Тестовое задание:

В БД:

- Подключиться к удаленной БД тестового сервера. Логин, пароль: postgres. Сервер: wb-go-team-dev.dev.wb.ru.
- Создать свою бд
- Настроить своего пользователя.
- Создать таблицы для хранения полученных данных.

В сервисе:

1. Подключение и подписка на канал в nats-streaming
2. Полученные данные писать в Postgres
3. Так же полученные данные сохранить in memory в сервисе (Кеш)
4. В случае падения сервиса восстанавливать Кеш из Postgres
5. Поднять http сервер и выдавать данные по id из кеша
6. Выложить сервис в kubernetes
7. Сделать простейший интерфейс отображения полученных данных, для их запроса по id используя TypeScript

Доп инфо

- Данные статичны, исходя из этого подумайте насчет модели хранения в Кеше и в pg
- В канал могут закинуть что угодно, подумайте как избежать проблем из-за этого
- Чтобы проверить работает ли подписка онлайн, сделайте себе отдельный скрипт, для публикации данных в канал
- Подумайте как не терять данные в случае ошибок или проблем с сервисом
- Мы убьем😈 ваш сервис, после восстановления все данные должны быть доступны для запроса
- Подключение к nats-streaming:

```
ClusterURLs = [
    "wbx-world-nats-stage.dp.wb.ru",
    "wbx-world-nats-stage.dl.wb.ru"
]
ClusterID = "world-nats-stage"
Subject = "go.test"
```

- Модель получаемых данных

```
type Order struct {
    OrderUID          string  `json:"order_uid"`
    Entry             string  `json:"entry"`
    InternalSignature string  `json:"internal_signature"`
    Payment           Payment `json:"payment"`
    Items             []Items `json:"items"`
    Locale            string  `json:"locale"`
    CustomerID        string  `json:"customer_id"`
    TrackNumber       string  `json:"track_number"`
    DeliveryService   string  `json:"delivery_service"`
    Shardkey          string  `json:"shardkey"`
    SmID              int     `json:"sm_id"`
}
type Payment struct {
    Transaction  string `json:"transaction"`
    Currency     string `json:"currency"`
    Provider     string `json:"provider"`
    Amount       int    `json:"amount"`
    PaymentDt    int    `json:"payment_dt"`
    Bank         string `json:"bank"`
    DeliveryCost int    `json:"delivery_cost"`
    GoodsTotal   int    `json:"goods_total"`
}
type Items struct {
    ChrtID     int    `json:"chrt_id"`
    Price      int    `json:"price"`
    Rid        string `json:"rid"`
    Name       string `json:"name"`
    Sale       int    `json:"sale"`
    Size       string `json:"size"`
    TotalPrice int    `json:"total_price"`
    NmID       int    `json:"nm_id"`
    Brand      string `json:"brand"`
}
```

- Модель для выдачи

```
type Order struct {
    OrderUID          string  `json:"order_uid"`
    Entry             string  `json:"entry"`
    TotalPrice int    `json:"total_price"` (товары + доставка)
    CustomerID        string  `json:"customer_id"`
    TrackNumber       string  `json:"track_number"`
    DeliveryService   string  `json:"delivery_service"`
}
```

Бонус задание

1. Покройте сервис автотестами. Будет плюсик вам в карму😊
2. Устройте вашему сервису стресс тест, выясните на что он способен
3. Выложите в куб несколько копий сервиса (имитируем несколько датацентров). Подумайте на тему синхронизации этих копий

Сборка докер-образов через CI

https://git.wildberries.ru/oer/go-trainee
скачать изменения из ветки master в свою (если хотите попробовать собрать собственный проект, то файлы кода в папке замените на свои);
если ваш сервис использует какие-то дополнительные файлы для работы, то их нужно положить в докер-контейнер; см Dockerfile
запушьте изменения и в разделе pipelines дождитесь выполнение стадии docker_build (приходится запускать вручную, если неудобно - исправьте в своей ветке в файле .gitlab-ci.yml)
подключитесь к dev кластеру kubernetes https://git.wildberries.ru/infrastructure/kube
В папке k8s вашего проекта лежит файл deployment.yaml, отредактируйте его и выложите сервис со своим неймспейсом
Несколько базовых команд:

- kubectl apply -f file.yaml / добавить содержимое файла
- kubectl delete -f file.yaml / удалить содержимое файла
- kubectl get pods -n namespace_name / получить состояние подов из неймспейса

Пароли хранятся в Secret блоке, закодированные в base64