

# Процесс разворачивания ПО "Сервис подписания договоров онлайн" в Яндекс Облаке

## Подготовка облака

[Создание Сервисного аккаунта](#)

[Создание Container Registry](#)

[Создание Serverless Container](#)

[Создание API Gateway](#)

[Создание Managed Service for YDB](#)

[Создание Object Storage для фронтенда](#)

[Создание Object Storage для сохранения файлов](#)

## Сборка и деплой бэкенда оферты в яндекс облако

[Настройка Serverless Containers](#)

[Обновление бэкенда](#)

[Обновления фронта на стейдже](#)

[Скрипт миграции YDB](#)

## Подготовка облака

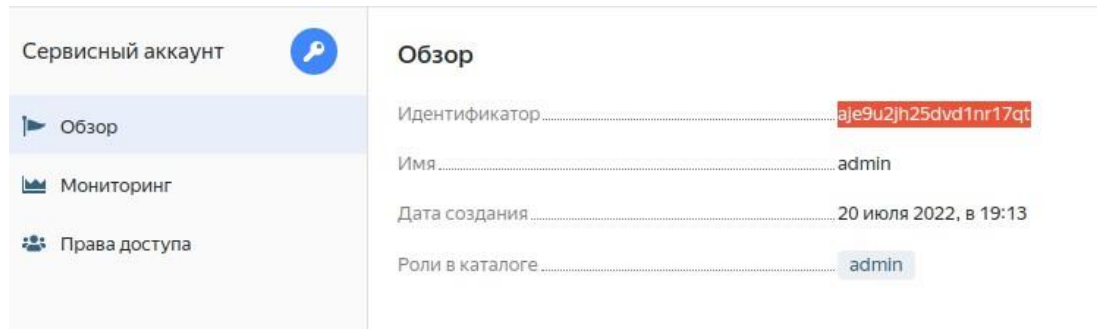
### Создание Сервисного аккаунта

Сервисный аккаунт нужен, что бы разные сервисы в нашем облаке могли получать приватный доступ друг к другу, а так же мы могли заливать новую версию кода в облако.

1. Нажимаем "Создать сервисный аккаунт", вводим любое имя и добавляем роль "admin"

Имя	Роли в каталоге	Дата создания	
admin	admin	20 июля 2022, в 19:13	⋮

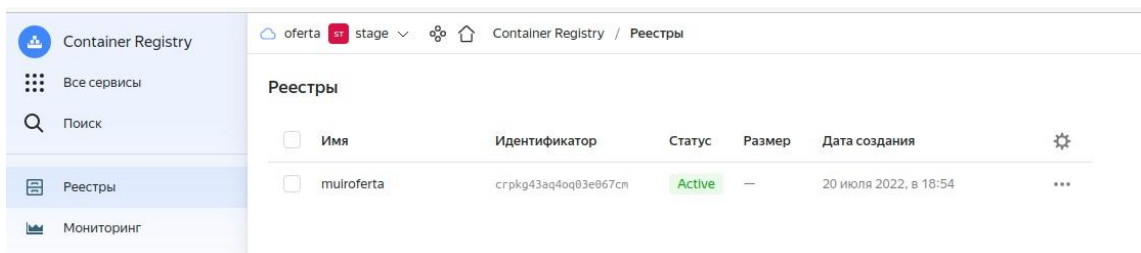
1. Кликаем на него и копируем идентификатор, он нам потребуется позже



## Создание Container Registry

Container Registry нужен что бы хранить докер-образы бэкенда из которых он уже разворачивается в облаке.

1. Выбираем ресурс "Container Registry"/"Реестр Docker-образов", нажимаем "Создать реестр", вводим любое имя, например muiroferta. Созданный реестр появится в списке

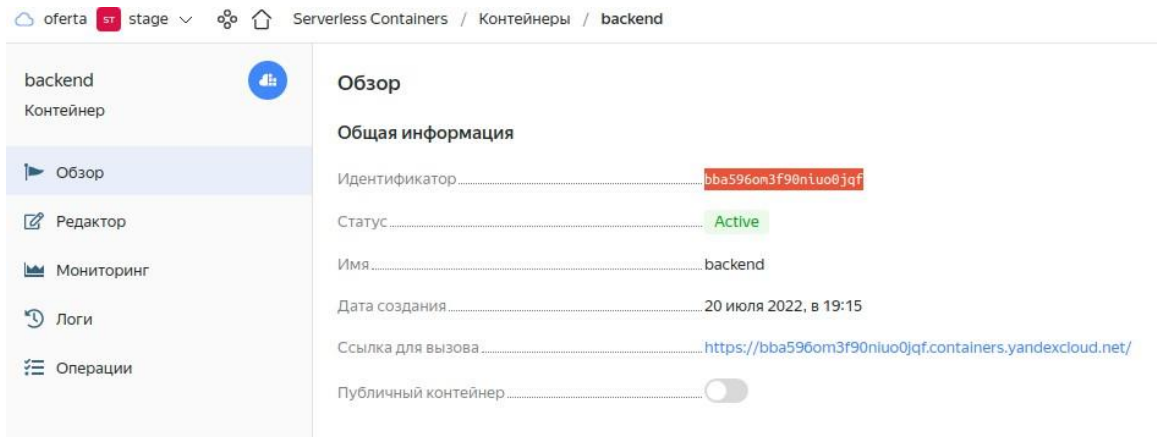


2. Копируем идентификатор, он понадобится нам на втором шаге, когда будем загружать код в облако.

## Создание Serverless Container

Serverless Container это ресурс, который будет запускать docker-образ с бэкендом оферты по запросу.

1. Выбираем ресурс "Serverless Container", нажимаем "Создать контейнер", вводим имя "backend"
2. Откроется окно с редактором, но нам пока оно не нужно, переходим на вкладку "Обзор" и копируем идентификатор



## Создание API Gateway

API Gateway проксирует все запросы к бэкенду, предоставляет единую точку входа и статический адрес (домен), который никогда не изменится. Без него отправлять запросы к бэкенду снаружи (например, из фронтенда), крайне сложно.

1. Выбираем ресурс "API Gateway"/"API-шлюз", появится окно с вводом имени, описания и спеки. Вводим любое имя, например "backend".
2. В поле "Спецификация" задается настройка как, куда и какие запросы проксировать. В нашем случае почти все запросы отправляются на бэкенд, только часть в файловое хранилище. И т.к. сам бэкенд берет на себя обязанность маршрутизации и аутентификации, заполняем спеку максимально обобщенно, что бы API Gateway пропускал все запросы без сложной логики.

**Нужно перечислить все ручки, которые обрабатывает бэкенд, если их тут не указать, запрос будет отбиваться с ошибкой**

В поле

`container_id` вставляем наш идентификатор Serverless Container'a, который мы скопировали на шаге [Создание Serverless Container] (#Создание Serverless Container), в моем случае это `bba596om3f90niu0jqf`

В поле `service_account_id` вставляем идентификатор сервисного аккаунта, который мы получили на шаге [Создание Сервисного аккаунта]

(#Создание Сервисного аккаунта) - `aje9u2jh25dvd1nr17qt`

```
paths:
  /:
    x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
```

```
    type: serverless_containers
    container_id: bba9o1d253mt4glimh4a
    service_account_id: ajeocu594jpv5dpnvhqg
/crm/v1/offers:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqg
/crm/v1/offers/{offer_id}:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqg
/crm/v1/offers/{offer_id}/resend:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqg
/front/v1/auth/{id}:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqg
/front/v1/services:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqg
/front/v1/sign:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
```

```
    service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/sign/verify:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/auth/refresh:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/pay:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba9old253mt4glimh4a
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v2/pay:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/paid:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba9old253mt4glimh4a
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v2/paid:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjetgn
      service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/signedUrlToUpload:
```

```
x-yc-apigateway-any-method:
  x-yc-apigateway-integration:
    type: serverless_containers
    container_id: bba9old253mt4glimh4a
    service_account_id: ajeocu594jpv5dpnvhqq
/front/v1/complete:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba9old253mt4glimh4a
      service_account_id: ajeocu594jpv5dpnvhqq
/payment/success:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqq
/payment/return:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqq
/payment/fail:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqq
/payment/ukassa-webhook:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba9old253mt4glimh4a
      service_account_id: ajeocu594jpv5dpnvhqq
/payment/tinkoff-webhook:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
```

```
    type: serverless_containers
    container_id: bba9old253mt4glimh4a
    service_account_id: ajeocu594jpv5dpnvhqg
/front/v1/status:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqg
/v1/payments/ukassa:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqg
/v1/payments/tinkoff-installment:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba1r70cbeqk397jjtgn
      service_account_id: ajeocu594jpv5dpnvhqg
/crm/v1/files:
  x-yc-apigateway-any-method:
    x-yc-apigateway-integration:
      type: serverless_containers
      container_id: bba9old253mt4glimh4a
      service_account_id: ajeocu594jpv5dpnvhqg
```

3. После создания нажимаем на него и попадаем на вкладку с информацией. Копируем домен в поле "Служебный домен", это будет адресом для доступа к нашему бэкенду офферты, по нему будем делать запросы из фронта.

oferta ST stage ⌵ 🏠 API Gateway / API-шлюз / backend

backend  
API-шлюз

🏠 Обзор

🌐 Домены

🕒 Логи

⚙️ Операции

📊 Мониторинг

Документация

[Как устроен сервис API Gateway](#)

[Начать работу с API Gateway](#)

[Тарифы API Gateway](#)

[Развернуть веб-приложение с Java Servlet API](#)

[Подключить домен](#)

### API-шлюз

**Общая информация**

Идентификатор.....d5dlssobhoc66t6gngce

Статус.....Active

Имя.....backend

Служебный домен.....d5dlssobhoc66t6gngce.apigw.yandexcloud.net

Дата создания.....20 июля 2022, в 20:43

**Спецификация**

```

openapi: 3.0.0
info:
  title: Sample API
  version: 1.0.0
servers:
  - url: https://d5dlssobhoc66t6gngce.apigw.yandexcloud.net
paths:
  /:
    x-yc-apigateway-any-method:
      x-yc-apigateway-integration:
        type: serverless_containers
        container_id: bba596om3f90niu0jqf
        service_account_id: aje9u2jh25dvd1nr17qt
  /crm/v1/offers:
    x-yc-apigateway-any-method:
      x-yc-apigateway-integration:
        type: serverless_containers
        container_id: bba596om3f90niu0jqf
        service_account_id: aje9u2jh25dvd1nr17qt
  /crm/v1/offers/{offer_id}:
    x-yc-apigateway-any-method:
      x-yc-apigateway-integration:
        type: serverless_containers
        container_id: bba596om3f90niu0jqf
        service_account_id: aje9u2jh25dvd1nr17qt

```

## Создание Managed Service for YDB

Это ресурс, который предоставляет доступ к базе данных Yandex DB.

1. Выбираем ресурс "Managed Service for YDB"/"База данных YDB", появится вкладка с настройками, можно все оставить по умолчанию. Нажимаем "Создать базу данных".
2. Кликаем на только что созданную бд и копируем два значения "Эндпоинт" и "Размещение базы данных"

oferta stage Managed Service for YDB / Базы данных / ydb577

ydb577

- Обзор
- Навигация
- Права доступа
- Резервные копии
- Диагностика
- Мониторинг
- Операции

**Обзор**

**Статус**

Идентификатор..... etnsvhtb9g18f4880klv  
Имя..... ydb577  
Тип..... Serverless  
Статус..... **Running**

**Соединение**

Эндпоинт..... **grpc://ydb.serverless.yandexcloud.net:2135**  
Размещение базы данных..... **/ru-central1/b1gmck8aqoka6mhksggha/etnsvhtb9g18f4880klv**

**Document API эндпоинт**

Эндпоинт..... https://docapi.serverless.yandexcloud.net/ru-central1/b1gmck8aqoka6mhksggha/etnsvhtb9g18f4880klv

**Ограничения ?**

Пропускная способность, RU/c..... 10  
Объем данных, ГБ..... 50

**Тарификация ?**

Выделенная пропускная способность, R... .. 0 **i**

Документация  
[Начать работу с Managed YDB](#)  
[Концепции Managed YDB](#)  
[Инструкции для работы с Managed YDB](#)

## Создание Object Storage для фронтенда

Для каждой академии создаем отдельный бакет, он же будет нашим фронтендом, к примеру создадим с названием `dev-muir` для стейджа:

В настройках выбираем "Хостинг" и настраиваем так же как на скриншоте:

Хостинг Переадресация Отключен

Главная страница\* ? index.html

Страница ошибки ? error.html

Ссылка <https://dev-muir.website.yandexcloud.net>

### Переадресация

#### Правило переадресации

Условие ?

Код ответа ? 404

Начало ключа key

#### Переадресация

Протокол  HTTPS  HTTP

Доменное имя dev-muir.website.yandexcloud.net

Код ответа 302

Заменять ключ  ▾

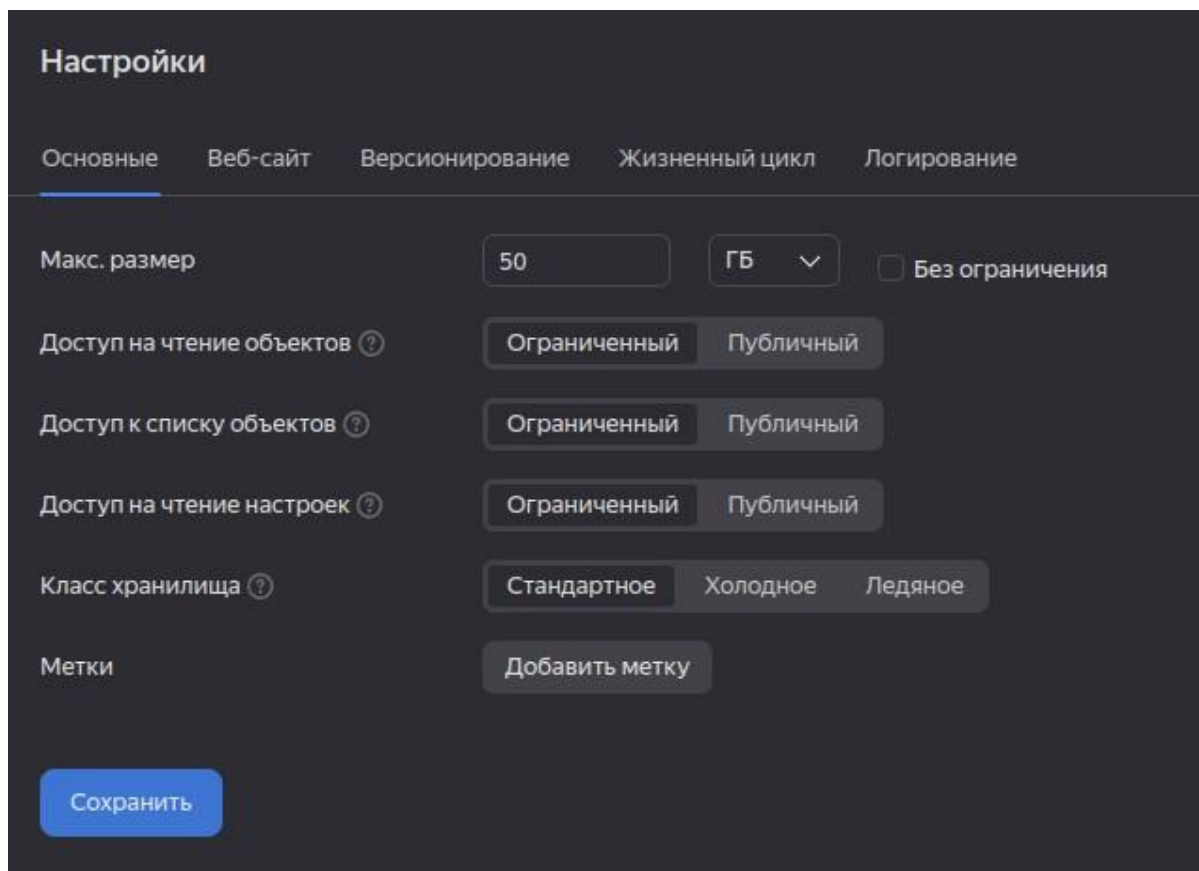
Новый ключ redirect.html

Добавить правило переадресации

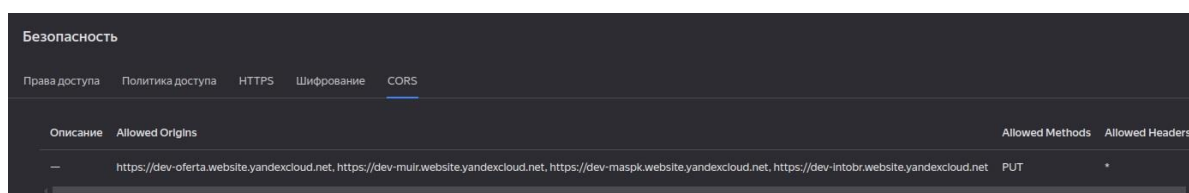
Сохранить

## Создание Object Storage для сохранения файлов

Создаем новый бакет с любым названием, изменяем в настройках все доступы на "ограниченный", как на скриншоте:



На вкладке "Безопасность" → CORS, настраиваем доступ для всех наших академий на метод PUT, чтобы они могли сохранять файлы:



## Сборка и деплой бэкенда oferty в яндекс облако

Для сборки и деплоя проекта нужно в гитлабе добавить новый репозиторий в переменные среды Setting \* CI/CD \* **Variables**:

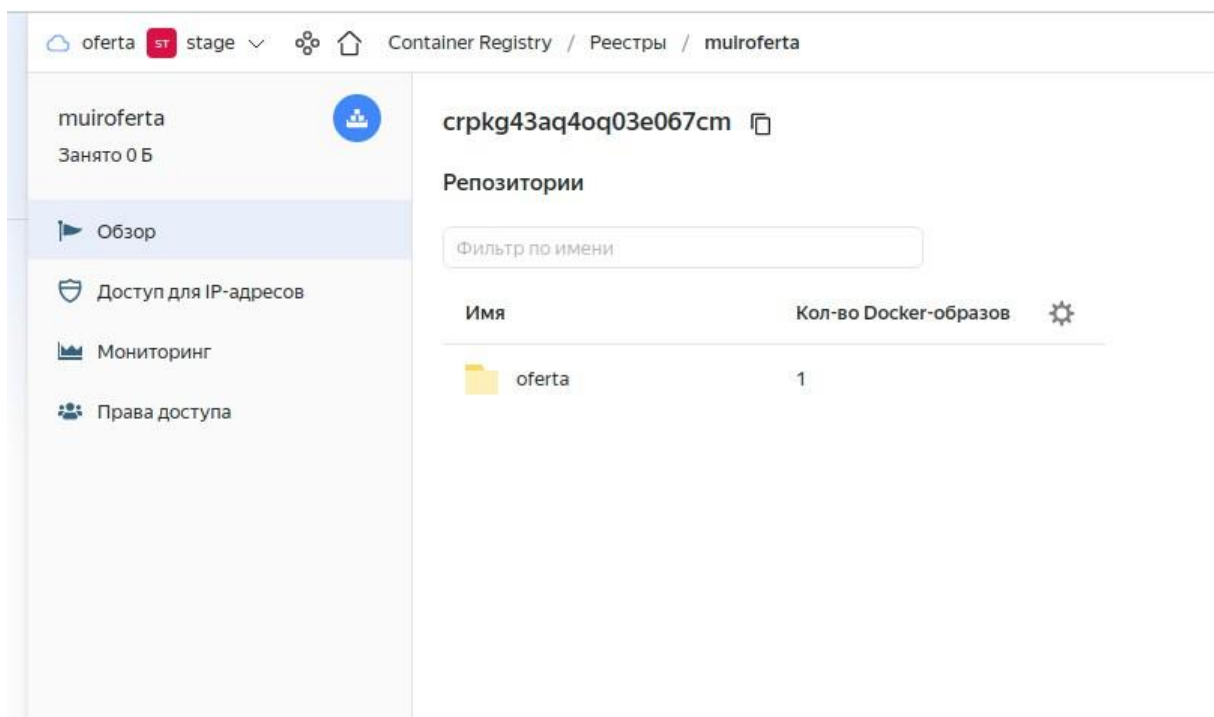
- PROD\_CI\_REGISTRY — идентификатор регистра который мы получили на шаге "Создание Container Registry"
- PROD\_CI\_REGISTRY\_KEY — секретный ключ для доступа к регистру, его можно получить выполнив локально команду ниже, предварительно

создав новый сервисный аккаунт `ci-backend-deployer` и настроив `yc sdk` <https://yandex.cloud/ru/docs/iam/operations/iam-token/create-for-sa>

```
yc iam key create --folder-name production --service-account-
```

После этого при выставление нового тега в гите и пуше изменений будет запускаться пайплайн, после его прохождения нужно будет вручную нажать на старт этапа деплоя в продакшен и изменения попадут в регистр.

После завершения отправки образа, можно зайти в облако, в Container Registry и увидеть там свежий образ:



## Настройка Serverless Containers

Теперь переходим в Serverless Containers → backend → редактор.

В "Параметры образа", поле URL образа кликаем на поле, в списке должен быть только один вариант, его и выбираем, это наш только что залитый образ:

Выбираем тег `latest` .

Следующим шагом задаем переменные окружения:

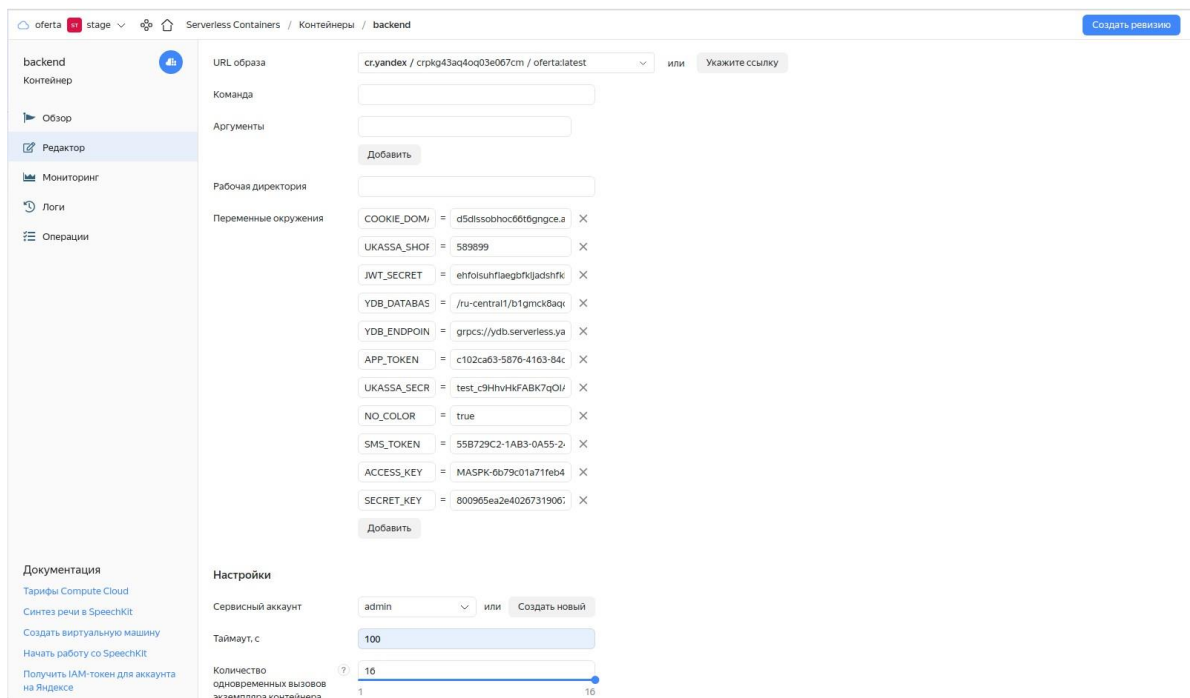
- `COOKIE_DOMAIN = d5dlssobhoc66t6gngce.apigw.yandexcloud.net` - это адрес API Gateway, полученный на шаге [\[x\] Создание API Gateway](#)(#Создание API Gateway)
- `JWT_SECRET = ehfoisuhflaegbfkljadshfklajsdhffewf324352tgs` - нужно указать любую рандомную строку
- `YDB_DATABASE = /ru-central1/b1gmck8aqoka6mhksgha/etnsvhtb9g18f4880klv` - указываем "Размещение базы данных", полученное на шаге [\[x\] Создание Managed Service for YDB](#)(#Создание Managed Service for YDB)
- `YDB_ENDPOINT = grpcs://ydb.serverless.yandexcloud.net:2135` - указываем "Эндпоинт" , полученный на шаге [\[x\]Создание Managed Service for YDB](#) (#Создание Managed Service for YDB)
- `APP_TOKEN = c102ca63-5876-4163-84c6-2b6b439e1f6d` - указываем любую рандомную строку, ее должен будет потом отправлять 1с, что бы он прошел аутентификацию. **не копируем значение с инструкции, обязательно генерируем новое**
- `LOG_LEVEL = info` - уровень логгирования, можно выставить любой от: trace, debug, info, warn, error
- `SMS_TOKEN = <real_token>` - токен для доступа к смс сервису
- `ACCESS_KEY = <real_token>` - первый из двух ключей для доступа к снилс сервису
- `SECRET_KEY = <real_token>` - второй из двух ключей для доступа к снилс сервису

- ♦ `JWT_EXPIRES_IN = 1h` - задаем время жизни access jwt токена, с которым фронт совершает запросы на бэкенд, время жизни refresh токена фиксировано 30 дням
- ♦ `AWS_REGION=ru-central1` - регион s3, для яндекса всегда `ru-central1`
- ♦ `S3_BUCKET=oferta-client-info-stage` - название бакета в котором будут храниться файлы
- ♦ `AWS_ACCESS_KEY_ID=<token>` и `AWS_SECRET_ACCESS_KEY=<token>` - для доступа к бакету с файлами <https://yandex.cloud/ru/docs/storage/s3/>
- ♦ `AWS_ENDPOINT_URL= https://storage.yandexcloud.net` - для яндекса всегда такое значение
- ♦ `GIN_MODE=release` и `WORK_MODE=production` - для работы в продакшен режиме

Далее выбираем сервисный аккаунт, которым будет управляться контейнер, у нас это "admin"

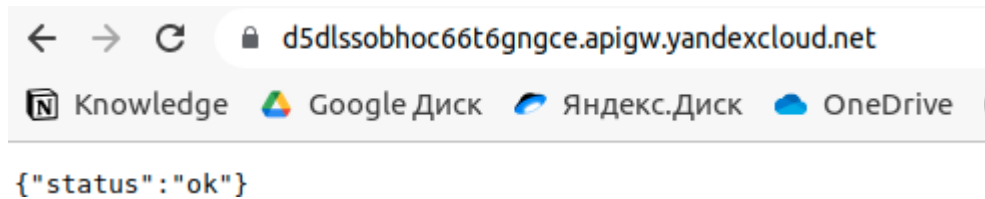
Ставим "Таймаут, с" в значение 100 и "Количество одновременных вызовов экземпляра контейнера" на максимальное значение. Это все нужно, что бы запросы быстрее обрабатывались.

В итоге должно получиться как-то так:



Теперь нажимаем "Создать ревизию" и ждем пока контейнер успешно создастся.

Проверяем, что все успешно настроено отправив get запрос на наш адрес `https://d5dlssobhoc66t6gngce.apigw.yandexcloud.net` (можно просто открыть его в браузере). Если все успешно должен вернуться ответ `{"status": "ok"}`



## Обновление бэкенда

Переходим в Serverless Containers → backend → Редактор. В поле "Параметры образа" → "URL образа", выбираем наш новый образ и нажимаем "Создать ревизию". Яндекс облако попытается запустить новый контейнер параллельно со старым, если все пройдет успешно, то оно остановит и удалит старый контейнер, не прерывая общей работы сервиса (все новые запросы будут редиректятся на новый контейнер). Если новый образ запустить не получилось, то продолжит работать старый.

## Обновления фронта на стейдже

Нужно залить новый код на ветку master, на стейдже он обновится автоматически, если сборка пройдет успешно. Для обновления на проде нужно будет вручную нажать на кнопку старта у соответствующей задаче после того как пайплайн пройдет успешно

## Скрипт миграции YDB

Актуальный скрипт, при обновлении схемы бд, его нужно вручную изменить:

```
CREATE TABLE cache
(
    key          Utf8,
    expiredAt   Timestamp,
    value        Utf8,
    PRIMARY KEY (key)
);
```

```
CREATE TABLE verified_phone
(
    clientId Utf8,
    phone    Utf8,
    PRIMARY KEY (clientId, phone)
);
```

```
CREATE TABLE client
(
    id                Utf8,
    birthDate         Date,
    educationLink     Utf8,
    email             Utf8,
    firstName         Utf8,
    lastName          Utf8,
    middleName        Utf8,
    passportIssuedBy Utf8,
    passportIssuedCode Utf8,
    passportIssuedDate Date,
    passportNumber    Utf8,
    passportSeries    Utf8,
    snils             Utf8,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE service
(
    offerId Utf8,
    name     Utf8,
    count    UInt32,
    price    Utf8,
    PRIMARY KEY (offerId, name)
);
```

```
CREATE TABLE ukassa
(
    id        Utf8,
    shopId    Utf8,
```

```

        secretKey Utf8,
        PRIMARY KEY (id)
    );

CREATE TABLE tinkoff_installment
(
    id          Utf8,
    shopId      Utf8,
    showcaseId Utf8,
    promoCode   Utf8,
    PRIMARY KEY (id)
);

CREATE TABLE offer
(
    id                    Utf8,
    academy               Utf8,
    clientId              Utf8,
    concludedDate         Timestamp,
    createdAt             Timestamp,
    crmId                 Utf8,
    newPhone              Utf8,
    paidAmount            Uint32,
    paidDate              Timestamp,
    phone                 Utf8,
    prevStatus            Utf8,
    sender                Utf8,
    smsCode               Utf8,
    smsSentDate           Timestamp,
    status                Utf8,
    updatedAt             Timestamp,
    verifiedDate          Timestamp,
    verifiedPhone         Utf8,
    paymentGatewayId     Utf8,
    tinkoffInstallmentId Utf8,
    PRIMARY KEY (id)
);

```

```
CREATE TABLE refresh_token
(
    token          Utf8,
    createdAt      Timestamp,
    expiresInSec   Uint32,
    fingerprint    Utf8,
    offerId        Utf8,
    PRIMARY KEY (token)
);

-- 1st migration

CREATE TABLE offer_payment
(
    offerId        Utf8,
    gatewayId      Utf8,
    PRIMARY KEY (offerId, gatewayId)
);

ALTER TABLE tinkoff_installment
    ADD COLUMN title Utf8;

ALTER TABLE tinkoff_installment
    ADD COLUMN password Utf8;
```

## Техническая поддержка

Для оказания технической поддержки по телефону необходимо позвонить: +79999872441.

Также заказчики и пользователи сервиса могут направлять возникающие вопросы на электронную почту технической поддержки по адресу: [panov@muir.info](mailto:panov@muir.info)  
График работы: 8 - 18.