

ё

G-MD2-1 «Платформа Атом.РИТА»   
Руководство по развертыванию

На 25 листах

Дата: 13.04.2022

Версия: v.02

Лист изменений

| Дата | Версия | Краткое описание изменений | ФИО |
| --- | --- | --- | --- |
| 01.04.2022 | v.01 | Документ создан | Поляков А.В. |
| 13.04.2022 | v.02 | Внесены дополнения | Матвеев Е.С. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Содержание

[1. Состав платформы 4](#_Toc100752313)

[2. Файлы, передаваемые заказчику 5](#_Toc100752314)

[3. Требования к окружению 6](#_Toc100752315)

[4. Развертывание платформы 7](#_Toc100752316)

[4.1. Развертывание модуля Атом.Рита Разработчик 7](#_Toc100752317)

[4.1.1. Docker-compose.yml 7](#_Toc100752318)

[4.1.2. Настройка .env 10](#_Toc100752319)

[4.1.3. Настройка PostgreSQL 10](#_Toc100752320)

[4.1.4. Подготовка файловой структуры. 11](#_Toc100752321)

[4.1.5. Запуск контейнеров 11](#_Toc100752322)

[4.1.6. Настройка авторизации через Active Directory 12](#_Toc100752323)

[4.1.7. Импорт файла справки 12](#_Toc100752324)

[4.2. Развертывание модуля Атом.Рита Администратор 13](#_Toc100752325)

[4.2.1. Docker-compose.yml 13](#_Toc100752326)

[4.2.2. Настройка .env 17](#_Toc100752327)

[4.2.3. Настройка PostgreSQL 18](#_Toc100752328)

[4.2.4. Настройка Rabbitmq 19](#_Toc100752329)

[4.2.5. Запуск контейнеров. 19](#_Toc100752330)

[4.2.6. Настройка авторизации через Active Directory 20](#_Toc100752331)

[4.3. Установка и настройка модулей сервиса РИТА.Голос 21](#_Toc100752332)

[4.3.1. Docker-compose.yml 21](#_Toc100752333)

[4.3.1. Настройка .env 24](#_Toc100752334)

[4.3.2. Запуск контейнеров. 24](#_Toc100752335)

[4.4. Установка и настройка модулей сервиса РИТА.Классификатор 25](#_Toc100752336)

[4.4.1. Docker-compose.yml 25](#_Toc100752337)

[4.4.2. Запуск контейнеров 26](#_Toc100752338)

# Состав платформы

Платформа Атом.РИТА предназначена для разработки и управления программными роботами, позволяющими автоматизировать бизнес-процессы путем воспроизведения действий пользователей в программах, установленных на рабочих станциях.

Платформа состоит из пяти модулей (см. рисунок 1):

* РИТА.Разработчик – модуль, предназначенный для создания, хранения, изменения и отладки программных роботов, реализованный как веб-приложение.
* РИТА.Администратор – модуль, предназначенный для автоматического запуска и управления роботами, созданными в модуле РИТА.Разработчик, а также мониторинга их состояния и аудита работы, реализованный как веб-приложение.
* РИТА.Сотрудник – модуль, предназначенный для выполнения программного робота, устанавливаемый на рабочую станцию.
* РИТА.Классификатор - модуль, предназначенный для автоматического определения классов (признаков, характеристик) текстовых объектов. Модуль возможно обучить для работы с различными текстами и определения выбранных пользователем классов (от 1 до 5).
* РИТА.Голос – модуль, предназначенный для распознавания речи и ее перевода в текст, а также синтеза речи на основании заданного текста.



Рисунок 1. Функциональная структура платформы Атом.РИТА

# Файлы, передаваемые заказчику

Платформа Атом.РИТА распространяется в виде набора собранных docker-образов (cерверная часть) и установочных файлов агента (РИТА.Сотрудник). Для передачи заказчику образы экспортированы в архивы. Состав компонентов платформы представлен в таблице ниже.

|  |  |  |
| --- | --- | --- |
| Модуль | Архив | Содержит |
| Сотрудник | rita\_worker.exe |  |
| Разработчик | rita\_razrab.tar.gz | ga\_rete |
| ga\_backend\_studio |
| ga\_backend\_app\_trace\_logs |
| ga\_authorize |
| studio-xxx.tar | файл справки |
| ga\_chrome\_web\_inspector\_v2.x.x.zip | плагин для браузера Google Chrome |
| Администратор | rita\_admin.tar.gz | ga\_frontend\_orchestrator |
| ga\_backend\_orchestrator |
| ga\_authorize |
| ga\_file\_sensor |
| ga\_timer |
| ga\_backend\_app\_trace\_logs |
| Голос | rita-golos-images-22.04.tgz | greenatom-rita-golos-api |
| greenatom-rita-golos-proxy |
| greenatom-rita-golos-redis |
| greenatom-rita-golos-worker |
| greenatom-rita-golos-callback |
| greenatom-rita-golos-cli |
| greenatom-rita-golos-web |
| rita-golos-models-22.04.tgz | файлы моделей |
| Классификатор | rita\_class.tar.gz | ga\_ritaclassifier:v2 |
| train\_classifier.csv | пример/семпл датасета |

# Требования к окружению

Целевая версия ОС: Linux Astra 1.6

Тестирование платформы выполнялось со следующими версиями ПО:

[ user ~]$ uname -a

Linux 1tgrp-s-grp01.novalocal 3.10.0-1127.el7.x86\_64 #1 SMP Tue Mar 31 23:36:51 UTC 2020 x86\_64 x86\_64 x86\_64 GNU/Linux

[user ~]$ docker --version

Docker version 20.10.7, build f0df350

[user ~]$ docker-compose --version

docker-compose version 1.29.2, build 5becea4c

Целевая версия СУБД: Postgres Pro Enterprise Certified

Для корректной работы через HTTPS/WSS требуется наличие PKI инфраструктуры и выпуск TLS/SSL сертификатов для доменов, которые планируется выделить под модули РИТА.Разработчик и РИТА.Администратор.

# Развертывание платформы

Все примеры развертывания платформы приведены для docker-compose.

## Развертывание модуля Атом.Рита Разработчик

Модуль Атом.Рита Разработчик имеет микросервисную архитектуру. Компоненты модуля распространяются в виде docker образов.

В модуле используются следующие образы:

|  |  |
| --- | --- |
| Имя | Описание |
| ga\_rete | Фронтенд-приложение РИТА.Разработчик. В качестве базового образа используется [nginx:alpine](https://hub.docker.com/_/nginx), обрабатывающий SSL-соединения и раздающий статику - бандл приложения. |
| ga\_backend\_studio | Бэкенд РИТА.Разработчик. Осуществляет маршрутизацию сообщений между браузером пользователя и агент-хостом, выполняет операции шифрования/дешифрования роботов. |
| ga\_backend\_app\_trace\_logs | Подробное логирование событий бэкенда |
| ga\_authorize | Сервис авторизации через AD |
| Minio\* | Хранение файлов справки |
| БД\* | Если планируется использовать СУБД в контейнере (не рекомендуется), то можно эмулировать Postgres Pro свободной версией pg 11. |

\* - образы не входят в передаваемый архив, являются свободно распространяемыми и доступны в docker registry.

Для загрузки образа докер образа postgres можно использовать

*docker pull postgres:11-alpine*

а для загрузки образа minio:

*docker pull minio/minio:latest*

Для импорта образов из переданного архива необходимо выполнить следующую команду:

*zcat rita\_studio.tar.gz | docker load*

## Docker-compose.yml

Пример файла docker-compose

|  |
| --- |
| *version: '3'*  *services:*  *ga\_rete:*  *image: ga\_rete:master*  *restart: unless-stopped*  *depends\_on:*  *- ga\_backend\_studio*  *volumes:*  *- ./default-frontend.conf:/etc/nginx/conf.d/default.conf*  *- ./cert:/etc/nginx/certs*  *- ./shared:/usr/src/app/files*  *ports:*  *- 80:3080*  *- 443:3443*  *networks:*  *- ga\_studio*  *ga\_backend\_studio:*  *image: ga\_backend\_studio:master*  *restart: unless-stopped*  *env\_file:*  *- .env*  *volumes:*  *- ./minio:/minio*  *networks:*  *- ga\_studio*  *ga\_backend\_app\_trace\_logs\_studio:*  *image: ga\_backend\_app\_trace\_logs:master*  *restart: unless-stopped*  *depends\_on:*  *- ga\_backend\_studio*  *env\_file:*  *- .env*  *networks:*  *- ga\_studio*  *ga\_authorize\_studio:*  *image: ga\_authorize:master*  *restart: unless-stopped*  *env\_file:*  *- .env*  *networks:*  *- ga\_studio*  *ga\_minio:*  *restart: unless-stopped*  *image: minio/minio:latest*  *env\_file:*  *- .env*  *volumes:*  *- ./minio:/data*  *networks:*  *- ga\_studio*  *ports:*  *- 9002:9000*  *command: server /data*  *ga\_backend\_pg:*  *restart: unless-stopped*  *image: postgres:11-alpine*  *env\_file:*  *- .env*  *networks:*  *- ga\_studio*  *volumes:*  *- ./pg-data:/var/lib/postgresql/data*  *networks:*  *ga\_studio:* |

## Настройка .env

Для подключения бэкенда к СУБД необходимо задать переменные POSTGRES\_\*, и сформировать новые значения для секретов.

* PG\_SECRET - 16 символов, используется для генерации доступов к БД
* JWT\_SECRET - ключ (32 символа) для шифровки/дешифровки авторизационного токена jwt
* ACCOUNTS\_SECRET - ключ (32 символа) для шифровки/дешифровки учетных записей из accounts
* APP\_PARAMS\_SECRET - ключ (32 символа) для шифровки/дешифровки параметров приложения из app\_parameter (настройки приложения)
* MINIO\_ROOT\_USER и MINIO\_ROOT\_PASSWORD - данные для авторизации в minio (хранение файлов справки)
* SECRET - указать лицензию (без этого РИТА.Разработчик не запустится)

Пример файла .env

|  |
| --- |
| *POSTGRES\_DB=test*  *POSTGRES\_HOST=ga\_backend\_pg*  *POSTGRES\_CONN\_PORT=5432*  *POSTGRES\_PASSWORD=change-me-test-pass*  *POSTGRES\_USER=test-user*  *PG\_SECRET=Abc12345dEfgh678*  *JWT\_SECRET=Abc12345dEfgh678Abc12345dEfgh678*  *ACCOUNTS\_SECRET=Abc12345dEfgh678Abc12345dEfgh678*  *APP\_PARAMS\_SECRET=Abc12345dEfgh678Abc12345dEfgh678*  *MINIO\_ROOT\_USER=minio\_user*  *MINIO\_ROOT\_PASSWORD=minio-user-password*  *SECRET=* |

## Настройка PostgreSQL

При использовании отдельного инстанса базы данных, если владелец не имеет прав SUPERUSER, то минимально необходимо предоставить следующие роли и включить расширения в БД:

ALTER ROLE rpaback WITH Create role

ALTER ROLE rpaback WITH Bypass RLS

CREATE EXTENSION pgcrypto;

CREATE EXTENSION uuid-ossp;

CREATE EXTENSION pg\_trgm;

## Подготовка файловой структуры.

В корневой директории проекта (место, где располагаются docker-compose.yml и .env) подготовить файловую структуру:

minio/

shared/

ga\_chrome\_web\_inspector\_v2.x.x.zip # Плагин google-chrome для web-инспектора

rita\_worker.exe # Установщик агента

cert/

ssl.conf # подключаемый конфиг nginx для работы по SSL, положить файлы сертификатов рядом

Пример файла ssl.conf (заменить STUDIO\_HOST\_SERVERNAME на имя сертификата)

|  |
| --- |
| *ssl\_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';*  *ssl\_protocols TLSv1.2 TLSv1.3;*  *ssl\_prefer\_server\_ciphers on;*  *add\_header Strict-Transport-Security "max-age=31536000; includeSubdomains; preload";*  *add\_header X-Frame-Options SAMEORIGIN;*  *add\_header X-XSS-Protection "1; mode=block";*  *add\_header X-Content-Type-Options nosniff;*  *add\_header Referrer-Policy "no-referrer";*  *ssl\_certificate /etc/nginx/certs/STUDIO\_HOST\_SERVERNAME.crt;*  *ssl\_certificate\_key /etc/nginx/certs/STUDIO\_HOST\_SERVERNAME.key;* |

## Запуск контейнеров

В качестве основы можно использовать приведённый в пункте 4.1.1 docker-compose.yml. Скорректировать пути до docker-образов в соответствии с загруженными архивами.

* Запустить docker-compose up -d, проверить логи на наличие ошибок.
* Дождаться выполнения всех миграций\* в backend-сервисах
* Открыть в браузере адрес РИТА.Разработчик и авторизоваться под q\_studioadmin - пароль любой.
* Проверить консоль браузера на наличие ошибок

\* Пример записи лога, сообщающей о выполнении миграции БД:

== 20220328103025-add-RITA\_ENV: migrated (14.170s)

## Настройка авторизации через Active Directory

Нужно перейти в Параметры приложения и указать данные для параметров с префиксом LDAP\_\*. Пример настроек для gk:

LDAP\_DN=OU=Rosatom,DC=gk,DC=rosatom,DC=local

LDAP\_SEARCH\_USER\_LOGIN=tech\_ad\_user

LDAP\_SEARCH\_USER\_PASS=pass

LDAP\_URL=ldaps://ldaps.gk.rosatom.local:636

LDAP\_USER\_CANONICAL\_DOMAIN=gk.rosatom.local

Затем запретить авторизацию без AD, для этого в параметре SKIP\_AD\_AUTH нужно указать **false**. Параметр настраивается в «Параметрах приложения» на странице «Администрирование»

## Импорт файла справки

Открыть в браузере адрес РИТА.Разработчик и авторизоваться под q\_editor - пароль любой. Перейти в раздел «Справка». Загрузить файл справки, выбрав вариант «Импортировать справку из архива».

## Развертывание модуля Атом.Рита Администратор

В модуле используются следующие образы:

|  |  |
| --- | --- |
| Имя | Описание |
| ga\_file\_sensor | Файловый сенсор, инициирует запуск роботов по факту событий на файловой системе |
| ga\_timer | Инициирует запуск роботов по cron-расписанию или интервалу |
| ga\_backend\_app\_trace\_logs | Подробное логгирование событий бэкенда |
| ga\_authorize | Сервис авторизации через AD |
| ga\_backend\_orchestrator | Бэкенд РИТА.Администратор, выполняет распределение заданий на выполнение роботов зарегистрированными агент-хостами. |
| ga\_frontend\_orchestrator | Фронтенд-приложение РИТА.Администратор. В качестве базового образа используется [nginx:alpine](https://hub.docker.com/_/nginx), обрабатывающий SSL-соединения и раздающий статику - бандл приложения. |
| Rabbitmq\* | Очередь сообщений для обмена данными между РИТА.Администратор и агент-хостами |
| Minio\* | Хранение файлов справки |
| БД\* | Если планируется использовать СУБД в контейнере (не рекомендуется), то можно эмулировать Postgres Pro свободной версией pg. |

\* - образы не входят в передаваемый архив, являются свободно распространяемыми и доступны в docker registry.

Для загрузки образа докер образа postgres можно использовать

*docker pull postgres:11-alpine*

Для загрузки образа minio

*docker pull minio/minio:latest*

Для загрузки образа rabbitmq:

*docker pull rabbitmq:management-alpine*

## Docker-compose.yml

|  |
| --- |
| *version: '3'*  *services:*  *ga\_frontend\_orchestrator:*  *image: ga\_frontend\_orchestrator:master*  *ports:*  *- 8081:3081*  *restart: unless-stopped*  *depends\_on:*  *- ga\_backend\_orchestrator*  *volumes:*  *- ./default-frontend.conf:/etc/nginx/conf.d/default.conf*  *networks:*  *- ga\_orchestrator*  *ga\_pg\_orchestrator:*  *restart: unless-stopped*  *image: postgres:11-alpine*  *ports:*  *- 2345:5432*  *env\_file:*  *- .env*  *volumes:*  *- ./ga\_pg\_data:/var/lib/postgresql/data*  *- /data/orchestrator/pg:/backup*  *networks:*  *- ga\_orchestrator*  *ga\_backend\_orchestrator:*  *image: ga\_backend\_orchestrator:master*  *restart: unless-stopped*  *depends\_on:*  *- ga\_pg\_orchestrator*  *env\_file:*  *- .env*  *ports:*  *- 3053:3053*  *networks:*  *- ga\_orchestrator*  *ga\_rabbitmq:*  *image: rabbitmq:3-management-alpine*  *restart: unless-stopped*  *ports:*  *- 5671:5671*  *- 5672:5672*  *- 15672:15672*  *networks:*  *- ga\_orchestrator*  *ga\_minio:*  *restart: unless-stopped*  *image: minio/minio:latest*  *env\_file:*  *- .env*  *volumes:*  *- ./minio:/data*  *networks:*  *- ga\_orchestrator*  *ports:*  *- 9003:9000*  *command: server /data*  *ga\_timer:*  *image: ga\_timer:development*  *restart: unless-stopped*  *command: 'python src/main.py'*  *env\_file:*  *- .env*  *networks:*  *- ga\_orchestrator*  *depends\_on:*  *- ga\_rabbitmq*  *- ga\_backend\_orchestrator*  *ga\_file\_sensor:*  *image: ga\_file\_sensor:development*  *restart: unless-stopped*  *command: 'python src/main.py'*  *env\_file:*  *- .env*  *networks:*  *- ga\_orchestrator*  *depends\_on:*  *- ga\_rabbitmq*  *- ga\_backend\_orchestrator*  *ga\_backend\_app\_trace\_logs\_orchestrator:*  *image: ga\_backend\_app\_trace\_logs:master*  *restart: unless-stopped*  *depends\_on:*  *- ga\_pg\_orchestrator*  *env\_file:*  *- .env*  *networks:*  *- ga\_orchestrator*  *ga\_authorize\_orchestrator:*  *image: ga\_authorize:master*  *restart: unless-stopped*  *depends\_on:*  *- ga\_pg\_orchestrator*  *env\_file:*  *- .env*  *networks:*  *- ga\_orchestrator*  *networks:*  *ga\_orchestrator:* |

## Настройка .env

Для подключения бэка к СУБД необходимо задать переменные POSTGRES\_\*, и сгенерировать новые значения для секретов:

* PG\_SECRET - 16 символов, используется для генерации доступов к БД
* JWT\_SECRET - ключ (32 символа) для шифровки/дешифровки авторизационного токена jwt
* ACCOUNTS\_SECRET - ключ (32 символа) для шифровки/дешифровки учетных записей из accounts
* APP\_PARAMS\_SECRET - ключ (32 символа) для шифровки/дешифровки параметров приложения из app\_parameter (настройки приложения)
* NODE\_TLS\_REJECT\_UNAUTHORIZED - игнорировать самоподписаный сертификат в ldaps/https.
* MINIO\_ROOT\_USER и MINIO\_ROOT\_PASSWORD - данные для авторизации в minio (хранение файлов справки)
* MOLECULER\_NAMESPACE - указывается studio если РИТА.Разработчик либо orchestrator если РИТА.Администратор
* RABBITMQ\_\* - указать данные для подключения модуля РИТА.Администратор к RabbitMQ
* BACKEND\_BASE\_URL - указать url РИТА.Администратор, например: http://ga\_backend\_orchestrator:3053
* INSTANCE\_ID - номер инстанса, для каждого ga\_timer/ga\_file/ga\_email нужно указывать номер от 1. Например если есть два ga\_timer, то у первого указать INSTANCE\_ID=1, а у второго INSTANCE\_ID=2. После запуска РИТА.Администратор нужно будет перейти в Параметры приложения (смотри п.3.6 документа «РИТА.Администратор. Руководство администратора») и указать число инстансов:

SENSOR\_FILE\_COUNT – число инстансов ga\_file

SENSOR\_TIMER\_COUNT – число инстансов ga\_timer

SENSOR\_EMAIL\_COUNT – число инстансов ga\_email

Пример файла .env

|  |
| --- |
| *POSTGRES\_DB=orchestrator*  *POSTGRES\_HOST=ga\_pg\_orchestrator*  *POSTGRES\_CONN\_PORT=5432*  *POSTGRES\_PASSWORD=orchestrator-pass*  *POSTGRES\_USER=orchestrator*  *PORT=3053*  *PG\_SECRET=Abc12345dEfgh678*  *JWT\_SECRET=Cba54321dEfgh876Abc54321dEfgh876*  *DOMAIN\_CREDENTIAL\_SECRET=Cba54321dEfgh876Abc54321dEfgh876*  *APP\_PARAMS\_SECRET=Cba54321dEfgh876Abc54321dEfgh876*  *TZ=Europe/Moscow*  *NODE\_TLS\_REJECT\_UNAUTHORIZED=0*  *MOLECULER\_NAMESPACE=orchestrator*  *COMPOSE\_IGNORE\_ORPHANS=True*  *MINIO\_ROOT\_USER=minio\_user*  *MINIO\_ROOT\_PASSWORD=minio-user-password*  *RABBITMQ\_HOST=ga\_rabbitmq*  *RABBITMQ\_PORT=5672*  *RABBITMQ\_USER=guest*  *RABBITMQ\_PASSWORD=guest*  *BACKEND\_BASE\_URL=http://ga\_backend\_orchestrator:3053*  *USER\_API=ga\_orchestrator*  *PASSWORD\_API=1* |

## Настройка PostgreSQL

При использовании отдельного инстанса базы данных, если владелец не имеет прав SUPERUSER, то минимально необходимо предоставить следующие роли и включить расширения в БД:

ALTER ROLE rpaback WITH Create role

ALTER ROLE rpaback WITH Bypass RLS

CREATE EXTENSION pgcrypto;

CREATE EXTENSION uuid-ossp;

CREATE EXTENSION pg\_trgm;

## Настройка Rabbitmq

При использовании отдельного инстанса очереди сообщений RabbitMQ,

необходимо добавить пользователя на сервере:

*rabbitmq-plugins enable rabbitmq\_management*

*rabbitmqctl list\_users*

*rabbitmqctl add\_user orc <пароль для пользователя оркестратора>*

*rabbitmqctl set\_user\_tags orc administrator*

*rabbitmqctl set\_permissions -p / orc ".\*" ".\*"* ".\*"

и скорректировать параметры RABBITMQ\_\* в файле .env.

## Запуск контейнеров.

В качестве основы можно использовать приведённый в пункте 4.2.1 docker-compose.yml.

* Запустить docker-compose up -d, проверить логи на наличие ошибок.
* Дождаться выполнения всех миграций в backend-сервисах
* Открыть в браузере адрес оркестратора и авторизоваться под q\_appadmin - пароль любой.
* Проверить консоль браузера на наличие ошибок.

## Настройка авторизации через Active Directory

Нужно перейти в Параметры приложения и указать данные для параметров с префиксом LDAP\_\*. Пример настроек для gk:

LDAP\_DN=OU=Rosatom,DC=gk,DC=rosatom,DC=local

LDAP\_SEARCH\_USER\_LOGIN=tech\_ad\_user

LDAP\_SEARCH\_USER\_PASS=pass

LDAP\_URL=ldaps://ldaps.gk.rosatom.local:636

LDAP\_USER\_CANONICAL\_DOMAIN=gk.rosatom.local

затем запретить авторизацию без AD, для этого в параметре SKIP\_AD\_AUTH нужно указать false

## Установка и настройка модулей сервиса РИТА.Голос

В состав сервиса РИТА.Голос входят следующие модули, каждый из которых содержится в отдельном Docker-образе:

| Модуль | Имя образа | Описание |
| --- | --- | --- |
| proxy | greenatom-rita-golos-proxy | Обратный реверс-прокси, распределяющий входящие HTTP-запросы по другим модулям сервиса |
| api | greenatom-rita-golos-api | Веб-сервер, принимающий и обрабатывающий API запросы по протоколу HTTP |
| redis | greenatom-rita-golos-redis | Оперативное хранилище, брокер сообщений |
| worker | greenatom-rita-golos-worker | Обработчик заданий, который непосредственно занимается преобразованиями ASR и TTS |
| callback | greenatom-rita-golos-callback | Микросервис обратного вызова, предназначенный для обработки отложенных результатов |
| cli | greenatom-rita-golos-cli | Интерфейс командной строки |
| web | greenatom-rita-golos-web | Веб-приложение для доступа к API сервиса через протокол HTTP |

Для непосредственной установки модулей сервиса РИТА.Голос необходимо последовательно выполнить команды:

docker load –-input=./releases/release-2204/greenatom-rita-golos-api-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-worker-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-web-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-cli-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-proxy-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-redis-0.0.5.tar

docker load –-input=./releases/release-2204/greenatom-rita-golos-callback-0.0.5.tar

## Docker-compose.yml

|  |
| --- |
| *version: "2.4"*  *services:*  *redis:*  *image: greenatom-rita-golos-redis:latest*  *restart: always*  *worker:*  *image: greenatom-rita-golos-worker:latest*  *restart: always*  *scale: 1*  *volumes:*  *- /data/rita.golos/data/audio:/data/audio*  *- /data/rita.golos/data/models:/data/models*  *env\_file:*  *- .env*  *command: celery -A main worker --concurrency 4 --loglevel=INFO*  *depends\_on:*  *- redis*  *api:*  *image: greenatom-rita-golos-api:latest*  *restart: always*  *scale: 1*  *env\_file:*  *- .env*  *ports:*  *- 8000*  *command: uvicorn --host 0.0.0.0 --port 8000 main:app*  *labels:*  *reproxy.enabled: true*  *reproxy.server: '\*'*  *reproxy.route: '^/api/v1/(.\*)'*  *reproxy.dest: '/$$1'*  *reproxy.port: 8000*  *depends\_on:*  *- redis*  *- worker*  *proxy:*  *image: greenatom-rita-golos-proxy:latest*  *restart: always*  *ports:*  *- "8888:8080"*  *volumes:*  *- /var/run/docker.sock:/var/run/docker.sock:ro*  *environment:*  *- TZ=Europe/Moscow*  *- LISTEN=0.0.0.0:8080*  *- MAX\_SIZE=64M*  *- DOCKER\_ENABLED=true*  *- DOCKER\_AUTO=false*  *- DEBUG=true*  *- HEADER=*  *X-Frame-Options:SAMEORIGIN,*  *X-XSS-Protection:1; mode=block;,*  *Content-Security-Policy:default-src 'self'; style-src 'self' 'unsafe-inline';*  *depends\_on:*  *- api*  *web:*  *image: greenatom-rita-golos-web:latest*  *restart: always*  *ports:*  *- 80:8501*  *command: streamlit run /opt/app/main.py*  *environment:*  *- API\_BASE\_URL=http://proxy:8080*  *depends\_on:*  *- proxy*  *callback:*  *image: greenatom-rita-golos-callback:latest*  *restart: always*  *environment:*  *- STORAGE\_URL=redis://redis:6379/3*  *ports:*  *- 8800*  *command: uvicorn --host 0.0.0.0 --port 8800 main:app*  *labels:*  *reproxy.enabled: true*  *reproxy.server: '\*'*  *reproxy.route: '^/data/(.\*)'*  *reproxy.dest: '/data/$$1'* |

## Настройка .env

|  |
| --- |
| *CELERY\_NAME=rita-golos*  *CELERY\_BROKER\_URL=redis://redis:6379/0*  *CELERY\_RESULT\_BACKEND\_URL=redis://redis:6379/1* |

## Запуск контейнеров.

Перед запуском модулей сервиса РИТА.Голос необходимо установить в каталог /data/rita.golos/data обученные модели распознавания и синтеза речи. Сами модели находятся в файле *rita-golos-models.tgz*. Для извлечения моделей из архива необходимо в каталоге /data/rita.golos выполнить команду tar -xf rita-golos-models.tgz –-directory=./data.

Для запуска модулей сервиса РИТА.Голос необходимо запустить команду docker-compose -f docker/docker-compose.yml up -d. Убедиться, что модули сервиса успешно запущены и работают корректно можно с помощью команды docker-compose -f docker/docker-compose.yml ps. Для просмотра журнала необходимо выполнить команду docker-compose -f docker/docker-compose.yml logs –-follow.

## Установка и настройка модулей сервиса РИТА.Классификатор

В модуле используются следующие образы:

| Модуль | Имя образа | Описание |
| --- | --- | --- |
| Классификатор | ga\_ritaclassifier:v2 |  |

## Docker-compose.yml

Пример docker-compose файла.

|  |
| --- |
| *version: '3.3'*  *services:*  *predictorr:*  *restart: always*  *image: ga\_ritaclassifier:v2*  *privileged: True*  *volumes:*  *- ./:/app/services/ml\_data*  *- ./ml\_data/ga.ml.services/logs/:/app/logs*  *ports:*  *- "80:8000"*  *environment:*  *APP\_LANDSCAPE\_ID: 'TEST'*  *PATH\_TO\_CONFIG: ''*  *PATH\_TO\_MODELS: '/app/services/ml\_data/'*  *APP\_LOG\_PATH: '/app/logs/'*  *APP\_LOG\_SIZE\_MB: '10'*  *APP\_LOG\_BACKUP\_COUNT: '100'*  *PREDICT\_COUNT: '5'*  *labels:*  *- "com.ouroboros.enable=true"*  *networks:*  *main:*  *aliases:*  *- predictor1*  *networks:*  *main:* |

## Запуск контейнеров

В качестве основы можно использовать приведённый в пункте 4.4.1 docker-compose.yml. Скорректировать пути до docker-образов в соответствии с загруженными архивами.

* Запустить docker-compose up -d, проверить логи на наличие ошибок.

Для запуска контейнера без ошибок, рядом с docker-compose.yml необходимо положить передаваемый с архивом файл - train\_classifier.csv