Руководство системного администратора

Это руководство описывает работы с TMS Test IT Enterprise , выполняемые системным администратором. Установка Test IT выполняется с помощью развертывания контейнеров с компонентами системы в Docker Compose и Kubernetes. Контейнеры с базами данных и сторонними сервисами поддерживают внешнее подключение — вы можете подключить свои базы данных.

Руководство содержит следующие разделы:

- Установка в Docker Compose и Kubernetes
- Настройка внешних подключений в Docker Compose и Kubernetes
- Настройка почтового SMTP-сервера для уведомлений
- Настройка внешних ссылок для перехода из Test IT
- Перезапуск системы в Docker Compose и Kubernetes
- Работа с компонентами Kubernetes:
  - Изменение выделенных ресурсов
  - Замена рабочего узла
  - Настройка SSL для внутренних подключений
  - Переход на новый кластер Kubernetes
  - Перезапуск подов и остановка компонентов Test IT
  - Переопределение переменных и настроек приложений
  - Переход из Docker в Kubernetes
- Обновление системы в Docker Compose и Kubernetes
- Резервное копирование в Docker Compose и Kubernetes
- Логирование пользовательских действий в Docker Compose
- Настройка HTTPS в Docker Compose и Kubernetes
- Добавление самоподписанных сертификатов в контейнеры в Docker Compose
- Удаление системы в Docker Compose и Kubernetes

Это руководство можно скачать в PDF.

Обновлено: 22.01.2026, 20:51:48

Установка в Docker Compose

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Требования

Минимальные системные требования

Минимальные системные требования указаны для системы, содержащей до 100 активных пользователей. Чтобы рассчитать требования для большего количества пользователей, обратитесь в техническую поддержку: (support@yoonion.ru).

- **CPU:** 4 ядра серверного класса с поддержкой виртуализации и тактовой частотой 2.2 ГГц и выше
- **RAM:** 16 ГБ
- **Network:** 100 Мбит/с
- **SSD:** 100 ГБ
- **SWAP:** отключен

Требования к программному обеспечению

- Docker Engine 17.12.0 и выше
- Docker Compose V2 и выше

Смотрите также: Руководство по установке Docker Compose

Состав поставки

- `.env` — конфигурационный файл, содержащий переменные, используемые для обращения к контейнерам Test IT
- `docker-compose.yml` — конфигурационный файл Docker Compose
- `docker-compose.elk.yml` — конфигурационный файл Docker Compose с базами данных Elasticsearch, Logstash и Kibana
- `backup.sh` — скрипт запуска резервного копирования
- `restore.sh` — скрипт восстановления из резервной копии
- `images.tar.gz` — архив с образами (только в архиве для автономной установки)
- `images.tar.gz` — архив с образами (только в архиве для автономной установки)
- `postgres-init.sql` — инициализационный файл для контейнера базы данных

Подготовка

1. Измените значения переменных по умолчанию в `.env` **-файле**.
2. Задайте параметры `vm.max_map_count=262144` и `vm.overcommit_memory=1` :

```sh
echo 'vm.max_map_count=262144' >> /etc/sysctl.conf
echo 'vm.overcommit_memory = 1' >> /etc/sysctl.conf
sysctl -p
```

3. Заблокируйте все порты, кроме порта 80, необходимого для доступа к пользовательскому интерфейсу.
4. **Опционально:** для обслуживания системы посредством протокола SSH, необходимо открыть порт 22 (может быть переназначено на конкретной конфигурации). Для работы по HTTPS необходимо открыть порт 443. Пример открытия доступа к портам для CentOS 7:

```sh
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --zone=public --add-port=22/tcp --permanent
firewall-cmd --zone=public --add-port=443/tcp --permanent
firewall-cmd --reload
```

5. **Опционально:** включите **логирование пользовательских действий**. По умолчанию оно отключено.

Автономная установка

Данный тип установки поможет установить продукт, если сервер изолирован от сети Internet и нет возможности получить Docker образы с публичных репозиториев.

1. Скачайте дистрибутив со страницы загрузок .
2. Распакуйте содержимое архива автономной установки, например в папку `~/testit` .
3. Выполните следующие команды:

   Для версий до v5.2.0:

```sh
cd ~/testit
docker load -i images.tar.gz
docker network create yoonion_network
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

   Начиная с версии v5.2.0:

```sh
cd ~/testit
mkdir images
tar -zxvf images.tar.gz -C images
for file in $(ls images/*.tar); do docker image load -i $file; done
docker network create yoonion_network
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Установка онлайн

1. Скачайте файлы online-установки со страницы загрузок .
2. Распакуйте содержимое архива online-установки, например в папку `~/testit` .
3. Выполните следующие команды:

```sh
cd ~/testit
docker network create yoonion_network
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Обновлено: 30.07.2025, 13:09:29

Описание .env-файла

- Репозиторий для скачивания образов установки Test IT:

```text
TMS_DOCKER_REGISTRY=registry.testit.software/testit
```

- Текущая версия программы:

```text
TMS_CONTAINER_VERSION=4.0.1
```

- Адрес Test IT используется в качестве обратной ссылки. Необходимо задать эту переменную, если вы разворачиваете Frontend и Backend на разных серверах или хотите настроить интеграцию с Jira.

```text
FRONTEND_URL=http://localhost
```

- Сертификат для настройки HTTPS, ключ для настройки HTTPS, true — редирект HTTP на HTTPS:

```text
## internal certificate path
#SSL_CERTIFICATE=/etc/nginx/ssl/testit.crt
#SSL_CERTIFICATE_KEY=/etc/nginx/ssl/testit.key
#REDIRECT_TO_HTTPS=true
```

- Принудительное отключение проверки сертификата для внешнего сервиса, например, в случае проблем подключения к Jira с самоподписанным сертификатом (причине не принимает цепочку сертификатов); вы можете указать несколько сервисов, используя в качестве разделителя ";":

```text
#INSECURE_REMOTES=example.com:443
```

- Ключи доступа к хранилищу прикрепляемых файлов в Test IT (minio):

```text
AWS_ACCESS_KEY=testitAccessKey
AWS_SECRET_KEY=testitSecretKey
AWS_CONNECTION_STRING=http://minio:9000
```

- Ключи доступа к хранилищу "avatars" в Test IT (minio):

```text
AVATARS_AWS_ACCESS_KEY=${AWS_ACCESS_KEY}
AVATARS_AWS_SECRET_KEY=${AWS_SECRET_KEY}
AVATARS_AWS_CONNECTION_STRING=${AWS_CONNECTION_STRING}
```

- Параметры подключения к RabbitMQ:

```text
RABBITMQ_DEFAULT_USER=testit
RABBITMQ_DEFAULT_PASS=F1rstL0g0N!
RABBITMQ_DEFAULT_VHOST=testitrabbit
RABBITMQ_DEFAULT_HOST=rabbitmq
RABBITMQ_DEFAULT_PORT=5672
RABBITMQ_AUTH_MODE=plain
RABBITMQ_CLIENT_CERT_PATH=/etc/rabbitmq/ssl/client/testit.pfx
#RABBITMQ_CLIENT_CERT_PASSPHRASE=
```

- Переменная включения SSL в RabbitMQ (Настройка внешнего подключения RabbitMQ):

```text
#RABBITMQ_SSL_ENABLED=true
```

- Параметры подключения к БД, при установке внешней БД, поменять на свои значения (Использование внешней БД (PostgreSQL)):

```text
DB_CONNECTION_STRING=Host=db;Port=5432;Database=testitdb;Username=p
Pool Size=130
```

- Данные для создания БД, пользователя и пароля в дефолтной поставке:

```text
POSTGRES_DB=testitdb
POSTGRES_USER=postgres
POSTGRES_PASSWORD=F1rstL0g0N!
```

- Аналогично для Auth DB:

```text
AUTH_CONNECTION_STRING=Host=db;Port=5432;Database=authdb;Username=p
Pool Size=130;Command Timeout=30
POSTGRES_AUTH_DB=authdb
POSTGRES_AUTH_USER=postgres
POSTGRES_AUTH_PASSWORD=F1rstL0g0N!
```

- Аналогично для Avatar DB:

```text
AVATARS_CONNECTION_STRING=Host=db;Port=5432;Database=avatarsdb;User
Timeout=30
POSTGRES_AVATARS_DB=avatarsdb
POSTGRES_AVATARS_USER=postgres
POSTGRES_AVATARS_PASSWORD=F1rstL0g0N!
```

- Адрес для подключения к InfluxDB (Настройка внешнего подключения базы данных InfluxDB)

```text
INFLUX_CONNECTION_STRING=http://influxdb:8086
#INFLUX_AUTH_ENABLED=true
#INFLUX_USERNAME=testit
#INFLUX_PASSWORD=password
#INFLUXDB_META_DIR=/var/lib/influxdb/meta2
```

- Параметры SSL соединения InfluxDB:

```text
#INFLUXDB_HTTP_HTTPS_ENABLED=true
#INFLUXDB_HTTP_HTTPS_CERTIFICATE=/var/lib/influxdb/tls/server.crt
#INFLUXDB_HTTP_HTTPS_PRIVATE_KEY=/var/lib/influxdb/tls/server.key
```

- Параметры конфигурирования Elasticsearch, Logstash, Kibana (Настройка внешнего подключения стека Elasticsearch, Logstash и Kibana (ELK)):

```text
ELASTICSEARCH_CONNECTION_STRING=http://elasticsearch:9200
LOGSTASH_CONNECTION_STRING=http://logstash:5044
ELASTICSEARCH_INDEX=testit
ELASTICSEARCH_LOGS_INDEX=action_logs
```

- Параметры SSL соединения Elasticsearch:

```text
#ELASTICSEARCH_SECURITY_ENABLED=true
#ELASTICSEARCH_SSL_ENABLED=true
#ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES=/usr/share/elasticsearch/
```

- Параметры аутентификации в Elasticsearch:

```text
#ELASTICSEARCH_AUTH_ENABLED=true
#ELASTICSEARCH_USERNAME=elastic
#ELASTICSEARCH_PASSWORD=pass
#KIBANA_SERVER_NAME=localhost
#KIBANA_ELASTICSEARCH_USERNAME=kibana_system
#KIBANA_ELASTICSEARCH_PASSWORD=pass
#LOGSTASH_ELASTICSEARCH_USERNAME=elastic
#LOGSTASH_ELASTICSEARCH_PASSWORD=pass
```

- Строка подключения к Redis (Настройка внешнего подключения базы данных Redis):

```sh
AUTH_CACHE_CONNECTION_STRING=auth-cache
```

- Пароль для аутентификации в Redis:

```text
#REDIS_PASSWORD=pass
```

- Параметры SSL соединения Redis:

```text
#REDIS_TLS_CERT_FILE=/tls/server.crt
#REDIS_TLS_KEY_FILE=/tls/server.key
```

- Системные параметры оставить без изменений:

```text
ASPNETCORE_ACCESS_TOKEN_EXPIRATION_MINUTES=8000
ASPNETCORE_REFRESH_TOKEN_EXPIRATION_MINUTES=88000
```

- Уровень логирования. Можно изменить Warning на Information для более детального логирования, что повысит нагрузку на систему:

```text
API_LOG_LEVEL=Warning
```

- Минимальный пул рабочих потоков на процессор. Используется для Webapi. Чем выше значение, тем большее количество пользователей будут одновременно обслуживаться, что равномерно повысит нагрузку:

```text
THREAD_PER_PROCESSOR=10
```

- Параметр PKCE. Можно переключить в "false", если OpenId провайдер не поддерживает PKCE:

```text
USE_PKCE=true
```

- Параметр, необходимый, если внешний сервис Jira заблокирован для исходящих подключений. Значение указано в секундах, время через которое Test IT инициирует входящее подключение к Jira для синхронизации данных:

```text
SYNC_RESULT_LINKS_EVERY_SEC=120
```

- Период хранения бизнес-логов по действиям пользователей в Elasticsearch:

```text
EVENT_LOG_MAX_AGE=30d
```

- Название бакета MinIO для подключения:

```text
TMS_FILE_BUCKET_NAME=testit
```

- Параметр настройки интервала расчета стабильности автотестов:

```text
CALCULATION_AUTOTESTS_STABILITYPERCENTAGE_DELAY_SECONDS=600
```

- Параметр настройки интервала работы фонового сервиса по удалению архивных данных:

```text
DELETE_ARCHIVE_DATA_DELAY_SECONDS=600
```

- Параметр, позволяющий включить поддержку продукта TeamStorm:

```text
CWM_ENABLED=false
```

Обновлено: 11.12.2023, 18:44:17

Установка в Kubernetes

Требования

Минимальные системные требования

Минимальные системные требования указаны для системы, содержащей до 100 активных пользователей. Чтобы рассчитать требования для большего количества пользователей, обратитесь в техническую поддержку: (support@yoonion.ru).

- **CPU:** 4 ядра серверного класса с поддержкой виртуализации и тактовой частотой 2.2 ГГц и выше
- **RAM:** 16 ГБ
- **Network:** 100 Мбит/с
- **SSD:** 100 ГБ
- **SWAP:** отключен

Требования к программному обеспечению

- Установленный в кластере ingress-контроллер, например Nginx Ingress Controller
- Настроенный поставщик Persistent Volumes
- Наличие Kubectl
- Наличие Helm версии 3.10.0 или более поздней

Состав поставки

- Содержимое сборки:
  - `testit_unichart/` — Helm chart для развертывания продукта
  - `scripts/` — папка, содержащая вспомогательные скрипты
  - `jobs/` — папка, содержащая вспомогательные объекты K8S
- Содержимое чарта:
  - `templates/` — шаблоны объектов K8S
  - `files/` — конфигурационные файлы для компонентов продукта
  - `Chart.yaml` — основной конфигурационный файл чарта
  - `values.yaml` — настройки компонентов и переменных по умолчанию

- `values-override.yaml` — опциональный файл для переопределения настроек из values.yaml
- `values-ssl.yaml` — пример переопределения для включения внутреннего SSL

Установка приложения

1. Распакуйте файлы приложений с помощью команды:

```sh
unzip <testit_archive_name> -d <installation_folder>
```

2. В случае необходимости переназначьте переменные/доп. конфигурацию в файле `values-override.yaml`, используя такие же отступы и иерархию, что и в файле `values.yaml`. В случае переопределения переменных добавьте флаг `-f values-override.yaml` во все команды helm, как показано в примере ниже.
3. Установите приложение с помощью команды:

```sh
cd <installation_folder>
helm upgrade --install -f testit_unichart/values-override.yaml
-n <my-namespace> --create-namespace testit testit_unichart/ --
wait --timeout 10m
# Дождитесь начала работы всех компонентов продукта.
watch -n 1 kubectl -n <my-namespace> get pods
```

4. Перейдите в приложение, используя адрес, указанный в `.frontend.ingress.main.host` в `testit_unichart/values.yaml` или `testit_unichart/values-override.yaml`:

```yml
# testit_frontend/values.yaml или testit_frontend/values-
override.yaml
#---
frontend:
#---
ingress:
main:
host: "my.hostname.com" # Set the desired hostname you own
#---
```

Обновлено: 30.07.2025, 13:09:29

Значения, используемые в файле "values"

- Настройки для развертывания:

```yml
frontend:
replicaCount: 2 # Количество копий
resources:
requests: # Минимальные выделяемые ресурсы
memory: "128Mi"
cpu: "200m"
limits: # Лимит выделяемых ресурсов
memory: "512Mi"
cpu: "500m"
```

- Публичное DNS-имя приложения:

```yml
frontend:
ingress:
main:
host: "my.hostname.com"
webapi:
ingress:
main:
host: "my.hostname.com"
```

- Общие конфигурационные переменные:

```yml
config:
NGINX_JSON_LOGGING: "false"
SSL_STRONG_CIPHERS: "false"
```

- Конфигурационные файлы:

```yml
# Определение конфигурационного файла в соответствии с
расположением относительно директории testit_unichart
configFile:
nginx-conf-cm:
nginx.conf: "files/nginx.conf"
# Расположение конфигурационного файла в контейнере frontend
volumes:
nginx-conf:
fromConfigMap: nginx-conf-cm
mounts:
frontend:
mountPath: /etc/nginx/nginx.conf
subPath: nginx.conf
```

- Общие приложения внутреннего интерфейса:
  - auth-cache
  - postgres
  - rabbitmq
  - influxdb
  - minio
  - gotenbeng-api

  Представленное описание применимо ко всем сторонним приложениям, в качестве примера используется authCache (Redis):

```yml
auth-ache:
enabled: true # Развертывание внутри кластера
resources:
requests: # Минимальные выделяемые ресурсы
memory: "128Mi"
cpu: "100m"
limits: # Лимит выделяемых ресурсов
memory: "256Mi"
cpu: "200m"
statefulset:
enabled: true # Приложение разворачивается в K8S как
statefulset
storage:
mountPaths:
- /data # Путь монтирования PVC в контейнер
spec: # Конфигурация PVC
accessModes: ["ReadWriteOnce"]
# Set storageClassName (nfs-client, local-path, etc.)
# storageClassName: "nfs-client"
resources:
requests:
storage: "256Mi"
```

- Общая конфигурация

```yml
default: # <-- Секция со значениями по умолчанию, применяется
если поле обязательное, но для конкретного сервиса не указано
general: # <-- Секция с общими значениями, применяется для всех
сервисов, если конкретно не указано, что для my-service этого
делать не нужно
config: # Переменные приложений (Test IT и стороннее ПО)
# <https://docs.testit.software/installation-
guide/description-.env-file.html>
```

Обновлено: 20.06.2025, 14:33:37

Подключение RabbitMQ в Docker Compose

> **Важно**
>
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

1. Укажите в файле `.env` для следующих параметров значения, установленные вами при настройке RabbitMQ (ниже указаны значения по умолчанию):

```text
RABBITMQ_DEFAULT_USER=testit
RABBITMQ_DEFAULT_PASS=password
RABBITMQ_DEFAULT_VHOST=testitrabbit
RABBITMQ_DEFAULT_HOST=external-server (где external-server – ip-адрес или DNS-имя вашего сервера с RabbitMQ)
RABBITMQ_DEFAULT_PORT=5672
RABBITMQ_AUTH_MODE=plain
RABBITMQ_CLIENT_CERT_PATH=/etc/rabbitmq/ssl/client/testit.pfx
#RABBITMQ_CLIENT_CERT_PASSPHRASE=
```

2. В файле `docker-compose.yml` закомментируйте секцию с сервисом rabbitmq, зависимости от него других контейнеров (все упоминания rabbitmq в блоках `depends_on`) и `rabbit-volume` , `rabbitmq-configuration-volume` , `rabbitmq-certificates-volume` в списке `volumes` .

3. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120 --remove-orphans
```

Настройка безопасного соединения

1. Подготовьте файлы сертификатов, используя CN `rabbitmq` :

- Корневой файл `rabbitmq_ca.pem`
- Сертификат `rabbitmq_cert.pem` , подписанный с помощью корневого файла `rabbitmq_ca.pem`
- Ключ сервера `rabbitmq_key.pem` , подписанный с помощью корневого файла `rabbitmq_ca.pem` Названия сертификатов и ключа могут быть любыми.

2. Задайте разрешения файлов:

- Для файлов сертификата и ключа для пользователя rabbitmq (100) в контейнере
- Для файла ключа — ограниченное разрешение. Используйте команды:

```sh
chown 100 rabbitmq_key.pem rabbitmq_ca.pem rabbitmq_cert.pem
chgrp 101 rabbitmq_key.pem rabbitmq_ca.pem rabbitmq_cert.pem
chmod 600 rabbitmq_key.pem
```

3. Скопируйте файлы `rabbitmq_key.pem` , `rabbitmq_ca.pem` и `rabbitmq_cert.pem` в вольюм с помощью команды:

```sh
server_certs=$(docker inspect testit_rabbitmq-certificates-volume --format '{{ .Mountpoint }}')
cp -p rabbitmq_key.pem rabbitmq_ca.pem rabbitmq_cert.pem ${server_certs}/
```

4. Скопируйте файл сертификата CA, с помощью которого были выписаны сертификаты серверов, в вольюм `trusted-certificates-volume` :

```sh
trusted_certs=$(docker inspect testit_trusted-certificates-volume --format '{{ .Mountpoint }}')
cp rabbitmq_ca.pem ${trusted_certs}/
```

5. Создайте файл конфигурации с расширением .conf, например 20-ssl.conf со следующим содержанием:

```sh
listeners.ssl.default = 5671
ssl_options.cacertfile = /etc/certs/rabbitmq_ca.pem
ssl_options.certfile = /etc/certs/rabbitmq_cert.pem
ssl_options.keyfile = /etc/certs/rabbitmq_key.pem
ssl_options.verify = verify_none
ssl_options.fail_if_no_peer_cert = false
```

6. Скопируйте файл в вольюм с конфигурацией `rabbitmq` :

```sh
rabbitmq_conf=$(docker inspect testit_rabbitmq-configuration-volume --format '{{ .Mountpoint }}')
cp 20-ssl.conf ${rabbitmq_conf}/conf.d
```

7. В файле `.env` раскомментируйте строку:

```sh
RABBITMQ_SSL_ENABLED=true
```

8. Измените порт для подключения по ssl на `5671` :

```sh
RABBITMQ_DEFAULT_PORT=5671
```

9. Чтобы изменения вступили в силу, перезапустите запущенные сервисы, а затем примените изменения в файле `.env` :

```sh
docker compose -f docker-compose.yml --project-name testit restart
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Обновлено: 14.10.2024, 16:15:16

Подключение стека Elasticsearch, Logstash и Kibana (ELK) в Docker Compose

> **Важно**
>
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Настройка внешнего подключения

1. При настройке стека ELK укажите в `.env` -файле следующие параметры соответственно вашей конфигурации:

```text
ELASTICSEARCH_CONNECTION_STRING=http://external-server:9200
(где external-server – IP-адрес или DNS-имя вашего сервера с
Elasticsearch)
ELASTICSEARCH_INDEX= (заданное вами имя индекса для Test IT)
ELASTICSEARCH_LOGS_INDEX= (заданное вами имя индекса логов)
LOGSTASH_CONNECTION_STRING=http://external-server:5044 (где
external-server – IP-адрес или DNS-имя вашего сервера с
Logstash)
```

2. В файле `docker-compose.yml` выполните следующие действия:
   - Добавьте следующую строку в секцию `webapi` , раздел `environment` :

```text
Serilog__UserActionAll__WriteTo__1__Args__requestUri:
"${LOGSTASH_CONNECTION_STRING:-http://logstash:5044}"
```

- Добавьте следующую строку в секцию `auth`, раздел `environment`:

```text
Serilog__AdminAll__WriteTo__1__Args__requestUri:
"${LOGSTASH_CONNECTION_STRING:-http://logstash:5044}"
```

- Закомментируйте секции с сервисами Elasticsearch, Logstash, Kibana, зависимости от него других контейнеров (все упоминания `elasticsearch`, `logstash`, `kibana` в блоках `depends_on`) и `elastic-volume` в списке `volumes`.

3. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120 --remove-orphans
```

Настройка безопасного соединения и аутентификации между компонентами ELK

После выполнения данной инструкции соединение между контейнерами `logstash`, `elasticsearch`, `kibana` и `webapi` будет происходить по TLS протоколу.

1. Создайте конфигурационные файлы для генерации самоподписанных сертификатов:

   - `create-certs.yml`

```sh
version: '2.2'

services:
create_certs:
image:
docker.elastic.co/elasticsearch/elasticsearch:${VERSION}
container_name: create_certs
command: >
bash -c '
yum install -y -q -e 0 unzip;
if [[ ! -f /certs/bundle.zip ]]; then
bin/elasticsearch-certutil cert --silent --pem --in
config/certificates/instances.yml -out /certs/bundle.zip;
unzip /certs/bundle.zip -d /certs;
fi;
chown -R 1000:0 /certs
'
working_dir: /usr/share/elasticsearch
volumes:
- ./certs:/certs
- .:/usr/share/elasticsearch/config/certificates
networks:
- elastic

volumes:
certs:
driver: local

networks:
elastic:
driver: bridge
```

- create-certs.env

```sh
COMPOSE_PROJECT_NAME=testit
VERSION=7.17.5
```

- instances.yml

```sh
instances:
- name: elasticsearch
dns:
- elasticsearch
- localhost
ip:
- 127.0.0.1
- name: logstash
dns:
- logstash
- localhost
ip:
- 127.0.0.1
## Раскомментируйте строки ниже, чтобы настроить HTTPS для Kibana
# - name: 'kibana'
# dns:
# - kibana
# - localhost
```

2. Сгенерируйте сертификаты.

```sh
docker compose -f create-certs.yml --env-file create-certs.env run --rm create_certs
```

После того, как инструкции в контейнере будут выполнены, в текущей директории появится папка `certs` с сгенерированными сертификатами. Там должны находиться директория `ca` и директории с названиями сервисов из `instances.yml`.

3. Конвертируйте ключ сертификата `logstash.key` в формат `pkcs8`.

```sh
openssl pkcs8 -in ./certs/logstash/logstash.key -topk8 -nocrypt -out ./certs/logstash/logstash.key
```

4. Измените владельца файлов `logstash.key` и `logstash.crt` на пользователя `logstash (uid 1000)`.

```sh
chown 1000 ./certs/logstash/logstash.*
```

5. Скопируйте содержимое директории `certs` в вольюм `elk-tls-volume` .

```sh
elk_certs=$(docker inspect testit_elk-tls-volume --format '{{
.Mountpoint }}')
cp -r certs/* ${elk_certs}/
```

6. Скопируйте корневой сертификат `certs/ca/ca.crt` в вольюм `trusted-certificates-volume` , переименовав его в `elk_ca.crt` , чтобы избежать перезаписи уже существующих сертификатов в данном вольюме.

```sh
trusted_certs=$(docker inspect testit_trusted-certificates-
volume --format '{{ .Mountpoint }}')
cp certs/ca/ca.crt ${trusted_certs}/elk_ca.crt
```

7. В файле `.env` раскомментируйте строки, конфигурирующие SSL, и замените протокол в переменной ELASTICSEARCH_CONNECTION_STRING и LOGSTASH_CONNECTION_STRING на HTTPS.

```sh
ELASTICSEARCH_CONNECTION_STRING=https://elasticsearch:9200
LOGSTASH_CONNECTION_STRING=https://logstash:5044
...

ELASTICSEARCH_SECURITY_ENABLED=true
ELASTICSEARCH_SSL_ENABLED=true
ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES=/usr/share/elasticsearch/c
```

8. В файле `docker-compose.elk.yml` раскомментируйте строки, указывающие пути к сертификатам.

   - `elasticsearch`

```yml
xpack.security.http.ssl.certificate_authorities:
"${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elastics
xpack.security.http.ssl.key:
/usr/share/elasticsearch/config/certificates/elasticsearch/elast
xpack.security.http.ssl.certificate:
/usr/share/elasticsearch/config/certificates/elasticsearch/elast
```

   - `logstash`

```sh
ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:
"${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elastics
SERVER_SSL_KEY: /usr/share/elasticsearch/config/certificates/log
SERVER_SSL_CERTIFICATE: /usr/share/elasticsearch/config/certific
```

- `kibana`

```sh
ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:
"${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elastics
SERVER_SSL_KEY: /usr/share/elasticsearch/config/certificates/kib
SERVER_SSL_CERTIFICATE: /usr/share/elasticsearch/config/certific
```

9. После применения описанных выше настроек Kibana, Logstash и контейнер `webapi` не будут иметь возможности подключиться к Elasticsearch. Для подключения необходимо сгенерировать пароли для системных пользователей `elasticsearch` с помощью утилиты `elasticsearch-setup-passwords` .

```sh
docker exec testit-elasticsearch-1 /bin/bash -c
"bin/elasticsearch-setup-passwords auto --batch --url
https://elasticsearch:9200"
```

> **Важно**
>
> Сохраните сгенерированные пароли в безопасном месте. Они понадобятся для конфигурации подключения контейнеров к `elasticsearch` , а также для аутентификации в Kibana.

10. Укажите сгенерированные пароли в файле `.env` , раскомментировав следующие строки.

```sh
ELASTICSEARCH_AUTH_ENABLED=true
ELASTICSEARCH_USERNAME=elastic
ELASTICSEARCH_PASSWORD=pass
KIBANA_ELASTICSEARCH_USERNAME=kibana_system
KIBANA_ELASTICSEARCH_PASSWORD=pass
LOGSTASH_ELASTICSEARCH_USERNAME=elastic
LOGSTASH_ELASTICSEARCH_PASSWORD=pass
```

В данном случае для доступа `logstash` в `elasticsearch` используется супер-пользователь `elastic`. Для дополнительной безопасности вы можете ограничить права `logstash`, создав отдельную учетную запись. Пример создания такой учетной записи можно найти в документации Elastic.

11. Измените конфигурацию пайплайна logstash в файле `/config_files/logstash/logstash.conf` с учетом необходимости безопасного соединения. Пример файла с настройками для безопасного соединения вы можете найти в разделе Возможность конфигурирования Logstash.

12. Примените настройки, перезапустив систему.

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Возможность конфигурирования Logstash

Для внесения изменений в конфигурацию сбора логов из Test IT, вы можете редактировать файл `./config_files/logstash/logstash.conf` в поставке Test IT по своему усмотрению.

> **Внимание**
>
> Не переименовывайте файл и не меняйте путь к нему! Файл примонтирован как вольюм в контейнер logstash.

1. После первого внесения изменений в файл, выполните команду:

```sh
chown 1000 ./config_files/logstash/logstash.conf
```

Это необходимо, чтобы пользователь logstash в контейнере получил к файлу доступ.

2. После внесения любых изменений в файл logstash.conf обязательно выполняйте перезапуск контейнера logstash:

```sh
docker restart testit-logstash-1
```

3. Обратите внимание, что при создании собственной ILM политики рекомендуется указывать кастомное имя и не использовать имя по умолчанию (action_logs_policy), так как политика с данным именем будет приведена к настройкам по умолчанию при каждом старте контейнера logstash. Чтобы применить новую политику, после ее создания измените имя политики в файле `./config_files/logstash/logstash.conf` в следующей строке:

```sh
ilm_policy => "<policy_name>"
где <policy_name> – имя созданной вами политики.
```

4. Затем удалите index template ("action_logs" по умолчанию), к которому хотите применить политику, и выполните рестарт контейнера logstash. Все создаваемые далее индексы будут использовать новую ILM политику. Если необходимо, чтобы текущий индекс использовал новую политику, укажите ее название в настройках индекса, например в веб-интерфейсе kibana.

> **Важно**
>
> Если вы планируете использовать ssl в стеке Elk, пожалуйста, обратитесь к руководству по настройке защищенного соединения.
>
> ▶ Пример конфигурации в файле logstash.conf с включенным ssl (развернуть)

Смотрите также

- Built-in Users
- Set Up HTTPS
- Run Docker with TLS

Обновлено: 14.10.2024, 16:15:16

Подключение MinIO в Docker Compose

> **Важно**
>
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Вы можете использовать одну базу данных для сервисов minio и avatars.minio.

1. Создайте два бакета. Например, `bucket1` на замену сервису minio и `bucket2` на замену сервису avatars.minio.
2. Для каждого из бакетов создайте пару Access Key и Secret Key.
3. В `.env` -файле:
   - Для сервиса minio установите следующие значения переменных:

```text
AWS_ACCESS_KEY: (Access key, установленный для bucket1)
AWS_SECRET_KEY: (Secret Key, установленный для bucket1)
AWS_CONNECTION_STRING: http://external-server:9000 (где
external-server – ip-адрес или DNS-имя вашего сервера с
minio)
FILE_BUCKET_NAME: (в данном примере, bucket1)
```

   - Для сервиса avatars.minio установите следующие значения переменных:

```text
AVATARS_AWS_ACCESS_KEY: (Access key, установленный для
bucket2)
AVATARS_AWS_SECRET_KEY: (Secret Key, установленный для
bucket2)
AVATARS_AWS_CONNECTION_STRING: http://external-server:9000
```

   - Добавьте для сервиса avatars.minio следующую переменную:

```text
AVATARS_FILE_BUCKET_NAME: (в данном примере, bucket2)
```

4. В файле `docker-compose.yml` закомментируйте секцию с сервисом `minio`, зависимости от него других контейнеров (все упоминания сервиса в блоках `depends_on`), и его вольюмы (minio-export-volume и minio-data-volume) в списке `volumes`.

5. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120 --remove-orphans
```

Настройка безопасного соединения

1. Подготовьте файлы сертификатов — корневой `ca.crt`, а также подписанные с помощью него сертификат и ключ сервера `server.crt` и `server.key`. CN сертификата сервера должен быть `minio`.

2. Скопируйте файлы server.crt и server.key для minio в вольюм:

```sh
server_certs=$(docker inspect yourproject_minio-tls-volume --format '{{ .Mountpoint }}')
cp server-minio.crt ${server_certs}/public.crt
cp server-minio.key ${server_certs}/private.key
```

   При копировании файлы сертификатов переименовываются в `public.crt` и `private.key` соответственно, так как `minio` ожидает такие файлы в соответствии с документацией. :::

3. Скопируйте файл сертификата CA, с помощью которого были выписаны сертификаты серверов, в вольюм `trusted-certificates-volume`:

```sh
trusted_certs=$(docker inspect yourproject_trusted-certificates-volume --format '{{ .Mountpoint }}')
cp ca.crt ${trusted_certs}/
```

4. Измените значение переменной для подключения к `minio` в файле `.env` для использования HTTPS:

```sh
AWS_CONNECTION_STRING=https://minio:9000
```

5. Для того, чтобы изменения вступили в силу, перезапустите сервис `minio`, а затем примените изменения в файле `.env` с помощью `docker compose`:

```sh
docker restart yourproject_minio_1
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Сервисы `webapi` и `avatars.api` также будут перезапущены с новой конфигурацией для подключения к `minio` по HTTPS.

Обновлено: 14.10.2024, 16:15:16

Подключение Redis в Docker Compose

> **Назовите свой проект**
>
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

1. При настройке внешней базы данных Redis установите следующий параметр:

```text
appendonly yes
```

2. В файле `docker-compose.yml` закомментируйте или удалите секцию с сервисом БД (`auth_cache`), который будет заменен на внешний сервис, зависимости от него других контейнеров (все упоминания сервиса БД в блоках `depends_on`), и его вольюм (`auth-cache-volume`) в списке `volumes` в файле `docker-compose.yml` .

3. В `.env` -файле укажите данные для подключения к внешней БД, где `external-server` — IP или DNS-имя хоста, на котором установлен Redis (без указания протокола и порта). `AUTH_CACHE_CONNECTION_STRING=external-server`

4. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120 --remove-orphans
```

Включение аутентификации в Redis

1. Добавьте пароль в базу данных Redis.

```sh
auth-cache:
<...>
command: >
redis-server
--appendonly yes
--requirepass "${REDIS_PASSWORD}"
```

2. Раскомментируйте в файле `.env` следующую строку:

```sh
REDIS_PASSWORD=<YOUR_PASSWORD>
```

3. Отредактируйте в файле `.env` строку `AUTH_CACHE_CONNECTION_STRING`, добавив в нее `password=<YOUR_PASSWORD>`, например:

```sh
AUTH_CACHE_CONNECTION_STRING=auth-cache:6379,password=
<YOUR_PASSWORD>
```

Настройка безопасного соединения

1. Подготовьте файлы сертификатов — корневой `ca.crt`, а также подписанные с помощью него сертификат и ключ сервера `server.crt` и `server.key`. CN сертификата сервера должен быть `auth-cache`. Названия сертификатов и ключа могут быть любыми.

2. Настройте права пользователя `redis (999)` в файле `server.key`:

```sh
chown 999 server.key
chgrp 999 server.key
chmod 600 server.key
```

3. Скопируйте файлы `server.crt` и `server.key` для `auth-cache` в вольюм:

```sh
server_certs=$(docker inspect testit_auth-cache-tls-volume --
format '{{ .Mountpoint }}')
cp server.crt ${server_certs}/
cp server.key ${server_certs}/
```

4. Скопируйте файл сертификата CA, с помощью которого были выписаны сертификаты серверов, в вольюм `trusted-certificates-volume` :

```sh
trusted_certs=$(docker inspect testit_trusted-certificates-volume --format '{{ .Mountpoint }}')
cp ca.crt ${trusted_certs}/
```

5. В файл `docker-compose.yml` :

   - Добавьте в секции `auth-cache` под секцией `command` следующие строки:

```sh
auth-cache:
<...>
command: >
<...>
--tls-port 6379
--port 0
--tls-cert-file "${REDIS_TLS_CERT_FILE}"
--tls-key-file "${REDIS_TLS_KEY_FILE}"
--tls-auth-clients no
```

   - В файле `.env` отредактируйте строку `AUTH_CACHE_CONNECTION_STRING` , добавив `ssl=true` :

```sh
AUTH_CACHE_CONNECTION_STRING=auth-cache:6379,password=<YOUR_PASSWORD>,ssl=true
```

   - Раскомментируйте следующие строки и отредактируйте при необходимости:

```sh
REDIS_TLS_CERT_FILE=/tls/server.crt
REDIS_TLS_KEY_FILE=/tls/server.key
```

Обновлено: 14.10.2024, 16:15:16

Подключение InfluxDB в Docker Compose

> **Назовите свой проект**
>
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

1. При настройке внешней базы данных InfluxDB установите следующие параметры:

```text
max-series-per-database = 0
max-values-per-tag = 0
```

2. Закомментируйте или удалите секцию с сервисом БД (`influxdb`), который будет заменен на внешний сервис, зависимости от него других контейнеров (все упоминания сервиса БД в блоках `depends_on`), и его вольюм (`influx-volume`) в списке `volumes` в файле `docker-compose.yml` .

3. В `.env` -файле укажите данные для подключения к внешней БД, где `external-server` — IP-адрес или DNS-имя вашего сервера с InfluxDB.

```
INFLUX_CONNECTION_STRING=http://external-server:8086
```

4. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120 --remove-orphans
```

Включение аутентификации в InfluxDB

1. В файле `.env` раскомментируйте и при необходимости отредактируйте следующие строки:

```text
INFLUX_AUTH_ENABLED=true
INFLUX_USERNAME=<USER_NAME>
INFLUX_PASSWORD=<YOUR_PASSWORD>
INFLUXDB_META_DIR=/var/lib/influxdb/meta2
```

2. Выполните следующую команду:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Настройка безопасного соединения

1. Подготовьте файлы сертификатов — корневой `ca.crt`, а также подписанные с помощью него сертификат и ключ сервера `server.crt` и `server.key`. CN сертификата сервера должен быть `influxdb`.

2. Скопируйте файлы `server.crt` и `server.key` для `influxdb` с вольюм:

```sh
server_certs=$(docker inspect testit_influx-tls-volume --format '{{ .Mountpoint }}')
cp server.crt ${server_certs}/
cp server.key ${server_certs}/
```

3. Скопируйте файл сертификата CA, с помощью которого были выписаны сертификаты серверов, в вольюм `trusted-certificates-volume`:

```sh
trusted_certs=$(docker inspect testit_trusted-certificates-volume --format '{{ .Mountpoint }}')
cp ca.crt ${trusted_certs}/
```

4. В файле `.env`:

   - Измените значение переменной для подключения к `influxdb` с использованием HTTPS:

```sh
INFLUX_CONNECTION_STRING=https://influxdb:8086
```

- Раскомментируйте и при необходимости отредактируйте следующие строки:

```text
INFLUXDB_HTTP_HTTPS_ENABLED=true
INFLUXDB_HTTP_HTTPS_CERTIFICATE=/var/lib/influxdb/tls/server.crt
INFLUXDB_HTTP_HTTPS_PRIVATE_KEY=/var/lib/influxdb/tls/server.key
```

5. Чтобы изменения вступили в силу, перезапустите сервис `influxdb`, а затем примените изменения в файле `.env` с помощью `docker compose`:

```sh
docker restart testit_influxdb_1
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Обновлено: 14.10.2024, 16:15:16

Подключение PostgreSQL в Docker Compose

> **Внимание**
>
> - Перед настройкой подключения убедитесь, что дата и время в установках сервера синхронизированы с датой и временем в установках внешней базы данных. Несовпадение даты и времени может привести к нарушениям работы лицензии.
> - Версия внешнего сервиса должна совпадать с версией, указанной в файле `docker-compose.yml` .

> **Назовите свой проект**
>
> - В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

1. Подготовьте внешнюю базу данных (PostgreSQL) для каждого сервиса. Имена баз данных: `testitdb` , `authdb` , `avatarsdb` , `backgrounddb` , `licensedb` и `globalsearchdb` . Используйте команду:

```text
yum install postgresql-contrib
psql -U postgres
create database testitdb;
create user tester with encrypted password 'tester';
grant all privileges on database testitdb to tester;
\connect testitdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
```

2. Для остальных БД (`authdb` , `avatarsdb` , `backgrounddb` , `licensedb` и `globalsearchdb`) нужно выполнить скрипт ниже подставив название БД вместо `testitdb` :

   - Для `authdb` :

```text
create database authdb;
grant all privileges on database authdb to tester;
\connect authdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
```

- Для `avatarsdb`:

```text
create database avatarsdb;
grant all privileges on database avatarsdb to tester;
\connect avatarsdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
```

- Для `backgrounddb`:

```text
create database backgrounddb;
grant all privileges on database backgrounddb to tester;
\connect backgrounddb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
```

- Для `licensedb`:

```text
create database licensedb;
grant all privileges on database licensedb to tester;
\connect licensedb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
```

- Для `globalsearchdb`:

```text
create database globalsearchdb;
grant all privileges on database globalsearchdb to tester;
\connect globalsearchdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
CREATE EXTENSION if not exists "pg_trgm" SCHEMA public;
```

3. Закомментируйте или удалите секцию с сервисом БД (`db`), который будет заменен на внешний сервис, зависимости от него других контейнеров (все упоминания сервиса БД в блоках `depends_on`), и его вольюм (`db-volume`) в списке `volumes` в файле `docker-compose.yml` .

4. В `.env` -файле укажите данные для подключения к внешней БД. (`ip-external server/dns-name` , `port` , `login` , `password`). В Host можно указать отдельные СУБД или одну и ту же СУБД для каждой БД.

```text
DB_CONNECTION_STRING=Host=external_server1;Port=5432;Database=testi
Pool Size=130
#POSTGRES_DB=testitdb
#POSTGRES_USER=postgres
#POSTGRES_PASSWORD=F1rstL0g0N!
...
AUTH_CONNECTION_STRING=Host=external_server2;Port=5432;Database=aut
Pool Size=130
#POSTGRES_AUTH_DB=authdb
#POSTGRES_AUTH_USER=postgres
#POSTGRES_AUTH_PASSWORD=F1rstL0g0N!
...
AVATARS_CONNECTION_STRING=Host=external_server3;Port=5432;Database=
#POSTGRES_AVATARS_DB=avatarsdb
#POSTGRES_AVATARS_USER=postgres
#POSTGRES_AVATARS_PASSWORD=F1rstL0g0N!
...
BACKGROUND_CONNECTION_STRING=Host=external_server3;Port=5432;Databc
#POSTGRES_BACKGROUND_DB=backgrounddb
#POSTGRES_BACKGROUND_USER=postgres
#POSTGRES_BACKGROUND_PASSWORD=F1rstL0g0N!
...
LICENSE_DB_CONNECTION_STRING=Host=external_server3;Port=5432;Databc
#POSTGRES_LICENSE_DB=licensedb
#POSTGRES_LICENSE_USER=postgres
#POSTGRES_LICENSE_PASSWORD=F1rstL0g0N!
...
GLOBALSEARCH_CONNECTION_STRING=Host=external_server3;Port=5432;Datc
#POSTGRES_GLOBALSEARCH_DB=globalsearchdb
#POSTGRES_GLOBALSEARCH_USER=postgres
#POSTGRES_GLOBALSEARCH_PASSWORD=F1rstL0g0N!
```

5. Сконфигурируйте PostgreSQL вручную для внешних подключений на сервере расположения PostgreSQL, так как по умолчанию она не сконфигурирована для внешних подключений.

> **Внимание**

> Данный шаг не актуален, если вы используете кластер БД или БД в вашем окружении предоставляется как управляемое решение (вы не администрируете сервер БД). В таком случае необходимо обеспечить доступ к БД из контейнеров Test IT доступными в вашей конфигурации способами.

Для этого выполните следующие действия:

- В файле `postgresql.conf` закомментируйте следующую строку:

```text
# listen_addresses = 'localhost'
```

- Откройте прослушивание на всех интерфейсах. Для этого добавьте следующую строку:

```text
listen_addresses = '*'
```

- Добавьте следующие строки в файл `pg_hba.conf` :

```text
host all all 0.0.0.0/0 md5
host all all ::/0 md5
```

  Первая строка — для адресов IPv4, вторая — для адресов IPv6. Такая конфигурация позволяет PostgreSQL принимать соединения из любых сетей.
- Если вы хотите ограничить возможность подключения определенной подсетью, укажите адрес подсети с маской в формате CIDR, например:

```text
host all all 192.168.101.20/24 md5
```

  Подсети, в которых находятся машины с сервисами `rabbitmq_consumer` , `webapi` , `auth` и `avatars.api` , должны быть добавлены сюда, если вы не используете `0.0.0.0/0` .
- Убедитесь, что порт, на котором слушает PostgreSQL (5432 по умолчанию), открыт в настройках брандмауэра вашей ОС.

6. Выполните установку Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Настройка безопасного соединения

1. Для настройки безопасного соединения с сервисом `db` : подготовьте сертификаты — корневой `ca.crt` , а также подписанные с помощью него сертификаты и ключи серверов `server.crt` и `server.key` . Если будет включена проверка доменного имени при создании соединения, то CN сертификатов должен быть `db` соответственно. Названия файлов сертификатов сервера могут быть любыми.

2. В файле `docker-compose.yml` добавьте в секцию `db` в строку command следующее:

```sh
db:
<...>
command: postgres -c 'max_connections=300' -c 'shared_buffers=256MB' -c 'ssl=on' -c 'ssl_cert_file=/var/lib/postgresql/tls/server.crt' -c 'ssl_key_file=/var/lib/postgresql/tls/server.key'
```

3. В файле `.env` отредактируйте следующие строки, добавив `Ssl Mode=VerifyFull` для проверки подписи CA и домена, либо `Ssl Mode=VerifyCA` для проверки только CA:

   - `DB_CONNECTION_STRING`
   - `AUTH_CONNECTION_STRING`
   - `AVATARS_CONNECTION_STRING`
   - `BACKGROUND_CONNECTION_STRING`
   - `LICENSE_DB_CONNECTION_STRING`
   - `GLOBALSEARCH_CONNECTION_STRING`

   Например:

```sh
DB_CONNECTION_STRING=Host=db;Port=5432;Database=testitdb;Username=p
Pool Size=130; Ssl Mode=VerifyFull
```

4. Выполните следующую команду:

```sh
trusted_certs=$(docker inspect yourproject_trusted-certificates-volume --format '{{ .Mountpoint }}')
cp ca.crt ${trusted_certs}/
```

`ca.crt` — один корневой сертификат. Если все сертификаты выписаны одним CA, выполните вышеуказанную команду несколько раз, заменяя ca.crt на те CA, которыми выписаны сертификаты серверов.

5. Настройте права пользователя `(999)` в файле `server.key` :

```sh
chown 999 server.key
chgrp 999 server.key
chmod 600 server.key
```

6. Скопируйте файлы `server.crt` и `server.key` в соответствующий вольюм контейнера `db` :

```sh
server_certs=$(docker inspect yourproject_db-tls-volume --format '{{ .Mountpoint }}')
cp server.key server.crt ${server_certs}/
```

7. Выполните следующие команды:

```sh
docker compose -f docker-compose.yml --project-name testit restart
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Обновлено: 14.10.2024, 16:15:16

Подключение RabbitMQ в Kubernetes

Перед настройкой необходимо создать virtualhost testitrabbit и пользователя testit с правами на virtualhost testitrabbit.

> **Проверьте совместимость версии БД**
>
> Перед началом работы убедитесь, что версия внешней базы данных RabbitMQ совпадает с версией, указанной в `.Values.rabbitmq.image.tag` .

1. Если система Test IT запущена, остановите все поды с помощью команды.
2. В файле `values-override.yaml` установите переменные среды в с параметрами внешней базы данных RabbitMQ:

```yml
general:
config:
RABBITMQ_DEFAULT_VHOST: "testitrabbit"
RABBITMQ_DEFAULT_HOST: "external-server" # IP or DNS of outside
RabbitMQ server
RABBITMQ_DEFAULT_PORT: "5672"
RABBITMQ_AUTH_MODE: "plain"
RABBITMQ_CLIENT_CERT_PATH:
"/etc/rabbitmq/ssl/client/testit.pfx"
RABBITMQ_CLIENT_CERT_PASSPHRASE:
RABBITMQ_SSL_ENABLED: "false"
secrets:
RABBITMQ_DEFAULT_USER: "testit"
RABBITMQ_DEFAULT_PASS: "password"
```

3. В файле `values-override.yaml` отключите контроллер statefulSet для внутренней базы данных RabbitMQ:

```yml
rabbitmq:
enabled: false
```

4. Примените изменения с помощью команды:

```sh
cd <installation_folder>
helm -n <namespace> -f testit_unichart/values-override.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Настройка SSL для внешних подключений

> **Важно**
>
> 1. Убедитесь, что конфигурация-SSL во внешних сервисах настроена на проверку только файла CA.crt.
> 2. Если для внешних сервисов используются другие файлы CA.crt, убедитесь, что все они добавлены в связку. Добавьте файл ca-bundle-*.crt в директорию `testit_unichart/files/ssl/` :
>
>    ```sh
>    # Номер файла задан для примера, можно использовать и другое наименование, например ca-bundle-rabbitmq.crt
>    # Главное, чтобы нужные путь/название файла указывались в шаге 3 следующей секции testit_unichart/files/ssl/ca-bundle-2.crt
>    ```
>
> 3. Прежде чем применять изменения, описанные в данной инструкции, убедитесь, что настроены внешние подключения к соответствующим сервисам.

1. Создайте SSL-сертификаты, используя доменное имя (CN) вашего сервера RabbitMQ и файл CA.crt:

- rabbitmq_key.pem
- rabbitmq_cert.pem
- rabbitmq_ca.pem

2. Перенесите сертификаты в соответствующую папку на вашем сервере RabbitMQ. Например, `/etc/certs` .
3. Создайте конфигурационный файл `.conf` . Например, `20-ssl.conf` :

```hcl
listeners.ssl.default = 5671
ssl_options.cacertfile = /etc/certs/rabbitmq_ca.pem
ssl_options.certfile = /etc/certs/rabbitmq_cert.pem
ssl_options.keyfile = /etc/certs/rabbitmq_key.pem
ssl_options.verify = verify_none
ssl_options.fail_if_no_peer_cert = false
```

4. Перенесите созданный файл в в соответствующую папку на вашем сервере RabbitMQ. Например, `/etc/rabbitmq/conf.d` .
5. Добавьте в секцию `general.configFile` файла `values-override.yaml` или `values-ssl.yaml` путь до нового CA.crt:

```yml
# testit_unichart/values-ssl.yaml или testit_unichart/values-override.yaml
general:
config:
RABBITMQ_DEFAULT_VHOST: "testitrabbit"
RABBITMQ_DEFAULT_HOST: "external-server" # IP or DNS of outside RabbitMQ server
RABBITMQ_DEFAULT_PORT: "5672"
RABBITMQ_AUTH_MODE: "plain"
RABBITMQ_CLIENT_CERT_PATH: "/etc/rabbitmq/ssl/client/testit.pfx"
RABBITMQ_CLIENT_CERT_PASSPHRASE:
RABBITMQ_SSL_ENABLED: "false"
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
secrets:
RABBITMQ_DEFAULT_USER: "testit"
RABBITMQ_DEFAULT_PASS: "password"
volumes:
## Use to enable custom certificates
ssl-ca-bundle-1:
fromConfigMap: ssl-ca-bundle-1
mounts:
mountPath: /app/certs/ca-bundle-1.crt
subPath: ca-bundle-1.crt
# Mount new file
ssl-ca-bundle-2:
fromConfigMap: ssl-ca-bundle-2
mounts:
mountPath: /app/certs/ca-bundle-2.crt
subPath: ca-bundle-2.crt
configFile:
## Use to enable custom certificates
# Already have been there (hypothetically)
ssl-ca-bundle-1:
ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
# New file
ssl-ca-bundle-2:
ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

6. Примените изменения с помощью команды:

```sh
cd <installation_folder>
# В зависимости от того, какие файлы (values-override.yaml или values-ssl.yaml) используется, прокидывайте в команду соответствующий файл через флаг -f
helm -n <namespace> -f testit_unichart/values-override.yaml -f testit_unichart/values-ssl.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Обновлено: 30.05.2025, 13:38:26

Подключение MinIO в Kubernetes

> **Проверьте совместимость версии БД**
>
> Перед началом работы убедитесь, что версия внешней базы данных MinIO совпадает с версией, указанной в `.Values.minio.image.tag` .

1. Если система Test IT запущена, остановите все поды с помощью **команды**.
2. Во внешнем сервисе MinIO создайте 2 бакета: для minio и avatars.minio (например *bucket1* и *bucket2*).
3. Создайте ключ доступа (access key) и секретный ключ (secret key) для доступа к каждому из бакетов.
4. В файле `values-override.yaml` отключите контроллер statefulSet для внутренней базы данных MinIO:

```yml
minio:
enabled: false
```

5. В файле `values-override.yaml` задайте необходимую переменную для подключения к внешней базе данных MinIO:

```yml
avatars-api:
config:
AVATARS_AWS_BUCKET_NAME: "bucket2"
general:
config:
AWS_CONNECTION_STRING: "http://minio-external:9000" # minio-
external could be either IP or DNS
TMS_BUCKET_NAME: "bucket1"
secrets:
AWS_ACCESS_KEY: "YourAccessKey"
AWS_SECRET_KEY: "YourSecretKey"
```

6. Примените изменения с помощью команды:

```sh
cd <installation_folder>
helm -n <namespace> -f testit_unichart/values-override.yaml upgrade testit testit_unichart/ --wait --timeout 10m
```

Настройка SSL для внешних подключений

> **Важно**
>
> 1. Убедитесь, что конфигурация-SSL во внешних сервисах настроена на проверку только файла CA.crt.
> 2. Если для внешних сервисов используются другие файлы CA.crt, убедитесь, что все они добавлены в связку. Добавьте файл ca-bundle-*.crt в директорию `testit_unichart/files/ssl/` :
>
>    ```sh
>    # Номер файла задан для примера, можно использовать и другое наименование, например ca-bundle-minio.crt
>    # Главное, чтобы нужные путь/название файла указывались в шаге 3 следующей секции testit_unichart/files/ssl/ca-bundle-2.crt
>    ```
>
> 3. Прежде чем применять изменения, описанные в данной инструкции, убедитесь, что **настроены внешние подключения к соответствующим сервисам**.

1. Создайте SSL-сертификаты, используя доменное имя (CN) вашего сервера MinIO и файл CA.crt:
   - ca.crt
   - server-minio.crt
   - server-minio.key
2. Перенесите сертификаты в соответствующую папку на вашем сервере MinIO. Например:
   - ~/.minio/certs/CAs/ca.crt
   - ~/.minio/certs/server-minio.crt
   - ~/.minio/certs/server-minio.key
3. Добавьте в секцию `general.configFile` файла `values-override.yaml` или `values-ssl.yaml` путь до нового CA.crt:

```yml
# testit_unichart/values-ssl.yaml или testit_unichart/values-override.yaml
general:
config:
AWS_CONNECTION_STRING: "https://minio-external:9000"
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
volumes:
## Use to enable custom certificates
ssl-ca-bundle-1:
fromConfigMap: ssl-ca-bundle-1
mounts:
mountPath: /app/certs/ca-bundle-1.crt
subPath: ca-bundle-1.crt
# Mount new file
ssl-ca-bundle-2:
fromConfigMap: ssl-ca-bundle-2
mounts:
mountPath: /app/certs/ca-bundle-2.crt
subPath: ca-bundle-2.crt
configFile:
## Use to enable custom certificates
# Already have been there (hypothetically)
ssl-ca-bundle-1:
ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
# New file
ssl-ca-bundle-2:
ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

4. Примените изменения с помощью команды:

```sh
cd <installation_folder>
# В зависимости от того, какие файлы (values-override.yaml или values-ssl.yaml) используется, прокидывайте в команду соответствующий файл через флаг -f
helm -n <namespace> -f testit_unichart/values-override.yaml -f testit_unichart/values-ssl.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Обновлено: 30.05.2025, 13:38:26

Подключение Redis в Kubernetes

> **Проверьте совместимость версии БД**
>
> Перед началом работы убедитесь, что версия внешней базы данных Redis совпадает с версией, указанной в `.Values.auch-cache.image.tag` .

1. Если система Test IT запущена, остановите все поды с помощью **команды**.
2. При настройке конфигурации внешней базы данных Redis убедитесь, что установлен следующий параметр:

```sh
appendonly yes
```

3. В файле `values-override.yaml` установите значения переменных среды со строкой подключения внешней базы данных Redis:

```yml
general:
secrets:
AUTH_CACHE_CONNECTION_STRING: "redis-external" # IP or DNS of outside Redis server
```

   Если внешний Redis защищен паролем, строка подключения будет иметь вид:

```text
AUTH_CACHE_CONNECTION_STRING=redis-external,password=<YOUR_PASSWORD>
```

4. В файле `values-override.yaml` отключите контроллер statefulSet для внутренней базы данных Redis:

```yml
auth-cache:
enabled: false
```

   Затем добавьте

```yml
auth:
envFromSecret:
AUTH_CACHE_CONNECTION_STRING:
secretName: general-secret
secretKey: AUTH_CACHE_CONNECTION_STRING
```

5. Примените изменения с помощью команды:

```sh
cd <installation_folder>
helm -n <namespace> -f testit_unichart/values-override.yaml
upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Настройка SSL для внешних подключений

> **Важно**
>
> 1. Убедитесь, что конфигурация-SSL во внешних сервисах настроена на проверку только файла CA.crt.
> 2. Если для внешних сервисов используются другие файлы CA.crt, убедитесь, что все они добавлены в связку. Добавьте файл ca-bundle-*.crt в директорию `testit_unichart/files/ssl/` :
>
>    ```sh
>    # Номер файла задан для примера, можно использовать и
>    другое наименование, например ca-bundle-redis.crt
>    # Главное, чтобы нужные путь/название файла указывались в
>    шаге 3 следующей секции
>    testit_unichart/files/ssl/ca-bundle-2.crt
>    ```
>
> 3. Прежде чем применять изменения, описанные в данной инструкции, убедитесь, что настроены внешние подключения к соответствующим сервисам.

1. Создайте SSL-сертификаты, используя доменное имя (CN) вашего сервера Redis и файл CA.crt:
   - ca.crt
   - redis.crt
   - redis.key

2. Перенесите сертификаты в соответствующую папку на вашем сервере Redis. Например:
   - /tls/ca.crt
   - /tls/redis.crt
   - /tls/redis.key
3. При запуске сервера Redis убедитесь, что добавлены следующие параметры запуска:

```sh
redis-server \
--appendonly yes \
--tls-port 6379 \
--port 0 \
--tls-cert-file /tls/redis.crt \
--tls-key-file /tls/redis.key \
--tls-ca-cert-file /tls/ca.crt \
--tls-auth-clients no
```

4. Добавьте в секцию `general.configFile` файла `values-override.yaml` или `values-ssl.yaml` путь до нового CA.crt:

```yml
# testit_unichart/values-ssl.yaml или testit_unichart/values-override.yaml
general:
config:
AUTH_CACHE_CONNECTION_STRING: "redis-external" # IP or DNS of outside Redis server
volumes:
## Use to enable custom certificates
ssl-ca-bundle-1:
fromConfigMap: ssl-ca-bundle-1
mounts:
mountPath: /app/certs/ca-bundle-1.crt
subPath: ca-bundle-1.crt
# Mount new file
ssl-ca-bundle-2:
fromConfigMap: ssl-ca-bundle-2
mounts:
mountPath: /app/certs/ca-bundle-2.crt
subPath: ca-bundle-2.crt
configFile:
## Use to enable custom certificates
# Already have been there (hypothetically)
ssl-ca-bundle-1:
ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
# New file
ssl-ca-bundle-2:
ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

5. Примените изменения с помощью команды:

```sh
cd <installation_folder>
# В зависимости от того, какие файлы (values-override.yaml или values-ssl.yaml) используется, прокидывайте в команду соответствующий файл через флаг -f
helm -n <namespace> -f testit_unichart/values-override.yaml -f testit_unichart/values-ssl.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Обновлено: 30.05.2025, 13:38:26

Подключение InfluxDB в Kubernetes

> **Проверьте совместимость версии БД**
>
> Перед началом работы убедитесь, что версия внешней базы данных InfluxDB совпадает с версией, указанной в `.Values.influxdb.image.tag` .

Настройка подключения

1. Если система Test IT запущена, остановите все поды с помощью **команды**.
2. При настройке внешнего сервера InfluxDB убедитесь, что установлены следующие параметры:

```sh
max-series-per-database = 0
max-values-per-tag = 0
```

3. В файле `values-override.yaml` отключите контроллер statefulSet для внутренней базы данных InfluxDB:

```yml
influxdb:
enabled: false
```

4. В файле `values-override.yaml` задайте необходимые переменные для подключения к внешней базе данных InfluxDB:
   - Если на вашем InfluxDB-сервере настроена авторизация, используйте следующую конфигурацию:

```yml
general:
config:
INFLUX_CONNECTION_STRING: "http://influxdb-external:8086" #
influxdb-external could be either IP or DNSservice
INFLUX_AUTH_ENABLED: "true"
secrets:
INFLUX_USERNAME: "influx" # your influx username
INFLUX_PASSWORD: "influxpass" # your influx password
```

- Если авторизация не настроена, используйте следующую конфигурацию:

```yml
general:
config:
INFLUX_CONNECTION_STRING: "http://influxdb-external:8086" #
influxdb-external could be either IP or DNSservice
INFLUX_AUTH_ENABLED: "false"
```

5. Примените изменения с помощью команды:

```sh
cd <installation_folder>
helm -n <namespace> -f testit_unichart/values-override.yaml
upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Настройка SSL для внешних подключений

> **Важно**
>
> 1. Убедитесь, что конфигурация-SSL во внешних сервисах настроена на проверку только файла CA.crt.
> 2. Если для внешних сервисов используются другие файлы CA.crt, убедитесь, что все они добавлены в связку. Добавьте файл ca-bundle-*.crt в директорию `testit_unichart/files/ssl/` :
>
> ```sh
> # Номер файла задан для примера, можно использовать и
> другое наименование, например ca-bundle-influxdb.crt
> # Главное, чтобы нужные путь/название файла указывались в
> шаге 3 следующей секции
> testit_unichart/files/ssl/ca-bundle-2.crt
> ```

> 3. Прежде чем применять изменения, описанные в данной инструкции, убедитесь, что **настроены внешние подключения к соответствующим сервисам**.

1. Создайте SSL-сертификаты, используя доменное имя (CN) вашего сервера InfluxDB и файл CA.crt:
   - ca.crt
   - server.crt
   - server.key
2. Перенесите сертификаты в соответствующую папку на вашем сервере InfluxDB. Например:
   - /var/lib/influxdb/tls/ca.crt
   - /var/lib/influxdb/tls/server.crt
   - /var/lib/influxdb/tls/server.key
3. Добавьте в секцию `general.configFile` файла `values-override.yaml` или `values-ssl.yaml` путь до нового CA.crt:

```yml
# testit_unichart/values-ssl.yaml или testit_unichart/values-override.yaml
general:
config:
INFLUX_CONNECTION_STRING: "https://influxdb:8086"
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
volumes:
## Use to enable custom certificates
ssl-ca-bundle-1:
fromConfigMap: ssl-ca-bundle-1
mounts:
mountPath: /app/certs/ca-bundle-1.crt
subPath: ca-bundle-1.crt
# Mount new file
ssl-ca-bundle-2:
fromConfigMap: ssl-ca-bundle-2
mounts:
mountPath: /app/certs/ca-bundle-2.crt
subPath: ca-bundle-2.crt
configFile:
## Use to enable custom certificates
# Already have been there (hypothetically)
ssl-ca-bundle-1:
ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
# New file
ssl-ca-bundle-2:
ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

4. Примените изменения с помощью команды:

```sh
cd <installation_folder>
# В зависимости от того, какие файлы (values-override.yaml или values-ssl.yaml) используется, прокидывайте в команду соответствующий файл через флаг -f
helm -n <namespace> -f testit_unichart/values-override.yaml -f testit_unichart/values-ssl.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Обновлено: 30.05.2025, 14:11:50

Подключение PostgreSQL в Kubernetes

> **Проверьте совместимость версии БД**
>
> Перед началом работы убедитесь, что версия внешней базы данных PostgreSQL совпадает с версией, указанной в `.Values.postgresql.image.tag`.

1. Если система Test IT запущена, остановите все поды с помощью команды.
2. Подготовьте внешнюю базу данных PostgreSQL для каждого из сервисов (`testitdb`, `authdb`, `avatarsdb`, `licensedb`, `backgrounddb`, `globalsearchdb`):

```sh
yum install postgresql-contrib
psql -U postgres
create database testitdb;
create user tester with encrypted password 'tester';
grant all privileges on database testitdb to tester;
\connect testitdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;

create database authdb;
grant all privileges on database authdb to tester;
\connect authdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;

create database avatarsdb;
grant all privileges on database avatarsdb to tester;
\connect avatarsdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;

create database backgrounddb;
grant all privileges on database backgrounddb to tester;
\connect backgrounddb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;

create database licensedb;
grant all privileges on database licensedb to tester;
\connect licensedb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;

create database globalsearchdb;
grant all privileges on database globalsearchdb to tester;
\connect globalsearchdb;
CREATE EXTENSION if not exists "uuid-ossp" SCHEMA public;
CREATE EXTENSION if not exists "pg_trgm" SCHEMA public;
```

3. В файле `values-override.yaml` задайте хост, порт, имя пользователя, пароль и корректные имена баз данных. Все строки подключения к БД могут быть найдены поиском строки `Host=postgres` по файлу `values.yaml`

```yml
# Пример для license-service
license-service:
config:
LICENSE_DB_CONNECTION_STRING: >
Host=my-external-postgres-host;
Port=5432;
Database=licensedb;
Username=my-username;
Password=my-password;
Command Timeout=30;
Target Session Attributes=any;
```

4. Примените изменения с помощью команды:

```sh
cd <installation_folder>
helm -n <namespace> -f testit_unichart/values-override.yaml
upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Настройка SSL для внешних подключений

> **Важно**
>
> 1. Убедитесь, что конфигурация-SSL во внешних сервисах настроена на проверку только файла CA.crt.
> 2. Если для внешних сервисов используются другие файлы CA.crt, убедитесь, что все они добавлены в связку. Добавьте файл ca-bundle-*.crt в директорию `testit_unichart/files/ssl/` :
>
>    ```sh
>    # Номер файла задан для примера, можно использовать и
>    другое наименование, например ca-bundle-postgres.crt
>    # Главное, чтобы нужные путь/название файла указывались в
>    шаге 3 следующей секции
>    testit_unichart/files/ssl/ca-bundle-2.crt
>    ```
>
> 3. Прежде чем применять изменения, описанные в данной инструкции, убедитесь, что настроены внешние подключения к соответствующим сервисам.

1. Создайте SSL-сертификаты, используя доменное имя (CN) вашего сервера Postgres и файл CA.crt:
   - ca.crt
   - server.crt
   - server.key
2. Перенесите сертификаты в соответствующую папку на вашем сервере Postgres. Например:
   - /var/lib/postgresql/tls/ca.crt
   - /var/lib/postgresql/tls/server.crt
   - /var/lib/postgresql/tls/server.key
3. При запуске сервера Postgres убедитесь, что расположение сертификатов передано верно:

```sh
postgres -c 'max_connections=300' -c 'shared_buffers=256MB' -c
'ssl=on' \
-c 'ssl_ca_file=/var/lib/postgresql/tls/ca.crt' \
-c 'ssl_cert_file=/var/lib/postgresql/tls/server.crt' \
-c 'ssl_key_file=/var/lib/postgresql/tls/server.key'
```

4. Добавьте в секцию `general.configFile` файла `values-override.yaml` или `values-ssl.yaml` путь до нового CA.crt:

```yml
# testit_unichart/values-ssl.yaml или testit_unichart/values-override.yaml
general:
config:
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
volumes:
## Use to enable custom certificates
ssl-ca-bundle-1:
fromConfigMap: ssl-ca-bundle-1
mounts:
mountPath: /app/certs/ca-bundle-1.crt
subPath: ca-bundle-1.crt
# Mount new file
ssl-ca-bundle-2:
fromConfigMap: ssl-ca-bundle-2
mounts:
mountPath: /app/certs/ca-bundle-2.crt
subPath: ca-bundle-2.crt
configFile:
## Use to enable custom certificates
# Already have been there (hypothetically)
ssl-ca-bundle-1:
ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
# New file
ssl-ca-bundle-2:
ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

5. Примените изменения с помощью команды:

```sh
cd <installation_folder>
# В зависимости от того, какие файлы (values-override.yaml или values-ssl.yaml) используется, прокидывайте в команду соответствующий файл через флаг -f
helm -n <namespace> -f testit_unichart/values-override.yaml -f testit_unichart/values-ssl.yaml upgrade --install testit testit_unichart/ --wait --timeout 10m
```

Обновлено: 15.04.2025, 13:00:12

Подключение хранилища Яндекс S3 в Kubernetes

Для подключения потребуется:

- Test IT Enterprise, развернутая в Kubernetes (рекомендуется использовать последнюю версию)
- Учетная запись Yandex Cloud

Настройка включает:

- Получение идентификатора и секретного ключа в Yandex Cloud
- Добавление идентификатора и секретного ключа в Test IT

Получение идентификатора и секретного ключа в Yandex Cloud

> **Сохраните идентификатор и секретный ключ**
>
> После закрытия диалога значение ключа будет недоступно.

1. В консоли управления на панели сверху нажмите v и выберите каталог, которому принадлежит сервисный аккаунт.
2. В списке сервисов выберите **Identity and Access Management**.
3. На панели слева выберите **Сервисные аккаунты**.
4. Выберите сервисный аккаунт, для которого вы хотите создать статический ключ доступа.
5. На панели сверху нажмите кнопку **Создать новый ключ** и выберите Создать статический ключ доступа.
6. Задайте описание ключа и нажмите кнопку **Создать**.
7. Сохраните идентификатор и секретный ключ.

Подробности читайте в официальной инструкции Yandex Cloud .

Добавление идентификатора и секретного ключа в Test IT

Изменения вносятся в файл `values.yaml`. Подробности работы с файлом смотрите в инструкции Подключение MinIO в Kubernetes.

1. Отключите MinIO:

```yml
minio:
enabled: false
```

2. Добавьте идентификатор доступа и секретный ключ в соответствующие поля в файл `values.yaml`:

```yml
# идентификатор (ключ доступа)
AWS_ACCESS_KEY: "<access key>"
```

```yml
# секретный ключ
AWS_SECRET_KEY: "<secret key>"
```

3. В строке `connection_string` укажите **https://storage.yandexcloud.net**.

```yml
AWS_CONNECTION_STRING: "https://storage.yandexcloud.net"
```

4. Если бакеты для `AVATARS_AWS` и `TMS` не создались автоматически, создайте их вручную:
   а. В файле `values.yaml` у переменной `AWS_CREATE_BUCKET_IF_REQUIRED` укажите значение `false`.
   б. В Yandex Cloud создайте бакеты вручную, следуя **инструкции Yandex Cloud**. При необходимости замените названия бакетов.
   в. В файл `values.yaml` добавьте названия созданных бакетов как значения переменных `AVATARS_AWS_BUCKET_NAME` и `TMS_BUCKET_NAME`.
5. Перезапустите поды с помощью команды:

```sh
kubectl delete pods -n <namespace> --all -
```

6. Убедитесь, что TMS работает корректно, изменив аватар и добавив вложение в тест-кейс. Файлы должны появиться в соответствующих бакетах.

Обновлено: 22.01.2026, 18:39:55

Описание микросервисов Test IT

Описание микросервисов для **Test IT 5.3 Reticulum** и более поздних версий.

#	Контейнер	Базовое ПО	Назначение
Сервисы Test IT			
1	auth	.NET	Аутентификация и авторизация пользователей
2	avatars.api	.NET	REST API для взаимодействия с аватарами пользователей
3	background-service	.NET	Импорт и экспорт проектных данных, периодическая очистка архивных данных и вложений
4	globalsearch-migrationtool-context	.NET	Миграция данных сервиса globalsearch-service
5	globalsearch-migrationtool-data	.NET	Миграция данных сервиса globalsearch-service
6	globalsearch-service	.NET	Глобальный поиск элементов
7	ldapwebapi	.NET	Интеграция с AD/LDAP
8	license-service	.NET	Лицензирование
9	webapi	.NET	Имплементация публичного REST API и бизнес-логики
Сторонние сервисы			
10	auth-cache	Valkey (в **Test IT 5.2** и более ранних версиях — Redis)	Кэш для работы с пользователями
11	db	PostgreSQL	Основная база данных приложения. Используется для хранения тест-кейсов, автотестов, описаний, ссылок и т.д.
12	gotenberg-api	Gotenberg	Генерация PDF для офисных и

			текстовых документов
13	imgproxy	imgproxy	Обработка изображений: изменение размера, конвертация, и др.
14	influxdb	InfluxDB	Хранение исторических данных по по количеству тестов, на основании которых строятся графики. Используется в одноименном контейнере для агрегации статистики.
15	minio	MinIO	Объектное хранилище, S3 API. Хранение файлов (вложения, логи и др.)
16	rabbitmq	RabbitMQ	Брокер сообщений. В него попадают действия, которые можно исполнить позже (по событию).

- Смотрите также: Подключение Gotenberg

Обновлено: 21.06.2025, 14:30:18

Подключение Gotenberg

- Для подключения к сервису, входящему в поставку, используйте конфигурацию:

```json
gotenberg-api:
enabled: true
excludeGeneralConfig: true
excludeGeneralVolumes: true
image:
repository: "gotenberg/gotenberg"
tag: "8.21"
pullPolicy: "IfNotPresent"
resources:
requests:
memory: "128Mi"
cpu: "250m"
limits:
memory: "500Mi"
cpu: "500m"
args:
- "gotenberg"
- "--api-port-from-env=API_PORT"
config:
API_PORT: "8080"
containerPort: 8080
service:
port: 8080
```

- Для внешнего подключения используйте конфигурацию:

```json
webapi:
enabled: true
image:
repository: "registry.testit.software/testit/testit"
resources:
requests:
memory: "512Mi"
cpu: "50m"
limits:
memory: "161"
cpu: "1"
lifecycle:
default: true
config:
AWS_FILE_EXPIRATION_IN_DAYS: "7"
Dotnet__Dll: "WebApi.dll"
Tms__DbTimeoutInSeconds: "608"
ASPNETCORE_ENVIRONMENT: "Production"
GotenbergApiUrl: "<YOUR_GOTENBERG_HOST>"
INSECURE_RENOTES: ""
USE_PKCE: "true"
```

Смотрите также:

- Официальная документация Gotenberg

Обновлено: 20.06.2025, 14:33:37

Настройка почтового SMTP-сервера для уведомлений

Вы можете настроить SMTP-сервер для уведомлений пользователей. При настройке в `.env-файле` используются следующие переменные:

#	Наименование	Описание	Тип	Знач
1	SMTP_ENABLE	Включает/ выключает отправку уведомлений через SMTP	логическое	true, false
2	SMTP_FROM	Устанавливает автора письма (от кого)	строка	произвол
3	SMTP_HOST	Устанавливает адрес для соединения с SMTP-сервером	строка	произвол
4	SMTP_PORT	Устанавливает порт для соединения с SMTP-сервером	число	произвол
5	SMTP_LOGIN	Устанавливает логин пользователя, осуществляющего соединение с SMTP-сервером	строка	произвол
6	SMTP_PASSWORD	Устанавливает пароль пользователя, осуществляющего соединение с SMTP-сервером	строка	произвол
7	SMTP_DOMAIN	Устанавливает домен пользователя, осуществляющего соединение с SMTP-сервером	строка	произвол

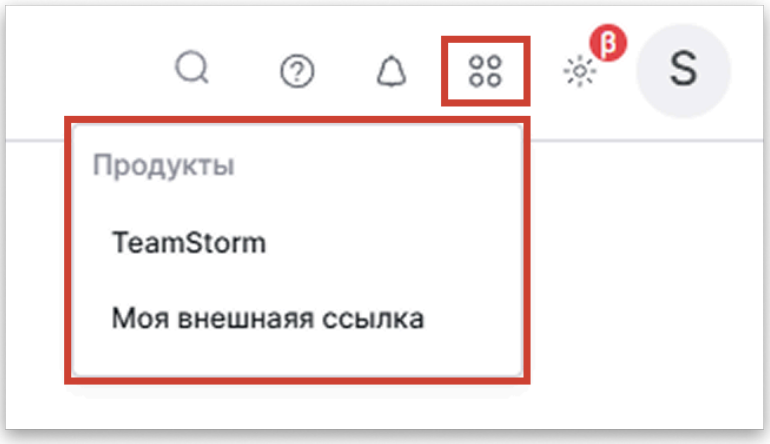
8	SMTP_CONNECTION	Устанавливает тип соединения	строка	• **none** – соедин шифро • **auto** – шифро станда сервер Если т соедин провод шифро • **ssl** — шифро SSL ил при со • **tls** — шифро TLS по привет Если с поддер START соедин будет заверш ошибк
9	SMTP_AUTHENTICATION	Устанавливает тип аутентификации	строка	• **ntlm** – аутент • **login** – аутент по лог паролн • **plain** – аутент
10	SMTP_SKIP_CERTIFICATE	Устанавливает флаг необходимости проверять	логическое	true, false

		сертификат при соединении		
11	SMTP_TLS_VERSION	Устанавливает версию TLS для соединения	строка	• **none** • **auto** • **ssl** • **tls**
12	SMTP_LOG_ENABLE	Устанавливает флаг необходимости логирования общения сервиса с SMTP-сервером	логическое	true, false

Обновлено: 16.10.2025, 18:35:17

Настройка внешних ссылок для перехода из Test IT

Вы можете настроить переход из интерфейса Test IT по внешним ссылкам — например, в другое приложение или на любой сайт. Переход осуществляется с помощью меню **Продукты** в правой части верхней навигационной панели.



Опция доступна, начиная с версии **Test IT Enterprise 5.3**.

Настройка в Docker Compose

1. В директории с продуктом создайте конфигурационный файл `external-products.json` .
   Пример директории: `config_files/external-products.json` .
2. Заполните файл по образцу:

```json
[
{ "name": "TeamStorm", "url": "https://work.teamstorm.io/" },
{ "name": "Моя внешняя ссылка", "url":
"https://docs.teamstorm.io/" }
]
```

   Значение параметров:
   - `name` — название, которое будет отображаться в меню **Продукты**
   - `url` — внешняя ссылка, которую вы хотите поместить в продукт
3. Сохраните файл.

4. В браузере перезагрузите страницу Test IT.

Настройка в Kubernetes

1. В директории с продуктом создайте конфигурационный файл `external-products.json` .
   Пример директории: `testit_unichart/files/external-products.json` .
2. Заполните файл по образцу:

```json
[
{ "name": "TeamStorm", "url": "https://work.teamstorm.io/" },
{ "name": "Моя внешняя ссылка", "url":
"https://docs.teamstorm.io/" }
]
```

   Значение параметров:
   - `name` — название, которое будет отображаться в меню **Продукты**
   - `url` — внешняя ссылка, которую вы хотите поместить в продукт
3. Сохраните файл.
4. В браузере перезагрузите страницу Test IT.
5. Обновите версию с помощью команды:

```sh
helm upgrade --install -f testit_unichart/values-override.yaml -n <my-namespace> --create-namespace testit testit_unichart/ --wait --timeout 10m
```

6. Выполните перезапуск развертывания фронтенда с помощью команды:

```sh
kubectl -n <my-namespace> rollout restart deployment frontend
```

Обновлено: 15.04.2025, 13:00:12

Перезапуск системы в Docker Compose

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

- Для перезапуска системы воспользуйтесь следующей командой:

```sh
docker compose -f docker-compose.yml --project-name testit restart --timeout 120
```

Обновлено: 07.02.2024, 17:36:29

Перезапуск системы в Kubernetes

- Для полного перезапуска системы используйте команду:

```sh
kubectl delete pods --all -n <namespace_name>
```

Обновлено: 27.03.2024, 17:54:43

Изменение выделенных ресурсов в Kubernetes

> **Важно**
>
> Рекомендуется вносить любые изменения только в файл `values-override.yaml` , а файл `values.yaml` оставить без изменений.

1. Внесите требуемые изменения.

- Чтобы внести изменения в приложения Test IT (frontend, auth, avatars-api, background-service, ldapwebapi, license-service, webapi и globalsearch-service) используйте следующий блок:

```yml
resources:
requests: # Минимальные выделенные ресурсы
memory: "3Gi"
cpu: "3"
limits: # Лимит выделенных ресурсов
memory: "5Gi"
cpu: "10"
```

- Чтобы внести изменения в сторонние приложения (auth-cache, postgres, minio, influxdb, rabbitmq и gotenberg-api) используйте следующий блок:

```yml
# Взят компонент postgres для примера, нерелевантные для смены ресурсов поля удалены из примера для прозрачности
postgres:
resources:
requests:
memory: "1Gi"
cpu: "1"
limits:
memory: "3Gi"
cpu: "3"
statefulset:
storage:
spec:
resources:
requests:
storage: "5Gi"
```

2. Примените изменения с помощью набора команд:

```sh
export NAMESPACE=<my-namespace>
helm -n $NAMESPACE list
helm -n $NAMESPACE -f testit_unichart/values-override.yaml upgrade testit testit_unichart/ --wait --timeout 10m
# Дождитесь запуска и готовности к работе всех подов.
kubectl -n $NAMESPACE get po --watch
```

Обновлено: 15.04.2025, 13:00:12

Замена рабочего узла (ноды) в Kubernetes

Подготовка ноды

1. Получите имя текущей ноды с помощью команды:

```sh
kubectl get nodes
```

2. Сохраните название текущей ноды с помощью команды:

```sh
export NODE=<old-node-name>
```

3. Создайте резервную копию Test IT.
4. Если новая нода еще не была добавлена в кластер, добавьте ее.
5. Отключите распределение новых рабочих нагрузок на старый узел с помощью команды:

```sh
kubectl cordon "{$NODE}"
```

Перенос рабочих нагрузок на новый узел

1. Остановите активные рабочие нагрузки с помощью команды:

```sh
cd <installation_folder>
export NAMESPACE=<my-namespace>
helm -n $NAMESPACE list
helm -n $NAMESPACE uninstall testit testit_unichart/
kubectl get po -n $NAMESPACE --watch
```

2. Переустановите приложение с помощью команды:

```sh
cd <installation_folder>
export NAMESPACE=<my-namespace>
helm upgrade --install -f testit_unichart/values-override.yaml
-n $NAMESPACE --create-namespace testit testit_unichart/ --wait
--timeout 10m
# Дождитесь начала работы всех компонентов продукта.
watch -n 1 kubectl -n $NAMESPACE get pods
```

3. Восстановите Test IT из резервной копии.
4. Удалите распределение новых рабочих нагрузок для старой ноды с помощью команды:

```sh
kubectl uncordon "{$NODE}"
```

Обновлено: 15.04.2025, 13:00:12

Настройка SSL для внутренних подключений в Kubernetes

Чтобы настроить внутреннее подключение:

1. Создайте требуемые SSL-сертификаты, ключи и связки для DNS-имен (CN) сервисов, для которых будет использоваться SSL.
2. Перенесите соответствующие сертификаты, ключи и центры сертификации в `testit_unichart/files/ssl` :
   - testit/testit_unichart/files/ssl/ca.crt
   - testit/testit_unichart/files/ssl/auth-cache.crt
   - testit/testit_unichart/files/ssl/auth-cache.key
   - testit/testit_unichart/files/ssl/gotenberg-api.crt
   - testit/testit_unichart/files/ssl/gotenberg-api.key
   - testit/testit_unichart/files/ssl/influxdb.crt
   - testit/testit_unichart/files/ssl/influxdb.key
   - testit/testit_unichart/files/ssl/minio.crt
   - testit/testit_unichart/files/ssl/minio.key
   - testit/testit_unichart/files/ssl/postgres.crt
   - testit/testit_unichart/files/ssl/postgres.key
   - testit/testit_unichart/files/ssl/rabbitmq.crt
   - testit/testit_unichart/files/ssl/rabbitmq.key
   - testit/testit_unichart/files/ssl/rabbitmq.conf
3. В файле `values-override.yaml` или `values-ssl.yaml` активируйте SSL для соответствующих сервисов. Нужные поля помечены комментарием с `internal SSL` . Пример для `auth-cache` :

```yml
auth-cache:
# Enable internal SSL for redis
args:
- "redis-server"
- "--appendonly"
- "yes"
- "--tls-port 6379"
- "--port 0"
- "--tls-cert-file /tls/redis.crt"
- "--tls-key-file /tls/redis.key"
- "--tls-ca-cert-file /tls/ca.crt"
- "--tls-auth-clients no"
configFile:
auth-cache-ssl-files:
redis.crt: "files/ssl/auth-cache.crt"
redis.key: "files/ssl/auth-cache.key"
ca.crt: "files/ssl/ca.crt"
volumes:
ssl:
fromConfigMap: auth-cache-ssl-files
mounts:
auth-cache:
mountPath: /tls
```

4. В файле `values-override.yaml` или `values-ssl.yaml` активируйте соответствующие переменные и настройки для SSL. Пример для `auth-cache` :

```yml
general:
config:
Redis__ConnectionString: "auth-cache:6379,ssl=true"
AUTH_CACHE_CONNECTION_STRING: "auth-cache:6379,ssl=true"
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
volumes:
## Use to enable internal SSL
ssl-ca-internal:
fromConfigMap: ssl-ca-internal
mounts:
mountPath: /app/certs/ca.crt
subPath: ca.crt
configFile:
## Use to enable internal SSL
ssl-ca-internal:
ca.crt: "files/ssl/ca.crt"
```

5. Примените изменения с помощью команды:

```sh
cd <installation_folder>
export NAMESPACE=<my-namespace>
helm upgrade --install -f testit_unichart/values-override.yaml
-n $NAMESPACE --create-namespace testit testit_unichart/ --wait
--timeout 10m
# Дождитесь начала работы всех компонентов продукта.
watch -n 1 kubectl -n $NAMESPACE get pods
```

Примечание: для `rabbitmq` также нужно будет создать дополнительный файл `testit_unichart/files/ssl/rabbitmq.conf` :

```text
listeners.ssl.default = 5671
ssl_options.cacertfile = /etc/certs/server_ca.pem
ssl_options.certfile = /etc/certs/server_crt.pem
ssl_options.keyfile = /etc/certs/server_key.pem
ssl_options.verify = verify_none
ssl_options.fail_if_no_peer_cert = false
```

Обновлено: 15.04.2025, 13:00:12

Добавление самоподписанных сертификатов в контейнеры (Kubernetes)

Для того, чтобы добавить самоподписанные сертификаты в контейнеры:

1. Создайте файл `ca-bundle.crt` в `testit_unichart/files/ssl/ca-bundle.crt` .
2. Активируйте настройки в `values-override.yaml` для передачи этого файла в контейнеры:

```yml
sslEnabled: true
general:
config:
APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
volumes:
## Use to enable custom certificates
ssl-ca-bundle:
fromConfigMap: ssl-ca-bundle
mounts:
mountPath: /app/certs/ca-bundle.crt
subPath: ca-bundle.crt
configFile:
## Use to enable custom certificates
ssl-ca-bundle:
ca-bundle.crt: "files/ssl/ca-bundle.crt"
```

3. Примените изменения с помощью команды:

sh

```sh
cd <installation_folder>
export NAMESPACE=<my-namespace>
helm upgrade --install -f testit_unichart/values-override.yaml
-n $NAMESPACE --create-namespace testit testit_unichart/ --wait
--timeout 10m
# Дождитесь начала работы всех компонентов продукта.
watch -n 1 kubectl -n $NAMESPACE get pods
```

Обновлено: 15.04.2025, 13:00:12

Переход на новый кластер Kubernetes

1. Создайте резервную копию Test IT.
2. Остановите все активные рабочие нагрузки с помощью команды:

```sh
cd <installation_folder>
export NAMESPACE=<my-namespace>
helm -n $NAMESPACE list
helm -n $NAMESPACE uninstall testit testit_unichart/
kubectl get po -n $NAMESPACE --watch
```

3. Перейдите на новый кластер с помощью команды:

```sh
kubectl config get-contexts
# Вывести список доступных контекстов (кластер/пользователь).
kubectl config use-context <context-name> # активировать выбранный контекст
```

4. Установите приложение на новый кластер.
5. Восстановите Test IT из резервной копии.

Обновлено: 27.03.2025, 13:54:57

Перезапуск подов и остановка компонентов Test IT в Kubernetes

Перезапуск подов

Вы можете перезапустить отдельный под или все поды приложения.

- Чтобы перезапустить отдельный под, используйте команду:

```sh
kubectl -n <namespace> get pods # выводит имена всех подов
kubectl -n <namespace> delete pod <podID> # удаляет выбранный под, новый создастся автоматически
```

- Чтобы перезапустить все поды приложения, используйте команду:

```sh
kubectl -n <namespace> get deployments # выводит имена всех deployment
kubectl -n <namespace> scale deployment <name> --replicas=0 # выключает все поды deployment-а
kubectl -n <namespace> scale deployment <name> --replicas=<1,2...> # создает новые поды в выбранном количестве
```

Остановка компонентов Test IT

- Чтобы остановить все компоненты Test IT, используйте команду:

```sh
kubectl -n <namespace> scale deployments --all --replicas 0
```

Обновлено: 14.10.2024, 16:47:10

Переопределение переменных и настроек приложений в Kubernetes

В случае, если есть необходимость переопределить переменные или настройки для компонентов Test IT (например, уровень логирования), а также есть требование не хранить чувствительные данные в переменных окружения,

- Отредактируйте соответствующее поле в `values.yaml` или `values-override.yaml` :

```yml
webapi:
configJson:
webapi-appsettings-json:
appsettings.json: # Name of the file in the configmap
# JSON contents:
AWS_ACCESS_KEY: "myAWSAccessKey"
AWS_SECRET_KEY: "myAWSSecretKey"
# Other variables you want to put in a file
# ---
Serilog:
System:
MinimumLevel: "Information" # Verbose, Debug, Information,
Warning, Error
# If you need the app to write to file
SystemAll:
MinimumLevel: "Information" # Verbose, Debug, Information,
Warning, Error
WriteTo:
- Args:
path: "/app/log/system.log" # SystemAll path logs
rollingInterval: "Day" # Rolling interval
retainedFileCountLimit: "7" # Retention period
UserAction:
MinimumLevel: "Information" # Verbose, Debug, Information,
Warning, Error
UserActionAll:
MinimumLevel: "Information" # Verbose, Debug, Information,
Warning, Error
WriteTo:
- Args:
path: "/app/log/user_actions.log" # AdminAll path logs
rollingInterval: "Day" # Rolling interval
retainedFileCountLimit: "7" # Retention period
```

Данный файл будет располагаться в директории приложений:

```yml
# values.yaml или values-override.yaml
webapi:
volumes:
appsettings-json:
fromConfigMap: webapi-appsettings-json
mounts:
webapi:
mountPath: /app/customs/appsettings.json
subPath: appsettings.json

general:
config:
APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
"/app/customs/appsettings.json"
```

При переопределении учитывайте:

- Дополнительный appsettings.json имеет приоритет выше, чем переменные окружения и основной appsettings.json.
- Дополнительный appsettings.json не является заменой основного файла, а действует как дополняющий или переопределяющий файл.
- Перечень переменных для компонентов Test IT расположен по пути `values.yaml:service-name.config` и `values.yaml:general.config` .

Переменные, название которых включает два "_" подряд, в файле .json структурно разделяются:

```yml
## пример для Hangfire__DbConnectionString
webapi:
configJson:
webapi-appsettings-json:
appsettings.json:
Handfire:
DbConnectionString:
"Host=postgres;Port=5432;Database=backgrounddb;Username=backgrounddbo
Pool Size=130;Command Timeout=30"
```

Обновлено: 27.03.2025, 13:54:57

Обновление Test IT в Docker Compose

> **Создайте резервную копию**
>
> Во избежание критичной потери данных перед обновлением необходимо создать резервную копию Test IT Enterprise.
>
> Скрипт для резервного копирования расположен в папке `scripts` с дистрибутивом Test IT.
>
> Смотрите также: Резервное копирование в Docker Compose

> **Обновляйте систему в соответствии с порядком версий**
>
> Мы рекомендуем обновлять систему последовательно — по порядку старших (мажорных) версий продукта. Например, чтобы обновить продукт с версии 5.1 до 5.3, рекомендуется:
>
> 1. Обновить продукт до Test IT 5.2
> 2. Обновить продукт до версии 5.3
>
> Непоследовательное обновление и игнорирование промежуточных мажорных версий может привести к ошибкам.

Test IT Enterprise можно обновить в режиме **онлайн** или **автономно**. Если вы используйте версии до 4.6. включительно, следуйте соответствующим инструкциям на этой странице.

Миграция данных автотестов при обновлении до версии 5.4

Миграция данных автотестов может занимать до 8 часов в зависимости от их объема. Это связано с необходимостью группировки результатов тестов в прогонах с учетом конфигурации и параметров автотестов, их нумерации и последующего распределения данных по биатлонам. Во время миграции работа в системе недоступна. Основные рекомендации:

1. Отслеживайте миграцию с помощью логов контейнера `webapi`. Минимальный уровень: **Information**.
    - Началом миграции служит запись вида: `Applying migration '20250313093240_TMS-31090_MigrateTestResultsToTestBiathlons'`.
    - При успешном окончании миграции отображается запись вида: `INSERT INTO "__EFMigrationsHistory" ("MigrationId", "ProductVersion") VALUES ('20250313093240_TMS-31090_MigrateTestResultsToTestBiathlons', '9.0.5');`
    - После превышения тайм-аута подключения к базе данных отобразится запись вида: `Database migration timed out. Operation will be retried. Retry number: 1.` После превышения тайм-аута миграция будет перезапущена автоматически. Данные, перенесенные в предыдущих попытках, затронуты не будут.
2. Для сокращения перезапусков на время миграции перед запуском обновления установите переменную окружения `DATABASE_TIMEOUT_SEC=3600` для сервиса `webapi` в файле `docker-compose.yml`.

    ▶ Пример указания тайм-аута

    По окончании миграций в логах отобразится сообщение: `Application started`.
3. По окончании миграции рекомендуется вернуть изначальное значение тайм-аута и перезапустить проект для снижения риска зависания соединений при штатной работе приложения.

4. При ошибке и прекращения работы контейнеров обновите систему повторно.
5. При возникновении проблем, связанных с миграцией, обратитесь в техническую поддержку: support@yoonion.ru.

Обновление онлайн

1. Создайте новую директорию, скачайте и распакуйте в ней файл для установки онлайн. Актуальная версия установочного файла доступна на странице загрузок .
2. Сравните содержимое файлов `.env` и `docker-compose.yml` и перенесите пользовательские значения переменных в соответствующие файлы новой версии.
3. В командной строке перейдите в директорию с новой версией и выполните следующую команду:

```sh
docker compose -f docker-compose.yml --project-name testit up -d --remove-orphans
```

Автономное обновление

1. Создайте новую директорию, скачайте и распакуйте в ней файл для автономной установки. Актуальная версия установочного файла доступна на странице загрузок .
2. Сравните содержимое файлов `.env` и `docker-compose.yml` и перенесите пользовательские значения переменных в соответствующие файлы новой версии.
3. В командной строке перейдите в директорию с новой версией, распакуйте архив для автономной установки и выполните следующую команду:
   - Для обновления до версии 5.2 и более поздних версий:

```sh
mkdir images
tar -zxvf images.tar.gz -C images
for file in $(ls images/*.tar); do docker image load -i $file; done
docker compose -f docker-compose.yml --project-name testit up -d --remove-orphans
```

- Для обновления до версии 5.1 и предыдущих версий:

```sh
docker load -i images.tar.gz
docker compose -f docker-compose.yml --project-name testit up -d --remove-orphans
```

> **Внимание!**
>
> Перед обновлением Test IT до версии 4.3 и выше **убедитесь, что дата и время в установках сервера синхронизированы с датой и временем в установках внешней базы данных**! Несовпадение даты и времени может привести к нарушениям работы лицензии.

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit`. Вы можете использовать другое название.

Если в файлах `.env` и `.yml` используются пользовательские значения переменных, перенесите их в файлы `.env` и `.yml` новой версии Test IT. Все значения в файлах `.env` и `.yml` новых версий Test IT заменяются на значения по умолчанию при обновлении.

Несовместимость CPU V1 с версией MinIO в Test IT 5.1 и выше

В связи с обновлением версии MinIO до **RELEASE.2024-07-16T23-46-41Z** в Test IT 5.1 и более поздних версиях при использовании старой версии CPU (V1) могут возникать ошибки вида `Fatal glibc error: CPU does not support x86-64-v2`. Для решения проблемы:

- При обновлении онлайн добавьте к версии образа постфикс `-cpuv1`. В результате должно получиться: `minio/minio:RELEASE.2024-07-16T23-46-41Z-cpuv1`.
- При автономном обновлении перенесите образ в оффлайн-среду самостоятельно.
- В качестве альтернативного решения для онлайн- или автономного обновления можно использовать вариант внешнего сервиса MinIO или аналогичного сервиса

S3.

Переход на Alpine в версии 4.6

▶ Развернуть/свернуть инструкцию

Подготовка к обновлению до версии 4.5

▶ Развернуть/свернуть инструкцию

Подготовка к обновлению до версии 4.2

▶ Развернуть/свернуть инструкцию

Подготовка к обновлению до версии 4.0.1

▶ Развернуть/свернуть инструкцию

Обновлено: 15.01.2026, 10:38:26

Структура файла docker-compose.yml в версии 4.5

В версии 4.5 структура файла `docker-compose.yml` отличается от предыдущих версий:

- Объединены контейнеры Postgres и MiniO
- Изменена группировка и возможности переиспользования переменных и других свойств сервисов

Объединение контейнеров Postgres и MiniO

> **Пользователям внешних БД и MiniO**
>
> Если вы используете внешние БД и/или MiniO, данные изменения не затронут обращение к ним. Подробнее читайте в соответствующих разделах настоящего руководства.

В версии 4.5 все базы данных и бакеты перемещены в контейнеры `db` и `minio` соответственно. Создание всех необходимых баз при запуске нового контейнера `db` осуществляется с помощью SQL-скрипта `postgres-init.sql`, для уже запущенного контейнера `db` — за счет выполнения миграции при обновлении до версии 4.5. Как следствие:

- Удаляются секции `avatars.minio`, `authdb`, `avatars.db`, `backgrounddb`, `licensedb` и `globalsearchdb`
- Все зависимости (`depends_on`) от `authdb`, `avatars.db`, `backgrounddb`, `licensedb` и `globalsearchdb` переходят в зависимость от `db`
- Все зависимости (`depends_on`) от `avatars.minio` переходят в зависимость от `minio`.
- Удаляются все вольюмы для `avatars.minio`, `authdb`, `avatars.db`, `backgrounddb`, `licensedb` и `globalsearchdb`
- В `.env`-файле для всех строк подключения к базам данных `*_CONNECTION_STRING значение Host` становится равно `db`.

Группировка и переиспользование переменных и других свойств сервисов

В версии 4.5 для переменных и других свойств в `docker-compose.yml` реализовано их объявление по отдельности и переиспользование для всех сервисов, которым это необходимо.
Например, общие для многих сервисов переменные объявляются в блоке:

```yml
x-tms-vars: &tms-vars
ASPNETCORE_ENVIRONMENT: "${ASPNETCORE_ENVIRONMENT:-Production}"
APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
"${APP_CONFIG_FILEPATH:-}"
DOTNET_ENVIRONMENT: "${ASPNETCORE_ENVIRONMENT:-Production}"
INSECURE_REMOTES: "${INSECURE_REMOTES:-}"
Serilog__System__MinimumLevel: "${API_LOG_LEVEL}"
SYSTEM_NAME: "${SYSTEM_NAME:-testit}"
TMS_BUCKET_NAME: "${TMS_BUCKET_NAME}"
USE_PKCE: "${USE_PKCE}"
AWS_CREATE_BUCKET_IF_REQUIRED: "${AWS_CREATE_BUCKET_IF_REQUIRED:-
true}"
```

Также переменные переиспользуются в сервисах следующим образом:

```yml
ldapwebapi:
image:
"${TMS_DOCKER_REGISTRY}/ldapwebapi:${TMS_CONTAINER_VERSION}"
<<: [*tms-service-defaults, *tms-ca-certs-volume]
environment:
<<: [*tms-vars]
```

Обновлено: 08.02.2024, 23:03:45

Обновление Test IT в Kubernetes

> **Создайте резервную копию**
>
> Во избежание критичной потери данных перед обновлением необходимо создать резервную копию Test IT Enterprise.
>
> Скрипт для резервного копирования расположен в папке `scripts` с дистрибутивом Test IT.
>
> Смотрите также: Резервное копирование в Kubernetes: Test IT 5.3 и более поздние версии

> **Обновляйте систему в соответствии с порядком версий**
>
> Мы рекомендуем обновлять систему последовательно — по порядку старших (мажорных) версий продукта. Например, чтобы обновить продукт с версии 5.1 до 5.3, рекомендуется:
>
> 1. Обновить продукт до Test IT 5.2
> 2. Обновить продукт до версии 5.3
>
> Непоследовательное обновление и игнорирование промежуточных мажорных версий может привести к ошибкам.

Если вы используйте версии Test IT Enterprise до 4.6 включительно, следуйте соответствующим инструкциям на этой странице.

- Миграция данных автотестов при обновлении до версии 5.4
- Обновление
- Переход на unichart в версии 5.3.0
- Переход на Alpine в версии 4.6
- Подготовка к обновлению до версии 4.5
  - Очистка backgrounddb через СУБД из состава поставки посредством скрипта
  - Очистка backgrounddb во внешней СУБД (не входит в поставку)

Миграция данных автотестов при обновлении до версии 5.4

Миграция данных автотестов может занимать до 8 часов в зависимости от их объема. Это связано с необходимостью группировки результатов тестов в прогонах с учетом конфигурации и параметров автотестов, их нумерации и последующего распределения данных по биатлонам. Во время миграции производительность системы снижается в связи с нагрузкой на базу данных. Основные рекомендации:

1. Отслеживайте миграцию следующим образом. В начале запуска обновления запускается job `webapi-migration`, порождающая соответствующий pod `webapi-migration-xxxx`. В логах этого пода периодически будет выводиться сообщение `Applying migration`. Минимальный уровень: **Information**.
   - Началом миграции служит запись вида: `Applying migration '20250313093240_TMS-31090_MigrateTestResultsToTestBiathlons'`.
   - При успешном окончании миграции отображается запись вида: `INSERT INTO "__EFMigrationsHistory" ("MigrationId", "ProductVersion") VALUES ('20250313093240_TMS-31090_MigrateTestResultsToTestBiathlons', '9.0.5');`
   - При превышении тайм-аута подключения к базе данных отобразится запись вида: `Database migration timed out. Operation will be retried. Retry number: 1.` После превышения тайм-аута миграция будет перезапущена автоматически. Данные, перенесенные в предыдущих попытках, затронуты не будут.
   - По окончании миграций в логах отобразится сообщение: `Application started`.
2. При ошибке и прекращения работы контейнеров обновите систему повторно.
3. При возникновении проблем, связанных с миграцией, обратитесь в техническую поддержку: support@yoonion.ru.

Обновление

> **Важно**
>
> При автономном обновлении предварительно загрузите образы из архива `images.tar.gz` в выбранный вами локальный репозиторий. Убедитесь, что

кластер имеет доступ к этому репозиторию.

1. Создайте новую директорию, скачайте и распакуйте в ней архив с новой поставкой Helm-чарта.
2. Сравните содержимое файлов `values.yaml` и `values-override.yaml` и перенесите пользовательские параметры в соответствующие файлы новой версии.
3. В командной строке перейдите в директорию с новой версией и выполните следующие команды:

```sh
# Обновите приложения.
helm upgrade --install -f testit_unichart/values-override.yaml
-n <my-namespace> --create-namespace testit testit_unichart/ --
wait --timeout 10h
# Дождитесь начала работы всех модулей.
watch -n 1 kubectl -n <namespace> get pods
```

Как обновлять Test IT до последних версий

- Если вы используете манифесты для Kubernetes, полученные от технической поддержки, вам доступно обновление до версии 4.4 Lupus .
- Чтобы своевременно обновлять Test IT до более поздних версий , требуется использовать helm-чарты. Для получения инструкций обратитесь в техническую поддержку: (support@yoonion.ru).

Переход на unichart в версии 5.3.0

Для случаев, когда Kubernetes-версия продукта обновляется с 5.2.x на 5.3.0, необходимо произвести миграцию объектов Kubernetes на новый чарт:

1. **Рекомендуется:** Выполните резервное копирование для сохранности данных, запустив `scripts/k8s_backup.sh` .
2. Удалите чарты предыдущей версии:

```sh
export NAMESPACE=<my-namespace>
# Убеждаемся, что в выбранном пространстве имен присутствуют
helm list -n $NAMESPACE | grep testit
# Удаляем установленные чарты testit
# NOTE: Названия чартов приведены для примера и могут
отличаться в зависимости от того, какими командами они
устанавливались
helm -n $NAMESPACE uninstall testit-backend
helm -n $NAMESPACE uninstall testit-frontend
# Рекомендуется удостовериться, что при удалении чартов не были
удалены тома памяти
kubectl -n $NAMESPACE get pvc
# NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
# auth-cache-pv-claim-auth-cache-0
# influxdb-pv-claim-influxdb-0
# minio-pv-claim-minio-0
# postgres-pv-claim-postgres-0
# rabbitmq-pv-claim-rabbitmq-0
```

3. Удалите pvc RabbitMQ.

```text
kubectl -n $NAMESPACE delete pvc rabbitmq-pv-claim-rabbitmq-0
```

   В поставку 5.3.0 вошел обновленный docker образ для RabbitMQ. В связи с этим, для избегания несовместимости, pvc rabbitmq-pv-claim-rabbitmq-0 нужно удалить. В противном случае при старте возникнет ошибка `Error during startup: {error,failed_to_initialize_feature_flags_registry}`

4. Если были какие-либо дополнительные настройки в `(testit_backend | testit_frontend)/values-override.yaml` - перенесите их в соответствующие поля в `testit_unichart/values-override.yaml` .
5. Выполните установку продукта через новый Helm-чарт:

```sh
cd <installation_folder>
helm upgrade --install -f testit_unichart/values-override.yaml
-n <my-namespace> --create-namespace testit testit_unichart/ --
wait --timeout 10m
# Дождитесь начала работы всех компонентов продукта.
watch -n 1 kubectl -n <my-namespace> get pods
```

6. **Опционально:** Выполните **восстановление** данных (из п.1) с помощью `scripts/k8s_restore.sh` , если тома памяти не сохранились во время миграции,

или замечена потеря данных.

Переход на Alpine в версии 4.6

▶ Развернуть/свернуть инструкцию

Подготовка к обновлению до версии 4.5

▶ Развернуть/свернуть инструкцию

Обновлено: 15.01.2026, 10:38:26

Обновление старых версий Test IT в Kubernetes

Создайте резервную копию

Во избежание критичной потери данных перед обновлением необходимо создать резервную копию Test IT Enterprise.

Скрипт для резервного копирования расположен в папке `scripts` с дистрибутивом Test IT.

Смотрите также: Резервное копирование в Kubernetes: Test IT 5.2 и более ранние версии

Обновляйте систему в соответствии с порядком версий

Мы рекомендуем обновлять систему последовательно — по порядку старших (мажорных) версий продукта. Например, чтобы обновить продукт с версии 5.1 до 5.3, рекомендуется:

1. Обновить продукт до Test IT 5.2
2. Обновить продукт до версии 5.3

Непоследовательное обновление и игнорирование промежуточных мажорных версий может привести к ошибкам.

Внимание

- Эта инструкция актуальна, если вы разворачивали Test IT в Kubernetes с помощью манифестов.
- В случае возникновения вопросов, рекомендуем обращаться в техническую поддержку: (support@yoonion.ru).

- Подготовка к обновлению до версии 4.2
- Миграция бакетов MinIO Kubernetes
- Миграция СУБД Postgres

- Обновление

Подготовка к обновлению до версии 4.2

> **Важно**
>
> Обновление до версии 4.2 осуществляется с версии 4.1.0 и выше.

Перед обновлением системы выполните следующие действия:

- Создайте резервную копию системы (рекомендуется).
- Убедитесь, что в кластере Kubernetes, в котором запущен Test IT, достаточно места для создания файлов дампа Postgres и новых бакетов MinIO.
- Если вы используете внешнее объектное хранилище MinIO и/или внешнюю СУБД Postgres, а не сервисы из поставки Test IT, пропустите соответствующие разделы этой инструкции.

Миграция бакетов MinIO Kubernetes

Начиная с версии 4.2 MinIO Gateway не поддерживается. Поэтому перед обновлением необходимо выполнить миграцию бакетов MinIO и их метаданных.

1. Перейдите в директорию с установочными файлами или скопируйте файл `scripts/k8s_minio_migrate.sh` и папку `jobs/minio/migrate` в удобную для вас директорию, где будет проходить миграция. Убедитесь, что на диске достаточно места для временного хранения содержимого `minio` и `avatars-minio` .

   > **Внимание**
   >
   > Для корректной работы скрипта сохраните структуру по директориям:

   ```sh
   scripts/
   k8s_minio_migrate.sh
   jobs/
   minio/
   migrate/
   job.yaml
   configmap.yaml
   pvc.yaml
   ```

2. Добавьте разрешение на использование скрипта:

```sh
chmod +x ./scripts/k8s_minio_migrate.sh
```

3. Определите пространство имен, в котором у вас запущен Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

4. Запустите скрипт, передав ему соответствующее пространство имен:

```sh
./scripts/k8s_minio_migrate.sh <namespace>
# Пример: ./scripts/k8s_minio_migrate.sh my-testit-namespace
# Для более развернутых логов и очистки временных файлов,
создаваемых при выполнении скрипта, его можно запустить с
флагами -d и -c соответственно.
# ./scripts/k8s_minio_migrate.sh -d my-testit-namespace
# или ./scripts/k8s_minio_migrate.sh -c my-testit-namespace
# или ./scripts/k8s_minio_migrate.sh -dc my-testit-namespace
```

Убедитесь, что миграция прошла успешно. При успешной миграции по окончании работы скрипта отобразится строка:

```sh
Migration successful!
```

В случае ошибки миграции логи сохранятся в текущую директорию:

```sh
Job 'job_name' failed. Logs: '/path/to/job_name-dd-mm-yyyy-
fail.log'.
```

В случае ошибки миграции устраните проблему самостоятельно или свяжитесь с технической поддержкой (support@yoonion.ru). По окончании миграции вы можете перейти к процессу миграции баз СУБД Postgres.

Миграция СУБД Postgres

Начиная с версии 4.2 в качестве основной системы управления базами данных (СУБД) используется Postgres 14. В процессе обновления необходимо выполнить дамп ваших баз данных (БД) с предыдущей версии Test IT и их восстановление в новой версии. Во время дампа и восстановления баз данных продукт будет недоступен.

1. Перейдите в директорию с установочными файлами или скопируйте в другую директорию файлы `scripts/k8s_postgres_migrate.sh`, `scripts/k8s_postgres_migrate.env` и папку `jobs/postgres`, сохраняя следующую структуру:

```sh
scripts/
k8s_postgres_migrate.sh
jobs/
postgres/
import/
job.yaml
configmap.yaml
pvc.yaml
export/
job.yaml
configmap.yaml
```

2. Задайте значение переменной, определяющей размер хранилища для .dump файлов баз `testitdb`, `avatarsdb`, `authdb`:

```sh
export DUMP_VOLUME_SIZE_GB=50
# ПРИМЕЧАНИЕ: данное значение в 50 Gb указано для примера, актуальное значение лучше выбрать в соответствии с действующим размером баз.
```

3. Добавьте разрешение на использование скрипта:

```sh
chmod +x ./scripts/k8s_postgres_migrate.sh
```

4. Определите пространство имен, в котором у вас запущен Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

5. Запустите скрипт, передав ему соответствующее пространство имен:

```sh
./scripts/k8s_postgres_migrate.sh <namespace>
# Пример: ./scripts/k8s_postgres_migrate.sh my-testit-namespace
# Для более развернутых логов и очистки временных файлов,
создаваемых при выполнении скрипта, его можно запустить с
флагами -d и -c соответственно.
# ./scripts/k8s_postgres_migrate.sh -d my-testit-namespace
# или ./scripts/k8s_postgres_migrate.sh -c my-testit-namespace
# или ./scripts/k8s_postgres_migrate.sh -dc my-testit-namespace
```

Убедитесь, что миграция прошла успешно. При успешной миграции по окончании работы скрипта отобразится строка:

```sh
Migration successful! Restarting Test IT...
```

В случае ошибки миграции отобразится строка:

```sh
Job 'job_name' failed. Logs: '/path/to/job_name-dd-mm-yyyy-
fail.log'.
```

Устраните проблему самостоятельно или свяжитесь с технической поддержкой (support@yoonion.ru).

Обновление

> **Важно**
>
> При офлайн-обновлении предварительно загрузите образы из архива images.tar.gz в выбранный вами локальный репозиторий. Убедитесь, что кластер имеет доступ к этому репозиторию.

1. В зависимости от вашей текущей версии Test IT, ознакомьтесь с `<old>-<new>_upgrade_plan.yaml` (`<old>` - версия продукта, с которой производится обновление).
2. Скопируйте старые конфигурационные файлы Kubernetes в папку для новой версии, например:

```sh
mkdir ./TestIT_4.2.4_k8s
cp ./TestIT_4.1.0_k8s/*.yaml ./TestIT_4.2.4_k8s/
```

3. В соответствии с `upgrage_plan`, примените соответствующие изменения к конфигурационным файлам `.yaml`. Например:

```yml
transfer-service: # <--- Название деплоймента
(deployments.yaml)
image: ...:4.2.4 # <--- новая версия образа
environment: # Edit configmap # <--- соответствующий
деплойменту конфигмап (configmap.yaml)
+ RABBITMQ_SSL_ENABLED: "false" # <--- переменная, которую
необходимо добавить/изменить/удалить
```

4. После успешного выполнения необходимых скриптов/миграций, описанных выше, примените изменения. Например:

```sh
kubectl apply -f ./TestIT_4.2.4_k8s/ --dry-run=client #
валидация конфигурации, без каких-либо фактических изменений в
кластере
# ^^ Опционально: можно добавить флаг -oyaml для вывода в
терминал будущих объектов k8s
kubectl apply -f ./TestIT_4.2.4_k8s/ # применение обновления
```

Обновлено: 15.01.2026, 10:38:26

Переход из Docker в Kubernetes: Test IT 5.3 и более поздние версии

> **Внимание**
>
> - Перед переходом необходимо обновить Test IT до последней `docker-версии`, соответствующей выходу сборки `Kubernetes`.
> - Во время перехода на Kubernetes продукт будет недоступен.

1. Задайте необходимые значения переменных для проекта:

```sh
# minio_vars
export TMS_BUCKET_NAME=testit
export AVATARS_AWS_BUCKET_NAME=avatars

# docker_vars
export COMPOSE_FILE_PATH=/absolute/path/to/testit/docker-compose.yml
export COMPOSE_PROJECT_PREFIX=prod
export COMPOSE_NETWORK_NAME=123123_yoonion_network_local
```

2. Из архива дистрибутива (сборки) Test IT в Kubernetes скопируйте файлы
   - `pre_k8s_backup.sh`
   - `docker-compose.minio-export.yml`

   Скопированные файлы поместите в папку scripts сборки Test IT в Docker Compose.
3. Выполните резервное копирование `docker`-версии Test IT:

```sh
chmod +x scripts/pre_k8s_backup.sh
./scripts/pre_k8s_backup.sh
```

4. По окончании резервного копирования сохраните путь, по которому расположен архив (он отобразится в сообщении об успешном окончании резервного копирования):

```sh
Backup successful! Archive saved at:
'/path/to/testit_docker_to_k8s.tar'
```

5. Сравните переменные из `.env` файла (в директории docker-версии продукта) с `values.yaml:general` в директории testit_unichart :

yml

```yaml
# testit_unichart/values.yaml
frontend:
enabled: true
image:
repository: "docker.testit.ru/tms-frontend/tms-frontend"
tag: v5.12.0 #<--env.TMS_FRONTEND_CONTAINER_VERSION
config:
NGINX_JSON_LOGGING: "true"
SSL_STRONG_CIPHERS: "false"
webapi:
enabled: true
image:
repository: "registry.testit.software/testit/testit"
tag: v5.10.1 #<--env.TMS_CONTAINER_VERSION
config:
USE_PKCE: "true"
INSECURE_REMOTES: ""
SYNC_RESULT_LINKS_EVERY_SEC: "120"
TEST_RESULT_LINK_REQUEST_LIFETIME_SEC: "600"
THREAD_PER_PROCESSOR: "10"
SMTP_ENABLE: "false"
SMTP_FROM: ""
SMTP_HOST: ""
SMTP_PORT: ""
SMTP_LOGIN: ""
SMTP_DOMAIN: ""
SMTP_PASSWORD: ""
SMTP_CONNECTION: ""
SMTP_AUTHENTICATION: ""
SMTP_SKIP_CERTIFICATE: "false"
SMTP_TLS_VERSION: "tls"
SMTP_LOG_ENABLE: "false"
PERF_MODULE_ENABLED: "false"
avatars-api:
enabled: true
image:
repository: "docker.testit.ru/avatars/avatars-api"
tag: v5.2.8 #<--env.AVATARS_CONTAINER_VERSION
config:
AVATARS_AWS_BUCKET_NAME: "avatars"
auth:
enabled: true
image:
repository: "docker.testit.ru/identity/authwebapi"
```

```
tag: v5.5.0 #<--env.AUTHWEBAPI_CONTAINER_VERSION
config:
ASPNETCORE_ACCESS_TOKEN_EXPIRATION_MINUTES: "120"
ASPNETCORE_REFRESH_TOKEN_EXPIRATION_MINUTES: "88000"
postgres:
config:
POSTGRES_PASSWORD: "F1rstL0g0N!"
POSTGRES_USER: "postgres"
POSTGRES_DB: "testitdb"
general:
config:
AWS_CONNECTION_STRING: "http://minio:9000"
AWS_CREATE_BUCKET_IF_REQUIRED: "true"
TMS_BUCKET_NAME: "testit"
COMPlus_EnableDiagnostics: "0"
INFLUX_CONNECTION_STRING: "http://influxdb:8086"
INFLUX_AUTH_ENABLED: "false"
RABBITMQ_DEFAULT_HOST: "rabbitmq"
RABBITMQ_SSL_ENABLED: "false"
RABBITMQ_DEFAULT_PORT: "5672"
RABBITMQ_DEFAULT_VHOST: "testitrabbit"
RABBITMQ_AUTH_MODE: "plain"
RABBITMQ_CLIENT_CERT_PASSPHRASE: ""
RABBITMQ_CLIENT_CERT_PATH: "/etc/rabbitmq/ssl/client/testit.pfx"
APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
"/app/customs/appsettings.json"
FRONTEND_URL: "http://frontend:8080"
CHECK_DB_READY: "true"
secrets:
AWS_ACCESS_KEY: "testitAccessKey"
AWS_SECRET_KEY: "testitSecretKey"
INFLUX_USERNAME: "influx"
INFLUX_PASSWORD: "influxpass"
RABBITMQ_DEFAULT_USER: "testit"
RABBITMQ_DEFAULT_PASS: "password"
AUTH_CACHE_CONNECTION_STRING: "auth-cache"
TESTIT_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=testitdb;
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
```

```
Maximum Pool Size=130;
BACKGROUND_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=backgrounddb;
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
Maximum Pool Size=130;
GLOBALSEARCH_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=globalsearchdb;
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
Maximum Pool Size=130;
AUTH_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=authdb;
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
Maximum Pool Size=130;
LICENSE_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=licensedb;
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
Maximum Pool Size=130;
AVATARS_DB_CONNECTION_STRING: >
Host=postgres;
Port=5432;
Database=avatarsdb;
```

```
Username=postgres;
Password=F1rstL0g0N!;
Command Timeout=30;
Target Session Attributes=any;
Pooling=true;
Maximum Pool Size=130;
```

6. **Опционально**: При необходимости, внесите изменения в отдельном файле `values-override.yaml` , раскомментировав только те строки, которые хотите поменять. Например:

```yml
# testit_unichart/values-override.yaml
# ...
general:
config:
# AUTH_CACHE_CONNECTION_STRING: "auth-cache:6379,ssl=true"
# INFLUX_CONNECTION_STRING: "https://influxdb:8086"
RABBITMQ_SSL_ENABLED: "true"
RABBITMQ_DEFAULT_PORT: "5671"
# APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION:
"/etc/ssl/certs"
frontend:
replicaCount: 3
# resources: {}
# ...
```

7. Установите Test IT Kubernetes.
8. Задайте необходимые значения переменных для проекта:

```sh
# Убедитесь, что вводите правильные значения переменных
(testit_unichart/values.yaml или testit_unichart/values-
override.yaml)
# minio_vars
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
export TESTIT_BUCKET="testit"
export AVATARS_BUCKET="avatars"

# postgres_vars
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
```

9. Восстановите данные из резервной копии, используя пространство имен, в котором создали `Test IT Kubernetes` и путь, который сохранили после успешного выполнения `scripts/pre_k8s_backup.sh` :

```sh
# Задавать эти переменные не обязательно, скрипт может
подтянуть из уже запущенных внутренних postgres и minio
export NAMESPACE=<my-namespace>
chmod +x scripts/move_to_k8s.sh
./scripts/move_to_k8s.sh $NAMESPACE
/path/to/testit_docker_to_k8s.tar
```

После успешного выполнения скрипта отобразится сообщение:

```text
Restore process completed successfully! Your Test IT is running
on Kubernetes now!
If you have SSL certificates (trusted-certificates-volume) or
any other configurations to apply, you can reference the Test
IT docs for guides.
```

Обновлено: 18.12.2025, 17:19:30

Переход из Docker в Kubernetes: Test IT 5.2 и более ранние версии

> **Внимание**
>
> - Перед переходом необходимо обновить Test IT до последней `docker-версии`, соответствующей выходу сборки `Kubernetes`.
> - Во время перехода на Kubernetes продукт будет недоступен.

1. Задайте необходимые значения переменных для проекта:

```sh
# minio_vars
export TMS_BUCKET_NAME=testit
export AVATARS_AWS_BUCKET_NAME=avatars

# docker_vars
export COMPOSE_FILE_PATH=/absolute/path/to/testit/docker-compose.yml
export COMPOSE_PROJECT_PREFIX=prod
export COMPOSE_NETWORK_NAME=123123_yoonion_network_local
```

2. Из архива дистрибутива (сборки) Test IT в Kubernetes скопируйте файлы
   - `pre_k8s_backup.sh`
   - `docker-compose.minio-export.yml`

   Скопированные файлы поместите в папку scripts сборки Test IT в Docker Compose.
3. Выполните резервное копирование `docker-версии` Test IT:

```sh
chmod +x scripts/pre_k8s_backup.sh
./scripts/pre_k8s_backup.sh
```

4. По окончании резервного копирования сохраните путь, по которому расположен архив (он отобразится в сообщении об успешном окончании резервного копирования):

```sh
Backup successful! Archive saved at:
'/path/to/testit_docker_to_k8s.tar'
```

5. Сравните переменные из `.env` файла (в директории docker-версии продукта) с `values.yaml:general` в директориях testit_backend и testit_frontend:

```yaml
# testit_frontend/values.yaml
general:
image:
repository: "docker.testit.ru/testit" # <--
env.TMS_DOCKER_REGISTRY
tag: 4.4.0 #<--env.TMS_CONTAINER_VERSION
pullPolicy: IfNotPresent # политика скачивания docker-образов
(IfNotPresent, Always, Never)
config:
CWM_ENABLED: "false"
CWM_S3_BUCKET_SECRET_KEY: "secretKey"
WIKI_ENABLED: "false"
WIKI_S3_BUCKET_SECRET_KEY: ""

# testit_backend/values.yaml
general:
image:
repository: "docker.testit.ru/testit" # <--
env.TMS_DOCKER_REGISTRY
tag: 4.4.0 #<--env.TMS_CONTAINER_VERSION
pullPolicy: IfNotPresent # политика скачивания docker-образов
(IfNotPresent, Always, Never)
config:
AUTH_CACHE_CONNECTION_STRING: "auth-cache" # должно
соответствовать .authCache.name при стандартном запуске
FRONTEND_URL: "http://localhost"
RABBITMQ_DEFAULT_USER: "testit"
RABBITMQ_DEFAULT_PASS: "password"
RABBITMQ_DEFAULT_VHOST: "testitrabbit"
RABBITMQ_DEFAULT_HOST: "rabbitmq"
RABBITMQ_DEFAULT_PORT: "5672"
RABBITMQ_AUTH_MODE: "plain"
RABBITMQ_CLIENT_CERT_PATH:
"/etc/rabbitmq/ssl/client/testit.pfx"
RABBITMQ_CLIENT_CERT_PASSPHRASE:
RABBITMQ_SSL_ENABLED: "false"
USE_PKCE: "true"
INSECURE_REMOTES: ""
SYNC_RESULT_LINKS_EVERY_SEC: "120"
AWS_CONNECTION_STRING: "http://minio:9000" # должно
соответствовать .minio.name при стандартном запуске
AWS_ACCESS_KEY: "testitAccessKey"
AWS_SECRET_KEY: "testitSecretKey"
INFLUX_CONNECTION_STRING: "http://influxdb:8086" # должно
```

```
соответствовать .influxdb.name при стандартном запуске
INFLUX_AUTH_ENABLED: "false"
INFLUX_USERNAME: "influx"
INFLUX_PASSWORD: "influxpass"
TEST_RESULT_LINK_REQUEST_LIFETIME_SEC: "600"
DATABASE_TIMEOUT_SEC: "600"
THREAD_PER_PROCESSOR: "10"
TMS_BUCKET_NAME: "testit"
AVATARS_AWS_BUCKET_NAME: "avatars"
SMTP_ENABLE: "false"
SMTP_FROM: ""
SMTP_HOST: ""
SMTP_PORT: ""
SMTP_LOGIN: ""
SMTP_DOMAIN: ""
SMTP_PASSWORD: ""
SMTP_CONNECTION: ""
SMTP_AUTHENTICATION: ""
SMTP_SKIP_CERTIFICATE: "false"
SMTP_TLS_VERSION: "tls"
SMTP_LOG_ENABLE: "false"
PERF_MODULE_ENABLED: "false"
POSTGRES_USER: "postgres" # данные для подключения к PostgreSQL
и названия баз данных
POSTGRES_PASSWORD: "F1rstL0g0N!"
POSTGRES_DB: "testitdb"
POSTGRES_AUTH_DB: "authdb"
POSTGRES_AVATARS_DB: "avatarsdb"
POSTGRES_BACKGROUND_DB: "backgrounddb"
POSTGRES_LICENSE_DB: "licensedb"
ASPNETCORE_ACCESS_TOKEN_EXPIRATION_MINUTES: "800"
ASPNETCORE_REFRESH_TOKEN_EXPIRATION_MINUTES: "8000"
CALCULATION_AUTOTESTS_STABILITYPERCENTAGE_DELAY_SECONDS: "600"
INFLUX_DISABLE_UPLOAD: "false"
APPLICATION__DEVELOPERMODE: "false"
```

6. **Опционально**: При необходимости, внесите изменения в отдельном файле `values-override.yaml` , раскомментировав только те строки, которые хотите поменять. Например:

```yml
# testit_backend/values-override.yaml
# ...
general:
config:
# ...
# SMTP_LOG_ENABLE:
# PERF_MODULE_ENABLED:
POSTGRES_USER: "testituser"
POSTGRES_PASSWORD: "8wkj9l!0asdk"
# POSTGRES_DB:
# POSTGRES_AUTH_DB:
# POSTGRES_AVATARS_DB:
# POSTGRES_BACKGROUND_DB:
# POSTGRES_LICENSE_DB:
# ...
# ...

# testit_frontend/values-override.yaml
# ...
frontend:
replicaCount: 3
# resources: {}
# ...
```

7. Установите Test IT Kubernetes.
8. Задайте необходимые значения переменных для проекта:

```sh
# Убедитесь, что вводите правильные значения переменных
(testit_backend/values.yaml или testit_backend/values-
override.yaml)
# minio_vars
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
export TESTIT_BUCKET="testit"
export AVATARS_BUCKET="avatars"

# postgres_vars
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
```

9. Восстановите данные из резервной копии, используя пространство имен, в котором создали `Test IT Kubernetes` и путь, который сохранили после успешного выполнения `scripts/pre_k8s_backup.sh` :

```sh
# Задавать эти переменные не обязательно, скрипт может подтянуть из уже запущенных внутренних postgres и minio
export NAMESPACE=<my-namespace>
chmod +x scripts/move_to_k8s.sh
./scripts/move_to_k8s.sh $NAMESPACE /path/to/testit_docker_to_k8s.tar
```

После успешного выполнения скрипта отобразится сообщение:

```text
Restore process completed successfully! Your Test IT is running on Kubernetes now!
If you have SSL certificates (trusted-certificates-volume) or any other configurations to apply, you can reference the Test IT docs for guides.
```

Обновлено: 18.12.2025, 17:19:30

Проверка лицензии в Docker Compose

Чтобы избежать безвозвратной потери лицензии из-за недостатка свободного места на жестком диске, контейнер `license-service` регулярно проверяет доступное место на диске. Для корректной работы системы рекомендуется иметь не менее 10% свободного места на жестком диске.

При настройках по умолчанию проверка свободного места на диске производится раз в час, а необходимый порог свободного места на диске для перезаписи файла составляет 10% от общего объема памяти диска. Данные настройки можно изменять в контейнере `license-service` :

- Интервал проверок устанавливается с помощью переменной `Monitoring__DiskSpaceCheckInterval` .
- Допустимый предел свободного места на диске устанавливается с помощью переменной `Monitoring__DiskSpacePercentageThreshold` . Значение указывается в виде целочисленной переменной в процентах.

Если свободного места на диске осталось меньше установленного порога, система переходит в режим просмотра, чтобы не потерять лицензию из-за недостатка места на диске. После того, как вы освободите необходимое место на жестком диске, вы можете выполнить одно из следующих действий:

- Дождитесь следующей проверки свободного места на диске.
- Перезапустите контейнер `license-service` .

Система автоматически перейдет из режима просмотра в режим редактирования.

Обновлено: 04.06.2024, 11:06:12

Проверка лицензии в Kubernetes

Чтобы избежать безвозвратной потери лицензии из-за недостатка свободного места на жестком диске, контейнер `license-service` регулярно проверяет доступное место на диске. Для корректной работы системы рекомендуется иметь не менее 10% свободного места на жестком диске. При настройках по умолчанию проверка свободного места на диске производится каждый час, а необходимый порог свободного места на диске для перезаписи файла составляет 10% от общего объема памяти диска. Данные настройки можно изменять в файле `values-override.yaml` в разделе `license` :

- Интервал проверок устанавливается с помощью переменной `DiskSpaceCheckInterval` .
- Допустимый предел свободного места на диске устанавливается с помощью переменной `DiskSpacePercentageTheshold` . Значение указывается в виде целочисленной переменной в процентах:

```yml
# testit_backend/values-override.yaml
license:
DiskSpaceCheckInterval: "01:00:00"
DiskSpacePercentageTheshold: 10
```

Если свободного места на диске осталось меньше установленного порога, система переходит в режим просмотра *(read-only)*, чтобы не потерять лицензию из-за недостатка места на диске. После того, как вы освободите необходимое место на жестком диске, вы можете выполнить одно из следующих действий:

- Дождитесь следующей проверки свободного места на диске.
- Перезапустите под `license-service` . Система автоматически перейдет из режима просмотра в режим редактирования.

Обновлено: 27.03.2025, 13:54:57

Резервное копирование в Docker Compose

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Создание резервных копий

> **Осторожно**
>
> Не рекомендуется создавать резервные копии от имени суперпользователя root (`sudo`) во избежание ошибок восстановления.

На время создания резервной копии продукт будет остановлен.

1. Перед выполнением скрипта на создание резервной копии перейдите в директорию, которая содержит `docker-compose.yml` файл с настройками текущей версии системы.

2. Для создания резервной копии выполните следующие команды:

```sh
chmod +x scripts/backup.sh
scripts/backup.sh docker-compose.yml testit
# Или
scripts/backup.sh docker-compose.yml testit --notar
# С флагом '--notar' архивы хранилищ docker не будут объединяться в один большой файл, что в моменте работы скрипта положительно скажется на утилизации места на диске в связи со спецификой работы утилиты tar
```

Система будет запущена после окончания процесса. В рабочей директории будет создан архив с резервной копией. Формат имени файла архива: `backup_{день}_{месяц}_{год}.tar` . Например, `backup_21_05_2019.tar` . С

использованием флага `--notar` будет создана директория `backup_{день}_{месяц}_{год}`, в которой будут содержаться архивы `volume_name.tar.bz2`.

> **Для внешних БД скрипт настраивается отдельно**
>
> Приведенный скрипт не распространяется на внешние БД (в случае их настройки и использования). Для внешних БД необходимо настроить резервное копирование отдельным шагом (штатными средствами PostgreSQL).

Восстановление из резервной копии

Продукт будет остановлен на время восстановления из резервной копии.

1. Перед выполнением скрипта на восстановление из резервной копии перейдите в директорию, которая содержит `docker-compose.yml` и `.env` -файлы с настройками текущей версии системы.
2. Для восстановления из резервной копии выполните следующие команды:

```sh
chmod +x scripts/restore.sh
scripts/restore.sh docker-compose.yml testit
backup_21_05_2019.tar
# Или
scripts/restore.sh docker-compose.yml testit backup_21_05_2019
--notar
# Запуск с флагом '--notar' сработает только для восстановления
из резервной копии, созданной с аналогичным флагом
```

Система будет запущена после окончания процесса.

> **Если вы переносите Test IT на новый сервер**
>
> При переносе продукта на другой сервер рекомендуется предварительно установить Test IT на новом сервере с настройками по умолчанию, затем восстановить данные системы из резервной копии.
> Для наилучшей совместимости на новом сервере рекомендуется устанавливать Test IT той же версии, которая содержится в резервной копии, переносимой из исходного сервера.

Обновлено: 01.03.2024, 11:13:10

Резервное копирование в Kubernetes: Test IT 5.3 и более поздние версии

Создание резервной копии

> **Внимание!**
>
> - Продукт будет остановлен на время создания резервной копии. Не используйте утилиту `sudo` для создания резервных копий.
> - Приведенный скрипт не распространяется на внешние БД (в случае их настройки и использования). Для внешних БД необходимо настроить резервное копирование отдельным шагом (штатными средствами PostgreSQL, Minio и т.д.)

1. Перед выполнением скрипта на создание резервной копии перейдите в директорию, где будут храниться резервные копии, папки scripts/ и jobs/:

```sh
cd my-testit-directory
ls
# Здесь должны присутствовать папки scripts/ и jobs/
```

2. Определите *namespace* (пространство имен), в котором запущено приложение Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

3. Создайте резервную копию с помощью набора команд, заменив `<namespace>` на нужное вам пространство имен:

```sh
# Задайте переменные для доступа на PostgreSQL и Minio:
# Значения для переменных располагаются в
testit_unichart/values.yaml:general:config или
testit_unichart/values-override.yaml:general:config
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
chmod +x scripts/k8s_backup.sh
./scripts/k8s_backup.sh <namespace>
# Пример: ./scripts/k8s_backup.sh my-testit-namespace
```

Система будет запущена после окончания процесса. В рабочей директории будет создана папка `backups`, содержащая папки с датами резервного копирования, внутри которых расположены архивы с резервными копиями. Формат отображения дат:

```sh
backups/
{день}_{месяц}_{год}/
{сервис1}_backup.tar
{сервис2}_backup.tar
...
# Например:
backups/
18_07_2023/
postgres_backup.tar
minio_backup.tar
...
```

Восстановление из резервной копии

> **Внимание!**
>
> - Продукт будет остановлен на время восстановления из резервной копии.
> - В директории со скриптами (в папке scripts/) должна находиться папка jobs/, в которой указаны ресурсы K8s, необходимые для восстановления из резервной копии.

> - При переносе продукта на другой сервер рекомендуется предварительно установить Test IT на новый сервер с настройками по умолчанию, затем восстановить данные системы из резервной копии. Для наилучшей совместимости на новом сервере рекомендуется устанавливать Test IT той же версии, которая содержится в резервной копии, переносимой из исходного сервера.

1. Перед выполнением скрипта на восстановление из резервной копии перейдите в директорию, где хранятся резервные копии, и выберите дату создания резервной копии для восстановления. Например:

```sh
# Перейдите в директорию
cd my-testit-directory
ls
# Здесь должны присутствовать папки backups/, scripts/ и jobs/
# Проверьте содержимое папки с резервными копиями:
ls backups/
# Здесь будут выведены папки с датами резервного копирования, например:
19_07_2023/ 20_07_2023/ 21_07_2023/
```

2. Определите namespace (пространство имен), в котором запущено Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

3. Восстановите систему из резервной копии с помощью набора команд:

```sh
# Задайте переменные для доступа на PostgreSQL и Minio:
# Значения для переменных располагаются в
testit_unichart/values.yaml:general:config или
testit_unichart/values-override.yaml:general:config
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
chmod +x scripts/k8s_restore.sh
./scripts/k8s_restore.sh <namespace> <backup_date>
# Пример: ./scripts/k8s_restore.sh my-testit-namespace
18_07_2023
```

Система будет запущена после окончания процесса.

Обновлено: 02.04.2025, 15:41:47

Резервное копирование в Kubernetes: Test IT 5.2 и более ранние версии

Создание резервной копии

> **Внимание!**
>
> - Продукт будет остановлен на время создания резервной копии. Не используйте утилиту `sudo` для создания резервных копий.
> - Приведенный скрипт не распространяется на внешние БД (в случае их настройки и использования). Для внешних БД необходимо настроить резервное копирование отдельным шагом (штатными средствами PostgreSQL, Minio и т.д.)

1. Перед выполнением скрипта на создание резервной копии перейдите в директорию, где будут храниться резервные копии, папки scripts/ и jobs/:

```sh
cd my-testit-directory
ls
# Здесь должны присутствовать папки scripts/ и jobs/
```

2. Определите *namespace* (пространство имен), в котором запущено приложение Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

3. Убедитесь, что версии Postgres и Minio/mc: совпадают с версиями, запущенными в пакете Test IT:

```yml
# testit_backend/values.yaml
postgres:
image:
name: <...>
tag: 14.8-bookworm

# /jobs/postgres/restore/job.yaml,
/jobs/postgres/backup/job.yaml
spec:
template:
spec:
containers:
- name: <...>
image: postgres:14.8-bookworm
#########################################################
# testit_backend/values.yaml
minio:
image:
name: <...>
tag: RELEASE.2023-05-04T21-44-30Z
```

4. Создайте резервную копию с помощью набора команд, заменив `<namespace>` на нужное вам пространство имен:

```sh
# Задайте переменные для доступа на PostgreSQL и Minio:
# Значения для переменных располагаются в
testit_backend/values.yaml:general:config или
testit_backend/values-override.yaml:general:config
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
chmod +x scripts/k8s_backup.sh
./scripts/k8s_backup.sh <namespace>
# Пример: ./scripts/k8s_backup.sh my-testit-namespace
```

Система будет запущена после окончания процесса. В рабочей директории будет создана папка `backups`, содержащая папки с датами резервного копирования,

внутри которых расположены архивы с резервными копиями. Формат отображения дат:

```sh
backups/
{день}_{месяц}_{год}/
{сервис1}_backup.tar
{сервис2}_backup.tar
...
# Например:
backups/
18_07_2023/
postgres_backup.tar
minio_backup.tar
...
```

Восстановление из резервной копии

> **Внимание!**
>
> - Продукт будет остановлен на время восстановления из резервной копии.
> - В директории со скриптами (в папке scripts/) должна находиться папка jobs/, в которой указаны ресурсы K8s, необходимые для восстановления из резервной копии.
> - При переносе продукта на другой сервер рекомендуется предварительно установить Test IT на новый сервер с настройками по умолчанию, затем восстановить данные системы из резервной копии. Для наилучшей совместимости на новом сервере рекомендуется устанавливать Test IT той же версии, которая содержится в резервной копии, переносимой из исходного сервера.

1. Перед выполнением скрипта на восстановление из резервной копии перейдите в директорию, где хранятся резервные копии, и выберите дату создания резервной копии для восстановления. Например:

```sh
# Перейдите в директорию
cd my-testit-directory
ls
# Здесь должны присутствовать папки backups/, scripts/ и jobs/
# Проверьте содержимое папки с резервными копиями:
ls backups/
# Здесь будут выведены папки с датами резервного копирования,
например:
19_07_2023/ 20_07_2023/ 21_07_2023/
```

2. Определите namespace (пространство имен), в котором запущено Test IT:

```sh
kubectl get ns
# Здесь будут отображены доступные пространства имен
```

3. Восстановите систему из резервной копии с помощью набора команд:

```sh
# Задайте переменные для доступа на PostgreSQL и Minio:
# Значения для переменных располагаются в
testit_backend/values.yaml:general:config или
testit_backend/values-override.yaml:general:config
export POSTGRES_HOST="postgres"
export POSTGRES_PORT="5432"
export POSTGRES_USER="postgres"
export POSTGRES_PASSWORD="F1rstL0g0N!"
export AWS_CONNECTION_STRING="http://minio:9000"
export AWS_ACCESS_KEY="testitAccessKey"
export AWS_SECRET_KEY="testitSecretKey"
chmod +x scripts/k8s_restore.sh
./scripts/k8s_restore.sh <namespace> <backup_date>
# Пример: ./scripts/k8s_restore.sh my-testit-namespace
18_07_2023
```

Система будет запущена после окончания процесса.

Обновлено: 08.04.2025, 12:42:56

Логирование пользовательских действий (Docker Compose)

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Подготовка

- Установите параметры `vm.max_map_count=262144` и `vm.overcommit_memory=1` на машине, где будет запускаться ELK стек:

```sh
echo 'vm.max_map_count=262144' >> /etc/sysctl.conf
echo 'vm.overcommit_memory = 1' >> /etc/sysctl.conf
sysctl -p
```

Для версий 5.3 и выше

> **Внимание**
>
> Начиная с версии 5.3.0 логи пользовательских действий поступают вместе со всеми другими служебными логами в stdout контейнеров по умолчанию.Формат логов - JSON. В связи с этим ELK стек выходит из состава поставки Test IT.

Вы можете использовать ELK стек или другие решения для централизованного сбора логов самостоятельно.

Если нужно выделять/фильтровать логи пользовательских действий среди других, сделать это можно, применив фильтр, который оставляет только те логи, в которых JSON-ключ `LogType` в разделе `Properties` имеет значение `UserAction` :

```json
// пример содержимого лога пользовательских действий
{
//...
"Properties": {
//...
"LogType": "UserAction",
//...
},
//...
}
```

> **Как временное решение (т.е. до полного перехода на ваши инструменты логирования), пример использования ELK+filebeat стека для логирования пользовательских действий представлен ниже.**
>
> ДИСКЛЕЙМЕР: это не production-ready, а концептуально-демонстрационное решение

Состав файлов для запуска:

```yml
- filebeat.sh
- filebeat.yml
- kibana.yml
- logstash.sh
- logstash.yml
- logstash.conf
- docker-compose.elk.yml
```

Ниже представлено содержимое файлов.

filebeat.sh

```sh
#!/bin/bash

cat /tmp/filebeat.yml > /usr/share/filebeat/filebeat.yml

chown -R root /usr/share/filebeat/
chmod -R go-w /usr/share/filebeat

/usr/local/bin/docker-entrypoint -environment container
```

filebeat.yml

```yml
filebeat.inputs:
- type: log
paths:
- /var/lib/docker/containers/*/*-json.log

output.logstash:
hosts: ["logstash:5044"]
```

kibana.yml

```yml
server.name: kibana
server.host: "0"
elasticsearch.hosts: ["http://elasticsearch:9200"]
xpack.monitoring.ui.container.elasticsearch.enabled: true
```

logstash.sh

sh

```sh
#! /bin/bash

generate_put_data()
{
cat <<EOF
{
"policy": {
"phases": {
"hot": {
"min_age": "0ms",
"actions": {
"rollover": {
"max_age": "24h"
}
}
},
"delete": {
"min_age": "${EVENT_LOG_MAX_AGE:-30d}",
"actions": {
"delete": {}
}
}
}
}
EOF
}

curl --retry 8 --location \
--request PUT
"${ELASTICSEARCH_CONNECTION_STRING}/_ilm/policy/testit_logs_policy?
pretty" \
--header 'Content-Type: application/json' \
--data "$(generate_put_data)"

cat /tmp/logstash.yml > /usr/share/logstash/config/logstash.yml
cat /tmp/logstash.conf > /usr/share/logstash/pipeline/logstash.conf

/usr/local/bin/docker-entrypoint
```

logstash.yml

```yml
http.host: "0.0.0.0"
```

logstash.conf

```json
input {
# Accept logs from filebeat, listening on port 5044
beats {
port => 5044
}
}

filter {
# Docker-level JSON logs parser
json {
source => "message"
tag_on_failure => ["json_parse_error"] # Error-handling for JSON
parsing
}
# Date format for Kibana
date {
match => ["time", "ISO8601"]
target => "@timestamp"
}
# Renaming fields for convenience
mutate {
rename => { "[log]" => "message" }
rename => { "[stream]" => "log_stream" }
}
#####################################---Optional Block---
#####################################
# #
# #
if [message] =~ /^\{.*\}$/ { # Check the message field for JSON-
compatibility
json {
source => "message"
tag_on_failure => ["nested_json_parse_error"] # Error-handling for
JSON parsing
}
} else {
drop { } # Non-JSON logs are dropped (necessary for the filter
below)
}

if [Properties][LogType] != "UserAction" {
drop { } # Non-"aciton" logs are dropped (optional)
}
# #
```

```
# #
#####################################---Optional Block---
#####################################
}

output {
elasticsearch {
hosts => "${ELASTICSEARCH_CONNECTION_STRING}"
ilm_enabled => true
ilm_rollover_alias => "testit_logs" # < Index name prefix in
ElasticSearch
ilm_pattern => "{now/d{dd.MM.yyyy}}-000001" # < Index name
postfix, containing date and number
ilm_policy => "testit_logs_policy" # < Log storage policy, created
during logstash init (logstash.sh)
}
}
```

docker-compose.elk.yml

```yml
version: '3.8'

x-logging: &default-logging
driver: "json-file"
options:
max-size: "500m"
max-file: "2"

networks:
testit_monitoring_network:
name: "testit_monitoring_stack"
external: true

x-networks: &default-networks
- "testit_monitoring_network"

x-elk-service: &elk-service-defaults
logging: *default-logging
networks: *default-networks
restart: unless-stopped

services:
elasticsearch:
image: elasticsearch:8.8.1
<<: *elk-service-defaults
environment:
ES_JAVA_OPTS: "-Xms1024m -Xmx1024m"
bootstrap.memory_lock: "true"
cluster.name: "elasticsearch"
discovery.type: "single-node"
network.host: "0.0.0.0"
http.port: "9200"
xpack.security.enabled: "false"
xpack.security.http.ssl.enabled: "false"
ulimits:
memlock:
soft: -1
hard: -1
healthcheck:
test: ["CMD", "curl", "-f",
"http://localhost:9200/_cluster/health"]
interval: 30s
timeout: 10s
retries: 5
```

```
volumes:
- elastic-volume:/usr/share/elasticsearch/data

logstash:
image: logstash:8.8.1
<<: *elk-service-defaults
environment:
PATH_CONFIG: "/usr/share/logstash/pipeline/logstash.conf"
ELASTICSEARCH_CONNECTION_STRING: "http://elasticsearch:9200"
ELASTICSEARCH_INDEX: "testit"
ELASTICSEARCH_LOGS_INDEX: "testit_logs"
LS_JAVA_OPTS: "-Xmx512m -Xms512m"
EVENT_LOG_MAX_AGE: 30d
ELASTICSEARCH_SSL_ENABLED: "false"
entrypoint: ["/bin/bash", "-c"]
command: ["/tmp/launch.sh"]
healthcheck:
test: ["CMD", "curl", "-f", "http://localhost:9600/_node"]
interval: 30s
timeout: 10s
retries: 3
volumes:
- ${PWD}/logstash.conf:/tmp/logstash.conf:ro
- ${PWD}/logstash.yml:/tmp/logstash.yml:ro
- ${PWD}/logstash.sh:/tmp/launch.sh:ro
- /var/lib/docker/containers:/var/lib/docker/containers:ro
depends_on:
elasticsearch:
condition: service_healthy

kibana:
image: "kibana:8.8.1"
<<: *elk-service-defaults
ports:
- ${KIBANA_EXPOSE_PORT:-5601}:5601
environment:
SERVER_NAME: "localhost"
SERVER_HOST: "0.0.0.0"
ELASTICSEARCH_URL: "http://elasticsearch:9200"
ELASTICSEARCH_HOSTS: "http://elasticsearch:9200"
SERVER_SSL_ENABLED: "false"
healthcheck:
test: ["CMD", "curl", "-f", "http://localhost:5601/api/status"]
interval: 30s
timeout: 10s
```

```
    retries: 3
    volumes:
    - ${PWD}/kibana.yml:/usr/share/kibana/config/kibana.yml:ro
    depends_on:
    elasticsearch:
    condition: service_healthy

    filebeat:
    image: elastic/filebeat:8.8.1
    user: root
    <<: *elk-service-defaults
    entrypoint: ["/bin/bash", "-c"]
    command: ["/tmp/launch.sh"]
    volumes:
    - ${PWD}/filebeat.yml:/tmp/filebeat.yml:ro
    - ${PWD}/filebeat.sh:/tmp/launch.sh:ro
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    healthcheck:
    test: ["CMD", "filebeat", "test", "output"]
    interval: 30s
    timeout: 10s
    retries: 3
    depends_on:
    logstash:
    condition: service_healthy


    volumes:
    elastic-volume:
```

Команды для развертывания ELK стека:

```sh
# Создаем в локальной директории все файлы из списка выше

# Добавляем возможность запуска для скриптов
chmod +x logstash.sh filebeat.sh

# Создаем сеть docker
docker network create testit_monitoring_stack

# Запускаем стек
docker compose -f docker-compose.elk.yml -p testit-monitoring up -
d

# После завершения команды запуска переходите на
http://localhost:5601 (:порт может отличаться если вы задали
другое значение переменной KIBANA_EXPOSE_PORT)
# Информацию по пользованию Kibana (веб-интерфейс ELK стека)
можете найти по ссылке ниже:
# https://www.elastic.co/guide/en/kibana/8.8/data-views.html
```

Команды для остановки ELK стека

```sh
# Временная остановка проекта с возможностью перезапуска
контейнеров
docker compose -f docker-compose.elk.yml -p testit-monitoring down

# Полная остановка с удалением данных, содержащихся в томе
elastic-volume
docker compose -f docker-compose.elk.yml -p testit-monitoring down
--volumes

# Удаление сети docker
docker network rm testit_monitoring_stack
```

Для версий 4.0 и выше

> **Внимание**
>
> Начиная с версии 4.0.0 структура файла `docker-compose.elk.yml` была изменена, в новых версиях в этом файле содержатся только сервисы ELK стека.

Для включения опции логирования пользовательских действий:

1. Выполните следующие команды:

```sh
cd ~/testit
cp docker-compose.yml docker-compose.yml.bak # резервная копия
docker compose -f docker-compose.yml -f docker-compose.elk.yml
config --no-interpolate > docker-compose.yaml
```

   Последняя команда объединяет содержимое файлов `docker-compose.yml` и `docker-compose.elk.yml` и записывает результат в файл `docker-compose.yml`, замещая предыдущее содержимое.
2. Выполните шаги из Инструкции по установке в Docker Compose или Руководства по обновлению.

> **Альтернативный способ запуска стека ELK**
>
> Альтернативным способом включения сервисов ELK-стека может быть выполнение команд docker compose с двумя ключами “-f”, при этом передается несколько файлов.
>
> - Например, для установки Test IT может быть выполнена следующая команда:
>
> ```sh
> docker compose -f docker-compose.yml -f docker-compose.elk.yml --project-name testit up --detach --timeout 120
> ```
>
> - Для создания резервной копии и восстановления из нее следующие команды:
>
> ```sh
> scripts/backup.sh "docker-compose.yml -f docker-compose.elk.yml" testit
> scripts/restore.sh "docker-compose.yml -f docker-compose.elk.yml" testit backup_21_05_2019.tar
> ```

Для версий ниже 4.0

Для включения опции логирования пользовательских действий:

1. Замените `docker-compose.yml` на содержимое `docker-compose.elk.yml` :

```sh
cd ~/testit
cp docker-compose.yml docker-compose.yml.bak # резервная копия
cp docker-compose.elk.yml docker-compose.yml
```

2. Выполните шаги из **Инструкции по установке в Docker Compose** или **Руководства по обновлению**.

Обновлено: 27.03.2025, 13:53:38

Настройка HTTPS в Docker Compose

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Перед настройкой https необходимо провести базовую настройку и установку Test IT.

1. В `.env` -файле раскомментируйте переменные SSL_CERTIFICATE и SSL_CERTIFICATE_KEY:

```text
## internal certificate path
SSL_CERTIFICATE=/etc/nginx/ssl/testit.crt
SSL_CERTIFICATE_KEY=/etc/nginx/ssl/testit.key
#REDIRECT_TO_HTTPS=true
```

2. Раскомментируйте REDIRECT_TO_HTTPS для редиректа `http` на `https` .

3. Чтобы сервис frontend был доступен по `https` протоколу, пропишите 443 порт в `docker-compose.yml` файле:

```text
services:
frontend:
image: "${DOCKER_REGISTRY}/frontend:${CONTAINER_VERSION}"
ports:
- 80:80
- 443:443
...
```

4. Подготовьте файлы с сертификатом и ключом. Для этого дайте им имена `testit.crt` и `testit.key` . Имена файлов сертификатов должны соответствовать значению переменных SSL_CERTIFICATE и SSL_CERTIFICATE_KEY в `.env` -файле.

5. Скопируйте подготовленные файлы в хранилище сертификатов:

```text
certs=$(docker inspect testit_ssl-volume --format '{{
.Mountpoint }}')

chown 1611 testit.crt testit.key

chgrp 0 testit.crt testit.key

cp -p testit.crt ${certs}/

cp -p testit.key ${certs}/
```

6. Примените изменения, выполнив команду:

```sh
docker compose -f docker-compose.yml --project-name testit up --detach --timeout 120
```

Обновлено: 28.10.2025, 13:38:21

Настройка HTTPS в Kubernetes

Перед настройкой HTTPS нет необходимости проводить установку Test IT.

1. Подготовьте файлы с сертификатом и ключом. Назвать их можно, например `tls.crt` и `tls.key` .
2. Переведите содержимое файлов в формат base64:

```sh
cat tls.crt | base64 | tr -d "\n" > tls-encoded.crt
cat tls.key | base64 | tr -d "\n" > tls-encoded.key
```

3. Создайте секрет с закодированными файлами в нужном пространстве имен:

```sh
kubectl -n <namespace> create secret tls my-tls-secret --
cert=tls-encoded.crt --key=tls-encoded.key
```

4. Настройте **ingress** для выбранного доменного имени и tls-секрета:

```yml
# testit_unichart/values-override.yaml
frontend:
ingress:
main:
ingressClassName: "nginx" # Set ingress class name
host: "my.hostname.com" # Set the desired hostname you own
path: /
pathType: Prefix
annotations:
nginx.ingress.kubernetes.io/ssl-redirect: "True"
nginx.ingress.kubernetes.io/proxy-connect-timeout: "300"
nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
nginx.ingress.kubernetes.io/proxy-body-size: 100m
nginx.ingress.kubernetes.io/proxy-buffer-size: 50m
nginx.ingress.kubernetes.io/proxy-buffers-number: "4"
tls: true
tlsSecretName: my-tls-secret
```

5. Если данная настройка проходит на этапе до установки Test IT, продолжайте следовать описанным шагам.
6. Если Test IT был уже установлен, примените изменения:

```sh
cd <installation_folder>
helm -n <my-namespace> -f testit_unichart/values-override.yaml
upgrade testit testit_unichart/ --wait --timeout 10m
```

7. **Опционально**: Вы можете использовать готовые решения для настройки HTTPS в K8s, например cert-manager + Letsencrypt . Для этого добавьте соответствующие флаги и аннотации в *ingress*:

```yml
# testit_unichart/values-override.yaml
frontend:
ingress:
main:
ingressClassName: "nginx" # Set ingress class name
host: "my.hostname.com" # Set the desired hostname you own
path: /
pathType: Prefix
annotations:
nginx.ingress.kubernetes.io/ssl-redirect: "True"
nginx.ingress.kubernetes.io/proxy-connect-timeout: "300"
nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
nginx.ingress.kubernetes.io/proxy-body-size: 100m
nginx.ingress.kubernetes.io/proxy-buffer-size: 50m
nginx.ingress.kubernetes.io/proxy-buffers-number: "4"
cert-manager.io/cluster-issuer: letsencrypt-production
cert-manager.io/common-name: my.hostname.com
tls: true
tlsSecretName: my-ssl-secret
```

Обновлено: 15.04.2025, 13:00:12

Добавление самоподписанных сертификатов в контейнеры (Docker Compose)

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Вам может понадобиться добавить самоподписанные сертификаты в контейнеры `webapi` и `rabbitmqconsumer` как доверенные. Обычно это необходимо при:

- Интеграции с клиентом Jira, если в нем используются самоподписанные сертификаты
- Миграции с TestRail, если между серверами отсутствует SSL соединение

Определение сертификатов для добавления

Данный шаг рассмотрен на примере Google Chrome. Вы можете повторить аналогичную процедуру в вашем браузере.

Чтобы определить, какие сертификаты вам необходимо добавить в список доверенных, откройте Jira или TestRail в вашем браузере и выполните следующие действия:

1. Откройте меню настроек подключения, нажав на значок безопасного подключения.
2. Выберите секцию **Безопасное подключение**.
3. Нажмите **Действительный сертификат**.
4. В открывшемся окне перейдите на вкладку **Certificate path**, чтобы найти сертификат, требующийся для интеграции. Если сертификатов больше, чем 2, вам нужны все сертификаты. Если сертификатов 2, вам нужен первый (корневой) сертификат.

Добавление сертификата вручную

1. Подготовьте сертификаты (в случаях с использованием промежуточного сертификата необходимо подготовить всю цепочку).
2. Выполните следующие команды:
   1. `trusted_certs=$(docker inspect testit_trusted-certificates-volume --format '{{ .Mountpoint }}')`
   2. `cp -p certificate.crt ${trusted_certs}/`
3. Перезапустите систему Test IT:

```sh
docker compose -f docker-compose.yml --project-name testit restart
```

Добавление сертификата с помощью bash-скрипта

Вы также можете создать и запустить bash-скрипт, чтобы автоматизировать данный процесс.

Чтобы сделать это, скопируйте в текстовый файл `copy_cert.sh` следующие команды:

```sh
#!/bin/bash
trusted_certs=$(docker inspect testit_trusted-certificates-volume
--format '{{ .Mountpoint }}')
Cert=$(find . -name "*.crt")
Cert_count=$(find . -name "*.crt" | wc -l)
echo "Найдено '${Cert_count}' файлов формата .crt"
cp -p $Cert ${trusted_certs}
echo "Сертификаты скопированы"
```

Чтобы добавить самоподписанные сертификаты при помощи скрипта:

1. Перейдите в директорию с сертификатами и скопируйте в нее скрипт `copy_cert .`
2. Настройте права на исполнение скрипта: `sudo chmod +x copy_cert.sh`
3. Запустите скрипт: `bash copy_cert.sh`

Если у вас нет доступа к вольюму, вы получите ошибку no permissons. В таком случае запустите скрипт под sudo: `sudo copy_cert.sh`

> **Внимание**
>
> Если добавление сертификатов в список доверенных не помогает или сертификаты отсутствуют, то **в крайнем случае** вы можете отключить их проверку для Jira. Для этого необходимо открыть `.env` -файл и поменять значение переменной `INSECURE_REMOTES` на ваш адрес Jira без протокола, но с указанием порта. Например, `INSECURE_REMOTES=example.com:443` .

Обновлено: 22.07.2024, 15:43:26

Удаление Test IT в Docker Compose

> **Назовите свой проект**
>
> В качестве примера в этой инструкции используется проект с именем `testit` . Вы можете использовать другое название.

Вы можете удалить систему Test IT с сохранением информации (тома и содержащиеся в них данные будут сохранены) или с потерей всех данных (полное удаление: все тома и содержащиеся в них данные будут потеряны).

Чтобы удалить систему с сохранением информации:

- Выполните команду:

```sh
docker compose -f docker-compose.yml --project-name testit down --timeout 120
```

Чтобы удалить систему полностью (с потерей данных):

- Выполните команду:

```sh
docker compose -f docker-compose.yml --project-name testit down --volumes --timeout 120
```

Обновлено: 07.02.2024, 17:36:29

Удаление Test IT в Kubernetes

> **Внимание!**
>
> Эта инструкция описывает **полное удаление системы с потерей всех данных**. Чтобы сохранить данные, перед удалением создайте резервную копию.

Чтобы полностью удалить Test IT:

- Используйте набор команд:

```sh
# Зафиксируйте наименование пространства имен.
export NAMESPACE=<my-namespace>
# Проверьте, имеются ли в пространстве имен установленные чарты.
helm -n $NAMESPACE list

# Удалите чарты внешнего интерфейса (frontend) и внутреннего интерфейса (backend).
helm -n $NAMESPACE uninstall testit

# Опционально: Дождитесь остановки подов.
kubectl -n $NAMESPACE get pods --watch
```

Обновлено: 27.03.2025, 13:54:57