Департамент образования города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа 3.2
по дисциплине «Интеграция и развертывание программного обеспечения с помощью контейнеров»

Тема: «Развертывание приложения в Kubernetes»

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-211
St_88

Москва
2025

ВВЕДЕНИЕ

Цель работы: освоить процесс развертывания приложения в Kubernetes с использованием Deployments и Services.

Задачи:

1. Создать Deployment для указанного приложения.
2. Создать Service для обеспечения доступа к приложению.
3. Проверить доступность приложения через созданный Service.
4. Выполнить индивидуальное задание.

Вариант 1.

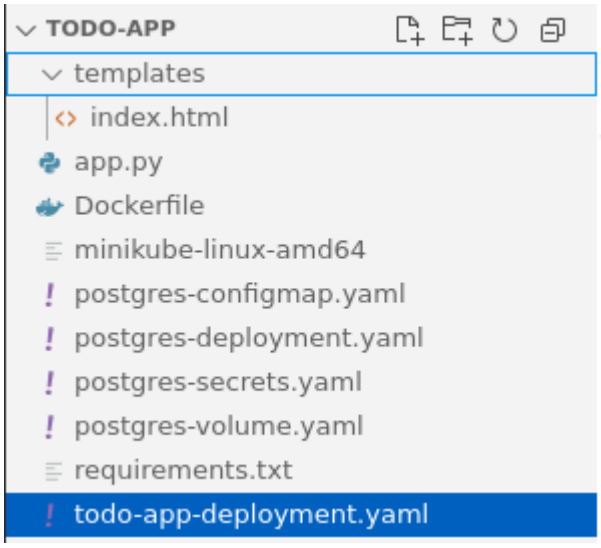
Разверните приложение на Flask, использующее базу данных PostgreSQL, в Kubernetes.

Создайте Deployment для Flask и PostgreSQL, а также Service для доступа к приложению.

ХОД РАБОТЫ

1. Создание приложения на Flask, которое будет представлять из себя веб-страницу с полем для ввода задач на день и кнопкой для их записи в базу данных PostgreSQL. Также записанные задачи на день будут отображаться в таблице на веб-странице.

Структура проекта полученного проекта (со всеми деплоиментами):



(Листинг файлов, относящихся именно к самому приложению, вынесен в Приложение)

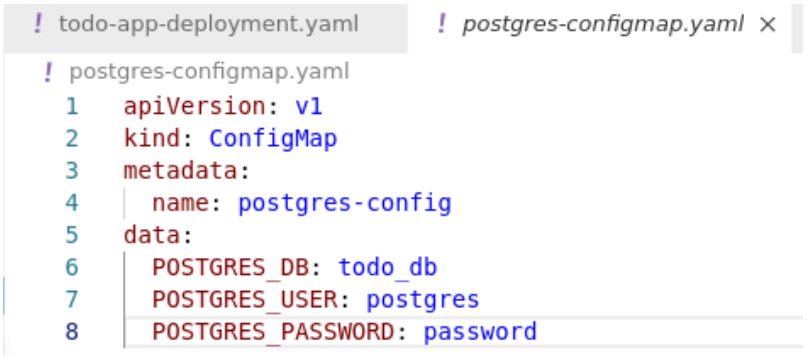
2. В файле *postgres-secrets.yaml* определяется Kubernetes Secret, который хранит конфиденциальные данные для доступа к PostgreSQL в закодированном виде:

todo-app-deployment.yaml | *postgres-secrets.yaml* ×

postgres-secrets.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secrets
type: Opaque
data:
  POSTGRES_USER: cG9zdGdyZXM=  # postgres
  POSTGRES_PASSWORD: cGFzc3dvcmQ=  # password
  POSTGRES_DB: dG9kb19kYg==  # todo_db
```

3. Файл *postgres-configmap.yaml* используется для настройки PostgreSQL:



4. Файл *postgres-volume.yaml* создает том для хранения данных PostgreSQL, чтобы сохранять данные в базу данных при перезапуске пода, гарантировать, что они не потеряются при сбоях:

PersistentVolume (PV) — это "физический" том в кластере Kubernetes, куда будут записываться данные;

PersistentVolumeClaim (PVC) — это "запрос" на выделение тома (PV) для пода.

5. Файл *postgres-deployment.yaml* создает диплоимент и сервис (внутренний) для PostgreSQL:

```
apiVersion: apps/v1 #версия
kind: Deployment
metadata:
  name: postgres #имя диплоимента
spec:
  replicas: 1 #создает 1 под с PostgreSQL
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres #лейбел для самого пода
    spec:
      containers:
        - name: postgres #имя контейнера
          image: postgres:13 #вресия используемого образа PostgreSQL
          ports:
            - containerPort: 5432 #порт для PostgreSQL (внутренний)
          envFrom:
            - configMapRef:
                name: postgres-config #указание, откуда берутся переменные
            - secretRef:
                name: postgres-secrets #указание, откуда берутся секретные переменные (логин и пароль от PostgreSQL)
          volumeMounts: #путь, куда монтируется том в контейнере
            - mountPath: /var/lib/postgresql/data
              name: postgres-data
      volumes: #подключение тома, созданного в postgres-volume.yaml
        - name: postgres-data
          persistentVolumeClaim:
            claimName: postgres-pvc
---
apiVersion: v1 #сервис для PostgreSQL
kind: Service
metadata:
  name: postgres
spec:
```

```
  selector:
    app: postgres
  ports:
    - protocol: TCP
      port: 5432
targetPort: 5432
```

6. Файл *todo-app-deployment.yaml* – диплоимент для созданного приложения на FLASK:

```
apiVersion: apps/v1 #версия
kind: Deployment
metadata:
  name: todo-app #имя диплоимента
spec:
  replicas: 1 #запускается 1 под
  selector:
    matchLabels:
      app: todo-app
  template:
    metadata:
      labels:
        app: todo-app #лейбел для пода
    spec:
      containers:
        - name: todo-app #имя контейнера
          image: todo-app:latest #версия локального образа
          imagePullPolicy: Never #запрет на скачивание образа из реестра, чтобы точно скачивался локальный (иначе падает в ошибку)
          ports:
            - containerPort: 5000 #порт приложения
          env: #переменные
            - name: POSTGRES_HOST
              value: "postgres"
            - name: POSTGRES_USER
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: POSTGRES_USER
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: POSTGRES_PASSWORD
```

```
            - name: POSTGRES_DB
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: POSTGRES_DB
---
apiVersion: v1 #прописан сервис для доступа к приложению
kind: Service
metadata:
  name: todo-app
spec:
  type: NodePort
  selector:
    app: todo-app
  ports:
    - protocol: TCP
      port: 5000 #внутренний порт сервиса
      targetPort: 5000 #порт контейнера
      nodePort: 30007 #фиксированный порт на ноде миникуба
```

7. Файла проекта сохранены и готовы. Запуск приложения:

7.1. Установка миникуба:

```
dev@dev-vm:~/todo-app$ curl -LO https://storage.googleapis.com/minikube/releases/latest/
minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0  13.5M      0  0:00:08  0:00:08 --:--:-- 25.2M
dev@dev-vm:~/todo-app$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

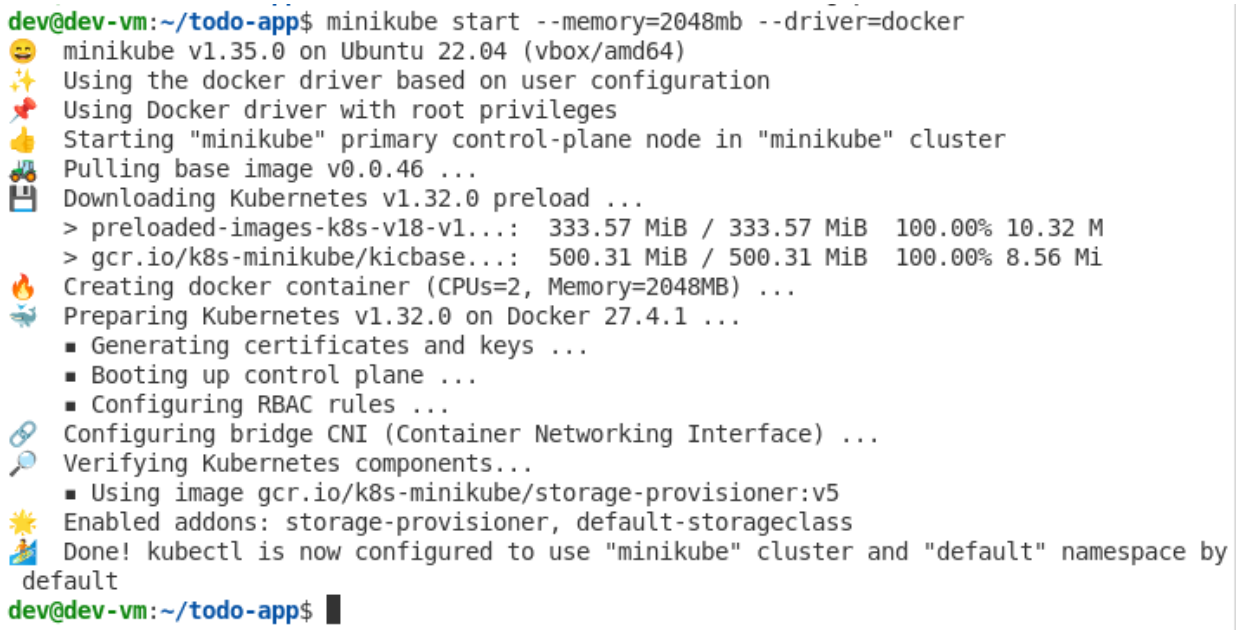
7.2. Добавление пользователя в группу docker:

```
dev@dev-vm:~/todo-app$ sudo usermod -aG docker $USER && newgrp docker
dev@dev-vm:~/todo-app$
```

7.3. Установка kubectl:

```
dev@dev-vm:~/todo-app$ sudo snap install kubectl --classic
[sudo] password for dev:
kubectl 1.32.3 from Canonical✓ installed
```

7.4. Запуск миникуба (с фиксировано выделенной памятью):



7.5. Используется для настройки окружения командной строки Ubuntu для работы с Docker, который управляется Minikube:

```
eval $(minikube docker-env)
```

7.6. Билд локального образа приложения:

```
dev@dev-vm:~/todo-app$ docker build -t todo-app:latest .
2025/04/10 22:43:23 in: []string{}
2025/04/10 22:43:23 Parsed entitlements: []
[+] Building 77.1s (10/10) FINISHED                                    docker:default
 => [internal] load build definition from Dockerfile                            0.4s
 => => transferring dockerfile: 185B                                            0.2s
 => [internal] load metadata for docker.io/library/python:3.9-slim              5.1s
 => [internal] load .dockerignore                                               0.1s
 => => transferring context: 2B                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:9aa5793609640ecea2f06451  34.9s
 => => resolve docker.io/library/python:3.9-slim@sha256:9aa5793609640ecea2f06451a  0.2s
 => => sha256:501f96d59d707efd12df137b6d001069e83068678ef241ca5bd 5.29kB / 5.29kB  0.0s
 => => sha256:9aa5793609640ecea2f06451a0d6f379330880b413f954933 10.41kB / 10.41kB  0.0s
 => => sha256:d6e2ddca278e1b84c1a934ca8f07bbbfe6b6ec936a0b68107c8 1.75kB / 1.75kB  0.0s
 => => sha256:8a628cdd7ccc83e90e5a95888fcb0ec24b991141176c515ad 28.23MB / 28.23MB  7.7s
 => => sha256:74018f7cfa8f2965fd86b13c38f71417bc846e071a5f5bb5ae5 3.51MB / 3.51MB  3.7s
 => => sha256:a0b0cfc480ce03c723a597904bcfbf28c71438c689e6d5097 14.93MB / 14.93MB  6.0s
 => => sha256:97d21b95fb00ac3b08975ab6f8709f3a7e35a05d75e2f9a70fa9534 250B / 250B  4.1s
 => => extracting sha256:8a628cdd7ccc83e90e5a95888fcb0ec24b991141176c515ad101f12  14.5s
 => => extracting sha256:74018f7cfa8f2965fd86b13c38f71417bc846e071a5f5bb5ae569ccb  1.5s
 => => extracting sha256:a0b0cfc480ce03c723a597904bcfbf28c71438c689e6d5097c233283  5.8s
 => => extracting sha256:97d21b95fb00ac3b08975ab6f8709f3a7e35a05d75e2f9a70fa95348  0.0s
 => [internal] load build context                                              16.2s
 => => transferring context: 125.30MB                                          16.0s
 => [2/5] WORKDIR /app                                                          0.9s
 => [3/5] COPY requirements.txt .                                               0.4s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                   27.6s
 => [5/5] COPY . .                                                              3.0s
 => exporting to image                                                          3.9s
 => => exporting layers                                                         3.8s
 => => writing image sha256:6a739e7721aa1a732249f1930435980af287cbe5301561964765f  0.0s
 => => naming to docker.io/library/todo-app:latest                              0.0s
dev@dev-vm:~/todo-app$
```

7.7. Создание и применение всех прописанных конфигураций:

```
dev@dev-vm:~/todo-app$ kubectl create -f postgres-secrets.yaml
secret/postgres-secrets created
dev@dev-vm:~/todo-app$ kubectl create -f postgres-configmap.yaml
configmap/postgres-config created
dev@dev-vm:~/todo-app$ kubectl create -f postgres-volume.yaml
persistentvolume/postgres-pv created
persistentvolumeclaim/postgres-pvc created
dev@dev-vm:~/todo-app$ kubectl create -f postgres-deployment.yaml
deployment.apps/postgres created
service/postgres created
dev@dev-vm:~/todo-app$ kubectl create -f todo-app-deployment.yaml
deployment.apps/todo-app created
service/todo-app created
```

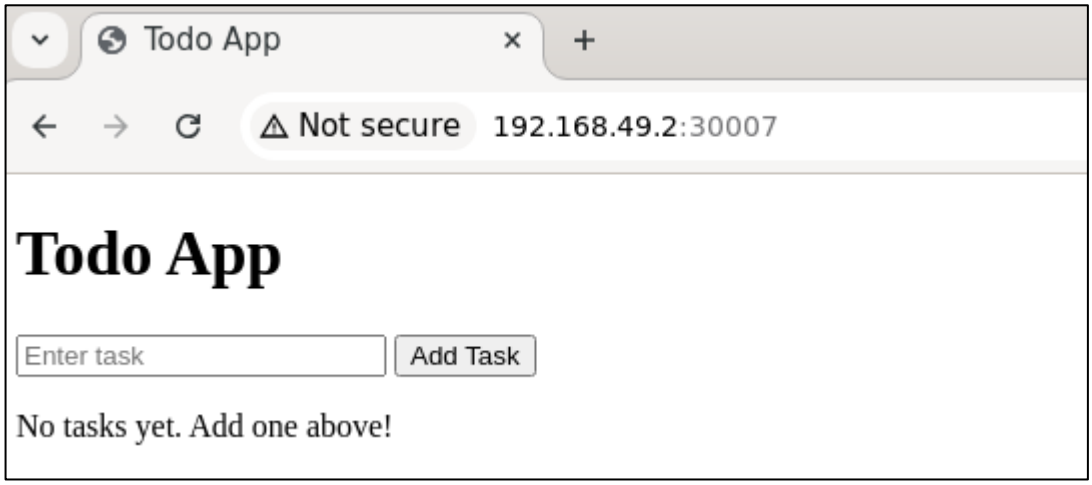
7.8. Проверка, что оба нода (базы данных и приложения) успешно работают:

```
dev@dev-vm:~/todo-app$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
postgres-5864b7c477-2hmfx   1/1     Running   0          17m
todo-app-7565995757-xxt27   1/1     Running   0          14s
dev@dev-vm:~/todo-app$
```

7.9. Вывод адреса, на котором доступен сервис приложения:

```
dev@dev-vm:~/todo-app$ minikube service todo-app --url
http://192.168.49.2:30007
```

8. Проверка доступности сервиса приложения:



Как видно, сервис доступен и свою функцию выполняет – записывает задачи на день в таблицу:

9. Когда все успешно отработало, удаляем миникуб:

```
dev@dev-vm:~/todo-app$ minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Removing /home/dev/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
dev@dev-vm:~/todo-app$
```

ВЫВОДЫ

В ходе выполнения лабораторной работы были реализованы все поставленные задачи, а именно:

- Создан диплоимент для приложения на Flask, использующее базу данных PostgreSQL, в Kubernetes;
- Создан, запущен и успешно протестирован сервис созданного приложения для записи задач на день.

Таким образом, была достигнута главная цель работы - освоить процесс развертывания приложения в Kubernetes с использованием Deployments и Services.

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Pod, Deployment и Service в Kubernetes?

Pod – наименьшая и самая простая единица в Kubernetes. Pod представляет собой один или несколько контейнеров, которые:

- разделяют общие ресурсы (сеть, хранилище).
- запускаются на одной ноде (виртуальной/физической машине).
- имеют общий ip-адрес и пространство имен.

Deployment - объект для управления жизненным циклом Pod’ов. Позволяет:

- развертывать и обновлять приложения.
- масштабировать количество реплик pod’ов.
- откатываться к предыдущим версиям.

Deploymen работает по принципу:

1. Создает Pod’ы по шаблону (template).
2. Следит за их состоянием (перезапускает при падении).
3. Позволяет обновлять приложение без downtime (стратегия rolling update).

Service - абстракция, которая обеспечивает постоянный доступ к Pod’ам, даже если те пересоздаются (и меняют IP). Решает две задачи: обнаружение сервисов — дает Pod’ам стабильное DNS-имя; балансировка нагрузки — распределяет трафик между Pod’ами.

Типы Service:

- ClusterIP (по умолчанию) — доступ только внутри кластера.
- NodePort — открывает порт на ноде для доступа снаружи (например, http://<IP-ноды>:30007).
- LoadBalancer — создает внешний балансировщик нагрузки (в облачных провайдерах).

2. Каково назначение Deployment в Kubernetes?

Deployment в Kubernetes — это контроллер, который управляет жизненным циклом приложений, обеспечивая их стабильность, масштабируемость и бесперебойную работу.

Развертывание приложений	Создает и запускает Pod’ы по заданному шаблону (`template`).
Обновление без downtime	Постепенно заменяет старые Pod’ы новыми (rolling update).
Откат при ошибке	Возвращает предыдущую версию, если новое развертывание не работает.
Масштабирование	Увеличивает или уменьшает количество реплик Pod’ов.
Самовосстановление	Автоматически перезапускает упавшие Pod’ы.

3. Каково назначение Service в Kubernetes?

Service в Kubernetes — это абстракция, которая обеспечивает стабильный доступ к приложению, работающему в виде набора подов.

Единая точка доступа	Pod'ы имеют временные IP-адреса, которые меняются при перезапуске. Service предоставляет постоянный DNS-адрес.
Балансировка нагрузки	Распределяет трафик между всеми Pod'ами, соответствующими селектору.
Обнаружение сервисов	Позволяет Pod'ам находить друг друга по имени, без hardcoded IP-адресов.
Публикация наружу	Обеспечивает доступ к приложению извне кластера (через NodePort/LoadBalancer).

4. Как создать Deployment в Kubernetes?

Создать YAML-файл (манифест):

```
apiVersion: apps/v1
kind: Deployment
metadata:
   name: my-app
spec:
     replicas: 3
     selector:
          matchLabels:
          app: my-app
     template:
          metadata:
               labels: app:
                    my-app
               spec:
                    containers:
                    - name: my-app
                    image: my-app:latest
                    ports:
                    - containerPort: 8080
```

Применить деплой с помощью команды:

kubectl apply -f deployment.yaml

5. Как создать Service в Kubernetes и какие типы Services существуют?

Создать YAML-файл:

```
apiVersion: v1
kind: Service
metadata:
  name: todo-app
spec:
  type: NodePort
  selector:
    app: todo-app
  ports:
    - protocol: TCP
      port: 5000 #внутренний порт сервиса
      targetPort: 5000 #порт контейнера
               nodePort: 30007 #фиксированный
```

Применить сервис с помощью команды :

kubectl apply -f service.yaml

Типы Services:

ClusterIP (по умолчанию): Внутренний IP для доступа внутри кластера;

NodePort: открывает порт на каждом узле кластера;

LoadBalancer: создает внешний балансировщик нагрузки (в облачных провайдерах);

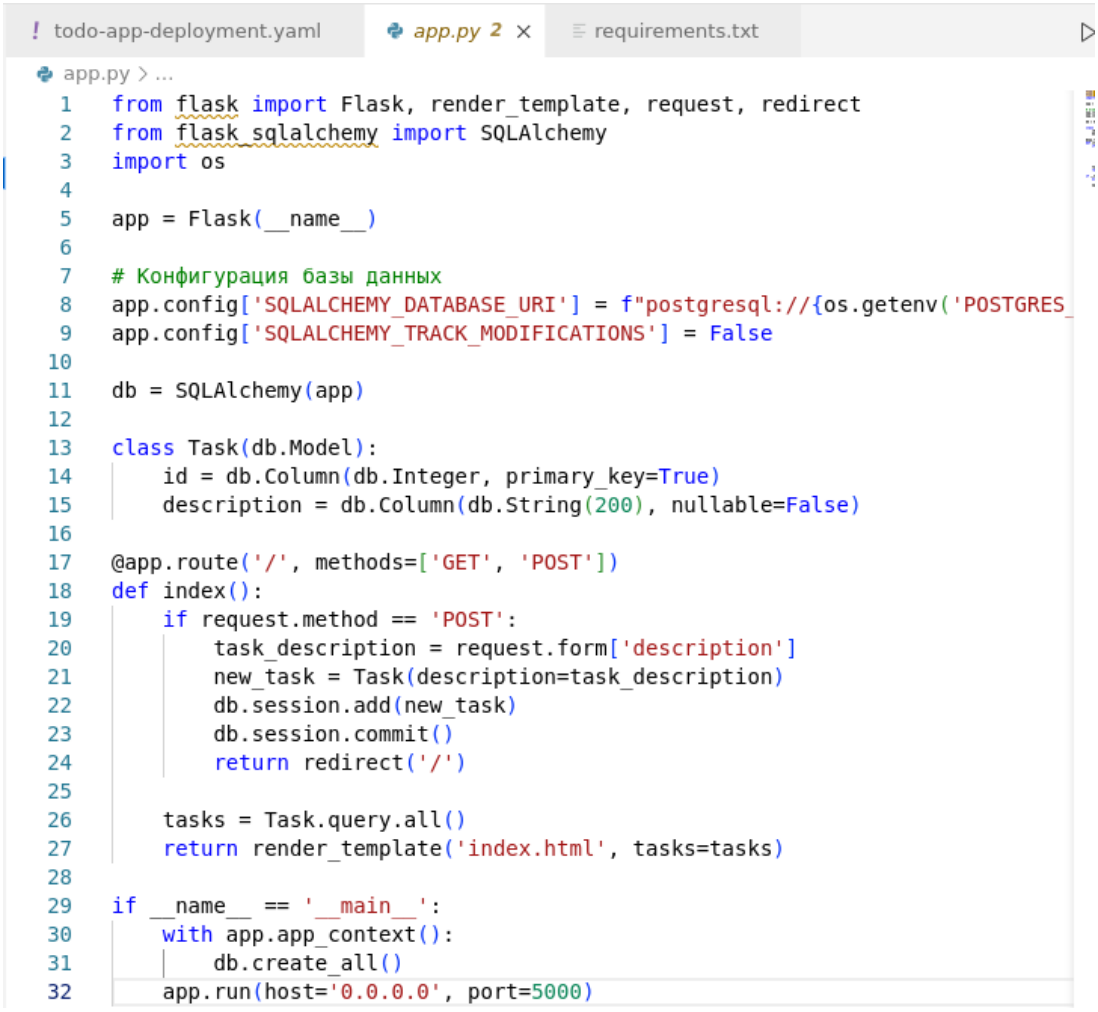
ExternalName: сопоставляет Service с внешним DNS-именем.

ПРИЛОЖЕНИЯ

1. Структура проекта приложения:

```
todo-app/
├── app.py
├── requirements.txt
├── templates/
│   └── index.html
└── Dockerfile
```

2. Файл app.py:



3. Файл requirements.txt:

requirements.txt

```
flask==2.0.3
werkzeug==2.0.3
flask-sqlalchemy==2.5.1
sqlalchemy==1.4.46
psycopg2-binary==2.9.3
python-dotenv==0.19.0
```

4. Файл index.html:

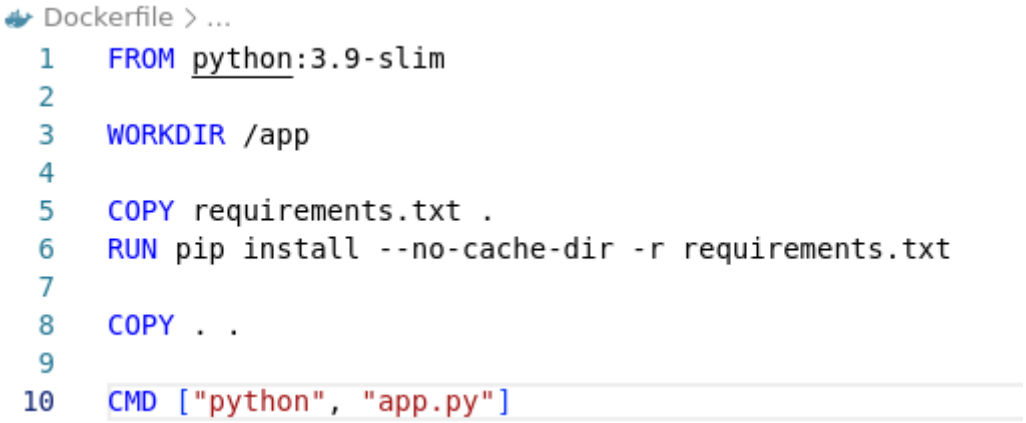
templates > index.html > html > body > form > input

```
<!DOCTYPE html>
<html>
<head>
    <title>Todo App</title>
    <style>
        table {
            width: 50%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
        form {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>Todo App</h1>

    <form method="POST">
        <input type="text" name="description" placeholder="Enter task" required>
        <button type="submit">Add Task</button>
    </form>

    {% if tasks %}
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Description</th>
            </tr>
        </thead>
        <tbody>
            {% for task in tasks %}
            <tr>
                <td>{{ task.id }}</td>
                <td>{{ task.description }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    {% else %}
    <p>No tasks yet. Add one above!</p>
    {% endif %}
</body>
</html>
```

5. Dockerfile:



6. Команды для развертывания приложения:

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64

sudo install minikube-linux-amd64 /usr/local/bin/minikube

sudo usermod -aG docker $USER && newgrp docker

sudo snap install kubectl --classic

minikube start --memory=2048mb --driver=docker

eval $(minikube docker-env)

docker build -t todo-app:latest .


kubectl create -f postgres-secrets.yaml

kubectl create -f postgres-configmap.yaml

kubectl create -f postgres-volume.yaml

kubectl create -f postgres-deployment.yaml

kubectl create -f todo-app-deployment.yaml


kubectl get pods

minikube service todo-app –url
```