Общество с ограниченной ответственностью «УМСКУЛ»
ОГРН 1181690104188
ИНН 1655411677

ИНСТРУКЦИЯ ПО УСТАНОВКЕ ЭКЗЕМПЛЯРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ “РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ПО ПЕРСОНАЛИЗИРОВАННОЙ ВЫДАЧЕ КОНТЕНТА И ПОСТРОЕНИЯ ИНДИВИДУАЛЬНОГО ОБРАЗОВАТЕЛЬНОГО ПУТИ”, ПРЕДОСТАВЛЕННОГО ДЛЯ ПРОВЕДЕНИЯ ЭКСПЕРТНОЙ ПРОВЕРКИ

2024

Локальный запуск (для локальной разработки)

Входные данные:
ОС: Ubuntu 22.04.3 LTS
Среда контейнеризации: Docker (+ docker-compose plugin)
БД SQL: Postgresql 13.1
БД NoSQL: Redis 7.2.3
Система обмена сообщениями: Kafka 3.2.3
Система контроля версий: Git 2.20.1

1. Для запуска приложения и необходимых компонентов требуется предварительно установить среду контейнеризации и систему контроля версий. Установка выполняется с использованием следующих команд:

```
sudo apt update
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common git

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io
sudo usermod -aG docker $USER

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose
```

2. Исходный код разработанного приложения, вместе с необходимыми конфигурационными файлами для локального запуска находится в репозитории Git репозитории Необходимо получить исходных код приложения из данного репозитория.
3. Используемый конфигурационный файл для сервиса Kafka (kafka_server_jaas.conf):

```
KafkaServer {
  org.apache.kafka.common.security.plain.PlainLoginModule required
  user_umschooluser="admin123";
};
```

4. Используемый конфигурационный файл для контейнера с разработанным сервисом:

```
# KAFKA
ACAD_PERF_KAFKA_BOOTSTRAP_SERVER=kafka://kafka:9092
ACAD_PERF_KAFKA_TOPIC=academic-performance

# LOCAL_STORAGE
ACAD_PERF_STORAGE_HOST=127.0.0.1
ACAD_PERF_STORAGE_PORT=2379

# Datasphere
ACAD_PERF_DATASPHERE_HIDDEN_STATES_LEN=256

ACAD_PERF_DATASPHERE_FOLDER_ID=12345
ACAD_PERF_DATASPHERE_NODE_ID=12345

ACAD_PERF_YANDEX_SERVICE_ACCOUNT_ID=12345
ACAD_PERF_YANDEX_KEY_ID=12345
ACAD_PERF_YANDEX_SECRET_KEY=12345
```

5. Файл сборки docker-контейнера с разработанным приложением:

```
# STAGE 1: Базовый образ
FROM docker-hub.umschool.net/umschool/umschool-python-build:3.11.4 as base

ENV PKGS_DIR=/install \
        PIP_NO_CACHE_DIR=off \
        PIP_DISABLE_PIP_VERSION_CHECK=on \
        PIP_DEFAULT_TIMEOUT=100


# STAGE 2: Образ сборки зависимостей
FROM base as builder

RUN apt update
RUN apt install -y gcc g++ python3-dev zlib1g-dev libbz2-dev liblz4-dev libzstd-dev libsnappy-dev librocksdb-dev
RUN pip install --upgrade pip
RUN pip install poetry

RUN mkdir $PKGS_DIR
RUN mkdir /code

WORKDIR /code

COPY poetry.lock pyproject.toml /code/
RUN poetry export --without lint --without-hashes -f requirements.txt --output ./requirements.txt
RUN --mount=type=secret,id=pip_extra_index_url \
        pip install --disable-pip-version-check --no-cache-dir --target=$PKGS_DIR -r
```

```
./requirements.txt --extra-index-url $(cat /run/secrets/pip_extra_index_url)

# STAGE 3: Основной образ для запуска сервиса
FROM base

ENV PYTHONPATH=/usr/local
COPY --from=builder /install /usr/local

# Копируем код сервиса
WORKDIR /app
COPY app/ .
```

6. Файл запуска приложения и необходимых компонент в среде контейнеризации (docker-compose.yml):

```
version: '3.9'

services:
  postgres:
        image: postgres:13.1
        restart: always
        environment:
        - POSTGRES_DB=aps_db
        - POSTGRES_USER=postgres
        - POSTGRES_PASSWORD=postgres
        ports:
        - "5435:5432"

  zookeeper:
        image: confluentinc/cp-zookeeper:7.3.0
        environment:
        ZOOKEEPER_CLIENT_PORT: 2181
        ZOOKEEPER_TICK_TIME: 2000

  kafka:
        image: confluentinc/cp-kafka:latest
        depends_on:
        - zookeeper
        ports:
        - "9092:9092"
        - "9093:9093"
        environment:
        KAFKA_INTER_BROKER_LISTENER_NAME: INTERNAL
        KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
INTERNAL:PLAINTEXT,PLAINTEXT:SASL_PLAINTEXT
```

```
        KAFKA_LISTENERS: INTERNAL://0.0.0.0:9092,PLAINTEXT://0.0.0.0:9093
        KAFKA_ADVERTISED_LISTENERS:
INTERNAL://kafka:9092,PLAINTEXT://localhost:9093
        ALLOW_PLAINTEXT_LISTENER: "yes"

        KAFKA_SASL_ENABLED_MECHANISMS: PLAIN,SCRAM-SHA-256,SCRAM-SHA-
512
        KAFKA_SASL_MECHANISM_INTER_BROKER_PROTOCOL: PLAIN
        KAFKA_OPTS: "-Djava.security.auth.login.config=/etc/kafka/kafka_server_jaas.conf"

        KAFKA_BROKER_ID: 1
        KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
        ZOOKEEPER_SASL_ENABLED: "false"

        KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

        KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"
        KAFKA_LOG4J_ROOT_LOGLEVEL: INFO
        KAFKA_GROUP_INITIAL_REBALANCE_DELAY_MS: 0
        KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
        KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
        KAFKA_MESSAGE_MAX_BYTES: 10485760
        KAFKA_SOCKET_REQUEST_MAX_BYTES: 100001200
        restart: always
        volumes:
        - ./kafka_server_jaas.conf:/etc/kafka/kafka_server_jaas.conf

 kafka-ui:
        image: provectuslabs/kafka-ui:latest
        ports:
        - "8080:8080"
        environment:
        KAFKA_CLUSTERS_0_NAME: "local"
        KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS: "kafka:9092"
        KAFKA_CLUSTERS_0_JMXUSERNAME: "umschooluser"
        KAFKA_CLUSTERS_0_JMXPASSWORD: "admin123"
        depends_on:
        - kafka

 etcd:
        image: 'bitnami/etcd:latest'
        restart: always
        environment:
        ETCD_ADVERTISE_CLIENT_URLS: http://etcd:2380
        ALLOW_NONE_AUTHENTICATION: yes
        ports:
```

```
        - "2379:2379"
        - "2380:2380"

  redis:
        image: redis:latest
        restart: always
        ports:
        - "6379:6379"
        environment:
        - REDIS_PORT=6379
        - REDIS_DATABASES=1

  app:
        build:
        dockerfile: Dockerfile
        secrets:
        - pip_extra_index_url
        environment:
        ACAD_PERF_LOG_LEVEL: info
        ACAD_PERF_SERVICE_PORT: 8000
        ACAD_PERF_STORAGE_HOST: redis
        ACAD_PERF_STORAGE_PORT: 6379
        ACAD_PERF_KAFKA_BOOTSTRAP_SERVER: kafka://kafka:9092
        ACAD_PERF_KAFKA_TOPIC: academic-performance
        ACAD_PERF_KAFKA_SASL_ENABLED: False
        ACAD_PERF_DB_NAME: aps_db
        ACAD_PERF_DB_USER: postgres
        ACAD_PERF_DB_PASS: postgres
        ACAD_PERF_DB_HOST: postgres
        ACAD_PERF_DB_PORT: 5432
        ports:
        - "8000:8000"
        command: >
        sh -c "alembic upgrade head &&
        python -m faust -A main worker -l $$ACAD_PERF_LOG_LEVEL -p
$$ACAD_PERF_SERVICE_PORT"
        depends_on:
        - etcd
        - kafka
        - postgres
        restart: always

secrets:
   pip_extra_index_url:
        file: ./secrets/pip_extra_index_url.txt
```

7. Далее необходимо находясь в каталоге с исходным кодом, собрать контейнер с приложением и запустить контейнер с приложением и всеми необходимыми компонентами с использованием среды контейнеризации используя следующую команду:

```
docker-compose up -d
```

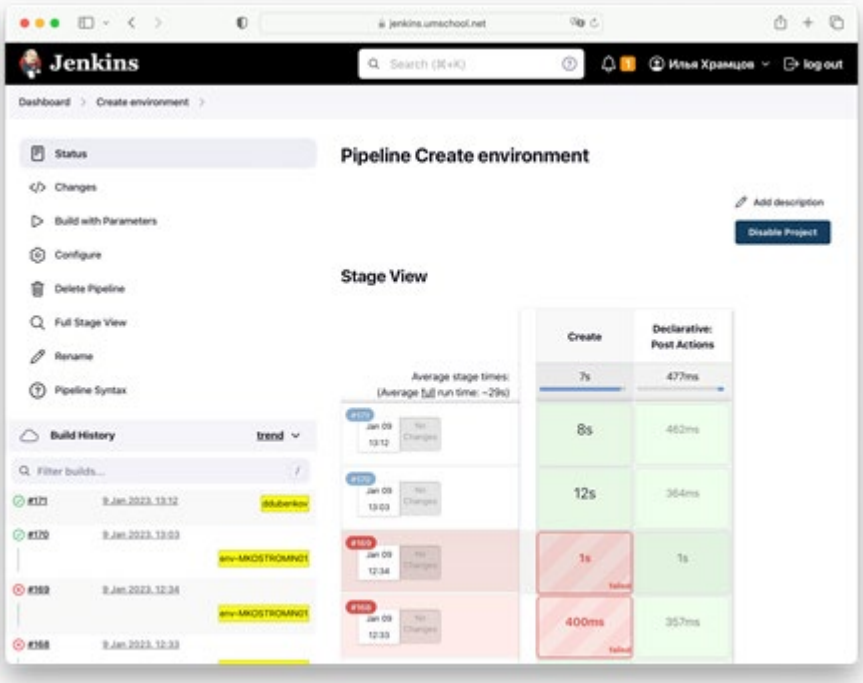
8. Проверяем работоспособность приложения пройдя по ссылке http://localhost:8000

Запуск для совместной разработки в рамках инфраструктуры ИТ-Департамента

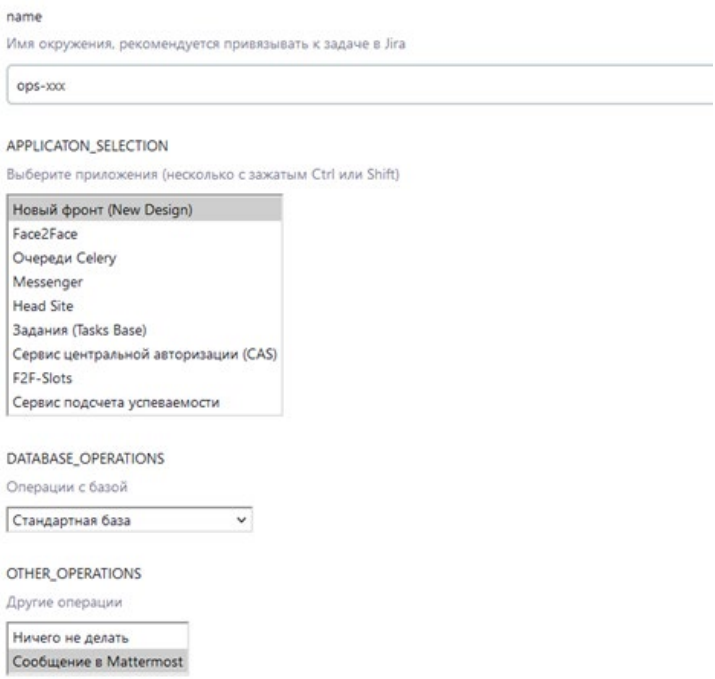
Входные данные:

Система оркестрации: Kubernetes
Среда контейнеризации: Docker (+ docker-compose plugin)
Системы CI/CD: Jenkins, ArgoCD
БД SQL: Postgresql 13.1
БД NoSQL: Redis 7.2.3
Система обмена сообщениями: Kafka 3.2.3
Система контроля версий: Git 2.20.1

1. Для создания стенда необходимо перейти в jenkins.umschool.dev и выбрать пайплайн **Create environment**



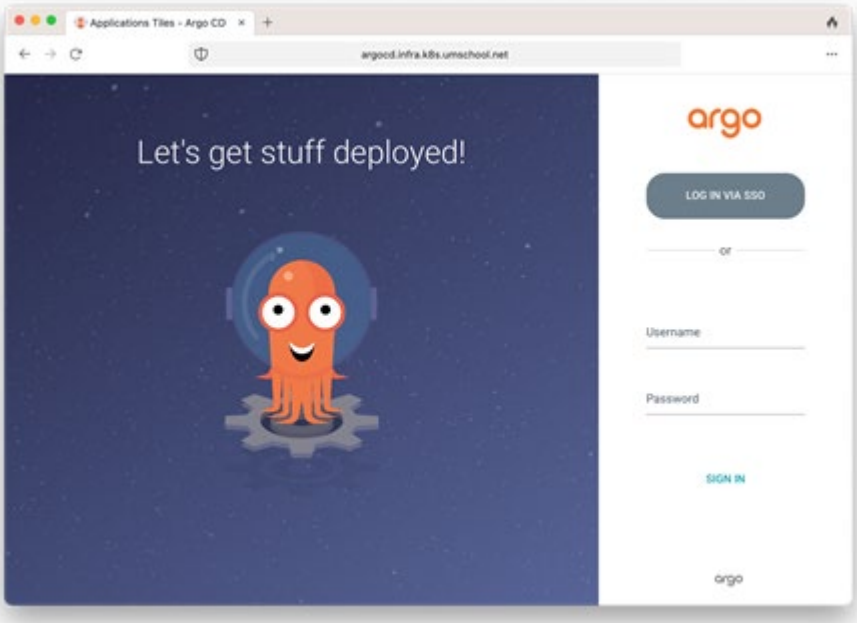
2. Нажать кнокпу "Собрать с параметрами" (Build with parameters) и ввести нужные параметры для сборки:



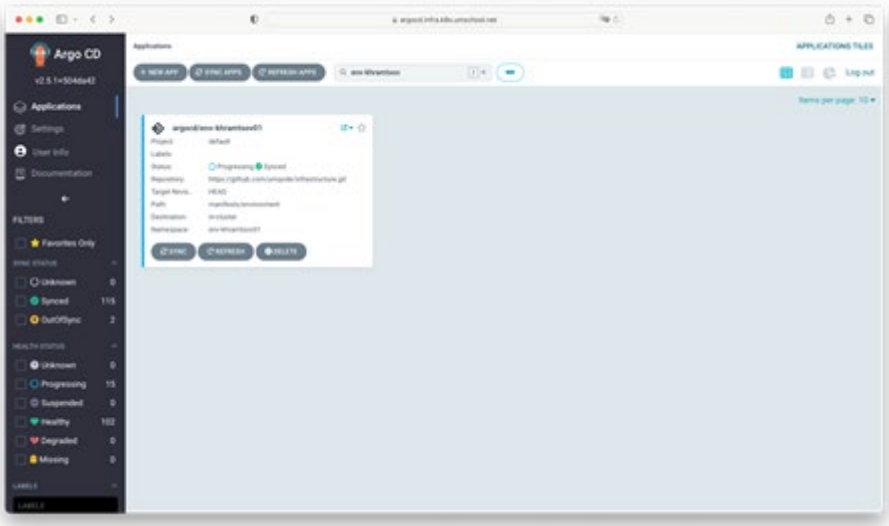
3. Описание параметров:

Имя параметра	Описание параметра
name	Необходимо указать ключ задачи в Jira (например ops-175)
APPLICATION_SELECTION	Необходимо выбрать необходимые приложения для развёртывания
DATABASE_OPERATIONS	Стандартная база (standart_db) - База восстанавливается из дампа текущей версии release стенда Перезалить базу (recreate_db) - Дропается база, затем заливается из дампа release Чистая база (clean_db) - Дропается база, и применяются только миграции для сервисов. Данных нет. Не проводить операций с базой (donttouch_db) - Не проводятся никакие операции с базой. Стенд не удалятся при запуске джобы. **ВАЖНО:** Если ранее создавался уже стенд с таким же именем, то в режиме standart_db данные останутся старыми.
OTHER_OPERATIONS	Ничего не делать Отправить сообщение в Mattermost.
cas_version	Версия CAS (не ветка, а именно версия)
acadperf_version	Версия Academic performance (не ветка, а именно версия)
TTL (Time to Live)	количество дней после которого созданный стенд будет удалён

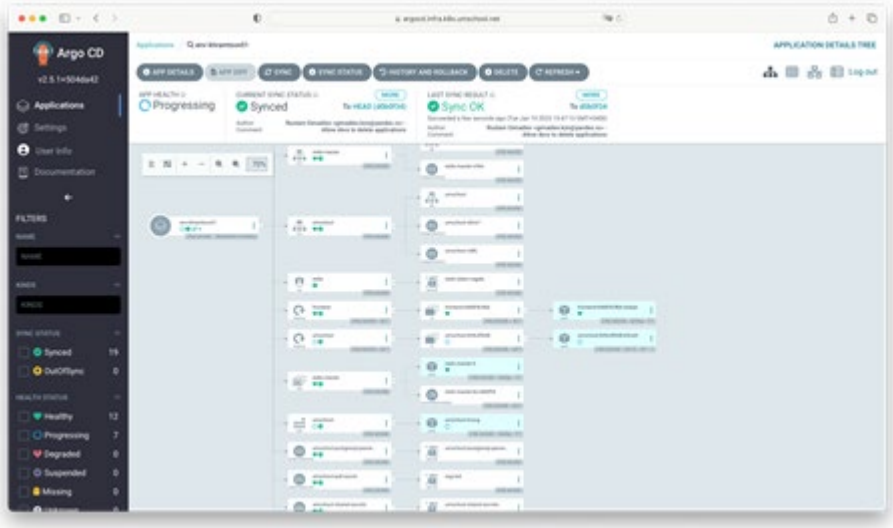
4. После запуска происходит установка стенда и первоначальная настройка, собирается статика и делаются миграции для базы данных, что по времени занимает 5-10 минут.
5. Переходим в argocd https://argocd.infra.k8s.umschool.net/applications



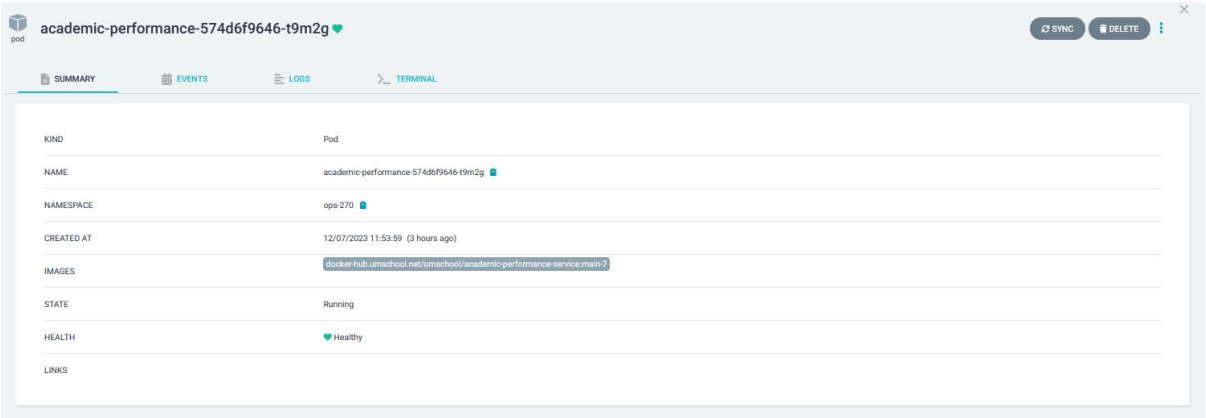
6. Нажимаем **LOG IN VIA SSO и входим под учетной записью LDAP**
7. В поиске вводим названия своего **стенда** и входим в него



8. Находим Под серсиса с именем **academic-performance** и переходим в него



9. В данном меню доступны вкладки Logs и Terminal в которых можно соответственно посмотреть логи приложения, а также попасть в консоль управления контейнером с приложением.



10. Проверить работоспособность сервиса можно с использованием ссылки формата https://acadperf-XXX.k8s.umschool.net/ , где XXX - это наименование стенда, указанное в пункте 2.