Программное обеспечение «Kolmogorov.ai Predicate»

Руководство по установке

ООО «Дата Сапиенс» | kolmogorov.ai

ПО «Kolmogorov.ai Predicate» | version 2.1.0

Содержание

1. Введение 3
2. Пререквизиты 4
   2.1 Kubernetes/OpenShift 4
   2.2 Keycloak 4
   2.3 S3 хранилище 4
3. Установка Predicate в Kubernetes 5
   3.1 Доступ к образам и чартам 5
      3.1.1 Пример получения docker образа из репозитория: 5
      3.1.2 Пример получения helm чарта из репозитория: 5
   3.2 Predicate 6
      3.2.1 Устройство чарта 6
      3.2.2 Переменные окружения 7
      3.2.3 Установка/обновление/удаление релиза Predicate 11
   3.3 Predicate UI 12
      3.3.1 Устройство чарта 12
      3.3.2 Переменные окружения 13
      3.3.3 Установка/обновление/удаление релиза Predicate UI 14
4. Установка и настройка Keycloak 15
5. Проверка работоспособности приложения 18
   5.1 Предварительные действия 18
   5.2 Первичная авторизация 18
      5.2.1 Авторизация 18
   5.3 Работа в приложении 19
   5.4 Регистрация источника данных 19
   5.5 Регистрация датасета 19
   5.6 Создание проекта 19
   5.7 Проверка исполнения проекта 21
6. Установка Predicate в Yandex Cloud 23
   6.1 Создание Managed Service for Kubernetes и его настройка 23
      6.1.1 Установка NFS сервера 25
      6.1.2 Установка Ingress Controller 26
   6.2 Создание S3 хранилища и настройка доступа к нему 26
   6.3 Установка Keycloak и настройка доступа к нему 27
   6.4 Установка Predicate 27

1. Введение

Predicate позволяет тестировать модели и проверять качество данных рассчитывая метрики по регламенту и формируя отчеты.

Основной сервис бэкенда - FastAPI сервер, который принимает запросы по адресу, описанному в ingress сервиса. Часть задач выполняется асинхронно с использованием Celery. Данные хранятся в БД Postgres. Для каждого проекта генерируется DAG Airflow, который позволяет мониторить данные, метрики и модели по регламенту.

Взаимодействие пользователя с приложением осуществляется через web-интерфейс. Доступ к функционалу открывается после авторизации.

Установка

Первичная установка программного обеспечения производится командой разработки и внедрения ООО «Дата Сапиенс» на сервер, предоставляемый заказчиком.

Обновление

Проверка, загрузка и установка обновлений программного обеспечения требует подключения к Интернету и выполняется вручную на сервере администратором ПО. Для обновления комонента ПО необходимо удалить его и установить повторно, после чего снова запустить Kolmogorov.ai **Predicate**.

Переустановка

Любую доступную версию Kolmogorov.ai **Predicate** можно загрузить и установить с помощью контейнеров, поставляемых разработчиком. Перед переустановкой допускается удаление существующего ПО.

2. Пререквизиты

Компоненты, которые не входят в дефолтную поставку приложения Predicate, однако необходимы для его работы.

2.1 Kubernetes/OpenShift

`K8s`-совместимый кластер с возможностью заведения отдельного namespace. В Данном namespace - возможность создания сервисной УЗ с правами на CRUD всех базовых типов объектов. Минимальная ресурсная квота - `8CPU, 16Gb RAM, 100Gb disk space`. Также, на локальной машине, должен быть установлен инструмент для управления Kubernetes приложениями - **Helm**. Дополнительно в кластере понадобятся:

- `NFS StorageClass` - RWM класс хранения в Kubernetes, который использует NFS-сервер.
- `Ingress Controller` - контроллер, который обеспечивает доступ к сервисам Kubernetes извне.

Установка Kubernetes на minikube

2.2 Keycloak

Сервис для авторизации и аутентификации **Keycloak**.

Рекомендуется использовать чарт codecentric. Описание установки и настройки сервиса можно найти здесь.

2.3 S3 хранилище

S3 хранилище (Minio или Yandex Object Storage) - хранилище, в котором сохраняются результаты метрик. В данном хранилище должна быть возможностью заведения отдельного бакета, минимально рекомендованный размер которого - `100Gb`. Для работы с данным бакетом нужен сервисный аккаунт с ролью `storage.editor` или `storage.admin` (GET/UPDATE/DELETE объектов).

Инструкция по установке Minio

3. Установка Predicate в Kubernetes

3.1 Доступ к образам и чартам

Важно

Необходимые учётные данные для доступа к репозиториям можно получить по запросу.

3.1.1 Пример получения docker образа из репозитория:

```
export USERNAME=
export PASSWORD=
export URL_DOCKER_REGISTRY=
export IMAGE=
docker login $URL_DOCKER_REGISTRY --username $USERNAME --password $PASSWORD
docker pull $IMAGE
```

3.1.2 Пример получения helm чарта из репозитория:

```
export URL_CHART_REGISTRY=
export CHART_NAME=
export CHART_VERSION=
helm repo add predicate $URL_CHART_REGISTRY --username $USERNAME --password $PASSWORD
helm repo update
helm search repo $CHART_NAME -l --devel
helm pull predicate --version $CHART_VERSION
```

Важно

Для успешного pull образов на кластере необходимо создать секрет `type: kubernetes.io/dockerconfigjson` с именем `regcred` в нужном namespace.

3.2 Predicate

3.2.1 Устройство чарта

Чарт содержит в себе следующие сервисы:

`predicate`

Основной сервис бэкенда. Внутри контейнера разворачивается FastAPI сервер. Также сервис содержит два init-container:

- `init-db` - применяющий миграции Alembic к базе данных;
- `metric-init` - инициализирует стандартные метрики Predicate. Если метрика уже существует, то она не будет создана и в лог выведется соответсвующее сообщение.

`predicate-manager`

Celery сервис, который отвечает за выполнение системных задач (создание рана проекта, создание необходимых нод, сохранение результатов в базу, загрузка файлов в S3 и т.д.).

`predicate-worker`

Celery сервис, который отвечает за выполнение расчётных задач (нод) проектов (выгрузка данных, применение преобразования над данными, применение метрик на данных)

`predicate-celery-monitor`

Вспомогательный сервис, который отвечает за мониторинг Celery задач и сохранения их состояния в базу данных.

`predicate-flower`

Celery Flower сервис, который отвечает за визуализацию Celery задач.

`predicate-postgresql`

Сервис, который отвечает за хранение базы данных Predicate.

`predicate-rabbitmq`

Сервис, который отвечает за хранение очереди задач для Celery.

3.2.2 Переменные окружения

Название	Описание	Обязательная	Значение по умолчанию
PREDICATE_ROOT_PATH	Базовый путь до API Predicate `(пример: "/api")`	Да	""
PREDICATE_HOSTNAME	Имя хоста для сервиса Predicate `(пример: "https://predicate-dev.k8s.datasapience.ru")`	Да	"http://localhost:8000"
PREDICATE_DB_URL	URL базы данных для Predicate `(пример: "postgresql+psycopg2://postgres:postgres@predicate-postgresql:5432/postgres")`	Да	
PREDICATE_MANAGER_CELERY_NAME	Имя для процесса для Celery менеджера `(пример: "predicate-manager")`	Нет	"predicate-manager"
PREDICATE_MANAGER_CELERY_BROKER_URL	URL брокера для Celery менеджера `(пример: "amqp://admin:adminq@predicate-rabbitmq:5672")`	Да	"redis://localhost:6379"
PREDICATE_MANAGER_CELERY_RESULT_BACKEND	Бэкенд результатов для Celery менеджера `(пример: "db+postgresql://postgres:postgres@predicate-postgresql:5432/postgres")`	Да	"redis://localhost:6379"
PREDICATE_MANAGER_QUEUE_DEFAULT	Очередь по умолчанию для Celery менеджера `(пример: "manager")`	Нет	"manager"
PREDICATE_MANAGER_MAX_RETRIES	Максимальное количество повторных попыток для Celery менеджера `(пример: "5")`	Нет	"5"
PREDICATE_MANAGER_RETRY_BACKOFF	Использовать ли отсрочку повторных попыток для Celery менеджера `(пример: "True")`	Нет	"True"
PREDICATE_MANAGER_DEFAULT_RETRY_DELAY	Задержка повторной попытки по умолчанию для Celery менеджера в секундах `(пример: "5")`	Нет	"5"
PREDICATE_WORKER_CELERY_NAME	Имя для процесса Celery воркера `(пример: "predicate-worker")`	Нет	"predicate-worker"
PREDICATE_WORKER_CELERY_BROKER_URL	URL брокера для Celery воркера `(пример: "amqp://admin:admin@predicate-rabbitmq:5672")`	Да	"redis://localhost:6379"
PREDICATE_WORKER_CELERY_RESULT_BACKEND	Бэкенд результатов для Celery воркера `(пример: "db+postgresql://postgres:postgres@predicate-postgresql:5432/postgres")`	Да	"redis://localhost:6379"
PREDICATE_WORKER_QUEUE_DEFAULT	Очередь по умолчанию для Celery воркера `(пример: "sandbox")`	Нет	"sandbox"
PREDICATE_WORKER_MAX_RETRIES	Максимальное количество повторных попыток для Celery воркера `(пример: "5")`	Нет	"5"

Название	Описание	Обязательная	Значение по умолчанию
PREDICATE_WORKER_RETRY_BACKOFF	Использовать ли отсрочку повторных попыток для Celery воркера `(пример: "True")`	Нет	"True"
PREDICATE_WORKER_DEFAULT_RETRY_DELAY	Задержка повторной попытки по умолчанию для Celery воркера в секундах `(пример: "5")`	Нет	"5"
PREDICATE_WORKER_LOG_PATH	Путь к логам для Celery воркера `(пример: "/tmp/logs")`	Да	"./var/logs"
PREDICATE_EXECUTOR_RESULT_DIR	Директория для сохранения результатов в рамках проекта `(пример: "/tmp/result")`	Да	"./var/result"
PREDICATE_EXECUTOR_DRIVER_PWD	Директория с jar файлами драйверов, для подключения к источникам данных `(пример: "/app/drivers")`	Да	"./var/drivers"
PREDICATE_EXECUTOR_METRIC_PWD	Директория с кодом метрик `(пример: "/tmp/share")`	Да	"./var/metrics"
PREDICATE_EXECUTOR_TRANSFORM_PWD	Директория с кодом преобразований `(пример: "/tmp/share")`	Да	"./var/transform"
PREDICATE_EXECUTOR_DATA_PWD	Директория для сохранения данных в рамках проекта `(пример: "/tmp/result")`	Да	"./var/result"
PREDICATE_EXECUTOR_S3_TMP_DIR	Временная директория S3 для исполнителя `(пример: "/tmp/s3")`	Да	"./var/s3"
PREDICATE_EXECUTOR_K8S_USE_LOCAL	Использовать ли локальный Kubernetes для исполнителя `(пример: "False")`	Да	"True"
KEYCLOAK_URL	URL для аутентификации Keycloak `(пример: "https://auth.k8s.datasapience.ru/auth")`	Да	
KEYCLOAK_REALM	Realm для Keycloak `(пример: "dev")`	Да	
KEYCLOAK_CLIENT_ID	ID клиента для Keycloak `(пример: "kolmogorov")`	Да	
KEYCLOAK_ADMIN_USERNAME	Имя пользователя администратора для Keycloak `(пример: "writer")`	Да	
KEYCLOAK_ADMIN_PASSWORD	Пароль администратора для Keycloak `(пример: "writer")`	Да	
PREDICATE_SENTRY_ENABLED	Включен ли Sentry для Predicate `(пример: "False")`	Нет	"False"
PREDICATE_SENTRY_DSN	DSN Sentry для Predicate `(пример: "")`	Нет	
PREDICATE_WORKER_SENTRY_ENABLED	Включен ли Sentry для Celery воркера `(пример: "False")`	Нет	"False"
PREDICATE_WORKER_SENTRY_DSN	DSN Sentry для Celery воркера `(пример: "")`	Нет	

Название	Описание	Обязательная	Значение по умолчанию
PREDICATE_S3_URL	URL S3 для Predicate `(пример: "https://storage.yandexcloud.net")`	Да	
PREDICATE_S3_ACCESS_KEY	Ключ доступа S3 для Predicate `(пример: "AccessKey")`	Да	
PREDICATE_S3_SECRET_ACCESS_KEY	Секретный ключ доступа S3 для Predicate `(пример: "AccessSecret")`	Да	
PREDICATE_S3_VERIFY	Проверять ли SSL-сертификаты S3 `(пример: "False")`	Нет	"False"
PREDICATE_S3_BUCKET	Имя бакета S3 для Predicate `(пример: "predicate")`	Да	
PREDICATE_S3_PREFIX	Префикс с именем папки S3 для Predicate `(пример: "predicate-dev")`	Нет	"predicate"
PREDICATE_LOCAL_STORAGE	Использовать ли локальное хранилище для Predicate `(пример: "False")`	Нет	"False"
TZ	Часовой пояс для приложения `(пример: "Europe/Moscow")`	Да	"Europe/Moscow"

3.2.3 Установка/обновление/удаление релиза Predicate

Установка/обновление

1. Получите дефолтный `values-predicate.yaml` из чарта:

```
export CHART_NAME=
export CHART_VERSION=
helm show values $CHART_NAME --version $CHART_VERSION >> values-predicate.yaml
```

2. Отредактируйте файл `values-predicate.yaml`:

- Укажите актуальный репозиторий образа и тег образа в `global.image.repository` и `global.image.tag`
- Подставьте актуальные данные для ingress и в следующих сервисах
- `service.ingress.host`, `service.ingress.baseDomain`
- `flower.ingress.host`, `flower.ingress.baseDomain`
- `rabbitmq.ingress.hostname`, `rabbitmq.ingress.extraTls.hosts`
- Укажите актуальные переменные окружения `service.secrets.data`, которые описаны в разделе Переменные окружения; минимально необходимо переопределить в `values-predicate.yaml` следующие переменные:
- `PREDICATE_HOSTNAME`
- `KEYCLOAK_URL`
- `KEYCLOAK_REALM`
- `KEYCLOAK_CLIENT_ID`
- `KEYCLOAK_ADMIN_USERNAME`
- `KEYCLOAK_ADMIN_PASSWORD`
- `PREDICATE_S3_URL`
- `PREDICATE_S3_ACCESS_KEY`
- `PREDICATE_S3_SECRET_ACCESS_KEY`
- `PREDICATE_S3_BUCKET`

1. Запустите команду для установки релиза

```
helm upgrade \
--namespace=<namespace> \
--install \
--atomic \
--values values-predicate.yaml \
<releasename> <path to chart>
```

где `namespace` - неймспейс k8s, в котором устанавливается релиз, `releasename` - название релиза, `path to chart` - путь до папки чарта или до архива

Удаление

Для удаления чарта запустите команду:

```
helm delete --namespace=<namespace> <releasename>
```

где `namespace` - неймспейс k8s, в котором устанавливается релиз, `releasename` - название релиза,

> **Важно**
>
> Для полного удаления релиза необходимо: 1. Удалить секреты, с именем всех существующих источников данных; 2. Удалить папку в бакете, указанную в переменной окружения `PREDICATE_S3_PREFIX`. 3. Запустить команду `helm delete --namespace=<namespace> <releasename> --purge`.

3.3 Predicate UI

3.3.1 Устройство чарта

Чарт содержит в себе один сервис:

`predicate-ui`

Сервис с файлами web-приложения. Также сервис содержит init-контейнер:

- `generate-configs` - настраивает переменные окружения и создает файл конфигурации.

3.3.2 Переменные окружения

Название	Описание	Обязательная
PREDICATE_API	Базовый путь до API Predicate `(пример: "/api")`	Да
TITLE	Заголовок приложения, отображаемый в интерфейсе логирования `(пример: "Predicate")`	Да
SUBTITLE	Подзаголовок приложения, отображаемый в интерфейсе логирования `(пример: "Metric Store - валидация ML Моделей")`	Да
KEYCLOAK_URL	Адрес Keycloak для авторизации, обычно совпадает с указанным в бекенде `(пример: "https://<keycloak-url>/auth")`	Да
KEYCLOAK_REALM	Название realm в Keycloak `(пример: "dev")`	Да
KEYCLOAK_CLIENT_ID	Идентификатор клиента в Keycloak `(пример: "kolmogorov")`	Да
KEYCLOAK_CLIENT_SECRET	Секретный ключ клиента Keycloak, по умолчанию пустой `(пример: "")`	Да
TZ	Часовой пояс `(пример: "Europe/Moscow")`	Да
SENTRY_ENABLED	Флаг, указывающий, включен ли сервис Sentry для отслеживания ошибок `(пример: "true")`	Да
SENTRY_DNS	DNS-адрес проекта Sentry `(пример: "https://<url-sentry-project>/<project_number>")`	Да

3.3.3 Установка/обновление/удаление релиза Predicate UI

Установка/обновление

1. Получите дефолтный `values-predicate-ui.yaml` из чарта:

```
export CHART_NAME=
export CHART_VERSION=
helm show values $CHART_NAME --version $CHART_VERSION >> values-predicate-ui.yaml
```

2. Отредактируйте файл `values-predicate-ui.yaml`

- Укажите актуальный репозиторий образа - `service.image.repository` и тег образа - `service.image.tag`;
- Укажите актуальный `service.ingress.host` и `service.ingress.baseDomain`;
- Укажите актуальные переменные окружения `service.secrets.data`, которые описаны в разделе Переменные окружения; минимально необходимо переопределить в `values-predicate-ui.yaml` следующие переменные:
- `KEYCLOAK_URL`
- `KEYCLOAK_REALM`
- `KEYCLOAK_CLIENT_ID`

3. Запустите команду

```
helm upgrade \
--namespace=<namespace> \
--install \
--atomic \
--values values-predicate-ui.yaml \
<releasename> <path to chart>
```

где `namespace` - неймспейс k8s, в котором устанавливается релиз, `releasename` - название релиза, `path to chart` - путь до папки чарта или до архива

Удаление

Для удаления чарта запустите команду:

```
helm delete --namespace=<namespace> <releasename>
```

где `namespace` - неймспейс k8s, в котором устанавливается релиз, `releasename` - название релиза,

Важно

Для полного удаления релиза необходимо запустить команду `helm delete --namespace=<namespace> <releasename> --purge`.

4. Установка и настройка Keycloak

4.0.1 Установка Keycloak

Важно

Если у вас уже есть Keycloak, то пропустите этот шаг и перейдите к его настройке.

Для установки Keycloak можете воспользоваться следующим чартом

Перед установкой необходимо создать следующий файл `values.yaml`, подставив везде свое значение в переменную `IP_LOADBALANCER`. Если у вас есть собственное доменное имя, то замените `IP_LOADBALANCER.nip.io` на свое доменное имя.

Файл `values.yaml`

```yaml
keycloakUser: "admin"
keycloakPassword: "admin"


databaseUser: "keycloak-user"
databasePassword: "dbpassword"

replicas: 1


extraEnv: |
 - name: JAVA_OPTS
   value: >-
     -XX:+UseContainerSupport
     -XX:MaxRAMPercentage=50.0
     -Djava.net.preferIPv4Stack=true
     -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS
     -Djava.awt.headless=true
     -Dkeycloak.profile.feature.upload_scripts=enabled
 - name: KEYCLOAK_LOGLEVEL
   value: INFO
 - name: PROXY_ADDRESS_FORWARDING
   value: "true"


extraEnvFrom: |
 - secretRef:
     name: '{{ include "keycloak.fullname" . }}-cred'
 - secretRef:
     name: '{{ include "keycloak.fullname" . }}-db'

secrets:
 cred:
   stringData:
     KEYCLOAK_USER: "{{ .Values.keycloakUser }}"
     KEYCLOAK_PASSWORD: "{{ .Values.keycloakPassword }}"
 db:
   stringData:
     DB_USER: '{{ .Values.databaseUser }}'
     DB_PASSWORD: '{{ .Values.databasePassword }}'


# resources:
#   requests:
#     cpu: 500m
#     memory: 1024Mi
#   limits:
#     cpu: 2000m
#     memory: 2048Mi


ingress:
 enabled: true
 ingressClassName: "nginx"
 servicePort: http
 annotations:
   ingress.kubernetes.io/ssl-redirect: "true"
   nginx.ingress.kubernetes.io/ssl-redirect: "true"
   nginx.ingress.kubernetes.io/proxy-body-size: "128k"
   nginx.ingress.kubernetes.io/server-snippet: |
     more_set_headers "Access-Control-Allow-Origin: $http_origin";
     location ~* /auth/realms/[^/]+/metrics {
         return 403;
     }
 rules:
   - host: "IP_LOADBALANCER.nip.io"
     paths:
       - path: /auth
         pathType: Prefix
 tls:
   - hosts:
       - "IP_LOADBALANCER.nip.io"
     secretName: dev-wildcard
 console:
   enabled: true
   ingressClassName: "nginx"
   annotations:
     ingress.kubernetes.io/ssl-redirect: "true"
     nginx.ingress.kubernetes.io/ssl-redirect: "true"
     nginx.ingress.kubernetes.io/proxy-body-size: "128k"
   rules:
     - host: "IP_LOADBALANCER.nip.io"
       paths:
         - path: /auth/admin/
           pathType: Prefix
   tls:
     - hosts:
         - "IP_LOADBALANCER.nip.io"
       secretName: dev-wildcard
```

После создания файла, выполните установку Keycloak:

```
helm repo add codecentric https://codecentric.github.io/helm-charts
helm repo update
helm upgrade --install keycloak codecentric/keycloak \
--values values.yaml \
--namespace default \
--kube-context $CONTEXT_NAME
```

4.0.2 Настройка доступа к Keycloak

1. Пройдите по адресу `https://IP_LOADBALANCER.nip.io/auth/admin/master/console/` и авторизуйтесь в системе.
2. Создайте клиента под именем `klmg` и добавить клиенту в поле Valid redirect URIs текущий host в формате `https://{HOST}/*` (для локальной установки - `https://IP_LOADBALANCER.nip.io/*`).
3. В настройках клиента поле `Access Type` должно быть в значении `public`. В более новых версиях Keycloak, аналогом поля `Access Type` являются поля `Client authentication` и `Authorization`, они должны быть в состоянии `off`.

   3.

   `В дефолтной версии:`

   В настройках клиента необходимо перейти в `Mappers`, нажать кнопку `Create`.

   `В новой версии:`

   В настройках клиента необходимо перейти в `Client scopes`. Из дефолтного списка `Assigned client scope` выбрать `<название клиента>-dedicated` и добавить маппер `by configuration` нажав `Add mapper`.

   В настройках указать:

   - Name: roles
   - Mapper type: User Client Role
   - Token Claim Name: roles

   Остальные настройки оставить по умолчанию, нажать `Save`.

1. В настройках клиента необходимо перейти в `Roles` и создать роль `predicate_admin` для предоставления выбранным пользователям прав админа (возможность видеть и изменять все сущности созданные на стенде любым пользователем) и `predicate_metric` (доступ до базовых метрик).
2. Создать необходимых пользователей и выдать им при необходимости роль `predicate_admin`. Если у пользователя нет роли `predicate_admin`, то он ему нужно будет выдать роль `predicate_metric`, чтобы он мог видеть базовые метрики. Для этого перейдите в раздел `Users`, выберите пользователя, перейдите в раздел `Role Mappings`.

   `В дефолтной версии:`

   В поле `Client Roles` выберите клиента `klmg`, в поле `Available Roles выберите роль` выберите `predicate_admin` и нажмите `Add selected`.

   `В новой версии:`

   Нажмите `Assign role` -> `Fileter by client`. В поиске введите `predicate_admin`, выберите роль и нажмите `Assign`.

5. Проверка работоспособности приложения

5.1 Предварительные действия

Для проверки разворота приложения **Kolmogorov.ai Predicate** необходимо:

1. Произвести установку **Kolmogorov.ai Predicate** со всеми компонентами, в том числе связать приложение с KeyCloak.
2. В KeyCloak создать пользователя с ролью `predicate_admin`.
3. Развернуть или обеспечить доступ к хранилищу S3 или базе данных подходящего типа. В хранилище S3 или БД должен быть хотя бы один файл формата `.csv` или таблица соответственно.

5.2 Первичная авторизация

Доступ пользователя к приложению Kolmogorov.ai **Predicate** осуществляется через веб-интерфейс браузера.

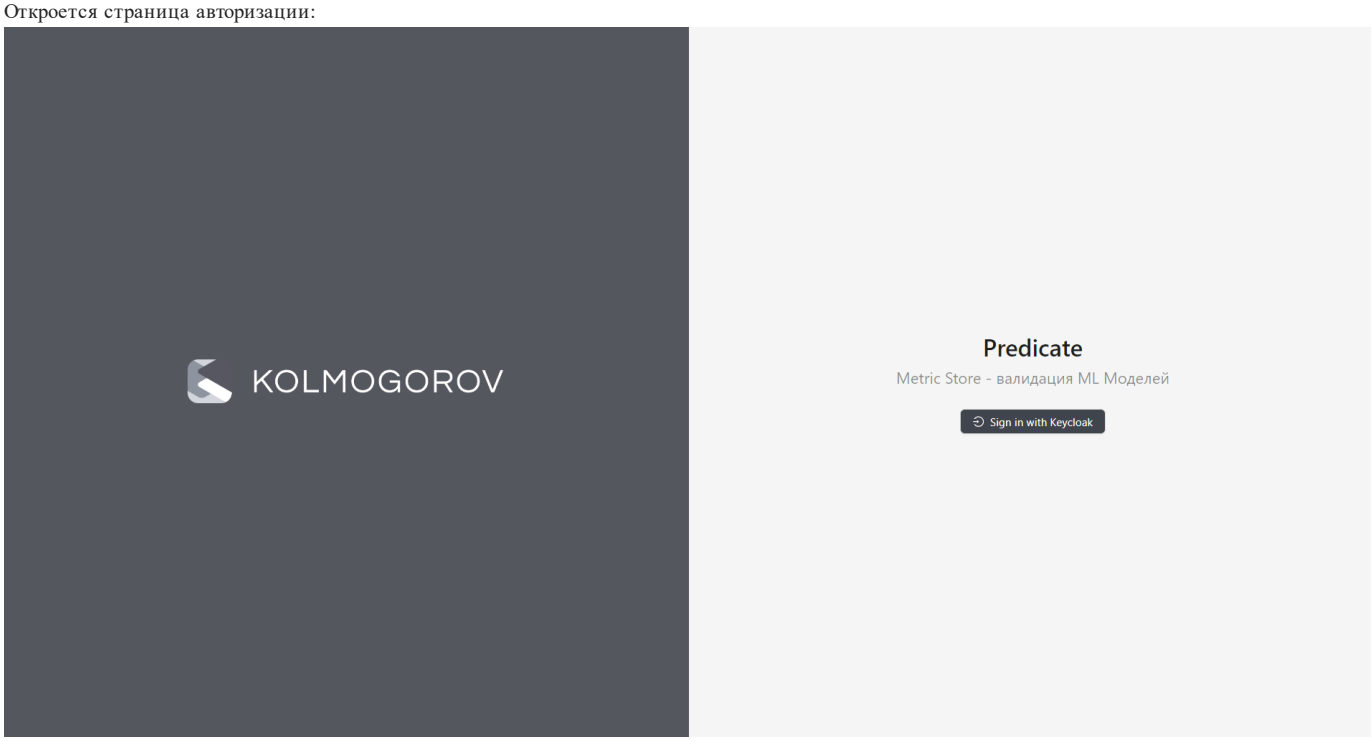
Для начала работы в приложении необходимо авторизоваться.

5.2.1 Авторизация

Вход в приложение

Для входа в приложение **Predicate** необходимо в браузере перейти по пути его разворота.

Откроется страница авторизации:



Нажмите на кнопку "Sing in with KeyCloak". В открывшемся окне введите логин (Username) и пароль (Password) созданного пользователя с ролью `predicate_admin` в соответствующие поля.

5.3 Работа в приложении

Приложение после установки должно содежать только заполненный каталог метрик (*Каталог > Метрики*). Остальные каталоги должны быть пусты. Для наполнения приложения и проверки его работоспособности необходимо проделать следующие действия:

1. Добавить источник данных.
2. Добавить датасет из источника.
3. Создать проект.

При успешном выполнении всех шагов можно считать, что разворот прошел успешно и приложение базово работоспособно.

5.4 Регистрация источника данных

Регистрация источника данных описана на этой странице руководства пользователя.

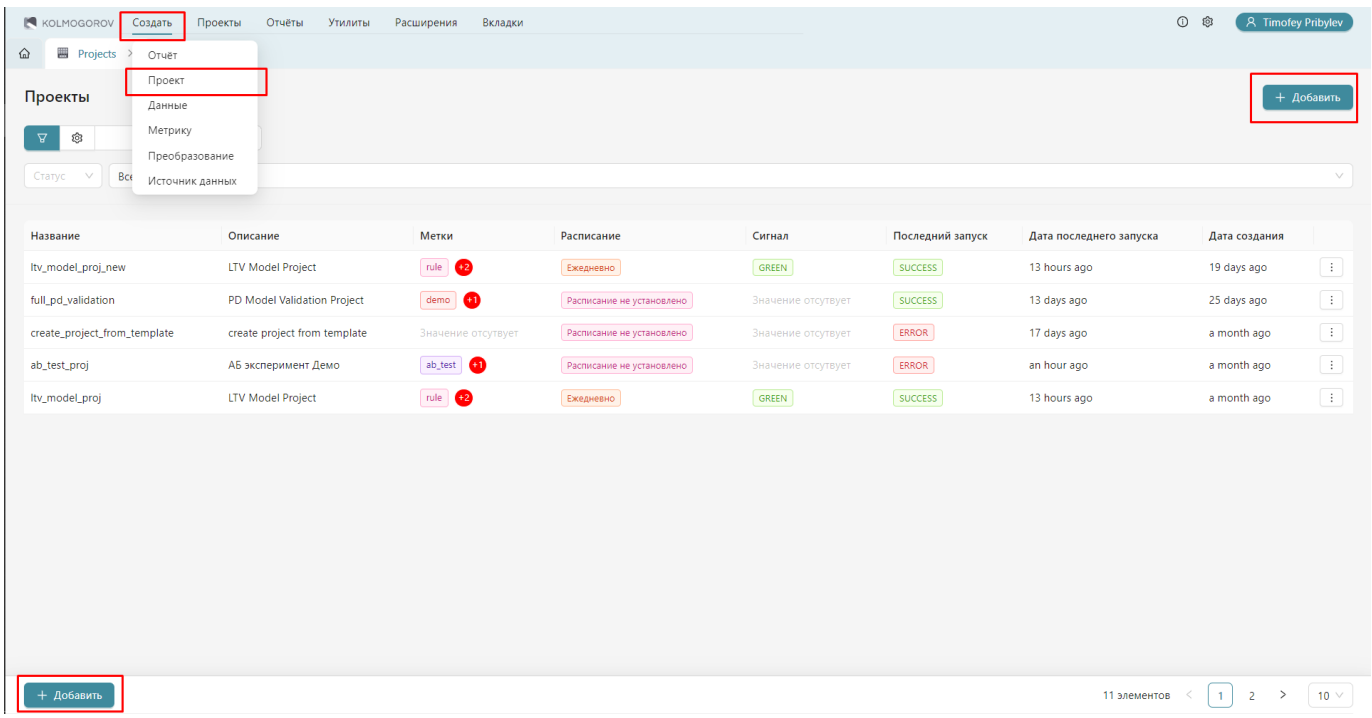
5.5 Регистрация датасета

Регистрация датасета описана на этой странице руководства пользователя.

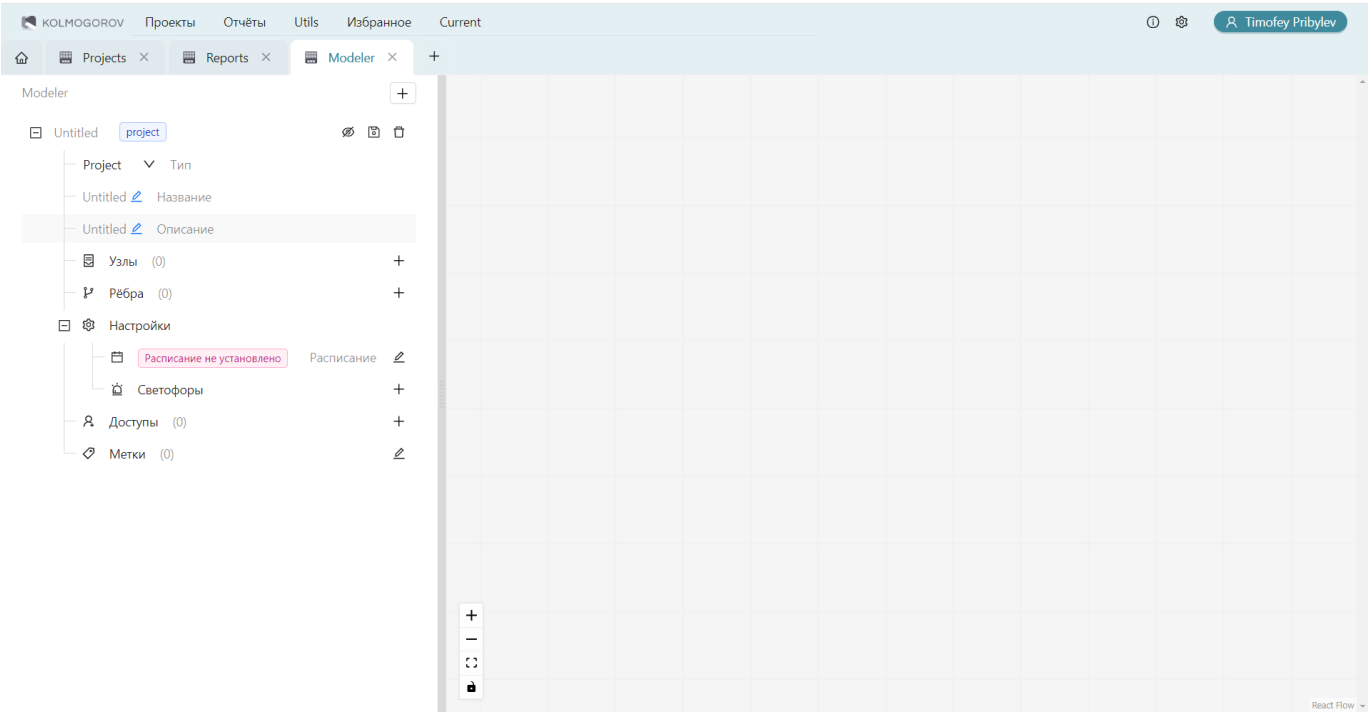
5.6 Создание проекта

После регистрации всех перечисленных выше объектов можно приступать к созданию проекта.

Переход к форме создания нового проекта осуществляется из каталога проектов (*Панель управления > Проекты*) нажатием на кнопку «Добавить»:



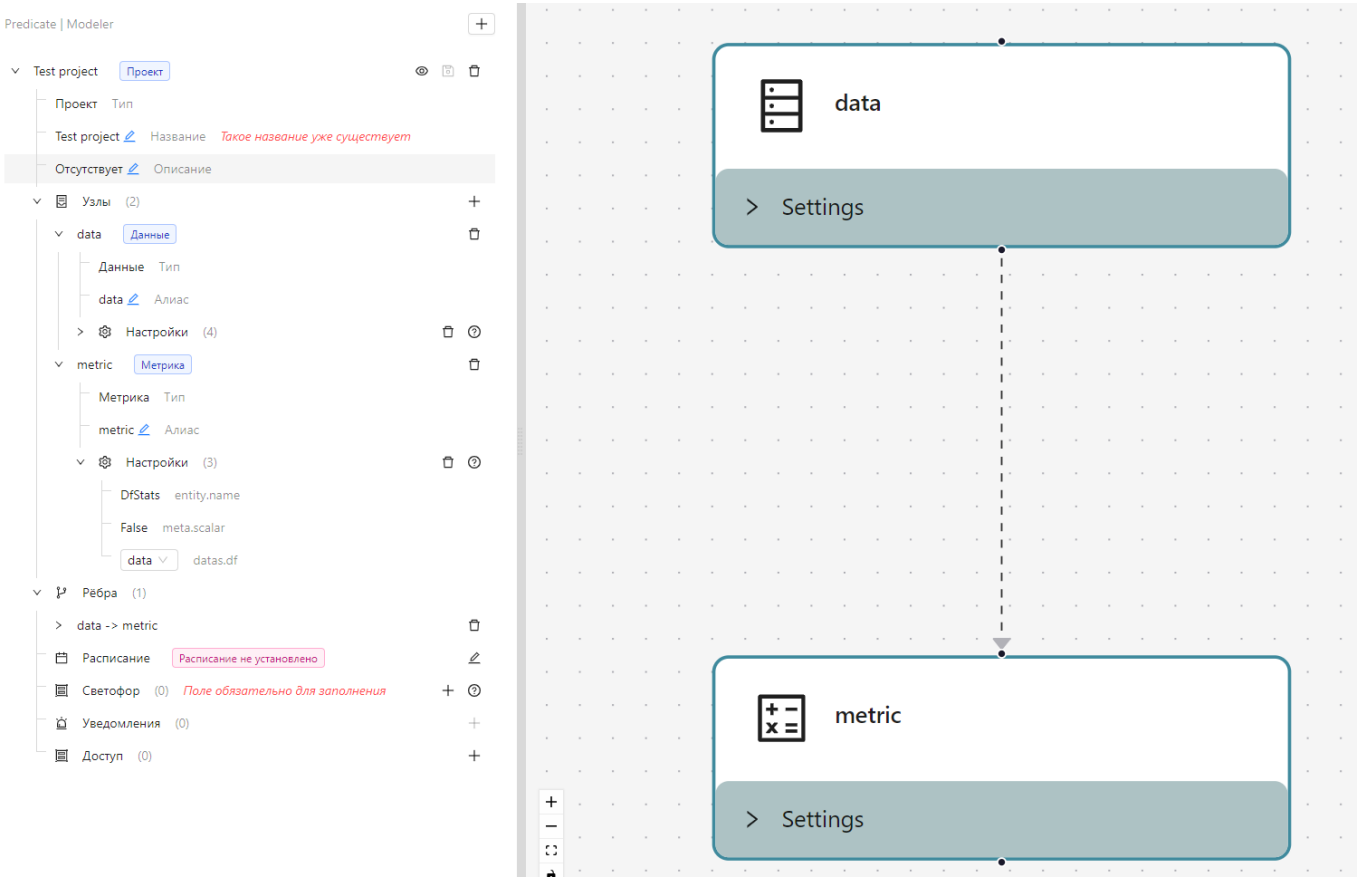
В результате откроется редактор для создания нового проекта:



Для, того, чтобы создать тестовый проект, необходимо:

1. Указать название проекта. Название проекта должно содержать только латинские буквы, цифры, знаки "дефис" и "нижнее подчеркивание".
2. В строке "Узлы" дважды нажать на "+" для добавления двух узлов - для данных и для метрики
3. Развернуть строку "Узлы".
4. Развернуть первый узел, указать в нем тип "Данные", Алиас - "data". В строке "Настройки" нажать на "+" и из каталога выбрать зарегистрированный ранее датасет.
5. Развернуть второй узел, указать в нем тип "Метрика", Алиас - "metric". В строке "Настройки" нажать на "+" и из каталога выбрать метрику "cd_2_1_Df_Stats".
6. Развернуть строку "Настройки" второго узла, заполнить параметры метрики (для метрики "cd_2_1_Df_Stats" - указать алиас первого узла для параметра "data.df").
7. В верхней строке объекта проекта нажать на значок глаза.
8. На появившейся справа диаграмме соединить узлы "data" и "metric" с помощью вытягивания линии из точки в нижней части узла "data" к точке в верхней части узла "metric".
9. В верхней строке объекта проекта нажать на значок сохранения. В появившейся форме подтвердить сохранение.

Заполненная форма проекта должна выглядеть следующим образом:

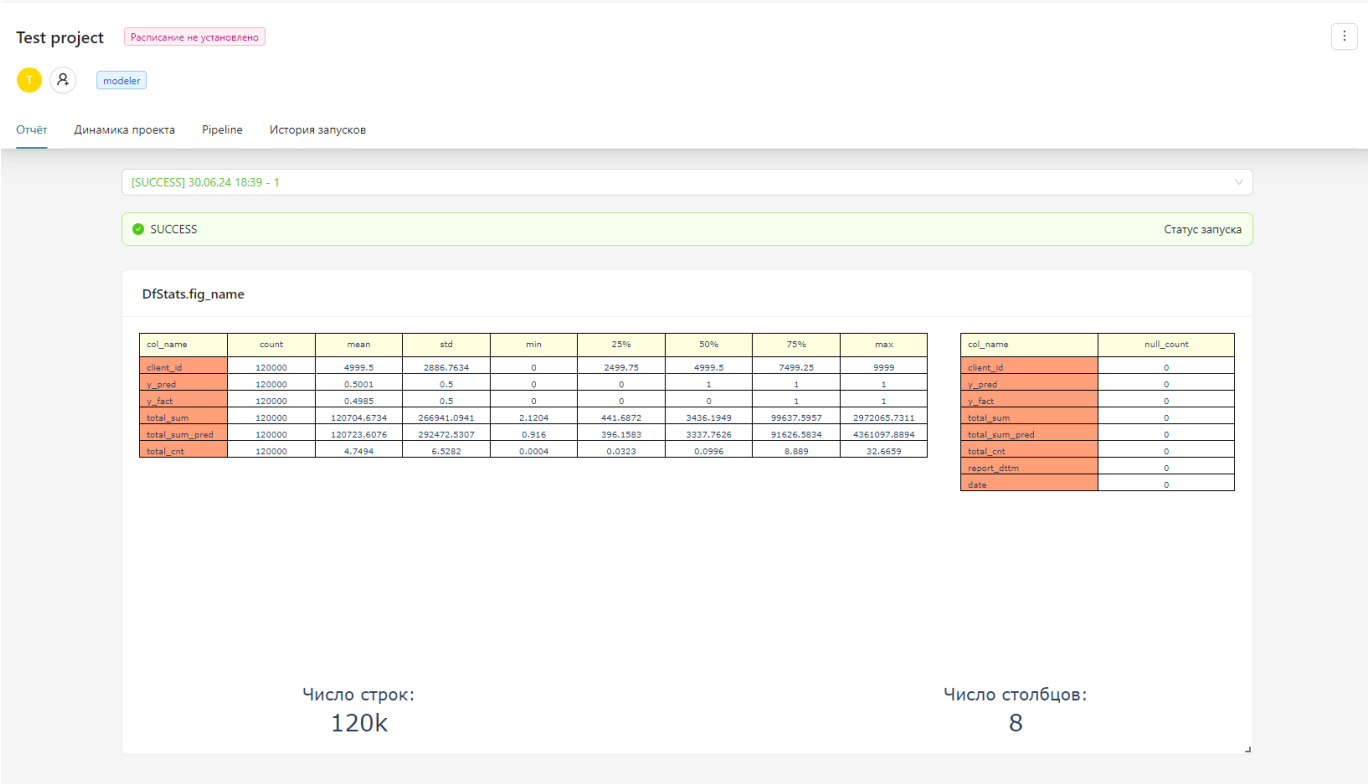


5.7 Проверка исполнения проекта

После создания проекта он должен появиться в каталоге. В него необходимо перейти двойным кликом по строке каталога.

Проект должен исполниться в течение нескольких десятков секунд (зависит от ресурсов worker-а, который был развернут при установке).

Отработавший проект будет содержать дашборд с одной метрикой. Если использовалась метрика Df_Stats, то это будет таблица со статистическими данными по датасету:



col_name	count	mean	std	min	25%	50%	75%	max
client_id	120000	4999.5	2886.7634	0	2499.75	4999.5	7499.25	9999
y_pred	120000	0.5001	0.5	0	0	1	1	1
y_fact	120000	0.4985	0.5	0	0	0	1	1
total_sum	120000	120704.6734	266941.0941	2.1204	441.6872	3436.1949	99637.5957	2972065.7311
total_sum_pred	120000	120723.6076	292472.5307	0.916	396.1583	3337.7626	91626.5834	4361097.8894
total_cnt	120000	4.7494	6.5282	0.0004	0.0323	0.0996	8.889	32.6659

col_name	null_count
client_id	0
y_pred	0
y_fact	0
total_sum	0
total_sum_pred	0
total_cnt	0
report_dttm	0
date	0

6. Установка Predicate в Yandex Cloud

6.1 Создание Managed Service for Kubernetes и его настройка

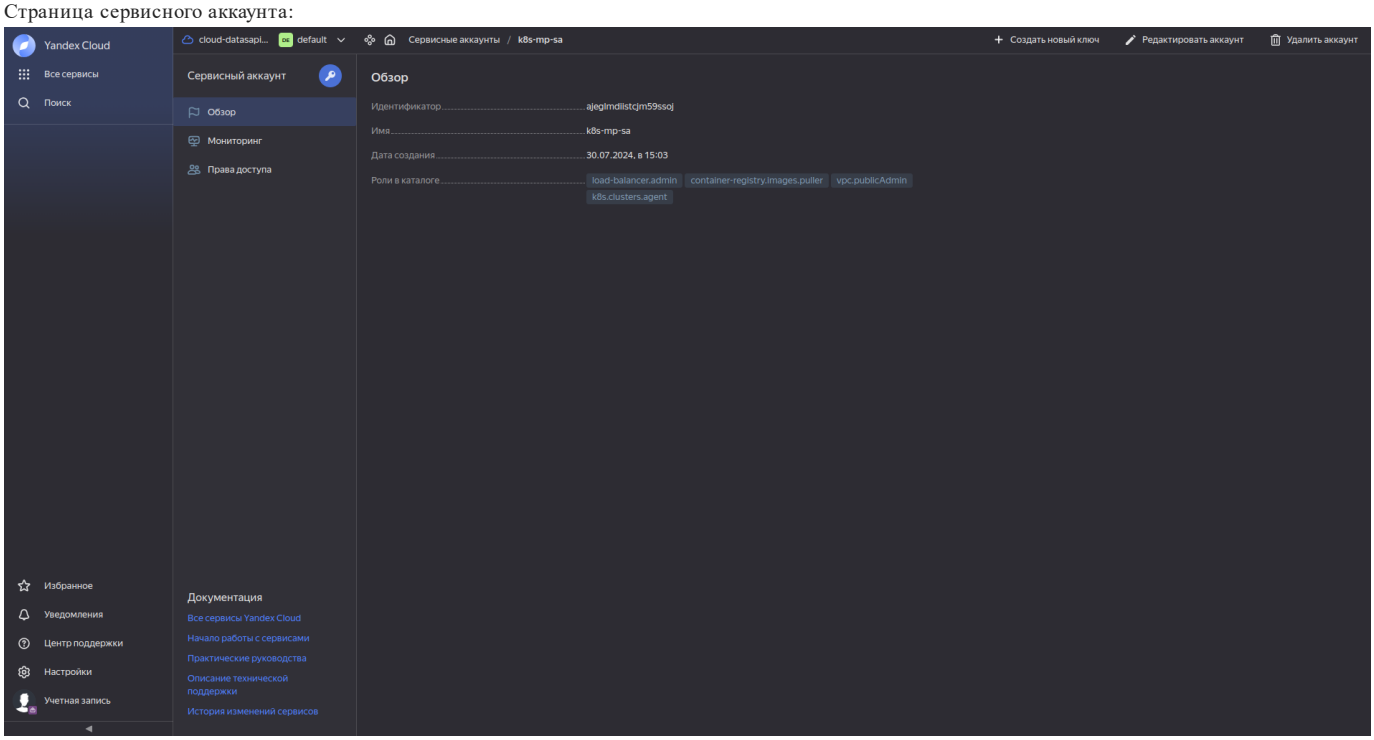
Для последующей работы с кластером, установите Yandex CLI, Helm, kubectl

Используя инструкцию создайте кластер Managed Service for Kubernetes.

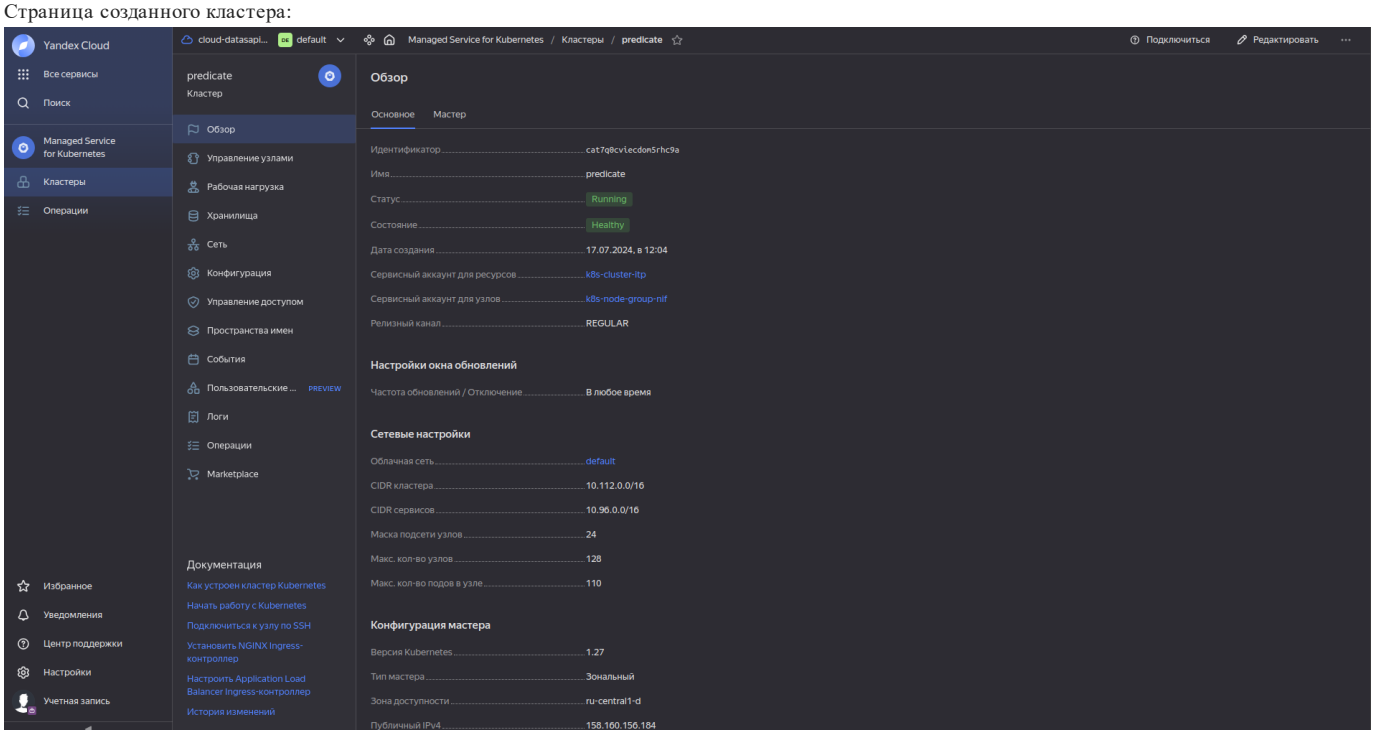
Важно

При создании сервисного аккаунта для кластера (пункт 8), необходимо дополнительно добавить роль `load-balancer.admin`.

Страница сервисного аккаунта:

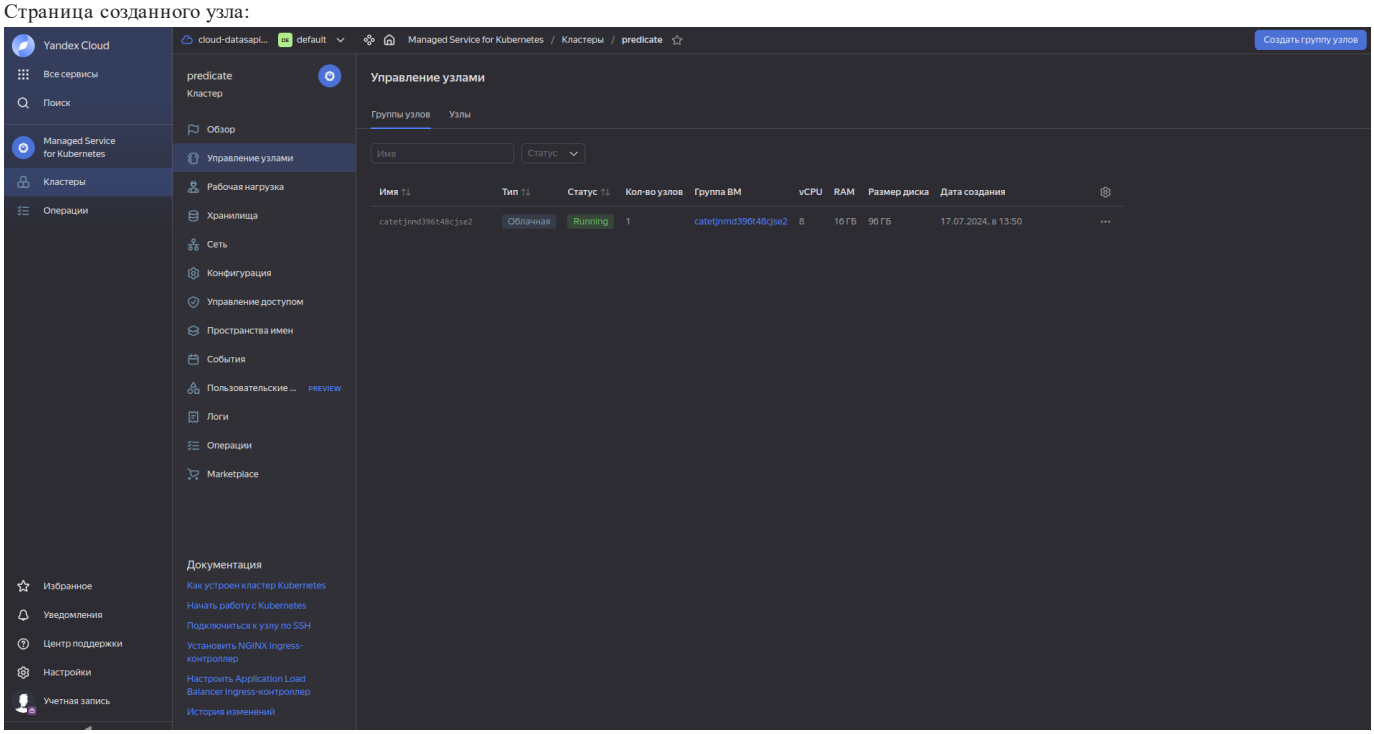


Страница созданного кластера:



После создания кластера, используя ту же инструкцию, создайте один узел с рекомендованными ресурсами.

Страница созданного узла:



Далее, выполните команду для получения учётных данных кластера, заменив `CLUSTER_NAME` на имя созданного кластера:

```
yc managed-kubernetes cluster get-credentials $CLUSTER_NAME --external
```

Чтобы узнать имя контекста кластера, выполните команду. После чего запомните имя созданного ранее контекста, которое будет использоваться в переменной `CONTEXT_NAME`:

```
kubectl config get-contexts
```

6.1.1 Установка NFS сервера

Важно

Если у вас уже есть NFS сервер, то пропустите этот шаг.

Установка будет производиться на базе провизора Ganesha. Для установки необходимо выполнить следующие шаги:

1. Добавить репозиторий

```
helm repo add nfs-ganesha-server-and-external-provisioner https://kubernetes-sigs.github.io/nfs-ganesha-server-and-external-provisioner
```

2. Установить NFS сервер, указав параметр `CONTEXT_NAME` - имя контекста кластера

```
helm upgrade --install nfs-ganesha \
nfs-ganesha-server-and-external-provisioner/nfs-server-provisioner \
--set persistence.enabled=true \
--set persistence.storageClass="yc-network-hdd" \
--set persistence.size=70Gi \
--set service.nameOverride='nfs-ganesha-svc' \
--set rbac.serviceAccount.name='nfs-ganesha-sa' \
--set fullnameOverride='nfs-ganesha' \
--namespace default \
--kube-context $CONTEXT_NAME
```

3. После установки необходимо проверить, что был создан StorageClass с именем `nfs`. Для этого выполните команду:

```
kubectl config use-context $CONTEXT_NAME
kubectl get sc
```

6.1.2 Установка Ingress Controller

Важно

Если у вас уже есть Ingress Controller, то пропустите этот шаг.

На данный момент, дефолтная поставка в YC не включает в себя поддержку Cert Manager, т.к. для его работы необходимо наличие доступа к DNS-серверу, а в поставке используется сервис nip.io для обеспечения доступа к сервисам.

Если у вас имеется собственное доменное имя, то вы можете использовать дополнительные шаги из следующей инструкции для установки Cert Manager. Для установки только Ingress Controller, выполните следующий шаг:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx && \
helm repo update && \
helm install ingress-nginx ingress-nginx/ingress-nginx \
--set controller.allowSnippetAnnotations=true \
--namespace default \
--kube-context $CONTEXT_NAME
```

Созданный контроллер будет установлен за Yandex Network Load Balancer, балансировщик создастся автоматически.

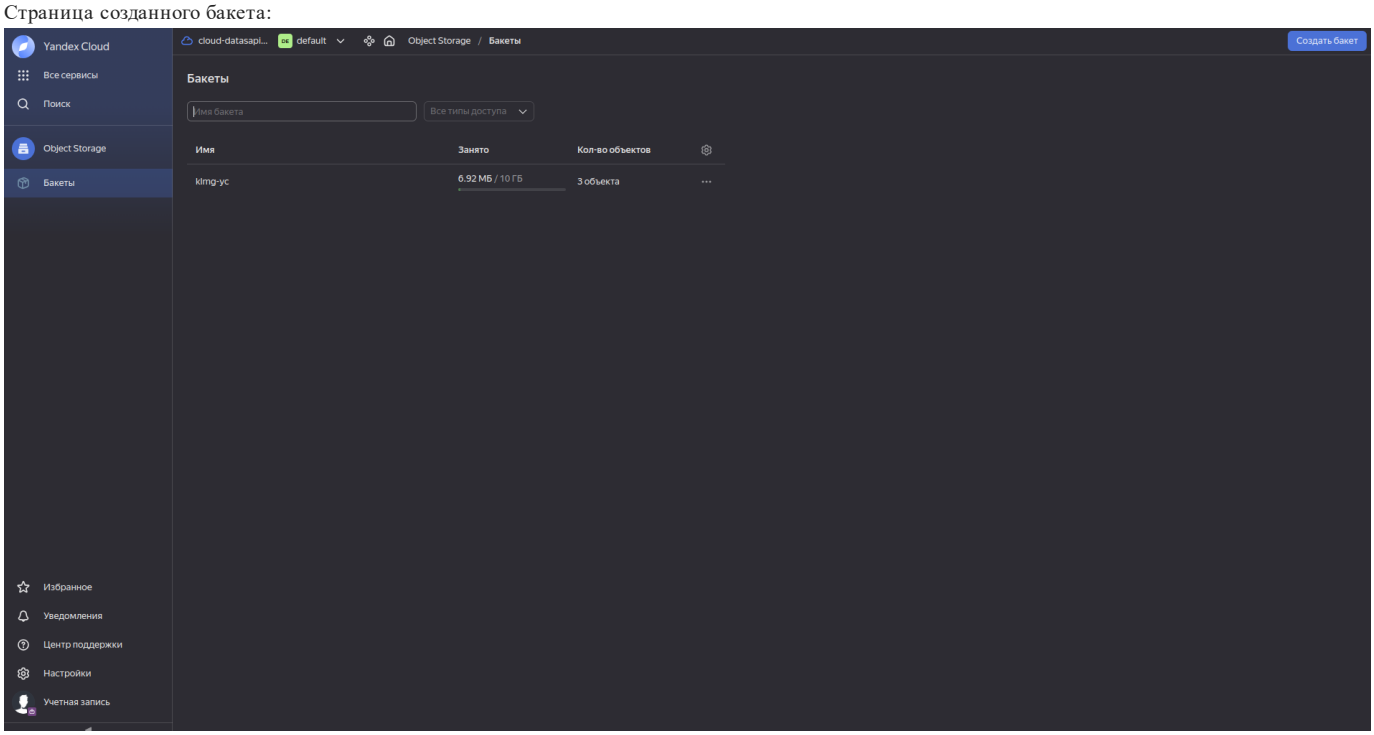
После установки Ingress Controller, необходимо узнать IP-адрес балансировщика, который будет использоваться для доступа к сервисам. Для этого выполните команду и запомните значение в поле EXTERNAL-IP:

```
kubectl config use-context $CONTEXT_NAME
kubectl get svc ingress-nginx-controller
```

6.2 Создание S3 хранилища и настройка доступа к нему

Для создания бакета в Yandex Object Storage, воспользуйтесь инструкцией. Рекомендованный размер бакета - 100Gb.

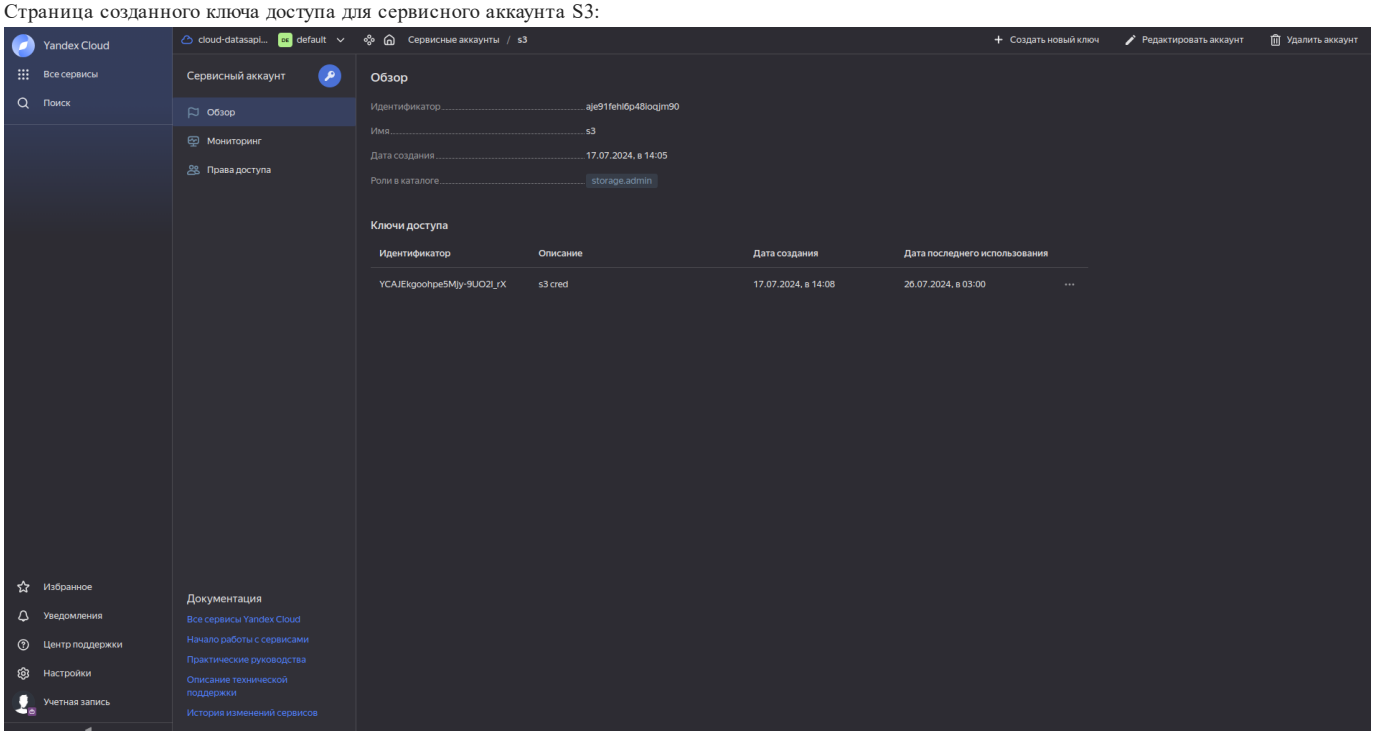
Страница созданного бакета:



Для дальнейшего доступа к бакету, создайте сервисный аккаунт с ролью `storage.editor` или `storage.admin` (GET/UPDATE/DELETE объектов). Для этого выполните следующие шаги:

1. В консоли управления выберите каталог.
2. Перейдите в раздел `Сервисные аккаунты` и нажмите `Создать сервисный аккаунт`.
3. Введите имя, добавьте роль `storage.editor` и нажмите `Создать`.
4. После создания сервисного аккаунта, нажмите на него и нажмите `Создать новый ключ -> Создать статический ключ доступа`.
5. Ведите необходимое описание (например s3-cred) и нажмите `Создать`.
6. Сохраните полученные данные (идентификатор ключа и секретный ключ) для дальнейшего использования.

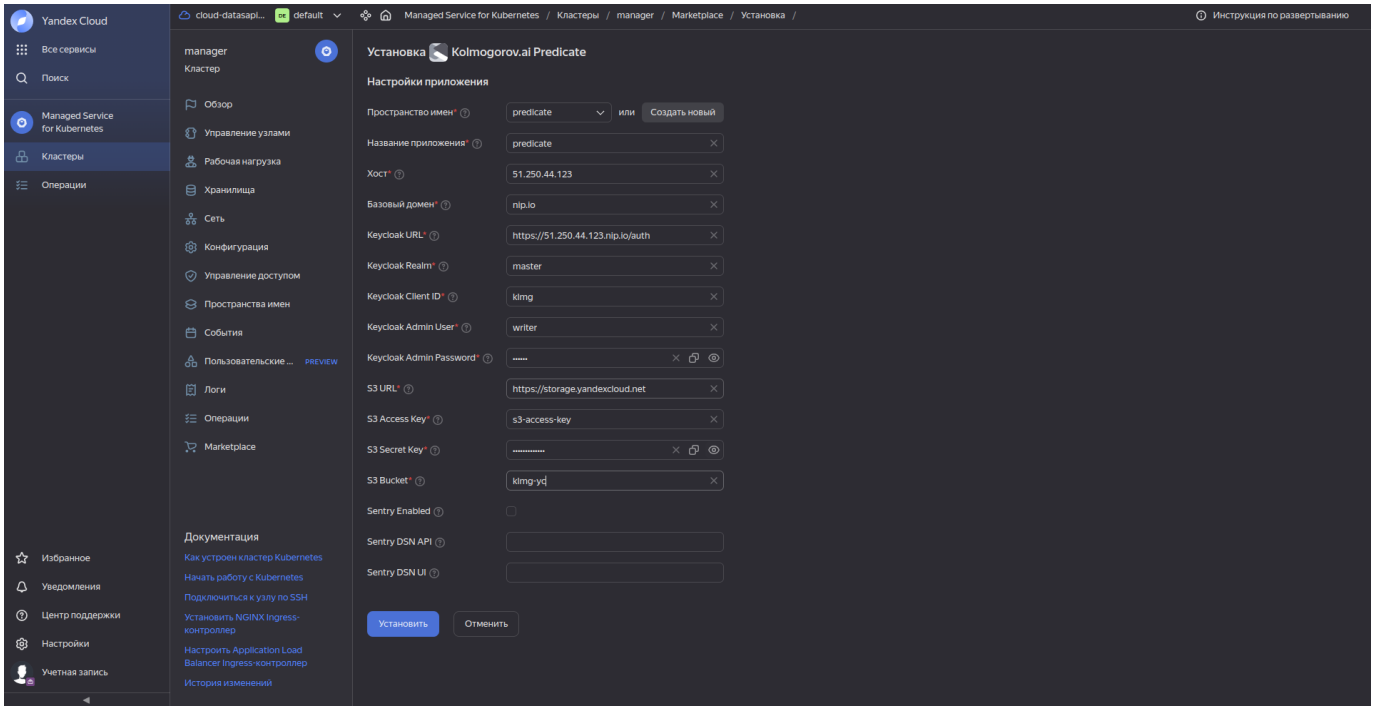
Страница созданного ключа доступа для сервисного аккаунта S3:



6.3

6.4 Установка Predicate

Важно

Перед установкой необходимо убедиться, что все предыдущие шаги были выполнены успешно.

Пример заполнения полей в форме установки приложения. Подробное описание полей можно узнать при наведении на знак вопроса рядом с названием поля: