# 11. Работа с Ingress (Локальное использование)

## 11.1. Введение в Ingress

Ingress предоставляет доступ к сервисам внутри кластера через HTTP и HTTPS, используя маршрутизацию на основе URL-адресов. В этой практике мы настроим Ingress-контроллер для локального тестирования без использования облачных сервисов.

## 11.2. Установка Ingress-контроллера

1. Устанавливаем Ingress-контроллер (например, Nginx):

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

2. Проверяем, что контроллер установлен:

```
kubectl get pods -n ingress-nginx
```

## 11.3. Создание приложения и сервиса

1. Создаем файл `deploy-ingress.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: web
        image: hashicorp/http-echo  # Использует контейнерный образ http-echo для демонстрации ответа с текстом
        args:
        - "-text=Hello, Ingress!"  # Текст ответа, который будет возвращаться при обращении к сервису
        ports:
        - containerPort: 5678  # Порт, на котором работает контейнер http-echo (по умолчанию)
---
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web
  ports:
  - protocol: TCP
    port: 80
```

```
    targetPort: 5678  # Перенаправляет трафик с порта 80 на порт контейнера
5678
  type: ClusterIP
  type: ClusterIP
```

2. Применяем манифест:

```
kubectl apply -f deploy-ingress.yaml
kubectl get svc
```

## 11.4. Настройка Ingress

*Пример 1: Простая маршрутизация*

1. Создаем файл `ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /  # Перезаписывает путь
запроса на корневой, чтобы сервис мог правильно обработать запрос
spec:
  rules:
  - host: local.ingress  # Определяем доменное имя, по которому будут
приниматься запросы
    http:
      paths:
      - path: /  # Обрабатываем все запросы, начинающиеся с /
        pathType: Prefix  # Тип маршрута, указывающий, что все пути с данным
префиксом будут маршрутизироваться на этот backend
        backend:
          service:
            name: web-service  # Название сервиса, к которому будут
перенаправляться запросы
            port:
              number: 80  # Порт, на котором работает сервис
```

2. Применяем манифест:

```
kubectl apply -f ingress.yaml
kubectl get ingress
```

*Пример 2: Маршрутизация по пути*

1. Создаем файл `ingress-path.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: path-ingress
spec:
  rules:
  - host: local.ingress
    http:
      paths:
      - path: /app1
        pathType: Prefix
        backend:
          service:
```

```
            name: web-service
            port:
              number: 80  # Перенаправляет запросы с путём /app1 к сервису
web-service на порту 80
      - path: /app2
        pathType: Prefix
        backend:
          service:
            name: web-service
            port:
              number: 80  # Перенаправляет запросы с путём /app2 к тому же
сервису, позволяя маршрутизировать разные пути на один сервис

# Маршрутизация по пути позволяет направлять запросы к разным сервисам или
разным экземплярам одного сервиса.
# В этом примере оба пути перенаправляют на один сервис, но в реальных
сценариях можно направлять запросы на разные сервисы или версии приложения.
      - path: /app2
        pathType: Prefix
        backend:
          service:
            name: web-service
            port:
              number: 80
```

2. Применяем манифест:

```
kubectl apply -f ingress-path.yaml
kubectl get ingress
```

3. Проверяем доступ:

```
curl http://local.ingress:8080/app1
curl http://local.ingress:8080/app2
```

## 11.5. Настройка hosts файла

1. Открываем файл `/etc/hosts` на локальном компьютере и добавляем строку:

```
127.0.0.1 local.ingress
```

## 11.6. HTTPS с самоподписанным сертификатом

1. Создаем самоподписанный сертификат:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
tls.crt -subj "/CN=local.ingress/O=local.ingress"
# Эта команда создает самоподписанный сертификат для домена local.ingress на
365 дней.
# Сертификат используется для шифрования HTTPS-трафика.
```

2. Создаем Secret для сертификата:

```
kubectl create secret tls web-tls --cert=tls.crt --key=tls.key
```

3. Создаем файл `ingress-https.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
metadata:
  name: web-ingress-https
spec:
  tls:
  - hosts:
    - local.ingress
    secretName: web-tls  # Указывает имя Secret, содержащего TLS-сертификат
для шифрования трафика
  rules:
  - host: local.ingress
    http:
      paths:
      - path: /
        pathType: Prefix  # Перенаправляет запросы с префиксом '/' на backend
        backend:
          service:
            name: web-service  # Название целевого сервиса
            port:
              number: 80  # Порт, используемый сервисом

# TLS используется для шифрования данных между клиентом и сервером,
обеспечивая безопасность передачи информации.
# Secret с сертификатами необходим, чтобы Ingress-контроллер мог установить
безопасное соединение.
```

4. Применяем манифест:

```
kubectl apply -f ingress-https.yaml
kubectl get ingress
```

5. Проверяем доступ через curl:

```
curl -k https://local.ingress:8080
```

## 11.7. Проверка работы

1. Перенаправляем порт с Nginx Ingress на локальную машину:

```
kubectl port-forward svc/ingress-nginx-controller -n ingress-nginx 8080:80
```

2. Проверяем работу через браузер или curl:

```
curl http://local.ingress:8080
curl http://local.ingress:8080/app1
curl http://local.ingress:8080/app2
```

Ожидается ответ: `Hello, Ingress!` или разные ответы в зависимости от пути.

## 11.8. Удаление ресурсов

1. Удаляем все созданные ресурсы:

```
kubectl delete -f deploy-ingress.yaml
kubectl delete -f ingress.yaml
kubectl delete -f ingress-path.yaml
kubectl delete -f ingress-https.yaml
kubectl delete secret web-tls
kubectl delete namespace ingress-nginx
```

---