# Project Generation Orders

## Генератор заказов 0.3.5

### 1. Описание решения

Приложение "Генератор заказов" реализует функционал по заказу товаров и услуг, например, для интернет-магазина. В нём реализована модульная система с возможностью репликации (Master-Replica) и партиционирования данных по городам в базе данных PostgreSQL.

Исходные файлы приложения доступны в общем репозитории на GitHub:

[GitHub Repository](https://github.com/nickfrolov2/orders-generator)

Локальный репозиторий для клонирования (необходима авторизация):

- SSH: git@gitlaboff.devopsf.ru:mygroup1/test\_project\_frolovna.git

- HTTPS: https://gitlaboff.devopsf.ru/mygroup1/test\_project\_frolovna.git

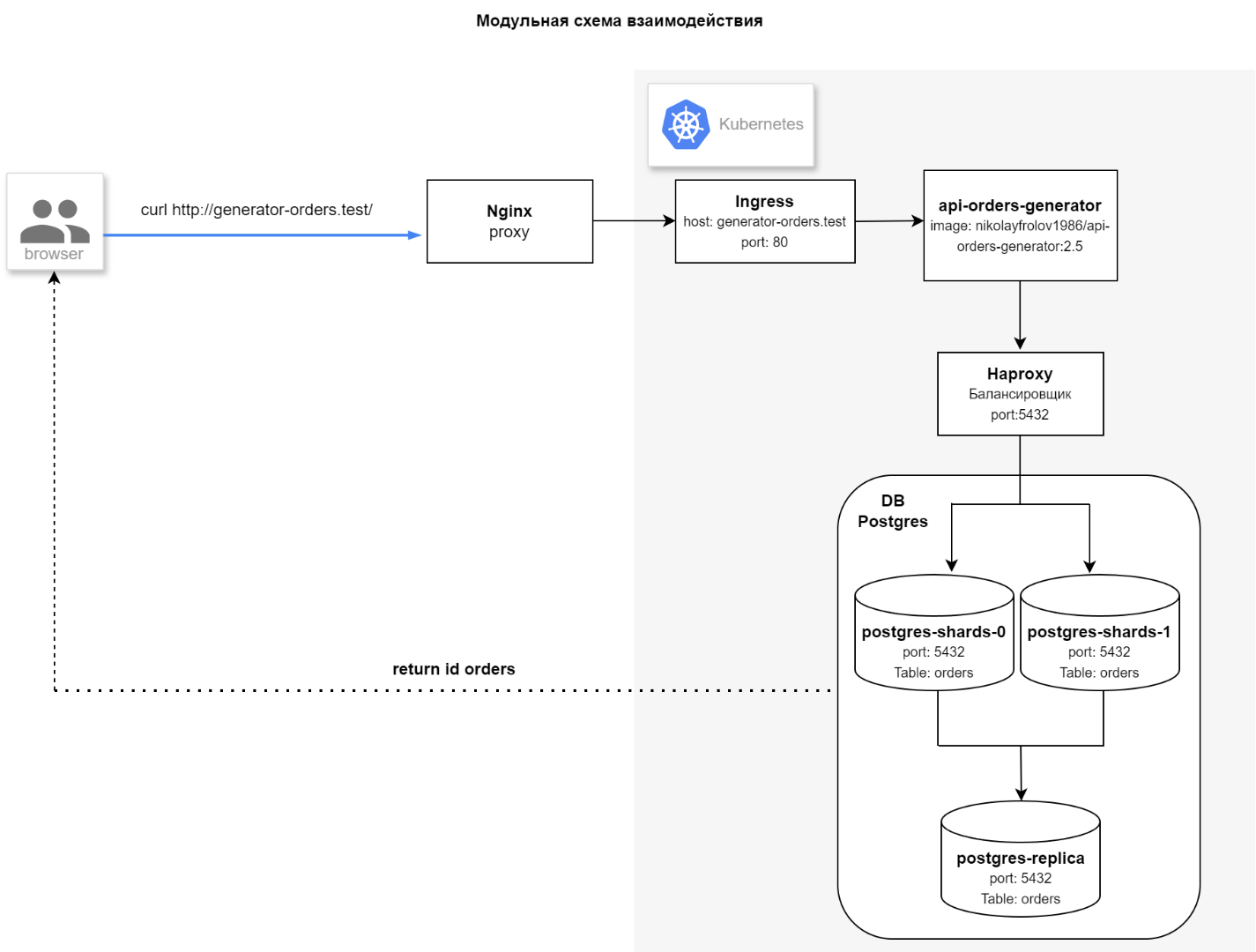
Подключение репозитория через Helm:

helm repo add orders\_generator https://raw.githubusercontent.com/nickfrolov2/orders-generator/refs/heads/main/

Запуск/установка производится с помощью команды:

helm install orders orders\_generator/project\_orders

### 2. Общая схема взаимодействия модулей



### 3. Описание компонентов и их взаимодействие

Исходные файлы проекта находятся в папке: \charts\templates

- api-deployment.yml

  Приложение API api-orders-generator разворачивается из образа Docker-контейнера nikolayfrolov1986/api-orders-generator:2.1, скачиваемого из публичного Docker Hub. Исходный файл с `Dockerfile` находится в корне проекта в папке GO.

- api-service.yml

  Сервис для API.

- ingress.yml

  Ingress, пробрасывает host и порт до `api-service`.

- haproxy-deployment.yml, haproxy-service.yml, haproxy-configmap.yml

  `HAProxy` выполняет роль балансировщика между двумя базами данных PostgreSQL: `postgres-shards-0` и `postgres-shards-1`.

- postgres-statefulset.yml, postgres-shards-service.yml, postgres-job1.yml, postgres-job2.yml, orders-replica-shards1-configmap.yml, orders-replica-shards2-configmap.yml

  В `StatefulSet` создаются два пода с контейнером PostgreSQL: `postgres-shards-0` и `postgres-shards-1`. Для каждого пода монтируется `postgres-storage` (PVC).

  - Job1 подключается к `postgres-shards-0`, выполняет команду на создание таблицы `orders`, делит её на партиции по городам, создавая отдельную таблицу для каждого города, устанавливает уровень logical и создаёт публикацию для таблицы `orders`.

  - Job2 работает аналогично для `postgres-shards-1`.

- postgres-replica-statefulset.yml, orders-replica-configmap.yml, postgres-job3.yml

  Создаётся база PostgreSQL `postgres-replica-0` для репликации данных из двух баз: `postgres-shards-0` и `postgres-shards-1`.

  - Job3 создаёт таблицу `orders`, разделяет её на партиции по городам, создавая отдельную таблицу для каждого города, устанавливает уровень replica и создаёт подписку на таблицу `orders` баз `postgres-shards-0` и `postgres-shards-1`.

После выполнения Job1, Job2, Job3 эти задания удаляются.

### 4. Разворачивание среды для приложения

Для работы приложения понадобятся инструменты:

- Helm

- Minikube (или аналогичные: Docker, Rancher, Kubeadm и т.д.)

Убедитесь, что addons ingress включен.

### 5. Запуск приложения

Подключение репозитория через Helm:

helm repo add orders https://raw.githubusercontent.com/nickfrolov2/orders-generator/refs/heads/main/

Запуск/установка производится с помощью команды:

helm install/upgrade orders orders\_generator/project\_orders

Вы должны увидеть успешную установку:

PS C:\Users\1> helm upgrade orders orders\_generator/project\_orders

Release "orders" has been upgraded. Happy Helming!

NAME: orders

LAST DEPLOYED: Fri Feb 28 21:33:29 2025

NAMESPACE: default

STATUS: deployed

REVISION: 2

TEST SUITE: None

NOTES:

###Project Orders by Frolov N.A. (DevOps-курс от RTK)

###Приложение "Генератор заказов" успешно установлено!

###Приятного использования! :-)

----------------------------------------------------

Для Windows/Linux/Mac:

В файле `hosts` добавляем строку:

<IP кластера> generator-orders.test

В браузере проверяем по ссылке: [http://generator-orders.test/](http://generator-orders.test/)

Пример работающего приложения доступен по ссылке: https://generator-orders.devopsf.ru/

### 6. Примечание

Убедитесь, что все компоненты настроены корректно и кластеры работают, прежде чем переходить к тестированию приложения.

### 7. Автоматический Deploy

Для реализации автоматического деплоя был разработан механизм, использующий GitLab. В процессе используются следующие инструменты:

- Helm — для упаковки приложений.

- Docker — для создания образов контейнеров.

- Container Registry — для хранения созданных образов.

- Kubernetes — для оркестрации контейнеров.

Pipelines описаны в примере файла `.gitlab-ci.yml`, который расположен в папке `artifacts`. В целях безопасности из файла были удалены пароли доступа к платформам, поэтому необходимо добавить свои учетные данные перед использованием.

Процесс CD-CD описан в документе: Artifacts/Инструкция по CI-CD.docx