DevOps

Л10. Kubernetes - basics

Виктор Моисеев
+7-902-83-145-30
t.me/v_paranoid
victorparanoid@gmail.com

План курса

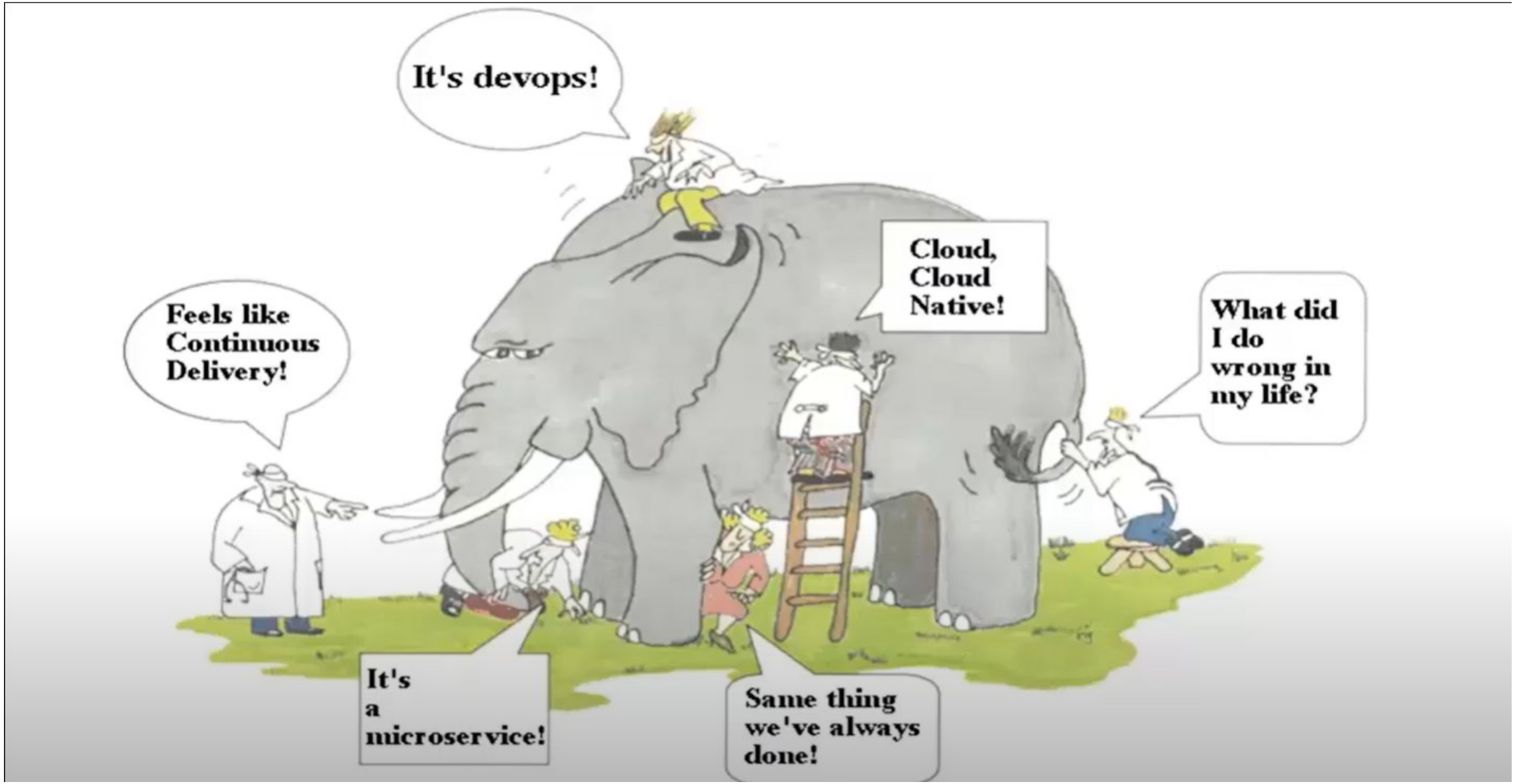
1. Введение в DevOps
2. Базовое администрирование Linux
3. Системы контроля версионности кода (git)
4. Оркестровка (Ansible)
5. Контейнеризация (docker)
6. Микросервисная архитектура и оркестровка контейнеров (k8s)
7. Непрерывная интеграция и доставка (CI/CD, Github Actions, ArgoCD)
8. Инфраструктура как код (IaC, Terraform)
9. Мониторинг (Prometheus)

10. Kubernetes - basics

1. Что такое k8s
2. Почему и зачем это всё?
3. Архитектура
4. Networking model
5. Установка
6. Развертывание нагрузки
7. pod/deployment (undo)/replicaset/statefulset
8. service
9. rolling update / rollout undo

10. service selector/labels (dev/prod/test)

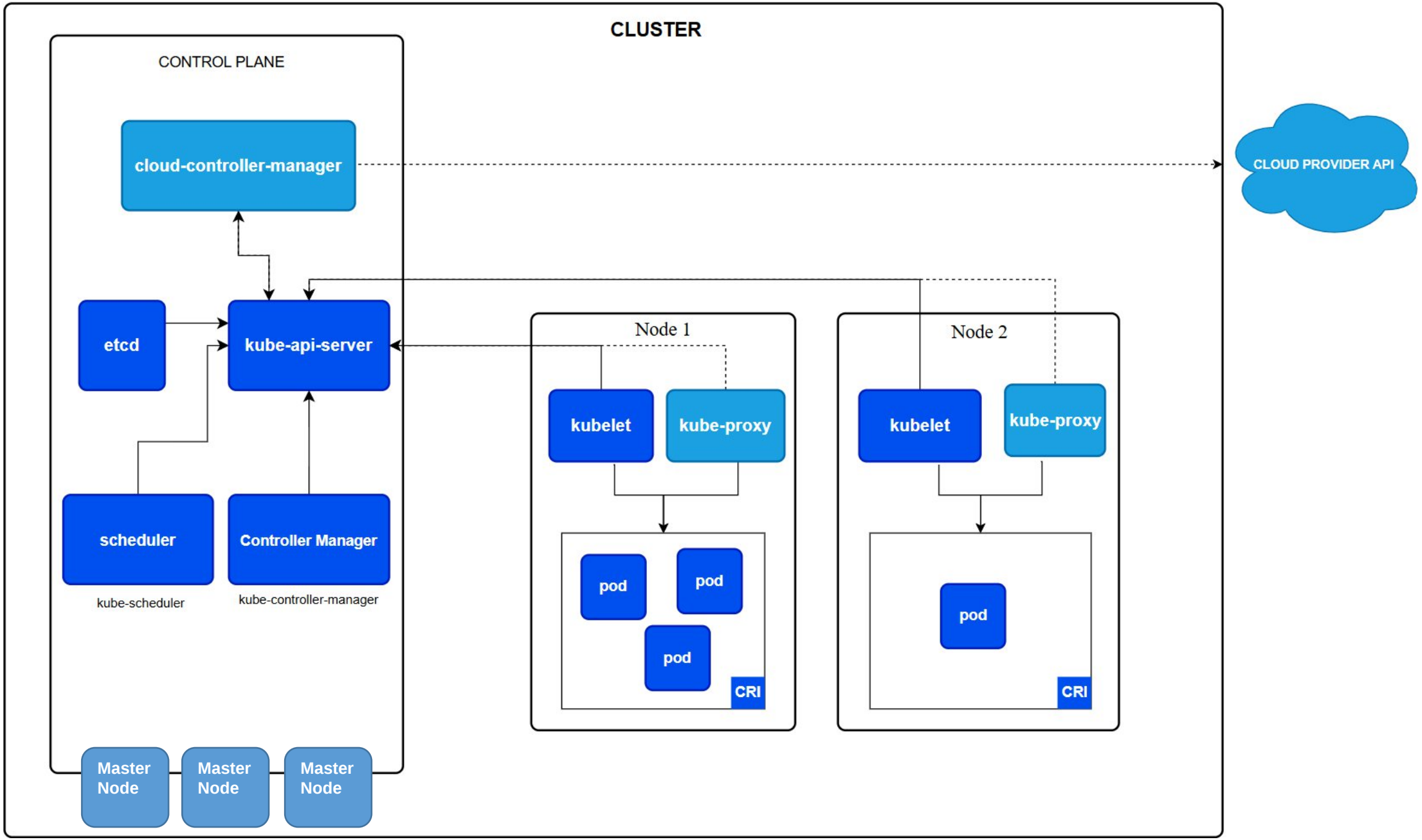
11. cicd commit to dev, tag to test, tag to prod
12. ingress / gateway



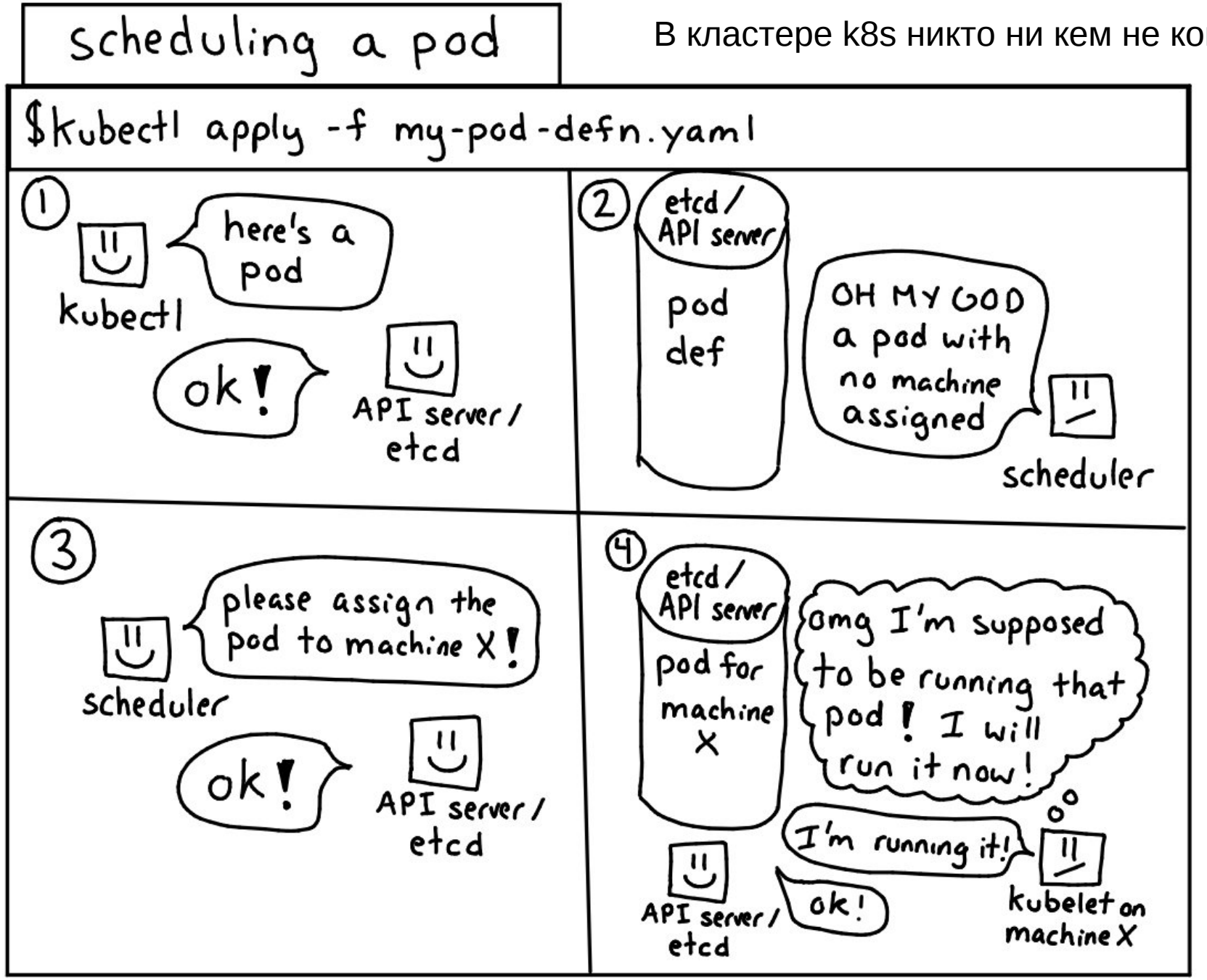
(c) Эндрю Шейфер

HighAvailability
Scalability
Visible
Decoupled

Предыдущие решения


CLUSTER
CONTROL PLANE
cloud-controller-manager
CLOUD PROVIDER API
etcd
kube-api-server
scheduler
Controller Manager
kube-scheduler
kube-controller-manager
Node 1
kubelet
kube-proxy
pod
pod
pod
CRI
Node 2
kubelet
kube-proxy
pod
CRI
Master Node
Master Node
Master Node

В кластере k8s никто ни кем не командует



Julia Evans
Scenes from Kubernetes

Пример Control-plain сервисов на мастер-нодах

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	calico-kube-controllers-5b97f5d8cf-hjdbj	1/1	Running	0	25m
kube-system	calico-node-6h6gc	1/1	Running	0	25m
kube-system	calico-node-fd5rr	1/1	Running	0	8m23s
kube-system	calico-node-kk5gm	1/1	Running	0	12m
kube-system	coredns-57575c5f89-pg7cm	1/1	Running	0	29m
kube-system	coredns-57575c5f89-tn8j4	1/1	Running	0	29m
kube-system	etcd-master-1.s065774.slurm.io	1/1	Running	0	29m
kube-system	etcd-master-2.s065774.slurm.io	1/1	Running	0	12m
kube-system	etcd-master-3.s065774.slurm.io	1/1	Running	0	8m13s
kube-system	kube-apiserver-master-1.s065774.slurm.io	1/1	Running	0	29m
kube-system	kube-apiserver-master-2.s065774.slurm.io	1/1	Running	0	11m
kube-system	kube-apiserver-master-3.s065774.slurm.io	1/1	Running	0	8m8s
kube-system	kube-controller-manager-master-1.s065774.slurm.io	1/1	Running	0	29m
kube-system	kube-controller-manager-master-2.s065774.slurm.io	1/1	Running	0	12m
kube-system	kube-controller-manager-master-3.s065774.slurm.io	1/1	Running	0	8m19s
kube-system	kube-proxy-5s85v	1/1	Running	0	8m23s
kube-system	kube-proxy-d4lkb	1/1	Running	0	29m
kube-system	kube-proxy-mbn75	1/1	Running	0	12m
kube-system	kube-scheduler-master-1.s065774.slurm.io	1/1	Running	0	29m
kube-system	kube-scheduler-master-2.s065774.slurm.io	1/1	Running	0	12m
kube-system	kube-scheduler-master-3.s065774.slurm.io	1/1	Running	0	8m11s

Workload

Deployment (ReplicaSet)
StatefulSet
DaemonSet
Job/Cronjob

Workload

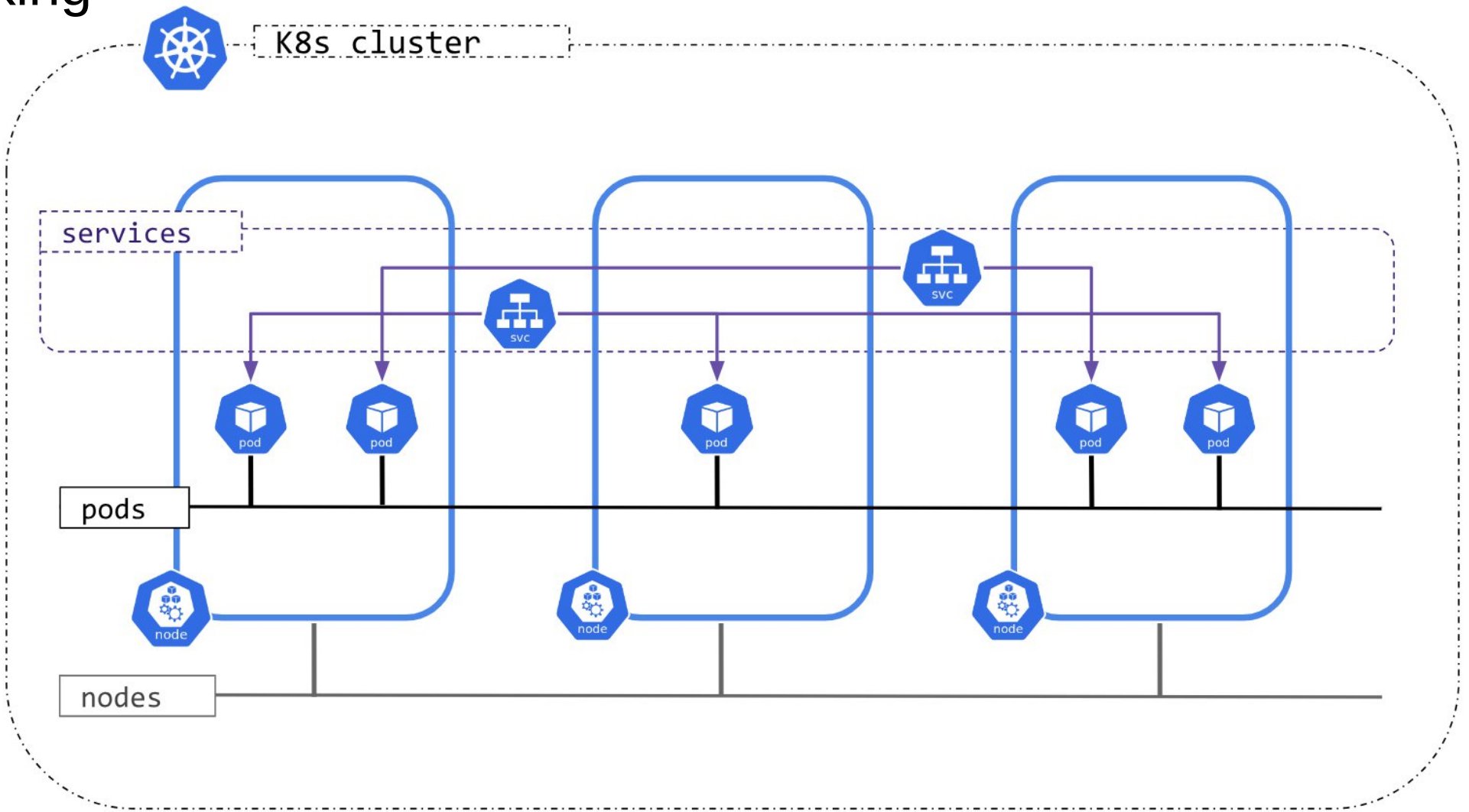
Deployment (ReplicaSet)

kubectl apply -f deploym.yml

kubectl delete -f deploym.yml

```yaml
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
```

Networking



CNI
Network plugin

Services – абстракция, чтобы под мог обращаться к другим подам неявно - по имени сервиса

Services

Реализация сервисов на совести сетевых плагинов (обычно они создают сложные конструкции для iptables и т.п.)

ClusterIP
Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default that is used if you don't explicitly specify a type for a Service. You can expose the Service to the public internet using an Ingress or a Gateway.

NodePort
Exposes the Service on each Node's IP at a static port (the NodePort). To make the node port available, Kubernetes sets up a cluster IP address, the same as if you had requested a Service of type: ClusterIP.

LoadBalancer
Exposes the Service externally using an external load balancer. Kubernetes does not directly offer a load balancing component; you must provide one, or you can integrate your Kubernetes cluster with a cloud provider.

ExternalName
Maps the Service to the contents of the externalName field (for example, to the hostname api.foo.bar.example). The mapping configures your cluster's DNS server to return a CNAME record with that external hostname value. No proxying of any kind is set up.

External >>> Ingress / Gateway

Services

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Можно объединять множество объектов в один yaml через разделитель ---
Например

---
kind: deployment
.....
---
king: service
.....

П10. Деплой сервисов в k8s (minikube)

Задача: «Куберизировать» приложение Flask+Redis из предыдущей практики и обновить

1. Разворачиваем на vm мини-кластер k8s - minikube
2. (опционально) Подключаемся к кластеру с хостовой машины через приложение Lens
3. Собираем образы будущих контейнеров и прокидываем их внутрь minikube
4. Создаем манифесты – описания приложений и сервисов
5. Разворачиваем deployment/ReplicaSet с application-серверами и БД и сервисы
6. Активируем балансировщик входящих запросов (служба LoadBalancer)
7. Проверяем работу сервисов
8. Обновляем веб сервисы на новую версию (RollingUpdate)

Примеры: https://github.com/vparanoid/devops11kube

Установка minikube на Ubuntu 24.04

Требования: 4GB RAM 10GB HDD FREESPACE (можно меньше через отдельную настройку)

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

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Скачиваем и инсталлируем minikube
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
sudo dpkg -i minikube_latest_amd64.deb

# Добавляем себе права
sudo usermod -aG docker $USER && newgrp docker

# Отключаем своп
sudo swapoff -a

# Запускаем minikube в режими однонодового кластера k8s
minikube start --vm-driver=docker

# Если ошибка доступа к сокету, то применить
sudo chmod 666 /var/run/docker.sock

# Перелогиниваемся для применения настроек окружения
su - $USER
```

Что делать, если нет места на диске

1. **Увеличить размер диска вм в гипервизоре**

2. **Увеличить диски внутри вм (пример для дефолтной инсталляции Ubuntu 24.04)**

смотрим разделы
sudo fdisk -l
victor@ubuntu:~$ sudo parted
(parted) print all

```
Warning: Not all of the space available to /dev/sda appears to be used, you can fix the GPT to use all
of the
space (an extra 2097152 blocks) or continue with the current setting?
```

(parted) Fix/Ignore? fix

(parted) quit

```
# расширяем раздел на все свободное пространство
```
sudo growpart /dev/sda 3

```
# расширяем логическии том ubuntu-lv
```
sudo lvs
sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv

```
# расширяем фаиловую систему (пример для ext4)
```
mount
sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu--lv

Проверка minikube и k8s кластера на вм

```
victor@ubuntu:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

```
victor@ubuntu:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   25m   v1.31.0
victor@ubuntu:~$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   25m
```

[**OPTIONAL**]: Настраиваем доступ к управлению кластером с хостовой машины

Скопировать конфиг k8s и сертификаты minikube из vm на хостовую машину:

```
scp -P <FORWARDED-PORT>  <VM-USER>@127.0.0.1:/home/<VM-USER> <LOCAL-HOST-FOLDER>

scp -P 2222 victor@127.0.0.1:/home/victor/.kube/config D:\victor\lections\
scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/profiles/minikube/client.crt D:\victor\lections\
scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/profiles/minikube/client.key D:\victor\lections\
scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/profiles/minikube/ca.crt D:\victor\lections\
scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/ca.crt D:\victor\lections\
```

Как вариант:
insecure-skip-tls-verify: true

```
PS C:\Users\Виктор> scp -P 2222 victor@127.0.0.1:/home/victor/.kube/config D:\victor\lections\
config                                                                      100%  827   252.0KB
PS C:\Users\Виктор> scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/profiles/minikube/client.crt D:\v

client.crt                                                                  100% 1147   331.8KB
PS C:\Users\Виктор> scp -P 2222 victor@127.0.0.1:/home/victor/.minikube/profiles/minikube/client.key D:\v

client.key                                                                  100% 1679   671.4KB
```

[**OPTIONAL**]: Исправляем ip адрес и пути до ключей для подключения к k8s внутри вм

D: > victor > lections > ! config

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: ca.crt
    extensions:
    - extension:
        last-update: Fri, 25 Oct 2024 05:08:52 UTC
        provider: minikube.sigs.k8s.io
        version: v1.34.0
      name: cluster_info
    server: https://127.0.0.1:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Fri, 25 Oct 2024 05:08:52 UTC
        provider: minikube.sigs.k8s.io
        version: v1.34.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: client.crt
    client-key: client.key
```

[**OPTIONAL**]: Пробрасываем порт для подключения к k8s внутри вм

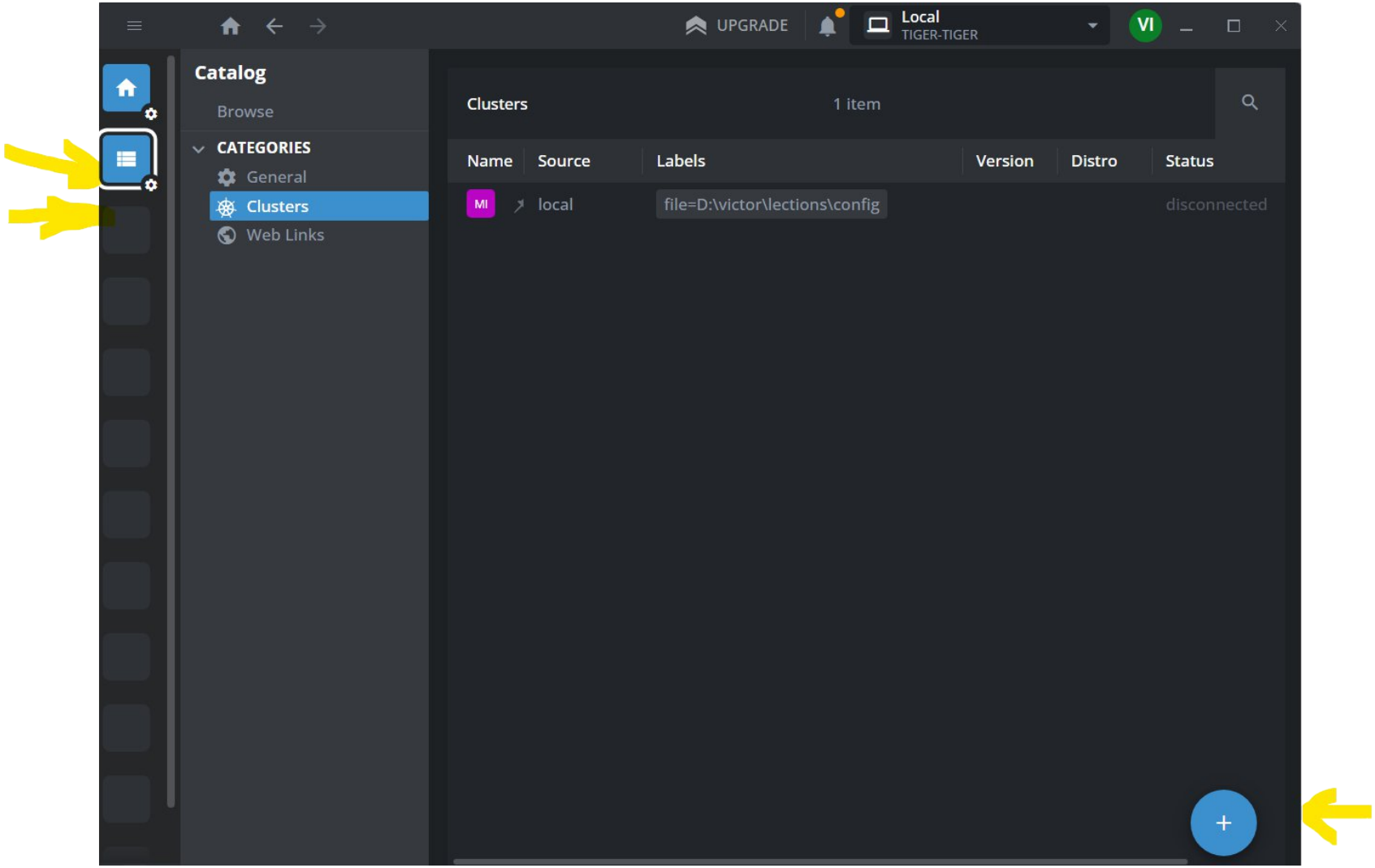
[**OPTIONAL**]: Пробрасываем порт внешнего адреса вм в сторону эмулятора k8s
Чтобы получить доступ с хостовой системы в кластер k8s

```
sudo iptables -t nat -A PREROUTING -d 10.0.2.15 -p tcp --dport 8443 -j DNAT --to 192.168.49.2:8443
```

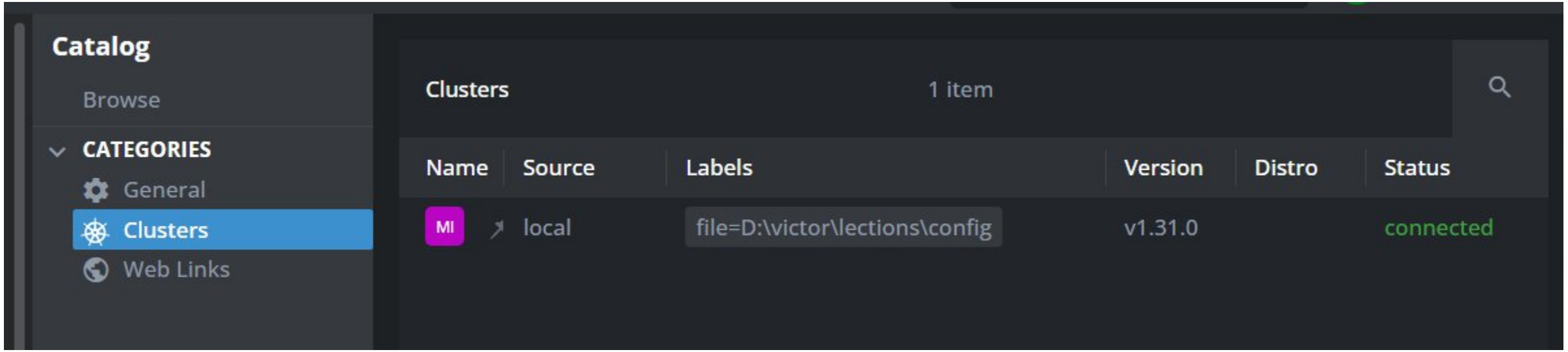
[**OPTIONAL**]: Скачать и установить Lens (by Mirantis)

https://k8slens.dev/

[**OPTIONAL**]: Добавить конфиг k8s в Lens

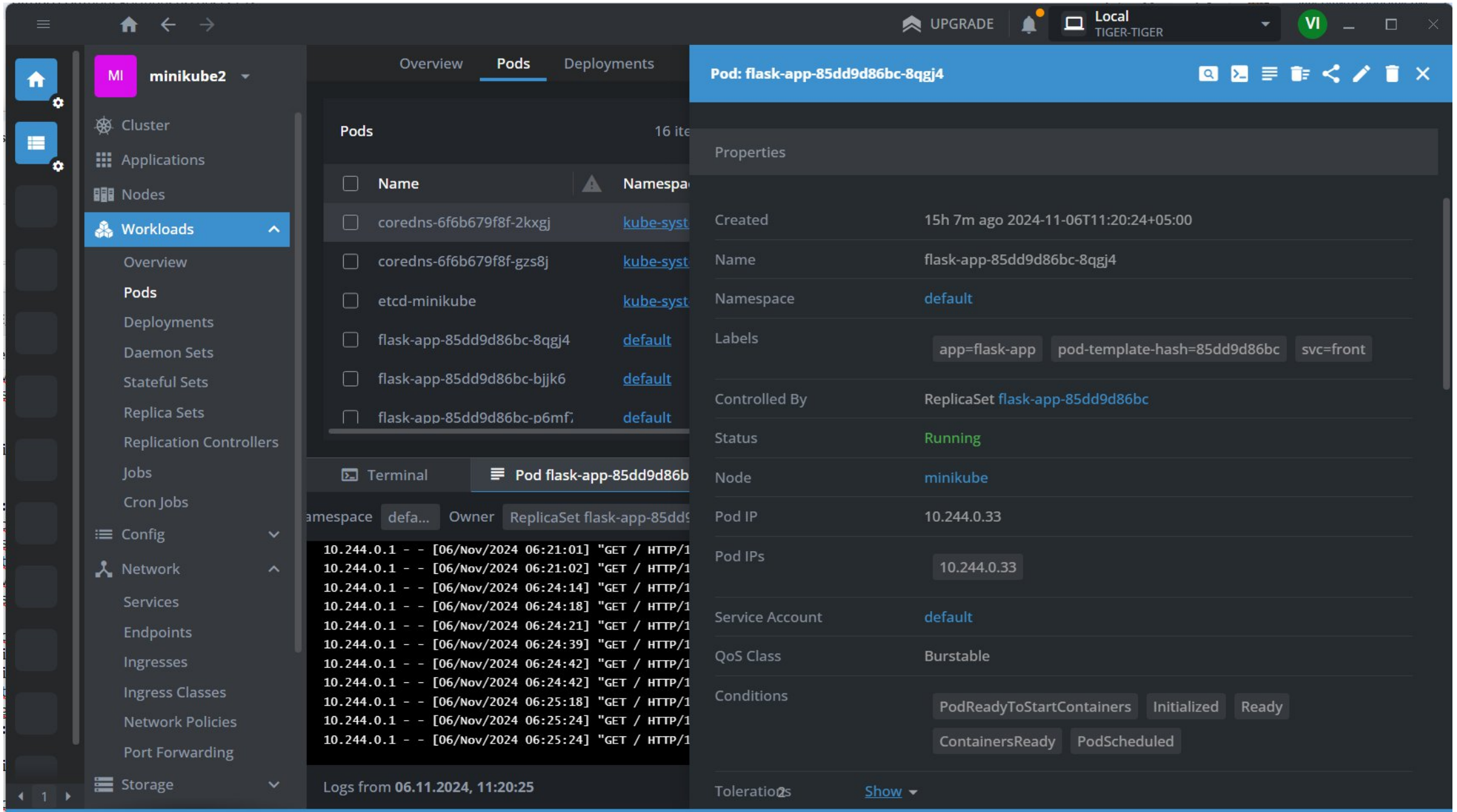


[**OPTIONAL**]: Если все порты проброшены и конфиги с сертификатами на месте, то Lens увидит кластер

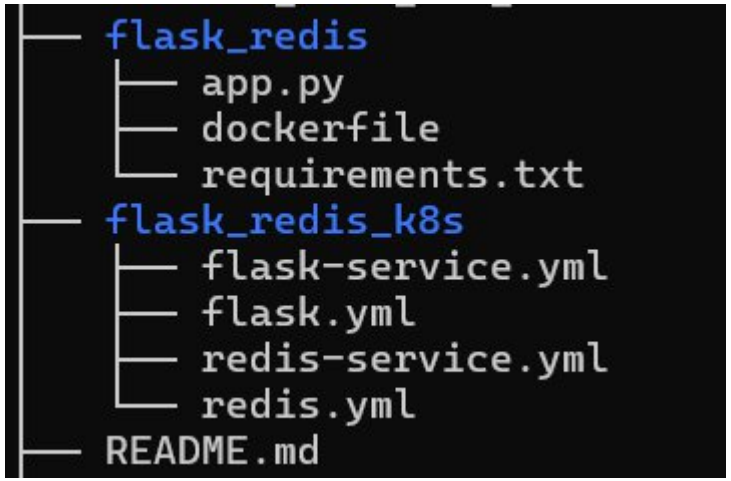


В Lens есть баг – если изменить конфиг, ленс не сможет больше подключаться – будет виснуть. Решается изменением всех названий кластера в конфиге (например поменять minikube на minikube2).

[**OPTIONAL**]: Пример подключения к поду



Итоговая структура каталогов будет выглядеть следующим образом:
Отдельный каталог для исходников приложения
Отдельный каталог для манифестов k8s



Модифицируем приложение Flask
Добавляем вывод имени реплики приложения на веб-страницу

```
  GNU nano 7.2                                                app.py
import time
import redis
import socket
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times. My name is: {}'.format(count, socket.gethostname())
```

Готовим образы контейнеров и делаем их доступными в кластере k8s

```
# Производим сборку образа из исходников прямо внутри minikube
minikube image build -t flask:v1 flask_redis/

# Загружаем готовый образ redis внутрь кластера
minikube image load redis:alpine

# Проверяем, что образы стали доступны внутри кластера
minikube image ls



# ИЛИ

# Устанавливаем переменные окружения нашего шелла, чтобы команды докера перенаправлялись в кластер
# eval $(minikube docker-env)
# сначала собираем образ, потом загружаем в миникуб
# docker build -t flask:v2 flask_redis/
# minikube image load flask:v2
```

Готовим описания (манифесты) для каждого сервиса - Flask

```
$ cat flask_redis_k8s/flask.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
  labels:
    app: flask-app
spec:
  replicas: 5
  selector:
    matchLabels:
      app: flask-app
      svc: front
  template:
    metadata:
      labels:
        app: flask-app
        svc: front
    spec:
      containers:
      - name: flask
        image: flask:v1
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 5000
        resources:
          limits:
            memory: "256Mi"
```

В отдельном каталоге

По этим меткам мы будем направлять сервис

Название образа - укажите корректный тег! (проверьте версию)

Брать образы из локального кеша

Сделать доступным этот порт контейнера

Готовим описания (манифесты) для каждого сервиса - Flask-SERVICE

```
$ cat flask_redis_k8s/flask-service.yml
```

В том же каталоге отдельным манифестом

```
apiVersion: v1
kind: Service
metadata:
  name: service-devops
  labels:
    app: flask-app
spec:
  type: LoadBalancer
  selector:
    app: flask-app
    svc: front
  ports:
  - port: 8000
    targetPort: 5000
  externalIPs:
  - 10.0.2.15
```

Селектором направляем трафик в сторону контейнера приложения

Внешний порт (может быть таки же как внутренний)

Внутренний порт контейнеров

Внешний ip кластера (если у вас не VirtualBox, то подставляйте свой реальный ip)

Готовим описания (манифесты) для каждого сервиса - Redis

```
$ cat flask_redis_k8s/redis.yml     # В том же каталоге отдельным манифестом

apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  labels:
    app: flask-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:                       # По этим меткам мы будем направлять сервис
        app: flask-app
        svc: db
    spec:
      containers:                   # Название контейнера будет зарегистрировано как DNS-имя внутри кластера
      - name: redis
        image: redis:alpine
        imagePullPolicy: IfNotPresent     # Брать образы из локального кеша
        ports:
        - containerPort: 6379             # Сделать доступным этот порт контейнера
```

Готовим описания (манифесты) для каждого сервиса - Redis-SERVICE

```
$  cat flask_redis_k8s/redis-service.yml
```

В том же каталоге отдельным манифестом

```
apiVersion: v1
kind: Service
metadata:
  name: redis
  labels:
    app: flask-app
spec:
  type: ClusterIP
  selector:
    app: flask-app
    svc: db
  ports:
  - port: 6379
    targetPort: 6379
```

Селектором направляем трафик в сторону контейнера приложения

Внешний порт (может быть таки же как внутренний)

Внутренний порт контейнеров

NOTE: Внешний ip кластера не задан, т.к. наша Redis не публикуется и доступна только внутри

Применяем манифесты k8s

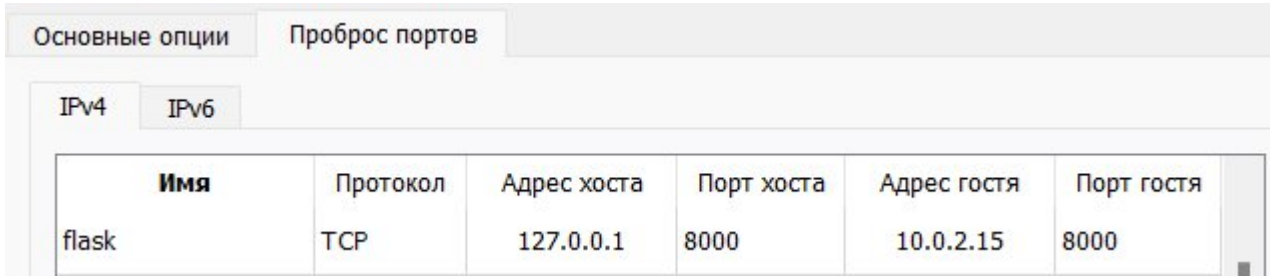
```
# применяем манифесты по отдельности файлами или весь каталог целиком
kubectl apply -f flask_redis_k8s/

# проверяем статус развертывания реплик
kubectl get pods

# проверяем сервисы
Kubectl get services
```

```
victor@ubuntu:~/devops11kube$ kubectl apply -f flask_redis_k8s/
service/service-devops created
deployment.apps/flask-app created
service/redis created
deployment.apps/redis created
victor@ubuntu:~/devops11kube$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
flask-app-55b977c4df-5smcr   1/1     Running   0          18s
flask-app-55b977c4df-967mj   1/1     Running   0          18s
flask-app-55b977c4df-btn5q   1/1     Running   0          18s
flask-app-55b977c4df-wcfmt   1/1     Running   0          18s
flask-app-55b977c4df-z2hqx   1/1     Running   0          18s
redis-577d9c666c-h99p5       1/1     Running   0          18s
victor@ubuntu:~/devops11kube$ kubectl get services
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP             PORT(S)          AGE
kubernetes       ClusterIP      10.96.0.1       <none>                  443/TCP          16d
redis            ClusterIP      10.100.90.4     <none>                  6379/TCP         21s
service-devops   LoadBalancer   10.110.98.229   127.0.0.1,10.0.2.15     8000:31015/TCP   21s
```

Проверка

У нас вложенная виртуализация – контейнеры внутри k8s внутри vm, поэтому настраиваем проброс портов на vm и на кластер внутри vm

```
# Пробрасываем с хостовой машины порт 8000 внутрь вм
```




```
# Внутри вм в отдельном окне разрешаем проброс трафика вм внутрь minikube
minikube tunnel --bind-address 10.0.2.15

# Оставляем эту ^ команду работать, не закрываем
# Проверяем в браузере http://127.0.0.1:8000 (обновите страницу несколько раз)
```

Hello World! I have been seen 136 times. My name is: flask-app-55b977c4df-c25vm

Hello World! I have been seen 93 times. My name is: flask-app-55b977c4df-wcfmt

Обновляем приложение на новую версию (Rolling Update)

– готовим новый образ

```
  GNU nano 7.2
import time
import redis
import socket
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World  VERSION 5! I have been se
```

Обновляем поды на новую версию (Rolling Update) – готовим новый образ

Пересобираем образ с новым тегом :v5
И делаем новый образ доступным в кластере minikube

```
victor@ubuntu:~/devops11kube$ minikube image build -t flask:v5 flask_redis/
#0 building with "default" instance using docker driver

#1 [internal] load build definition from dockerfile
#1 transferring dockerfile: 337B done
#1 DONE 0.0s

#2 resolve image config for docker-image://docker.io/docker/dockerfile:1
#2 DONE 1.2s
```

Обновляем поды на новую версию – патчим деплоймент

```
# Можно обновить файл манифеста и применить его заново,
# Можно пропатчить описание деплоймента прямо в БД k8s:

kubectl set image deployments/flask-app flask=flask:v5

# deployment.apps/flask-app image updated

Наблюдаем, как пересоздаются реплики подов на новую версию

kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
flask-app-bfb978db8-6qdjd    1/1     Running   0          9s
flask-app-bfb978db8-6vd5b    1/1     Running   0          7s
flask-app-bfb978db8-nxqxz    1/1     Running   0          8s
flask-app-bfb978db8-s4lhc    1/1     Running   0          9s
flask-app-bfb978db8-zxw9z    1/1     Running   0          9s
redis-577d9c666c-h99p5       1/1     Running   0          11m
kubectl describe deployment flask-app
```


```
victor@ubuntu:~/devops11kube$ kubectl rollout status deployment flask-app
deployment "flask-app" successfully rolled out
victor@ubuntu:~/devops11kube$
```

```
victor@ubuntu:~/devops11kube$ kubectl describe deployment flask-app
Name:                   flask-app
Namespace:              default
CreationTimestamp:      Thu, 21 Nov 2024 06:12:33 +0000
Labels:                 app=flask-app
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=flask-app,svc=front
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=flask-app
           svc=front
  Containers:
   flask:
    Image:      flask:v5
    Port:       5000/TCP
    Host Port:  0/TCP
    Limits:
      memory:      256Mi
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  flask-app-55b977c4df (0/0 replicas created)
NewReplicaSet:   flask-app-bfb978db8 (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  12m   deployment-controller  Scaled up replica set flask-app-55b977c4df to 5
  Normal  ScalingReplicaSet  43s   deployment-controller  Scaled up replica set flask-app-bfb978db8 to 2
  Normal  ScalingReplicaSet  43s   deployment-controller  Scaled down replica set flask-app-55b977c4df to 4 from 5
  Normal  ScalingReplicaSet  43s   deployment-controller  Scaled up replica set flask-app-bfb978db8 to 3 from 2
  Normal  ScalingReplicaSet  42s   deployment-controller  Scaled down replica set flask-app-55b977c4df to 3 from 4
  Normal  ScalingReplicaSet  42s   deployment-controller  Scaled up replica set flask-app-bfb978db8 to 4 from 3
```

<<< Происходит замена старых версий подов на новые

[**OPTIONAL**] Посмотрим как это всё выглядит через Lens

Hello World! I have been seen 127 times. My name is: flask-app-85dd9d86bc-8qgj4