Модуль «Консьерж»

Документация, содержащая информацию, необходимую для установки и эксплуатации программного обеспечения

Установка модуля “Консьерж” 4
Установка сервиса RoBERTa 4
Предварительные требования 4
Шаг 1. Настройка доступа к Docker Registry 4
Шаг 2. Создайте манифесты Kubernetes 4
Шаг 3. Порядок развертывания 5
Шаг 4. Проверка развертывания 5
Установка сервиса автоматической векторизации знаний 6
Предварительные требования 6
Порядок развертывания 6
Шаг 1. Подготовка секретов для сервиса 6
Шаг 2. Применение манифестов 7
Шаг 3. Проверка развертывания 8
Шаг 4. Проверка работоспособности 8
Критические моменты 8
Шаг 5. Настройка авторизационных данных для доступа к базам знаний по каждому проекту 9
Установка сервиса авторизационных cookie (опционально) 11
Предварительные требования 11
Этапы установки 11
Шаг 1. Создание секретов для сервиса 11
Шаг 2. Установка сервиса через Helm 11
Шаг 3. Проверка установки 12

Установка модуля «Консьерж»

Установка сервиса RoBERTa

Предварительные требования

1. Доступ к кластеру Kubernetes с правами в нужном namespace
2. Доступ к приватному registry
3. Требования к инфраструктуре: деплой сервиса RoBERTa требует наличия GPU в кластере

Шаг 1. Настройка доступа к Docker Registry

Создайте секрет для авторизации в приватном реестре:

```
kubectl apply -f secretchabopl.yaml
```

- *Проверьте доступ к приватному Docker Registry. Если доступ отсутствует, обратитесь к вашему администратору.*

Шаг 2. Создайте манифесты Kubernetes

Deployment манифест **должен** содержать:

- Указание GPU ресурсов:
```
resources:
limits:
  nvidia.com/gpu: 1
```

- Подключение Secret приватного регистри:
```
imagePullSecrets:
  - name: chabopl-secret
```

Переменные окружения, указываемые в секрете:

- MODEL_PATH - путь к моделям (используйте PersistentVolume)
- REDIS_CLIENT_URL - из Secret
- S3_ENDPOINT_URL - из Secret
- AWS_ACCESS_KEY_ID - из Secret
- AWS_SECRET_ACCESS_KEY - из Secret

Volume для моделей: Используйте PersistentVolumeClaim для хранения моделей

Service для доступа внутри кластера

Ingress для внешнего доступа с настройкой timeout (300 секунд)

Шаг 3. Порядок развертывания

Подготовка Secrets -> Создание PersistentVolumeClaim для хранения моделей (50+ GB) -> Применение манифестов в следующем порядке:

- [ ] Namespace (если нужен)
- [ ] Secrets
- [ ] PersistentVolumeClaim
- [ ] Deployment
- [ ] Service
- [ ] Ingress

Шаг 4. Проверка развертывания

Команды для проверки:

Проверка состояния подов

```
kubectl get pods -o wide

# Проверка логов
kubectl logs -f deployment/имя-деплоймента

# Проверка GPU
kubectl exec deployment/имя-деплоймента -- nvidia-smi

# Проверка доступности сервиса
curl http://ваш-домен/health
```

---

1. GPU квоты: Убедитесь, что в Namespace есть квоты на GPU
2. Сеть: Проверьте настройки NetworkPolicy если они используются
3. Хранилище: Используйте правильный StorageClass для PersistentVolume

---

Установка сервиса автоматической векторизации знаний

Предварительные требования

1. Доступ к кластеру Kubernetes с правами в нужном namespace
2. Доступ к приватному registry

- *Проверьте доступ к приватному Docker Registry. Если доступ отсутствует, обратитесь к вашему администратору.*

Порядок развертывания

Шаг 1. Подготовка секретов для сервиса

Создайте или обновите Secret worker-shtrih с корректными значениями:

```
kubectl get secret worker-shtrih -n ваш-namespace
```

```
# Обновите или создайте секрет
kubectl create secret generic worker-shtrih \
  --from-literal=DB_URL="postgresql://..." \
  --from-literal=MODEL_URL="https://..." \
  --from-literal=URL_BERT="https://..." \
  --from-literal=URL_SERVICE="http://worker-shtrih:8080" \
  --from-literal=FERNET_KEY="..." \
  --from-literal=SENDER="ваш-email.ru" \
  --from-literal=SENDER_PASSWORD="пароль" \
  --from-literal=DOMAINMAIL="ваш-домен.ru" \
  -n ваш-namespace
```

Обязательные поля для заполнения:

- DB_URL - строка подключения к PostgreSQL
- MODEL_URL - URL модели для загрузки
- URL_BERT - URL BERT-сервиса
- FERNET_KEY - ключ шифрования (32 байта в base64)
- SENDER - email отправителя
- SENDER_PASSWORD - пароль от email
- DOMAINMAIL - домен почтового сервера

Шаг 2. Применение манифестов

Примените манифесты в нужном namespace:

```
kubectl apply -f service.yaml -n ваш-namespace

kubectl apply -f deployment.yaml -n ваш-namespace

kubectl apply -f secret.yaml -n ваш-namespace

kubectl apply -f cronjob.yaml -n ваш-namespace
```

Шаг 3. Проверка развертывания

Проверка подов
```
kubectl get pods -n ваш-namespace -l app=worker-shtrih
```

Проверка логов
```
kubectl logs -f deployment/worker-shtrih -n ваш-namespace
```

Проверка сервиса
```
kubectl get svc worker-shtrih -n ваш-namespace
```

Проверка CronJob
```
kubectl get cronjob vectorization-status -n ваш-namespace
```

Шаг 4. Проверка работоспособности

Проверка health-check
```
kubectl exec deployment/worker-shtrih -n ваш-namespace -- curl
-s http://localhost:8080/health_check
```

Проверка запуска CronJob вручную
```
kubectl create job --from=cronjob/vectorization-status
vectorization-test -n ваш-namespace
```

Критические моменты

1. Все ресурсы должны быть в одном *namespace*
2. Email настройки: Почтовый сервер должен разрешать отправку с указанных креденшиалов

Шаг 5. Настройка авторизационных данных для доступа к базам знаний по каждому проекту

Автоматическая векторизация запускается каждый час в :30 минут (например: 10:30, 11:30 и т.д.).

Важно: *Перед запуском векторизации необходимо добавить проекты из базы знаний!*

Порядок действий после установки

Шаг 1. Проверка доступности сервиса

Создать порт-форвард для удобной работы с API в браузере:

```
kubectl port-forward svc/worker-shtrih 8080 -n monitoring
```

```
curl http://localhost:8080/health_check
```

Ожидаемый ответ: OK

Шаг 2. Добавление проектов из базы знаний

Все проекты из базы знаний, которые будут использоваться для формирования ответов, необходимо предоставить сервису (до запуска векторизации в 30 минут каждого часа, не во время процесса векторизации)

Параметры запроса:

project_name — название проекта.
project_id — соответствующий идентификатор проекта (project_id) в базе знаний.
channel — канал ответа в базе знаний, который будет использоваться ботом.
url — URL API используемой Базы знаний.

login — логин для доступа к базе знаний.
password — пароль для доступа к базе знаний; хранится в зашифрованном виде, не отображается пользователям и используется только сервисом.

Для каждого проекта выполните:

```
curl -X POST http://localhost:8080/add_project \
  -H "Content-Type: application/json" \
  -d '{
    "project_name": "консьерж мосру",
    "project_id": 5,
    "channel": "Виртуальный_помощник_(бот)",
    "url": "https://ваш-api-базы-знаний.ru/api",
    "login": "consierge_mos_api@optimalcity.ru",
    "password": "актуальный_пароль"
  }'
```

Шаг 3. Проверка добавленных проектов

```
curl http://localhost:8080/projects
```

Шаг 4. Если добавить проекты вовремя не получается, следует временно остановить CronJob:

```
kubectl scale cronjob vectorization-status --replicas=0 -n ваш-namespace
```

Добавьте проекты, затем запустите снова:

```
kubectl scale cronjob vectorization-status --replicas=1 -n ваш-namespace
```

Принудительный запуск векторизации

```
kubectl create job manual-vectorization --from=cronjob/vectorization-status -n ваш-namespace
```

Установка сервиса авторизационных cookie (опционально)

Если LLM, используемая “Консьержем”, требует авторизации по данным из файла cookie, то рекомендуется установить данный сервис для проксирования запросов к LLM с актуальными авторизационными данными.

Предварительные требования

1. Доступ к кластеру Kubernetes с правами в нужном namespace
2. Доступ к приватному registry

- *Проверьте доступ к приватному Docker Registry. Если доступ отсутствует, обратитесь к вашему администратору.*

Этапы установки

Шаг 1. Создание секретов для сервиса

Заполните актуальные переменные окружения, необходимые для работы сервиса, и создайте секрет с переменными окружения для package-cookie:

```
kubectl apply -f secret.yaml -n ваш-namespace
```

Шаг 2. Установка сервиса через Helm

Используйте файл values.yaml, при необходимости изменив те или иные параметры, и установите сервис:

```
helm upgrade --install package-cookie ./ваш-chart \
  -f values.yaml \
  -n chat-platform
```

Шаг 3. Проверка установки

Проверьте состояние пода:

```
kubectl get pods -n chat-platform -l
"app.kubernetes.io/name=package-cookie"
```

Проверьте логи:

```
kubectl logs -f deployment/package-cookie -n chat-platform
```

Проверьте health-check:

```
kubectl exec deployment/package-cookie -n chat-platform -- curl
-s http://localhost/health
```