Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



Лабораторная работа №4

по дисциплине «Автоматизация разработки и эксплуатации ПО»

«Развертывание кластера Kubernetes»

ИСПОЛНИТЕЛЬ:

студент ИУ5-73Б

Алёшин А.Д.

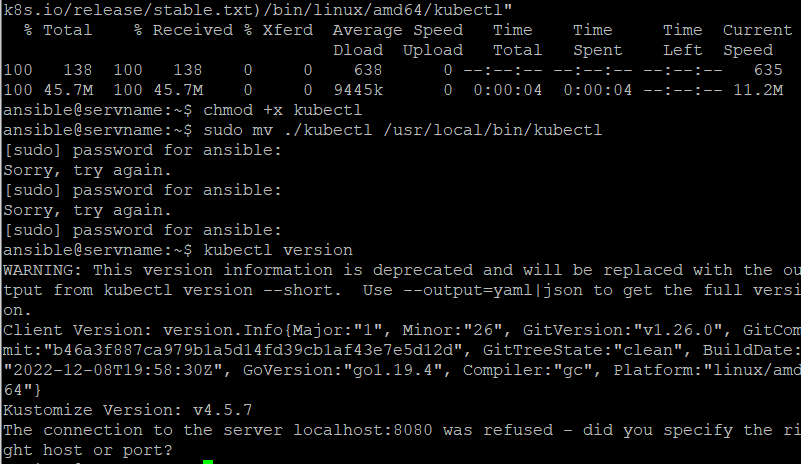
1. Установка kubectl

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version



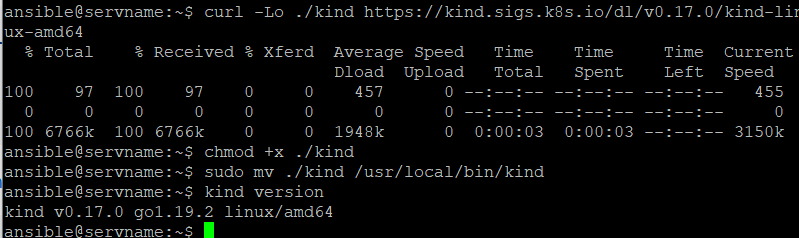
1. Установка Kind

curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.17.0/kind-linux-amd64

chmod +x ./kind

sudo mv ./kind /usr/local/bin/kind

kind version



1. Создаем кластер

cat <<EOF | kind create cluster --config=-

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

kubeadmConfigPatches:

- |

kind: InitConfiguration

nodeRegistration:

kubeletExtraArgs:

node-labels: "ingress-ready=true"

extraPortMappings:

- containerPort: 80

hostPort: 80

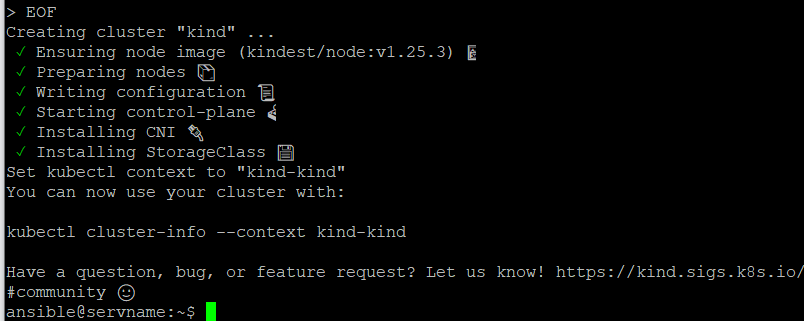
protocol: TCP

- containerPort: 443

hostPort: 443

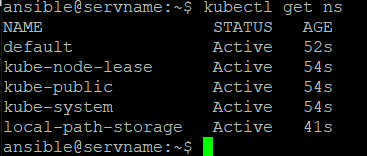
protocol: TCP

EOF



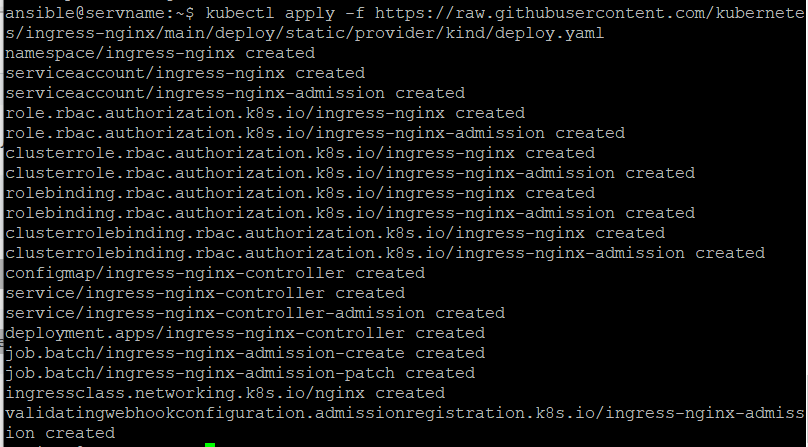
проверим что все ок, командой

kubectl get ns



получаем список namespace, ставим ingress

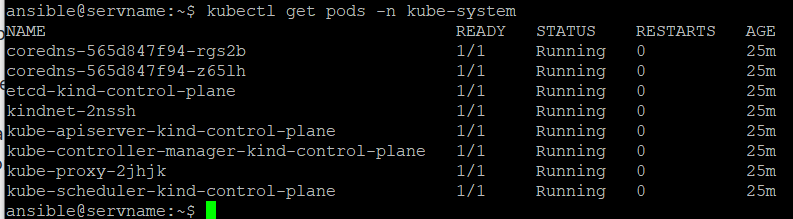
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml



## **Знакомство с kubernetes**

Посмотрим список подов в kube-system

kubectl get pods -n kube-system



## **Работа с kubernetes**

Создадим под c простым nginx, для этого выполним команды

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

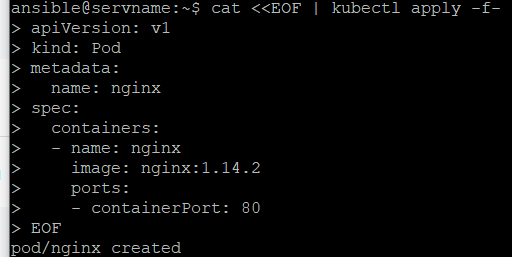
- name: nginx

image: nginx:1.14.2

ports:

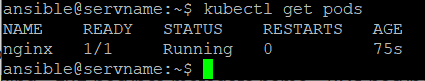
- containerPort: 80

EOF



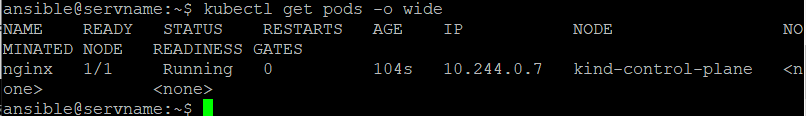
Проверим, что наш под создался.

kubectl get pods



Если указать параметр -o wide можно увидеть доп информацию

kubectl get pods -o wide



Создадим еще 1 под и подключимся к нему в интерактивном режиме

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: Pod

metadata:

name: netshoot

spec:

containers:

- name: netshoot

image: nicolaka/netshoot

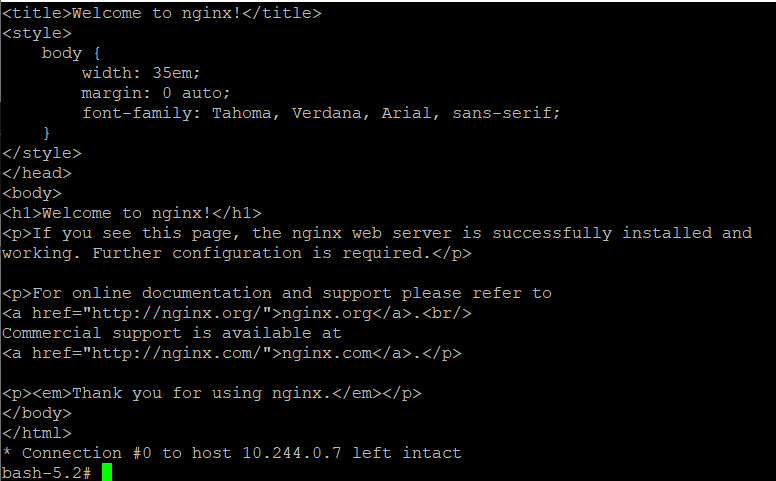
command: ["sleep", "infinity"]

EOF

kubectl exec -it netshoot -- bash

Дернем из этого пода созданный ранее nginx

curl -v 10.244.0.7:80



Удалим под nginx и пересоздадим его буквально той же самой командой

kubect delete pod nginx

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

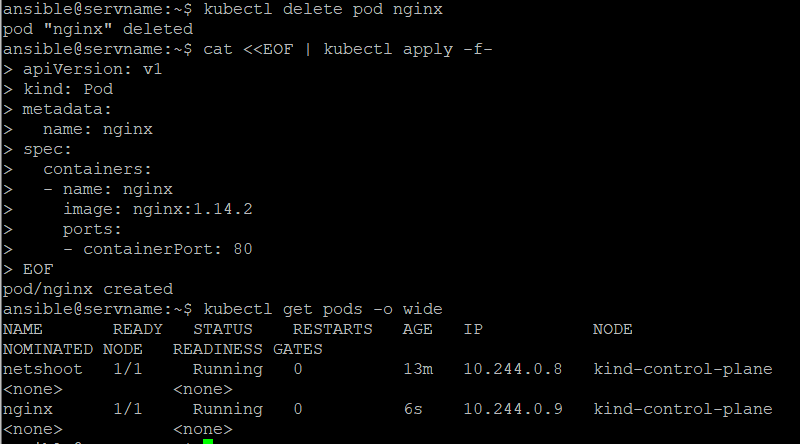
image: nginx:1.14.2

ports:

- containerPort: 80

EOF

kubectl get pods -o wide



Создадим сразу два типа сервисов CluesterIp и Headless

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

ports:

- protocol: TCP

port: 80

EOF

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: Service

metadata:

name: nginx-service-headless

spec:

clusterIP: None

selector:

app: nginx

ports:

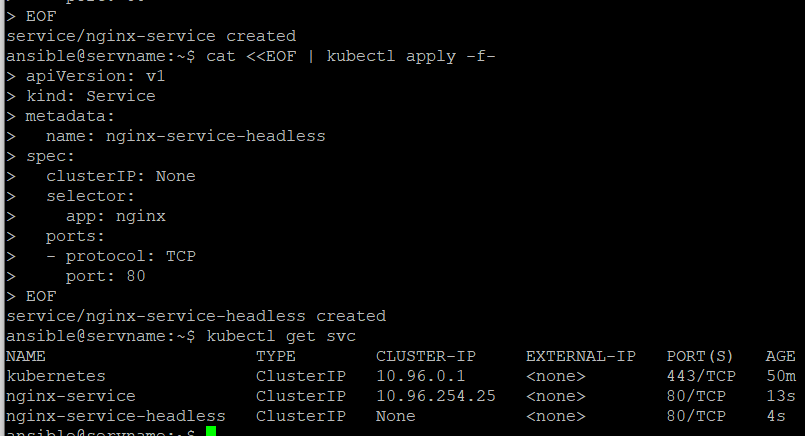
- protocol: TCP

port: 80

EOF

Проверяем, что наши сервсисы создались

kubectl get svc



Вернемся в под netshoot и проверим работоспособность нашего сервиса

curl 10.96.99.34

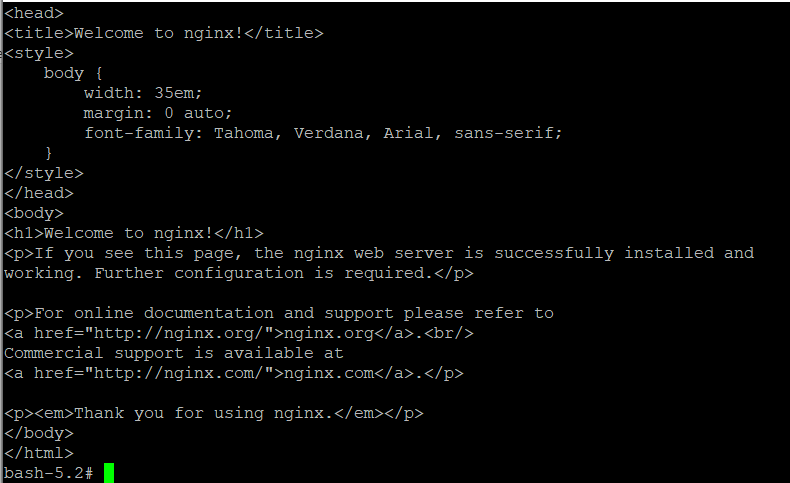
curl nginx-service

curl nginx-service.default

curl nginx-service.default.svc

curl nginx-service.default.svc.cluster.local

curl nginx-service-headless



Создадим Ingress для нашего сервиса nginx-service

cat <<EOF | kubectl apply -f-

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: minimal-ingress

annotations:

ingress.kubernetes.io/rewrite-target: /

spec:

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

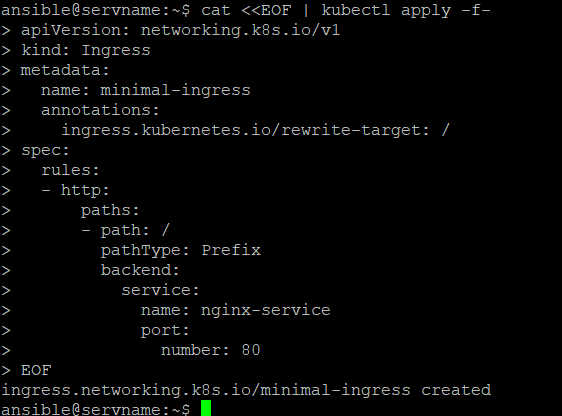
service:

name: nginx-service

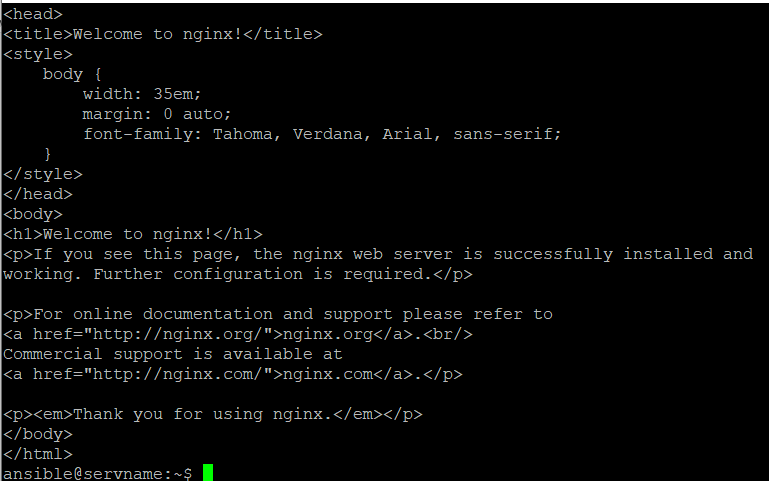
port:

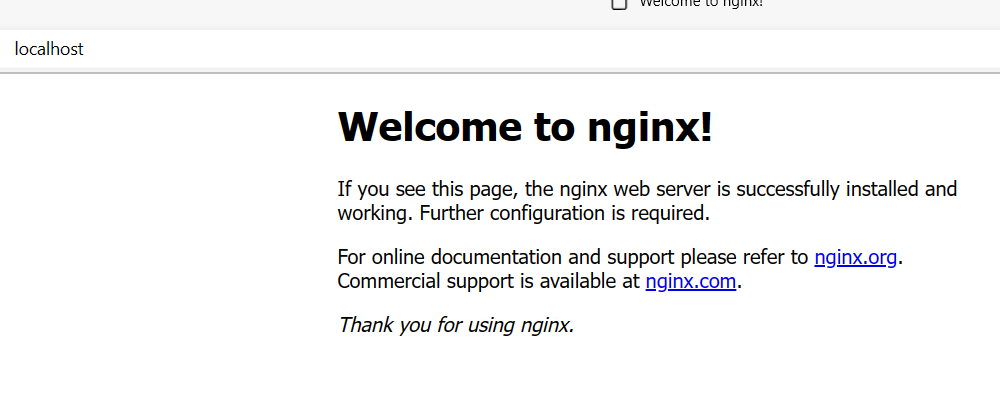
number: 80

EOF



Выполним curl localhost:80 напрямую с виртуальной машины





Удалим наш старый под nginx и создадим deployment

cat <<EOF | kubectl apply -f-

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

EOF

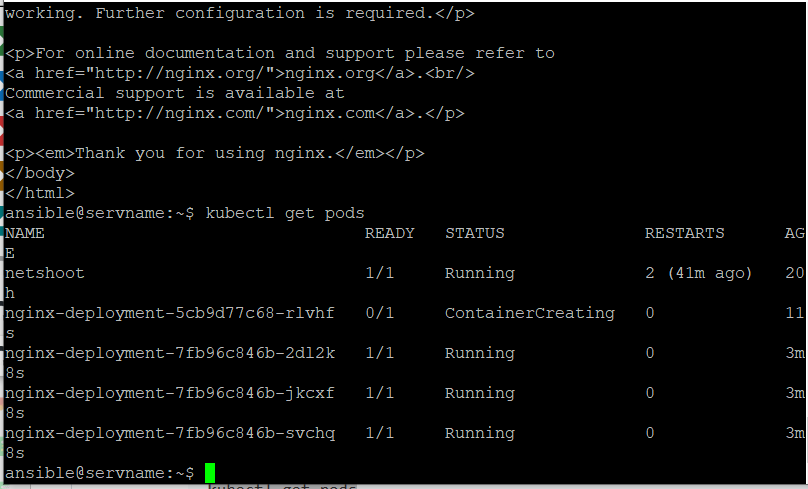


Проверим механизмы обновления deployment. Попытаемся обновить образ на несуществующий

kubectl set image deploy/nginx-deployment nginx=nginx:iam-not-exists

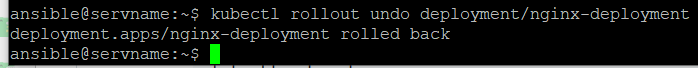
curl localhost:80

kubectl get pods



Откатимся на предыдущую версию

kubectl rollout undo deployment/nginx-deployment



Применяем http пробу к нашему деплойменту

cat <<EOF | kubectl apply -f-

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

livenessProbe:

httpGet:

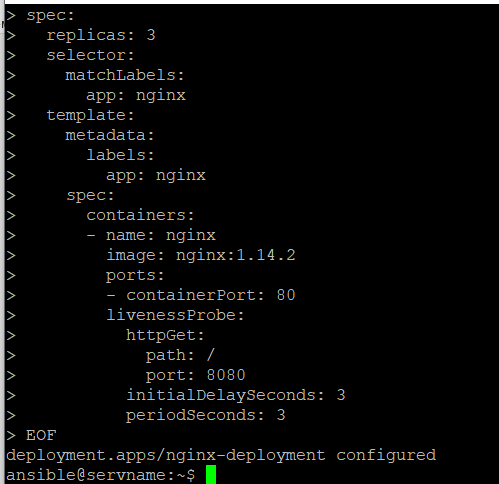
path: /

port: 8080

initialDelaySeconds: 3

periodSeconds: 3

EOF



ConfigMap

cat <<EOF | kubectl apply -f-

apiVersion: v1

kind: ConfigMap

metadata:

name: nginx-configmap

data:

student-server.conf: |

server {

listen 80 default\_server;

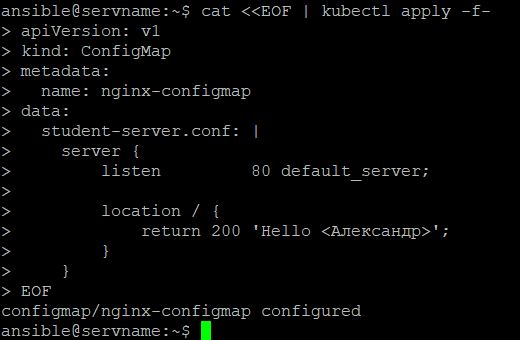
location / {

return 200 'Hello <Александр>';

}

}

EOF



И подклюаем конфиг к поду

cat <<EOF | kubectl apply -f-

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

livenessProbe:

httpGet:

path: /

port: 8080

initialDelaySeconds: 3

periodSeconds: 3

volumeMounts:

- name: nginx-cm

mountPath: "/etc/nginx/conf.d"

readOnly: true

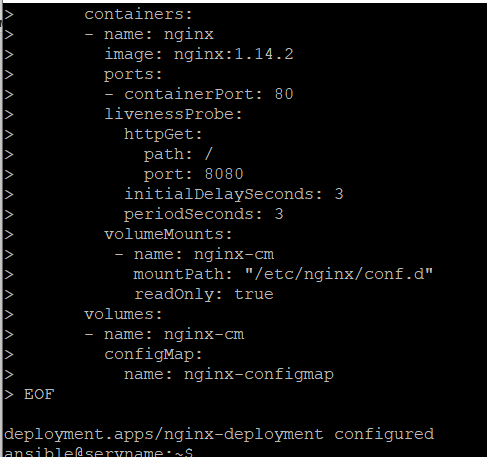
volumes:

- name: nginx-cm

configMap:

name: nginx-configmap

EOF



Проверяем что все ок через curl localhost:80 и получаем наш ответ, который мы задали в конфиге.

