Руководство администратора Платформы речевая аналитика

Оглавление

[1 Введение 3](#_Toc95183212)

[2 Оборудование и программное обеспечение 3](#_Toc95183213)

[2.1 Перечень программного обеспечения 3](#_Toc95183214)

[2.2 Создание нового кластера ПРА 3](#_Toc95183215)

[2.3 Добавление рабочих нод ПРА 3](#_Toc95183216)

[2.4 Настройка нового кластера ПРА 4](#_Toc95183217)

[2.5 Запуск кластера ПРА 4](#_Toc95183218)

[2.6 Перезапуск и обновление контейнера 4](#_Toc95183219)

[2.7 Масштабирование контейнеров 5](#_Toc95183220)

[3 Основная конфигурация 5](#_Toc95183221)

[3.1 Конфигурационный файл 5](#_Toc95183222)

[4 Программное обеспечение WEB 13](#_Toc95183223)

[4.1 Конфигурационный файл 13](#_Toc95183224)

[5 Программное обеспечение App 14](#_Toc95183225)

[5.1 конфигурационный файл 14](#_Toc95183226)

[6 Программное обеспечение SA 16](#_Toc95183227)

[6.1 Конфигурационный файл 16](#_Toc95183228)

[7 Программное обеспечение ASR PROXY 17](#_Toc95183229)

[7.1 Конфигурационный файл 18](#_Toc95183230)

[8 Мониторинг 19](#_Toc95183231)

[8.1 Управление нагрузкой 19](#_Toc95183232)

# Введение

Данное руководство описывает основные сценарии по управлению **Платформой Речевая аналитика** (далее веб-интерфейс).

# Оборудование и программное обеспечение

ПРА использует программное обеспечение Docker для управления компонентами Системы.

В качестве основного управляющего оркестратора используется программное обеспечение Docker Swarm в режиме кластера.

Для обеспечения минимальной отказоустойчивости требуется:

Сервер WEB приложений в количестве 2 шт.

Сервер App приложений и Речевого анализа в количестве 2 шт.

Сервер управления в количестве 1 шт.

## Перечень программного обеспечения

Docker v.1.19.03 ПО должно быть установлено на всех серверах.

Nginx v.1.16.1

Postgres v.12.2

MongoDB v.4.2.3

Программное обеспечение ПРА поставляется в виде архива с конфигурациями и образами контейнеров.

## Создание нового кластера ПРА

Для создания кластера на сервере управления необходимо выполнить команду инициализации:

docker swarm init

После успешного выполнения команды появится команда для добавления рабочих нод.

docker swarm join –token XXXXXXX

Для добавления контролирующей ноды выполните команду:

docker swarm join-token manager

После успешного выполнения команды появится команда для добавления управляющих нод.

docker swarm join –token XXXXXXX

Выполните команды добавления управляющих нод на серверах приложений:

На всех остальных серверах выполните команды по добавлению рабочих нод.

## Добавление рабочих нод ПРА

Для разметки автоматического размещения контейнеров необходимо использовать метки:

Для нод sa использовать метку sa со значением true.

Для нод sa\_high высоко производительных использовать метку sa\_high со значением true.

Для нод хранилища minio использовать метку minio со значением true.

На мастер сервере docker swarm или ноде необходимо выполнить команду:

docker node update --label-add sa=true p6z0gpr70moncmrn3y0ly4o8q

docker node update --label-add sa\_high=true p6z0gpr70moncmrn3y0ly4o8q

docker node update --label-add minio=true p6z0gpr70moncmrn3y0ly4o8q

, где sa=true – метка со значением, p6z0gpr70moncmrn3y0ly4o8q – идентификатор хоста на котором будут применены параметры.

Идентификатор хоста можно получить с помощью команды docker node ls.

## Настройка нового кластера ПРА

Для первичной настройки на каждый сервер необходимо скопировать архив с параметрами и образами контейнеров по пути /opt/Stack/

Структура папок /opt/Stack/ содержит:

/app/ - директория хранения данных по APP приложению

/app/api\_log/ - директория хранения лог файлов

/app/api\_master/ - конфигурационный файл Основного сервера приложений

/app/api\_slave/ - конфигурационный файл Резервного сервера приложений

/minio/data/ - данные встроенного файлового хранилища

/mongo/conf/ - данные конфигурации СУБД MongoDB

/mongo/data/ - данные СУБД MongoDB

/nginx/ - конфигурационные файлы nginx прокси для обработки внутренних взаимодействий

/pg/conf/ - данные конфигурации СУБД PostgreSQL

/pg/data/ - данные СУБД PostgreSQL

/sa/sa\_conf/ - конфигурационные файлы серверов речевого анализа

/sa/sa\_log/ - директория хранения лог файлов сервера речевого анализа

/web/web\_log/ - директория хранения лог файлов web сервера

/web/web\_conf/ - конфигурационные файлы веб серверов.

/asr\_proxy/ - конфигурационные файлы балансировщика нагрузки на asr сервера.

## Запуск кластера ПРА

Для запуска необходимо выполнить команду:

docker stack deploy --compose-file /opt/Stack/docker-compose.yaml sa\_app

где /opt/Stack/docker-compose.yaml – путь к конфигурационному файлу структуры приложений.

sa\_app – наименование приложений в кластере.

## Перезапуск и обновление контейнера

Для обновления контейнера необходимо загрузить архив нового контейнера в файловую систему необходимого сервера, и выполнить команду

docker load < /opt/app/Images/rtk\_sa\_v102.tar ,

где /opt/app/Images/rtk\_sa\_v102.tar – архив с контейнеров системы.

Данную команду необходимо выполнить на всех серверах с обновляемым компонентом

Для перезапуска контейнера необходимо:

Получить ID контейнера в кластере, выполнив команду:

docker service ls

Затем выполнить команду:

docker service update --force wc9ld294nis9, где

wc9ld294nis9 – ID необходимого контейнера полученный на прошлом шаге.

## Масштабирование контейнеров

Для масштабирования контейнеров используется команда docker service scale.

docker service scale sgk2qock80hs=14

где sgk2qock80hs – идентификатор сервиса.

14 – количество запущенных экземпляров приложения.

Идентификатор сервиса можно получить используя команду docker service ls.

# Основная конфигурация

Основная конфигурационная структура описана в формате docker compose YAML, версии 3.4.

## Конфигурационный файл

version: '3.8'

services:

nginx:

image: nginx:latest

hostname: nginx

networks:

- sa\_network

volumes:

- /opt/app/Stack/nginx/nginx.conf:/etc/nginx/nginx.conf # путь к конфигурации nginx

- /opt/app/Stack/nginx/logs/:/var/log/nginx/ # путь к логам nginx

ports:

- target: 80 # порт src

published: 8000 # порт dest

protocol: tcp

mode: host

deploy:

mode: global

restart\_policy:

condition: on-failure

redis:

image: redis

hostname: redis

networks:

- sa\_network

depends\_on:

- web1

- web2

ports:

- target: 6379

published: 6379

protocol: tcp

mode: host

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

nats:

image: nats

hostname: nats

networks:

- sa\_network

ports:

- target: 4222

published: 4222

protocol: tcp

mode: host

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints: [node.hostname == p0a8i82z0ansp03.unix.vats.local]

web1:

image: rtk\_web

hostname: web1

networks:

- sa\_network

volumes:

- /opt/app/Stack/web/web\_conf/conf.yaml:/app/conf/conf.yaml # порт к конфигурации web

- /opt/app/Stack/web/web1\_log/:/app/log/ # порт к логам web1

ports:

- target: 8080

published: 8080

protocol: tcp

mode: host

healthcheck:

test: curl -sS http://127.0.0.1:8080 || echo 1

interval: 5s

timeout: 10s

retries: 3

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints: [node.hostname == p0a8i41ansp01.unix.local] # указание запуска на определенном сервере

web2:

image: rtk\_web

hostname: web2

networks:

- sa\_network

volumes:

- /opt/app/Stack/web/web\_conf/conf.yaml:/app/conf/conf.yaml # порт к конфигурации web

- /opt/app/Stack/web/web2\_log/:/app/log/ # порт к логам web2

ports:

- target: 8080

published: 8081

protocol: tcp

mode: host

healthcheck:

test: curl -sS http://127.0.0.1:8080 || echo 1

interval: 5s

timeout: 10s

retries: 3

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints: [node.hostname == p0a8i41ansp02.unix.local] # указание запуска на определенном сервере

postgres:

image: postgres

hostname: postgres

networks:

- sa\_network

environment:

# POSTGRES\_PASSWORD: postgres

POSTGRES\_HOST\_AUTH\_METHOD: trust # изменить при открытом доступе

volumes:

- /opt/app/Stack/pg/data/:/var/lib/postgresql/data/

ports:

- target: 5432

published: 5432

protocol: tcp

mode: host

healthcheck:

test: ["CMD-SHELL", "pg\_isready -U postgres"]

interval: 10s

timeout: 5s

retries: 5

start\_period: 15s

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints: [node.hostname == p0a8i41ansp03.unix.local] # указание запуска на определенном сервере

mongo:

image: mongo

hostname: mongo

# environment:

# MONGO\_INITDB\_ROOT\_USERNAME: root

# MONGO\_INITDB\_ROOT\_PASSWORD: example

volumes:

- /opt/Stack/mongo/data/:/data/db

networks:

- sa\_network

ports:

- target: 27017

published: 27017

protocol: tcp

mode: host

healthcheck:

# test: ["CMD", "mongo --eval db.adminCommand('ping')"]

# interval: 10s

# timeout: 5s

# retries: 5

test: echo 'db.runCommand("ping").ok' | mongo mongo:27017/test --quiet

interval: 10s

timeout: 10s

retries: 5

start\_period: 40s

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints: [node.hostname == p0a8i41ansp03.unix.local] # указание запуска на определенном сервере

minio:

image: minio/minio

hostname: minio

networks:

- sa\_network

environment:

MINIO\_ACCESS\_KEY: hd78347gdncw09823hd2x5134

MINIO\_SECRET\_KEY: xm34gaxkfgaw8m232x3rmh8f9f

volumes:

- /ansp\_01/Stack/minio/data/:/data # монтируем с NFS хранилища

ports:

- target: 9000

published: 9000

protocol: tcp

mode: host

command: server /data

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]

interval: 30s

timeout: 20s

retries: 3

deploy:

replicas: 2

restart\_policy:

condition: on-failure

placement:

constraints:

- node.labels.minio == true # указание запуска на группе серверов с меткой minio

app1:

image: raapi\_dev

hostname: app

environment:

POSTGRES\_DB: rtk\_ra

networks:

- sa\_network

volumes:

- /opt/Stack/app/api\_master/:/app/conf/ # порт к конфигурации app

- /opt/Stack/app/api\_log/:/app/log/ # порт к логам app1

- /var/lib/docker/sftp/:/app/upload # порт к sftp папке получения аудио данных от Svetets vPBX

ports:

- target: 8675

published: 8675

protocol: tcp

mode: host

healthcheck:

test: curl -sS http://localhost:8675/check || echo 1

interval: 5s

timeout: 10s

retries: 3

depends\_on:

- postgres

deploy:

replicas: 1

restart\_policy:

condition: on-failure

delay: 5s

max\_attempts: 5

window: 20s

placement:

constraints: [node.hostname == p0a8i41ansp03.unix.local] # указание запуска на определенном сервере

app2:

image: raapi\_dev

hostname: app2

environment:

POSTGRES\_DB: rtk\_ra

networks:

- sa\_network

volumes:

- /opt/Stack/app/api\_slave/:/app/conf/ # порт к конфигурации app

- /opt/Stack/app/api2\_log/:/app/log/ # порт к логам app2

- /var/lib/docker/sftp/:/app/upload

ports:

- target: 8675

published: 8676

protocol: tcp

mode: host

healthcheck:

test: curl -sS http://localhost:8675/check || echo 1

interval: 5s

timeout: 10s

retries: 3

depends\_on:

- postgres

deploy:

replicas: 1

restart\_policy:

condition: on-failure

delay: 5s

max\_attempts: 5

window: 20s

placement:

constraints: [node.hostname == p0a8i41ansp04.unix.local] # указание запуска на определенном сервере

sa:

image: rtk\_sa

hostname: sa

networks:

- sa\_network

volumes:

- /opt/app/Stack/sa/sa\_conf/conf.yaml:/speech-analytics/conf/conf.yaml # путь к конфигурации sa

- /opt/app/Stack/sa/sa\_log/:/speech-analytics/log/ путь к логам sa

depends\_on:

- app1

deploy:

replicas: 4 # максимальное количество реплик

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 2 # максимальное количество реплик на одну ноду

constraints:

- node.labels.sa == true

sa\_high:

image: rtk\_sa

hostname: sa\_high

networks:

- sa\_network

volumes:

- /opt/app/Stack/sa/sa\_conf/conf.yaml:/speech-analytics/conf/conf.yaml путь к конфигурации sa (конфигурация sa и sa\_high отличаются)

- /opt/app/Stack/sa/sa\_log/:/speech-analytics/log/ путь к логам sa

depends\_on:

- app1

deploy:

replicas: 12

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 6 # максимальное количество реплик на одну ноду

constraints:

- node.labels.sa\_high == true# указание запуска на сервере с меткой sa\_high=true

asr1:

image: ruasr

hostname: asr1

networks:

- sa\_network

ports:

- target: 2700

published: 2700

protocol: tcp

mode: host

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 1

constraints: [node.hostname == p0a8i82z0ansp03.unix.vats.local] # указание запуска на определенном сервере

asr2:

image: ruasr

hostname: asr2

networks:

- sa\_network

ports:

- target: 2700

published: 2700

protocol: tcp

mode: host

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 1

constraints: [node.hostname == p0a8i82z0ansp04.unix.vats.local] # указание запуска на определенном сервере

asr3:

image: ruasr

hostname: asr2

networks:

- sa\_network

ports:

- target: 2700

published: 2700

protocol: tcp

mode: host

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 1

constraints: [node.hostname == p0a8i82z0ansp06.unix.vats.local] # указание запуска на определенном сервере

asr4:

image: ruasr

hostname: asr2

networks:

- sa\_network

ports:

- target: 2700

published: 2700

protocol: tcp

mode: host

deploy:

replicas: 1

restart\_policy:

condition: on-failure

placement:

max\_replicas\_per\_node: 1

constraints: [node.hostname == p0a8i82z0ansp07.unix.vats.local] # указание запуска на определенном сервере

asr\_proxy:

image: nginx:latest

hostname: asrproxy

networks:

- sa\_network

volumes:

- /opt/app/Stack/asr\_proxy/nginx.conf:/etc/nginx/nginx.conf # путь к конфигурации asr\_proxy

ports:

- target: 2600

published: 2600

protocol: tcp

mode: host

deploy:

mode: global

restart\_policy:

condition: on-failure

networks:

sa\_network:

driver: overlay

attachable: true

# Программное обеспечение WEB

Доступ к контейнерам происходит путем проксирования запросов от WEB сервера NGINX к контейнерам с WEB приложениями.

## Конфигурационный файл

*# Zone check for SSL: prod with SSL, test without SSL, rtk – iframe rtk*zone: rtk – режим работы шифрования (для продуктивной среды РТК с iframe - rtk)port: 8080 – порт запуска web приложений  
  
*#Manual Files upload "Отображение раздела Загрузить файлы", 0 – не отображать, 1 - отображать*manual\_upload: 0

*# Настройки доступа к хранению сессий пользователей*  
redis\_ip: redisredis\_port: 6379  
redis\_login: login  
redis\_password: password  
redis\_db: 0

*# Настройки доступа к APP серверам приложений*api\_ip: nginx  
api\_port: 8675

*# Настройки доступа к контейнеру хранения аудио контента*minio\_host: minio:9000  
minio\_access\_key: hd78347gdncw09823hd2x5134  
minio\_secret\_key: xm34gaxkfgaw8m232x3rmh8f9f  
minio\_files\_bucket: speech-files  
minio\_temp\_files\_bucket: temp-files  
minio\_web\_files: web  
minio\_secure: 0  
  
default\_mode: 1 - режим работы языковой модели (всегда 1)  
  
external\_content: 1 – использование СХДП для получения аудио контента  
external\_content\_ip: IP адрес WEB сервера c которого отправляется запрос  
external\_content\_nginx: IP адрес WEB сервера c которого отправляется запрос

external\_content\_vats\_url: <http://188.254.33.70:8080/svetets/orchestrator/speech/> : путь к получению контента с СХДП  
  
converted\_content: mp3 – поиск альтернативного формата контента (mp3)  
  
vocab\_description\_length: 120 – одна строка 120 символов (Длина фильтрации по количеству слов в словарях)default\_xlsx\_sheet: Sheet1 – (Вкладка по умолчанию в excel файле со скриптом )  
  
scripts\_percent: 70 – Значение по умолчанию для соответствия скрипту

# Программное обеспечение App

Программное обеспечение APP является ключевым сервером приложений обеспечивающих обработки логики и взаимодействие между компонентами ПРА.

## конфигурационный файл

*# Main app port, Основной порт работы приложения*port: 8675  
  
*# Cluster Params настройки работы режима кластера, текущий режим получения контента не позволяется использовать кластерный режим, значение 0 по-умолчанию.*cluster\_mode: 0  
cluster\_nodes:  
 - app1:8675- app2:8675 *# MongoDB настройки подключения*mongo\_ip: mongo  
mongo\_port: 27017  
mongo\_db: rtk\_analytics  
mongo\_collection\_data: data  
mongo\_collection\_jobs: jobs  
mongo\_collection\_queue: job\_queue  
mongo\_collection\_load\_queue: load\_queue  
  
*# Postgres DB настройки подключения*postgres\_ip: postgres  
postgres\_port: 5432  
postgres\_db: rtk\_ra  
postgres\_name: postgres  
postgres\_password: postgres  
  
*# Redis DB настройки подключения*redis\_ip: redis  
redis\_port: 6379  
redis\_db: 1  
redis\_name:  
redis\_password:  
  
*# Create empty Postgres DB – сервисный блок настроек, использовать только на тестовой среде. Используется для переиниациализации структуры данных БД.*create\_db: 0  
*# Initialize Postgres DB Scheme with drop all tables*init\_db: 0  
*# Add default content to Postgres DB*default\_content: 0

*# PG DB Limits*pg\_max\_connections: 1000  
pg\_max\_idle\_connections: 10  
*# PG Connection Lifetime in seconds*pg\_connection\_lifetime: 10  
  
*# Logfile path*logfile\_path: log/api\_log.log  
  
*# upload\_folder\_path*upload\_folder\_path: upload  
  
*# S3 storage path*minio\_host: minio:9000  
minio\_access\_key: hd78347gdncw09823hd2x5134  
minio\_secret\_key: xm34gaxkfgaw8m232x3rmh8f9f  
minio\_files\_bucket: speech-files  
minio\_temp\_files\_bucket: temp-files  
minio\_web\_files: web  
  
*# RTK VATS Svetets API*vats\_api\_url: <http://10.243.192.29:8500/svetets/orchestrator/speech> - URL запроса к основному API Svetetsvats\_api\_auth\_url: <http://10.243.192.29:8500/svetets/orchestrator/speech> - URL запроса к API Svetets для авторизации пользователей  
  
*# Worker params*timeout: 1 *# 1 время ожидания между запросами для обработки очереди, секунд*wait\_progress\_timeout: 60 *# 3 время ожидания обработки задания в состоянии in\_progress, минут*

wait\_queue\_processing\_timeout: 5 # *– время ожидания для обработки ожидающих в очереди заданий*wait\_queue\_timeout: 3 *# 3 – время ожидания для резервирования средств, минут*wait\_charging\_timeout: 60 *# 60 - время ожидания для списания средств, минут*wait\_attempts: 3 *# количество попыток по каждому действию*queue\_batch\_size: 10 *# количество одновременно обрабатываемых запросов*upload\_wait\_timeout: 720 *# 720 – время ожидания на повтор загрузки метаданных, минут*upload\_wait\_attempts: 30 *# количество попыток ожидания загрузки метаданных*local\_ip: 10.36.153.148 – IP адреса веб сервера для установки промежуточного Header X-Real-IP для получения аудио контента с СХДП.

nats\_host: nats – путь к nats серверу очередей   
nats\_port: 4222 – порт к nats серверу очередей  
nats\_version: 1 - # выбор версии 0 – версия без async очереди nats, 1 – версия с async очередью nats  
nats\_queue: sa – наименование канала в nats  
max\_parallel\_sa\_streams: 20 – максимальное количество параллельных заданий

# Программное обеспечение SA

Программное обеспечение SA является ключевым сервером анализа аудил и текстовых данных.

## Конфигурационный файл

app\_ip: # App IP заполняется только если нет доступа по hostname контейнера  
app\_port: 8980

app\_mode: 0 # 0 - режим отладки выключен, 1 - режим отладки включен  
node\_name: # Уникальное наименование ноды SA, если не заполнено используется hostname контейнера

*# Process limit* – количество одновременных потоков анализаprocess\_limit: 4

# Multiprocessing 0 - disable, 1 – enable, обработка каналов аудио записи параллельно  
mpc: 0  
   
*# Log file – путь к лог файлу*log\_file: log/speech\_api.log  
  
*#storage path внутренний путь хранения временных файлов*storage: storage  
minio\_host: minio:9000  
minio\_access\_key: hd78347gdncw09823hd2x5134  
minio\_secret\_key: xm34gaxkfgaw8m232x3rmh8f9f  
minio\_files\_bucket: speech-files  
minio\_temp\_files\_bucket: temp-files  
minio\_web\_files: web  
  
*# temp folders*temp\_folder: temp  
  
*# wave audio form path*audioform: /usr/bin/audiowaveform – генератор аудио волны для web приложений  
  
*# emotion\_detection model – модель определения эмоций*em\_model: classifier4.gmmvocab\_synon: synonyms\_only\_adjective.txt – указание на файл разметки словаря синонимов

# Set eng version 1 - enable, 0 – disable

eng\_asr: 0 – указание на включение англоязычную версию (1 – вкл, 0 - выкл), требуется наличие модели.

# Save export file format. wav/mp3

file\_format: mp3 – формат сохранения файлов

# engine version sync/async

engine: async – тип режима работы (требуется аналогичный с APP, async – версия с nats, sync – без nats)

# NATS Config

nats\_ip: nats

nats\_port: 4222 – порт к nats серверу очередей

nats\_queue: sa – наименование канала в nats

nats\_subqueue: queue1 – наименование очереди в канале в nats

# Redis Config

redis\_ip: redis – путь к redis серверу

redis\_port: 6379 – порт к redis серверу

redis\_login: – логин к redis серверу

redis\_password: – пароль к redis серверу

redis\_db: 5 – БД redis сервера

redis\_exp: 300 – время хранения информации о записи

# ASR Engine

asr\_engine: external – тип ASR (external – внешний, internal - внутренний)

asr\_ip: asr\_proxy – имя хоста с ASR

asr\_port: 2600 – порт хоста с ASR

asr\_ip\_eng: asrproxy – имя хоста с английской версией ASR

asr\_port\_eng: 2601– порт хоста с английской версией ASR

# Программное обеспечение ASR PROXY

Программное обеспечение ASR PROXY является сервером балансировки запросов перевода речи в текст.

## Конфигурационный файл

#user nobody;

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

sendfile on;

keepalive\_timeout 65;

server {

listen 7810;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

map $http\_upgrade $connection\_upgrade {

default upgrade;

'' close;

}

upstream asrserver {

server asr1:2700 max\_conns=6; - # указание по лимиту подключений на asr1

server asr2:2700 max\_conns=6; - # указание по лимиту подключений на asr2

server asr3:2700 max\_conns=15; - # указание по лимиту подключений на asr3

server asr4:2700 max\_conns=15; - # указание по лимиту подключений на asr4

}

server {

listen 2600;

server\_name 0.0.0.0;

location / {

proxy\_pass http://asrserver;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection $connection\_upgrade;

proxy\_read\_timeout 950s;

}

}

}

# Мониторинг

Каждый компонент ПРА имеет встроенные функции мониторинга доступности.

Управление состояние производится встроенными средствами Docker swarm.

## Управление нагрузкой

Для управления масштабирование ПРА при превышении нагрузи на какой либо компонент, предусмотрена возможность проверки состояние размера очередей с запросов в APP компонент:

**Очередь на загрузку метаданных и аудиофайлов в систему из ВАТС.**

**Запрос:** GET <http://app/monitoring/load_length> , где app – URL app приложения

**Ответ:** цифровое значение длины очереди.  
**Очередь на обработку загруженных в систему аудиоданных.**

**Запрос:** GET <http://app/monitoring/sa_length> , где app – URL app приложения

**Ответ:** цифровое значение длины очереди

**Очередь на обработку загруженных в систему аудиоданных в минутах.**

**Запрос:** GET <http://app/monitoring/sa_length_minute/> , где app – URL app приложения

**Ответ:** цифровое значение длины очереди в минутах.

**Очередь на обработку загруженных в систему аудиоданных старше 12 часов.**

**Запрос:** GET <http://app/monitoring/sa_length_12/> , где app – URL app приложения

**Ответ:** цифровое значение длины очереди более 12 часов.

**Очередь на обработку загруженных в систему аудиоданных в минутах старше 12 часов.**

**Запрос:** GET <http://app/monitoring/sa_length_minute_12/> , где app – URL app приложения

**Ответ:** цифровое значение длины очереди более 12 часов в минутах.

**Текущая производительность системы в записях.**

**Запрос:** GET <http://app/monitoring/performance/> , где app – URL app приложения

**Ответ:** цифровое значение количества одновременно обрабатываемых записей.

**Текущая производительность системы в минутах.**

**Запрос:** GET <http://app/monitoring/performance_minute/> , где app – URL app приложения

**Ответ:** цифровое значение количества одновременно обрабатываемых записей в минутах.