterRoleBinding  
metadata:  
 name: prometheus  
roleRef:  
 apiGroup: rbac.authorization.k8s.io  
 kind: ClusterRole  
 name: prometheus  
subjects:  
- kind: ServiceAccount  
 name: prometheus  
 namespace: default

### Создайте StatefulSet для Prometheus сервера из образа prom/prometheus:v2.19.2 с одной репликой

В нем должнен быть описан порт 9090 TCP

volumeClaimTemplate - ReadWriteOnce, 5Gi, подключенный по пути /prometheus Подключение конфигмапа с настройками выше по пути /etc/prometheus

Так же в этом стейтфулсете нужно объявить initContainer для изменения прав на вольюм data на 777. См пример из лекции 4: practice/4.resources-and-persistence/persistence/deployment.yaml

**! Не забудьте указать обязательное поле serviceName**

Так же укажите поле serviceAccount: prometheus на одном уровне с containers, initContainers, volumes См пример с rabbitmq из материалов лекции.

Создайте service и ingress для этого стейтфулсета, так чтобы запросы с любым доменом на белый IP вашего сервиса nginx-ingress-controller (тот что в нэймспэйсе ingress-nginx с типом LoadBalancer) шли на приложение

Проверьте что при обращении из браузера на белый IP вы видите открывшееся приложение Prometheus

В этом же неймспэйсе создайте DaemonSet node-exporter как в примере к лекции: practice/7.advanced-abstractions/daemonset.yaml

Откройте в браузере интерфейс Prometheus. Попробуйте открыть Status -> Targets Тут вы должны увидеть все ноды своего кластера, которые Prometheus смог определить и собирает с ним метрики.

Так же можете попробовать на вкладке Graph выполнить запрос node\_load1 - это минутный Load Average для каждой из нод в кластере.

# Реализация

## configmap.yml

---

apiVersion: v1

kind: ConfigMap

metadata:

name: prometheus-config

data:

prometheus.yml: |

global:

scrape\_interval: 30s

scrape\_configs:

- job\_name: 'prometheus'

static\_configs:

- targets: ['localhost:9090']

- job\_name: 'kubernetes-nodes'

kubernetes\_sd\_configs:

- role: node

relabel\_configs:

- source\_labels: [\_\_address\_\_]

regex: (.+):(.+)

target\_label: \_\_address\_\_

replacement: ${1}:9101

## Oбъекты для авторизации Prometheus сервера в Kubernetes-API: serviceaccount.yml

### serviceaccount.yml

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: prometheus

namespace: default

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: prometheus

rules:

- apiGroups: [""]

resources:

- nodes

verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: prometheus

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: prometheus

subjects:

- kind: ServiceAccount

name: prometheus

namespace: default

## StatefulSet для Prometheus сервера из образа prom/prometheus:v2.19.2 с одной репликой: statefulset.yml

…

*В нем должнен быть описан порт 9090 TCP*

*volumeClaimTemplate - ReadWriteOnce, 5Gi, подключенный по пути /prometheus*

*Подключение конфигмапа с настройками выше по пути /etc/prometheus*

*Так же в этом стейтфулсете нужно объявить initContainer для изменения прав на вольюм data на 777. См пример из лекции 4: practice/4.resources-and-persistence/persistence/deployment.yaml*

*Не забудьте указать обязательное поле serviceName*

*Также укажите поле serviceAccount: prometheus на одном уровне с containers, initContainers, volumes См пример с rabbitmq из материалов лекции.*

…

### statefulset.yml

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: prometheus

spec:

serviceName: prometheus

replicas: 1

selector:

matchLabels:

app: prometheus

template:

metadata:

labels:

app: prometheus

spec:

serviceAccount: prometheus

initContainers:

- image: busybox

name: mount-permissions-fix

command: ["sh", "-c", "chmod 777 /data"]

volumeMounts:

- name: data

mountPath: /data

terminationGracePeriodSeconds: 10

containers:

- name: prometheus

image: prom/prometheus:v2.19.2

ports:

- name: admin

protocol: TCP

containerPort: 9090

imagePullPolicy: Always

volumeMounts:

- name: config-volume

mountPath: /etc/prometheus

volumes:

- name: config-volume

configMap:

name: prometheus-config

items:

- key: prometheus.yml

path: prometheus.yml

volumeClaimTemplates:

- metadata:

name: data

spec:

accessModes: ["ReadWriteOnce"]

resources:

requests:

storage: 5Gi

storageClassName: csi-ceph-hdd-dp1

## Service для стейтфулсета: service.yml

…

*Создайте service и ingress для этого стейтфулсета, так чтобы запросы с любым доменом на белый IP вашего сервиса nginx-ingress-controller (тот что в нэймспэйсе ingress-nginx с типом LoadBalancer) шли на приложение*

…

### service.yml

---

kind: Service

apiVersion: v1

metadata:

name: prometheus

labels:

app: prometheus

spec:

clusterIP: None

ports:

- name: admin

protocol: TCP

port: 9090

targetPort: 9090

selector:

app: prometheus

## Ingress для стейтфулсета: ingress.yml

…

*Создайте service и ingress для этого стейтфулсета, так чтобы запросы с любым доменом на белый IP вашего сервиса nginx-ingress-controller (тот что в нэймспэйсе ingress-nginx с типом LoadBalancer) шли на приложение*

…

### ingress.yml

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-gateway

annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: prometheus

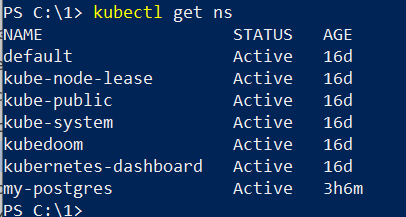
port:

number: 9090

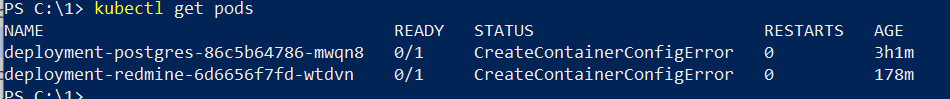
## НЕУСПЕХ! Развёртывание

Удалим объекты из предыдущего домашнего задания

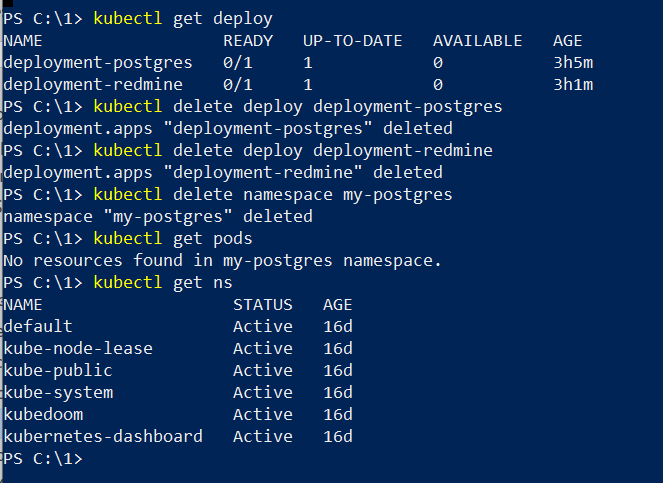
kubectl get ns



kubectl get pods



Удаляем объекты-сущности предыдущего задания

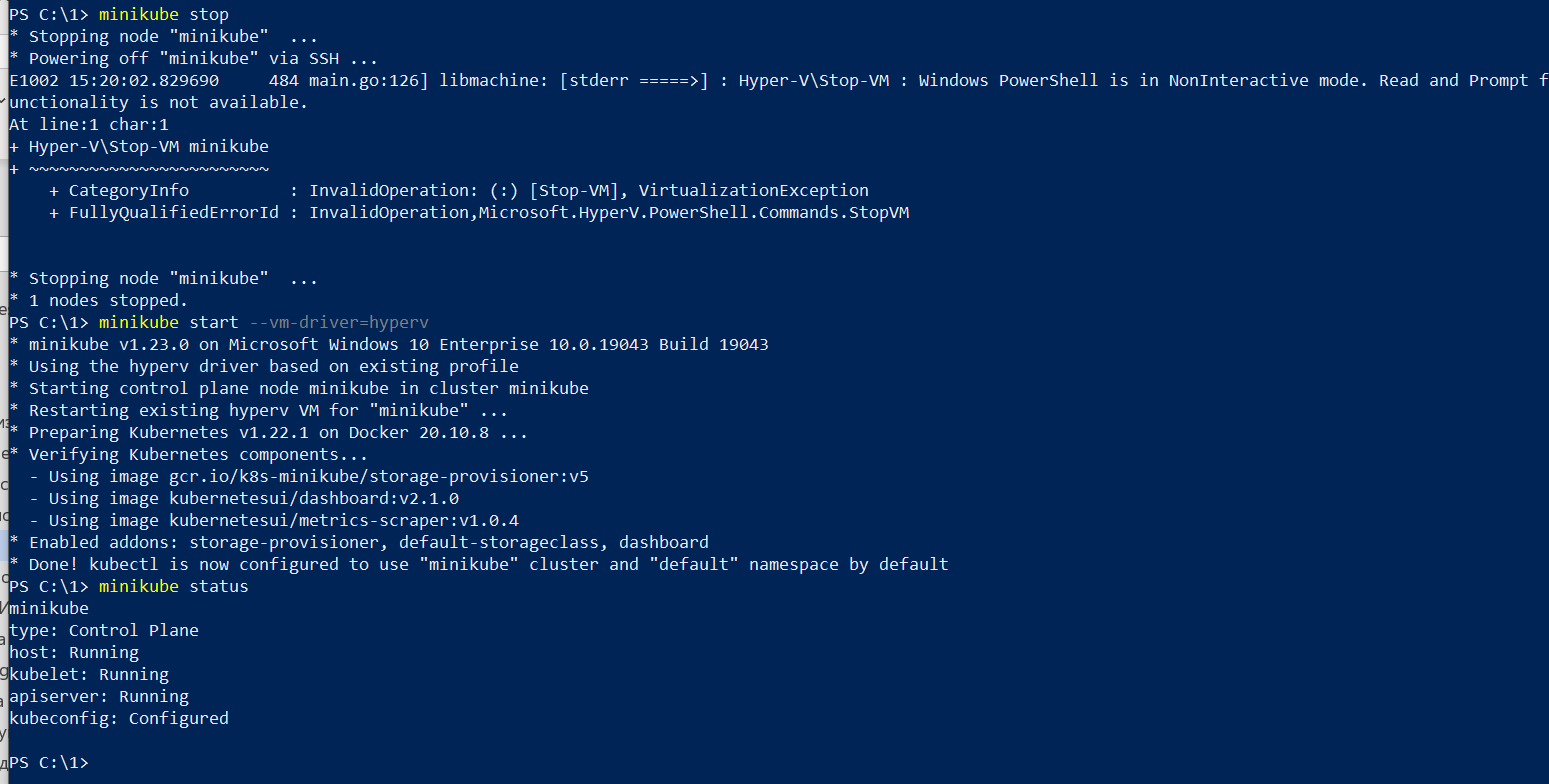


И даже всё перезапустим

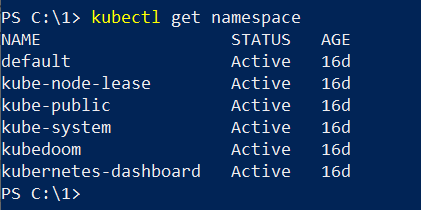
minikube stop

minikube start --vm-driver=hyperv

minikube status

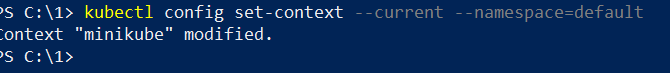


kubectl get namespace



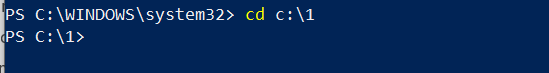
**Установка контекста в namespace default**

kubectl config set-context --current --namespace=default



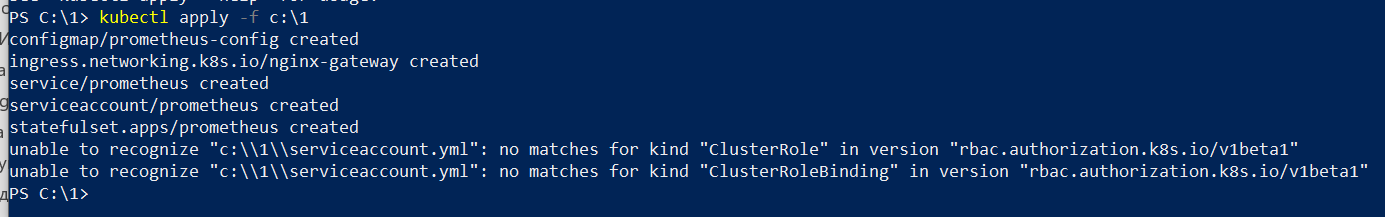
Переходим в папку размещения yml файлов

cd C:\1



Применяем манифесты

kubectl apply -f c:\1

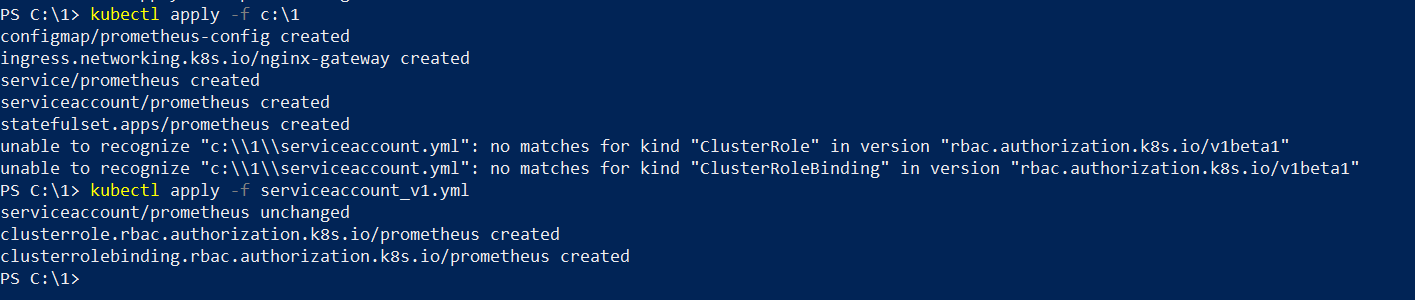


Изучаем ошибку <https://kubernetes.io/docs/reference/access-authn-authz/rbac/> , меняем строчки в serviceaccount\_v1.yml на

apiVersion: rbac.authorization.k8s.io/v1

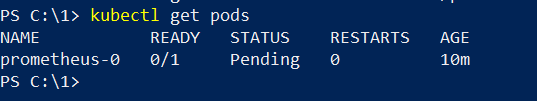
снова применяем - успех

kubectl apply -f serviceaccount\_v1.yml

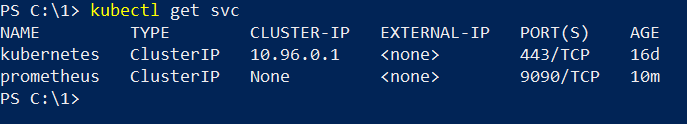


Проверяем

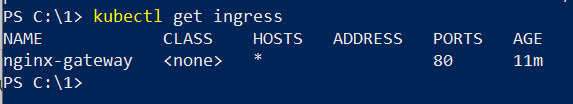
kubectl get pods



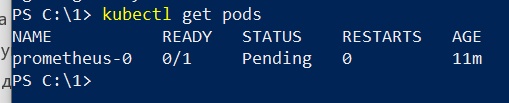
kubectl get svc



kubectl get ingress



kubectl get pods



НЕУСПЕХ!!! Под НЕ запускается (статус pending)

## НЕУСПЕХ! Проверьте что при обращении из браузера на белый IP вы видите открывшееся приложение Prometheus

Под в статусе pending

Я не знаю как получить белый IP

## Создание DaemonSet node-exporter: daemonset.yml

### daemonset.yml

---

apiVersion: apps/v1

kind: DaemonSet

metadata:

labels:

app: node-exporter

name: node-exporter

spec:

updateStrategy:

rollingUpdate:

maxUnavailable: 1

type: RollingUpdate

selector:

matchLabels:

app: node-exporter

template:

metadata:

labels:

app: node-exporter

spec:

containers:

- args:

- --web.listen-address=0.0.0.0:9101

- --path.procfs=/host/proc

- --path.sysfs=/host/sys

- --collector.filesystem.ignored-mount-points=^/(dev|proc|sys|var/lib/docker/.+)($|/)

- --collector.filesystem.ignored-fs-types=^(autofs|binfmt\_misc|cgroup|configfs|debugfs|devpts|devtmpfs|fusectl|hugetlbfs|mqueue|overlay|proc|procfs|pstore|rpc\_pipefs|securityfs|sysfs|tracefs)$

image: quay.io/prometheus/node-exporter:v0.16.0

imagePullPolicy: IfNotPresent

name: node-exporter

volumeMounts:

- mountPath: /host/proc

name: proc

- mountPath: /host/sys

name: sys

- mountPath: /host/root

name: root

readOnly: true

hostNetwork: true

hostPID: true

tolerations:

- effect: NoSchedule

operator: Exists

nodeSelector:

beta.kubernetes.io/os: linux

volumes:

- hostPath:

path: /proc

type: ""

name: proc

- hostPath:

path: /sys

type: ""

name: sys

- hostPath:

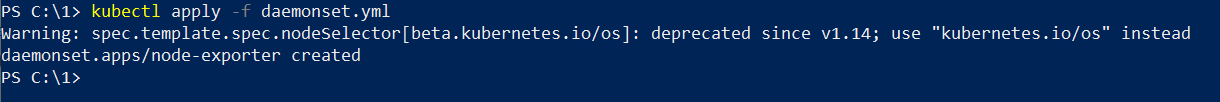
path: /

type: ""

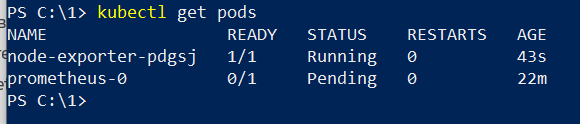
name: root

### Развёртывание daemonset.yml

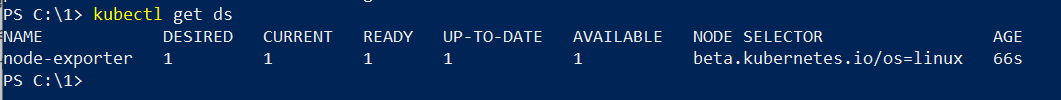
kubectl apply -f daemonset.yml



kubectl get pods



kubectl get ds



## НЕУСПЕХ! Откройте в браузере интерфейс Prometheus

Под в статусе pending

Я не знаю как получить белый IP