

Руководство системного администратора

Данный раздел описывает процессы работы с системой 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 уже существующие у вас базы данных.

Данное руководство содержит следующие разделы:

- Установка в Docker Compose и Kubernetes
- Настройка внешних подключений в Docker Compose и Kubernetes
- Работа с компонентами Kubernetes:
 - Изменение выделенных ресурсов
 - Замена рабочего узла
 - Настройка SSL для внутренних подключений
 - Переход на новый кластер K8s
 - Перезапуск подов и остановка компонентов Test IT
 - Переопределение переменных и настроек приложений
 - Переход из Docker в Kubernetes
- Перезапуск системы в Docker Compose и 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.

Установка в Docker Compose

- Установка в Docker Compose
 - Требования
 - Состав поставки
- Подготовка
- Автономная установка
- Установка онлайн

Назовите свой проект

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

Требования

- 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` :

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

sh

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:

```
1 firewall-cmd --zone=public --add-port=80/tcp --permanent
2 firewall-cmd --zone=public --add-port=22/tcp --permanent
3 firewall-cmd --zone=public --add-port=443/tcp --permanent
4 firewall-cmd --reload
```

sh

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 репозиториях.

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:

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

sh

Начиная с версии v5.2.0:

```
1 cd ~/testit sh
2 mkdir images
3 tar -zxvf images.tar.gz -C images
4 for file in $(ls images/*.tar); do docker image load -i $file;
5 done
6 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:

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

Обновлено: 27.03.2025, 13:54:57

Описание .env-файла

- Репозиторий для скачивания образов установки Test IT:

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

- Текущая версия программы:

```
1 | TMS_CONTAINER_VERSION=4.0.1 | text
```

- 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**.

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

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

```
1 | ## internal certificate path | text
2 | #SSL_CERTIFICATE=/etc/nginx/ssl/testit.crt
3 | #SSL_CERTIFICATE_KEY=/etc/nginx/ssl/testit.key
4 | #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 ";"

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

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

```
1 | AWS_ACCESS_KEY=testitAccessKey | text
2 | AWS_SECRET_KEY=testitSecretKey
3 | AWS_CONNECTION_STRING=http://minio:9000
```

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

```
1 | AVATARS_AWS_ACCESS_KEY=${AWS_ACCESS_KEY} | text
2 | AVATARS_AWS_SECRET_KEY=${AWS_SECRET_KEY}
3 | AVATARS_AWS_CONNECTION_STRING=${AWS_CONNECTION_STRING}
```

- Параметры подключения к RabbitMQ:

```
1 | RABBITMQ_DEFAULT_USER=testit | text
2 | RABBITMQ_DEFAULT_PASS=F1rstL0g0N!
3 | RABBITMQ_DEFAULT_VHOST=testitrabbit
4 | RABBITMQ_DEFAULT_HOST=rabbitmq
5 | RABBITMQ_DEFAULT_PORT=5672
6 | RABBITMQ_AUTH_MODE=plain
7 | RABBITMQ_CLIENT_CERT_PATH=/etc/rabbitmq/ssl/client/testit.pfx
8 | #RABBITMQ_CLIENT_CERT_PASSPHRASE=
```

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

```
1 | #RABBITMQ_SSL_ENABLED=true | text
```

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

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

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

```
1 | POSTGRES_DB=testitdb | text
2 | POSTGRES_USER=postgres
3 | POSTGRES_PASSWORD=F1rstL0g0N!
```

- Аналогично для Auth DB:

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

- Аналогично для Avatar DB:

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

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

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

- Параметры SSL соединения InfluxDB:

```
1 #INFLUXDB_HTTP_HTTPS_ENABLED=true text
2 #INFLUXDB_HTTP_HTTPS_CERTIFICATE=/var/lib/influxdb/tls/server.crt
3 #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)):

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

- Параметры SSL соединения Elasticsearch:

```
1 | #ELASTICSEARCH_SECURITY_ENABLED=true | text
2 | #ELASTICSEARCH_SSL_ENABLED=true
3 | #ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES=/usr/share/elasticsearch/
```

- Параметры аутентификации в Elasticsearch:

```
1 | #ELASTICSEARCH_AUTH_ENABLED=true | text
2 | #ELASTICSEARCH_USERNAME=elastic
3 | #ELASTICSEARCH_PASSWORD=pass
4 | #KIBANA_SERVER_NAME=localhost
5 | #KIBANA_ELASTICSEARCH_USERNAME=kibana_system
6 | #KIBANA_ELASTICSEARCH_PASSWORD=pass
7 | #LOGSTASH_ELASTICSEARCH_USERNAME=elastic
8 | #LOGSTASH_ELASTICSEARCH_PASSWORD=pass
```

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

```
1 | AUTH_CACHE_CONNECTION_STRING=auth-cache | sh
```

- Пароль для аутентификации в Redis:

```
1 | #REDIS_PASSWORD=pass | text
```

- Параметры SSL соединения Redis:

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

- Системные параметры оставить без изменений:

```
1 | ASPNETCORE_ACCESS_TOKEN_EXPIRATION_MINUTES=8000 | text
2 | ASPNETCORE_REFRESH_TOKEN_EXPIRATION_MINUTES=88000
```

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

1		API_LOG_LEVEL=Warning	text
---	--	-----------------------	------

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

1		THREAD_PER_PROCESSOR=10	text
---	--	-------------------------	------

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

1		USE_PKCE=true	text
---	--	---------------	------

- 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:

1		SYNC_RESULT_LINKS_EVERY_SEC=120	text
---	--	---------------------------------	------

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

1		EVENT_LOG_MAX_AGE=30d	text
---	--	-----------------------	------

- Название бакета MinIO для подключения:

1		TMS_FILE_BUCKET_NAME=testit	text
---	--	-----------------------------	------

- Параметр настройки интервала расчета стабильности автотестов:

1		CALCULATION_AUTOTESTS_STABILITYPERCENTAGE_DELAY_SECONDS=600	text
---	--	---	------

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

1		DELETE_ARCHIVE_DATA_DELAY_SECONDS=600	text
---	--	---------------------------------------	------

- Параметр, позволяющий включить поддержку продукта TeamStorm:

1		CWM_ENABLED=false	text
---	--	-------------------	------

Обновлено: 11.12.2023, 18:44:17

Установка в Kubernetes

Требования

- 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:

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

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:

```
1 cd <installation_folder> sh
2 helm upgrade --install -f testit_unichart/values-override.yaml
3 -n <my-namespace> --create-namespace testit testit_unichart/ --
4 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 :

```
1 # testit_frontend/values.yaml или testit_frontend/values- yml
2 override.yaml
3 #---
4 frontend:
5 #---
6 ingress:
7 main:
8 host: "my.hostname.com" # Set the desired hostname you own
#---
```

Значения, используемые в файле "values"

- Настройки для развертывания:

```
1  frontend:
2  replicaCount: 2 # Количество копий
3  resources:
4  requests: # Минимальные выделяемые ресурсы
5  memory: "128Mi"
6  cpu: "200m"
7  limits: # Лимит выделяемых ресурсов
8  memory: "512Mi"
9  cpu: "500m"
```

yaml

- Публичное DNS-имя приложения:

```
1  frontend:
2  ingress:
3  main:
4  host: "my.hostname.com"
5  webapi:
6  ingress:
7  main:
8  host: "my.hostname.com"
```

yaml

- Общие конфигурационные переменные:

```
1  config:
2  NGINX_JSON_LOGGING: "false"
3  SSL_STRONG_CIPHERS: "false"
```

yaml

- Конфигурационные файлы:

```
1 # Определение конфигурационного файла в соответствии с
2 расположением относительно директории testit_unichart
3 configFile:
4 nginx-conf-cm:
5 nginx.conf: "files/nginx.conf"
6 # Расположение конфигурационного файла в контейнере frontend
7 volumes:
8 nginx-conf:
9 fromConfigMap: nginx-conf-cm
10 mounts:
11 frontend:
12 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):

```
1 auth-ache:
2 enabled: true # Развертывание внутри кластера
3 resources:
4 requests: # Минимальные выделяемые ресурсы
5 memory: "128Mi"
6 cpu: "100m"
7 limits: # Лимит выделяемых ресурсов
8 memory: "256Mi"
9 cpu: "200m"
10 statefulset:
11 enabled: true # Приложение разворачивается в K8S как
12 statefulset
13 storage:
14 mountPaths:
15 - /data # Путь монтирования PVC в контейнер
16 spec: # Конфигурация PVC
17 accessModes: ["ReadWriteOnce"]
18 # Set storageClassName (nfs-client, local-path, etc.)
19 # storageClassName: "nfs-client"
20 resources:
21 requests:
22 storage: "256Mi"
```

- Общая конфигурация

```
1 default: # <-- Секция со значениями по умолчанию, применяется yml
2 если поле обязательное, но для конкретного сервиса не указано
3 general: # <-- Секция с общими значениями, применяется для всех
4 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>
```

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):

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

2. В файле `docker-compose.yml` прокомментируйте 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:

```
1 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:

```
1 | chown 100 rabbitmq_key.pem rabbitmq_ca.pem rabbitmq_cert.pem | sh
2 | chgrp 101 rabbitmq_key.pem rabbitmq_ca.pem rabbitmq_cert.pem
3 | chmod 600 rabbitmq_key.pem
```

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

```
1 | server_certs=$(docker inspect testit_rabbitmq-certificates- | sh
2 | 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` :

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

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

```
1 listeners.ssl.default = 5671 sh
2 ssl_options.cacertfile = /etc/certs/rabbitmq_ca.pem
3 ssl_options.certfile = /etc/certs/rabbitmq_cert.pem
4 ssl_options.keyfile = /etc/certs/rabbitmq_key.pem
5 ssl_options.verify = verify_none
6 ssl_options.fail_if_no_peer_cert = false
```

6. Скопируйте файл в вольюм с конфигурацией rabbitmq :

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

7. В файле .env раскомментируйте строку:

```
1 RABBITMQ_SSL_ENABLED=true sh
```

8. Измените порт для подключения по ssl на 5671 :

```
1 RABBITMQ_DEFAULT_PORT=5671 sh
```

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

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

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

- Настройка внешнего подключения
- Настройка безопасного соединения и аутентификации между компонентами ELK
- Возможность конфигурирования Logstash

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:

```
1 ELASTICSEARCH_CONNECTION_STRING=http://external-server:9200
2 (где external-server – IP-адрес или DNS-имя вашего сервера с
3 Elasticsearch)
4 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)
```

text

2. В файле `docker-compose.yml` выполните следующие действия:

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

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

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

```
1 | Serilog__AdminAll__WriteTo__1__Args__requestUri: text
   | "${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:

```
1 | docker compose -f docker-compose.yml --project-name testit up - sh
   | -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`

```
1 version: '2.2'
2
3 services:
4   create_certs:
5     image:
6     docker.elastic.co/elasticsearch/elasticsearch:${VERSION}
7     container_name: create_certs
8     command: >
9     bash -c '
10    yum install -y -q -e 0 unzip;
11    if [[ ! -f /certs/bundle.zip ]]; then
12    bin/elasticsearch-certutil cert --silent --pem --in
13    config/certificates/instances.yml -out /certs/bundle.zip;
14    unzip /certs/bundle.zip -d /certs;
15    fi;
16    chown -R 1000:0 /certs
17    '
18    working_dir: /usr/share/elasticsearch
19    volumes:
20    - ./certs:/certs
21    - ./usr/share/elasticsearch/config/certificates
22    networks:
23    - elastic
24
25    volumes:
26    certs:
27    driver: local
28
29    networks:
    elastic:
    driver: bridge
```

sh

- create-certs.env

```
1 COMPOSE_PROJECT_NAME=testit
2 VERSION=7.17.5
```

sh

- instances.yml

```
1 instances:
2   - name: elasticsearch
3   dns:
4     - elasticsearch
5     - localhost
6   ip:
7     - 127.0.0.1
8   - name: logstash
9   dns:
10    - logstash
11    - localhost
12   ip:
13    - 127.0.0.1
14   ## Раскомментируйте строки ниже, чтобы настроить HTTPS для
15   Kibana
16   # - name: 'kibana'
17   # dns:
18   # - kibana
19   # - localhost
```

2. Сгенерируйте сертификаты.

```
1 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`.

```
1 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).

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

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

```
1   elk_certs=$(docker inspect testit_elk-tls-volume --format '{{
2   .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.

```
1   trusted_certs=$(docker inspect testit_trusted-certificates-
2   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`.

```
1   ELASTICSEARCH_CONNECTION_STRING=https://elasticsearch:9200
2   LOGSTASH_CONNECTION_STRING=https://logstash:5044
3   ...
4
5   ELASTICSEARCH_SECURITY_ENABLED=true
6   ELASTICSEARCH_SSL_ENABLED=true
7   ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES=/usr/share/elasticsearch/c
```

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

- `elasticsearch`

```
1   xpack.security.http.ssl.certificate_authorities:
2   "${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elasticse
3   xpack.security.http.ssl.key:
   /usr/share/elasticsearch/config/certificates/elasticsearch/elasti
4   xpack.security.http.ssl.certificate:
   /usr/share/elasticsearch/config/certificates/elasticsearch/elasti
```

- `logstash`

```
1 ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES: sh
2 "${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elasticsearch}
3 SERVER_SSL_KEY: /usr/share/elasticsearch/config/certificates/log
SERVER_SSL_CERTIFICATE: /usr/share/elasticsearch/config/certific
```

- o kibana

```
1 ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES: sh
2 "${ELASTICSEARCH_SSL_CERTIFICATEAUTHORITIES:-/usr/share/elasticsearch}
3 SERVER_SSL_KEY: /usr/share/elasticsearch/config/certificates/kibana
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`.

```
1 docker exec testit-elasticsearch-1 /bin/bash -c sh
  "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.

```
1 ELASTICSEARCH_AUTH_ENABLED=true sh
2 ELASTICSEARCH_USERNAME=elastic
3 ELASTICSEARCH_PASSWORD=pass
4 KIBANA_ELASTICSEARCH_USERNAME=kibana_system
5 KIBANA_ELASTICSEARCH_PASSWORD=pass
6 LOGSTASH_ELASTICSEARCH_USERNAME=elastic
7 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.

```
1 | docker compose -f docker-compose.yml --project-name testit up -sh  
  | -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:

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

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`:

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

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:

```
1 ilm_policy => "<policy_name>" sh
2 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](#)

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:

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

text

- Для сервиса `avatars.minio` установите следующие значения переменных:

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

text

- Добавьте для сервиса `avatars.minio` следующую переменную:

1

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

text

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:

1

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

sh

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:

1

```
server_certs=$(docker inspect yourproject_minio-tls-volume --
```

sh

2

```
format '{{ .Mountpoint }}')
```

3

```
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`:

1

```
trusted_certs=$(docker inspect yourproject_trusted-
```

sh

2

```
certificates-volume --format '{{ .Mountpoint }}')
```

```
cp ca.crt ${trusted_certs}/
```

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

```
1 | AWS_CONNECTION_STRING=https://minio:9000
```

sh

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

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

sh

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:

```
1 | appendonly yes
```

text

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:

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

Включение аутентификации в Redis

1. Добавьте пароль в базу данных Redis.

```
1 auth-cache: sh
2 <...>
3 command: >
4 redis-server
5 --appendonly yes
6 --requirepass "${REDIS_PASSWORD}"
```

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

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

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

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

```
1 chown 999 server.key sh
2 chgrp 999 server.key
3 chmod 600 server.key
```

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

```
1 server_certs=$(docker inspect testit_auth-cache-tls-volume -- sh
2 format '{{ .Mountpoint }}')
3 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` :

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

5. В файл `docker-compose.yml` :

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

```
1 | auth-cache: | sh
2 | <...>
3 | command: >
4 | <...>
5 | --tls-port 6379
6 | --port 0
7 | --tls-cert-file "${REDIS_TLS_CERT_FILE}"
8 | --tls-key-file "${REDIS_TLS_KEY_FILE}"
9 | --tls-auth-clients no
```

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

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

- Раскомментируйте следующие строки и отредактируйте при необходимости:

```
1 | REDIS_TLS_CERT_FILE=/tls/server.crt | sh
2 | REDIS_TLS_KEY_FILE=/tls/server.key
```

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:

```
1 | max-series-per-database = 0 | text
2 | 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:

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

Включение аутентификации в InfluxDB

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

```
1 | INFLUX_AUTH_ENABLED=true | text
2 | INFLUX_USERNAME=<USER_NAME>
3 | INFLUX_PASSWORD=<YOUR_PASSWORD>
4 | INFLUXDB_META_DIR=/var/lib/influxdb/meta2
```

2. Выполните следующую команду:

```
1 | docker compose -f docker-compose.yml --project-name testit up - sh
   | -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:

```
1 | server_certs=$(docker inspect testit_influx-tls-volume --format sh
2 | '{{ .Mountpoint }}')
3 | 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`:

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

4. В файле `.env`:

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

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

- Раскомментируйте и при необходимости отредактируйте следующие строки:

```
1 | INFLUXDB_HTTP_HTTPS_ENABLED=true | text
2 | INFLUXDB_HTTP_HTTPS_CERTIFICATE=/var/lib/influxdb/tls/server.crt
3 | 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`:

```
1 | docker restart testit_influxdb_1 | sh
2 | 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:

```
1 yum install postgresql-contrib
2 psql -U postgres
3 create database testitdb;
4 create user tester with encrypted password 'tester';
5 grant all privileges on database testitdb to tester;
6 \connect testitdb;
7 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
```

text

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

- Для `authdb` :

```
1 create database authdb;
2 grant all privileges on database authdb to tester;
3 \connect authdb;
4 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
```

◦ Для `avatarsdb` :

```
1 create database avatarsdb;
2 grant all privileges on database avatarsdb to tester;
3 \connect avatarsdb;
4 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
```

▪ Для `backgrounddb` :

```
1 create database backgrounddb;
2 grant all privileges on database backgrounddb to tester;
3 \connect backgrounddb;
4 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
```

▪ Для `licensedb` :

```
1 create database licensedb;
2 grant all privileges on database licensedb to tester;
3 \connect licensedb;
4 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
```

▪ Для `globalsearchdb` :

```
1 create database globalsearchdb;
2 grant all privileges on database globalsearchdb to tester;
3 \connect globalsearchdb;
4 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
5 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.

```
1 DB_CONNECTION_STRING=Host=external_server1;Port=5432;Database=testi
2 Pool Size=130
3 #POSTGRES_DB=testitdb
4 #POSTGRES_USER=postgres
5 #POSTGRES_PASSWORD=FirstLog0N!
6 ...
7 AUTH_CONNECTION_STRING=Host=external_server2;Port=5432;Database=aut
8 Pool Size=130
9 #POSTGRES_AUTH_DB=authdb
10 #POSTGRES_AUTH_USER=postgres
11 #POSTGRES_AUTH_PASSWORD=FirstLog0N!
12 ...
13 AVATARS_CONNECTION_STRING=Host=external_server3;Port=5432;Database=
14 #POSTGRES_AVATARS_DB=avatarsdb
15 #POSTGRES_AVATARS_USER=postgres
16 #POSTGRES_AVATARS_PASSWORD=FirstLog0N!
17 ...
18 BACKGROUND_CONNECTION_STRING=Host=external_server3;Port=5432;Databc
19 #POSTGRES_BACKGROUND_DB=backgrounddb
20 #POSTGRES_BACKGROUND_USER=postgres
21 #POSTGRES_BACKGROUND_PASSWORD=FirstLog0N!
22 ...
23 LICENSE_DB_CONNECTION_STRING=Host=external_server3;Port=5432;Databc
24 #POSTGRES_LICENSE_DB=licensedb
25 #POSTGRES_LICENSE_USER=postgres
26 #POSTGRES_LICENSE_PASSWORD=FirstLog0N!
27 ...
28 GLOBALSEARCH_CONNECTION_STRING=Host=external_server3;Port=5432;Datc
29 #POSTGRES_GLOBALSEARCH_DB=globalsearchdb
#POSTGRES_GLOBALSEARCH_USER=postgres
#POSTGRES_GLOBALSEARCH_PASSWORD=FirstLog0N!
```

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:

```
1 | # listen_addresses = 'localhost' | text
```

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

```
1 | listen_addresses = '*' | text
```

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

```
1 | host all all 0.0.0.0/0 md5 | text
2 | host all all ::/0 md5 | text
```

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:

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

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:

```
1 | docker compose -f docker-compose.yml --project-name testit up - sh  
  | -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:

```
1 | db: sh  
2 | <...>  
3 | 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:

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

4. Выполните следующую команду:

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

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

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

```
1 chown 999 server.key sh
2 chgrp 999 server.key
3 chmod 600 server.key
```

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

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

7. Выполните следующие команды:

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

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 с параметрами внешней базы данных RabbitMQ:

```
1  general:
2  config:
3  RABBITMQ_DEFAULT_VHOST: "testitrabbit"
4  RABBITMQ_DEFAULT_HOST: "external-server" # IP or DNS of outside
5  RabbitMQ server
6  RABBITMQ_DEFAULT_PORT: "5672"
7  RABBITMQ_AUTH_MODE: "plain"
8  RABBITMQ_CLIENT_CERT_PATH:
9  "/etc/rabbitmq/ssl/client/testit.pfx"
10 RABBITMQ_CLIENT_CERT_PASSPHRASE:
11 RABBITMQ_SSL_ENABLED: "false"
12 secrets:
   RABBITMQ_DEFAULT_USER: "testit"
   RABBITMQ_DEFAULT_PASS: "password"
                                                                    yml
```

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

```
1  rabbitmq:
2  enabled: false
                                                                    yml
```

4. Примените изменения с помощью команды:

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

sh

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/` :

```
1 # Номер файла задан для примера, можно использовать и
2 другое наименование, например ca-bundle-rabbitmq.crt
3 # Главное, чтобы нужные путь/название файла указывались в
  шаге 3 следующей секции
  testit_unichart/files/ssl/ca-bundle-2.crt
```

sh

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` :

```
1 listeners.ssl.default = 5671
2 ssl_options.cacertfile = /etc/certs/rabbitmq_ca.pem
3 ssl_options.certfile = /etc/certs/rabbitmq_cert.pem
4 ssl_options.keyfile = /etc/certs/rabbitmq_key.pem
5 ssl_options.verify = verify_none
6 ssl_options.fail_if_no_peer_cert = false
```

4. Перенесите созданный файл 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:

```
1 # testit_unichart/values-ssl.yaml или testit_unichart/values- yml
2 override.yaml
3 general:
4 config:
5 RABBITMQ_DEFAULT_VHOST: "testitrabbit"
6 RABBITMQ_DEFAULT_HOST: "external-server" # IP or DNS of outside
7 RabbitMQ server
8 RABBITMQ_DEFAULT_PORT: "5672"
9 RABBITMQ_AUTH_MODE: "plain"
10 RABBITMQ_CLIENT_CERT_PATH:
11 "/etc/rabbitmq/ssl/client/testit.pfx"
12 RABBITMQ_CLIENT_CERT_PASSPHRASE:
13 RABBITMQ_SSL_ENABLED: "false"
14 APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
15 secrets:
16 RABBITMQ_DEFAULT_USER: "testit"
17 RABBITMQ_DEFAULT_PASS: "password"
18 volumes:
19 ## Use to enable custom certificates
20 ssl-ca-bundle-1:
21 fromConfigMap: ssl-ca-bundle-1
22 mounts:
23 mountPath: /app/certs/ca-bundle-1.crt
24 subPath: ca-bundle-1.crt
25 # Mount new file
26 ssl-ca-bundle-2:
27 fromConfigMap: ssl-ca-bundle-2
28 mounts:
29 mountPath: /app/certs/ca-bundle-2.crt
30 subPath: ca-bundle-2.crt
31 configFile:
32 ## Use to enable custom certificates
33 # Already have been there (hypothetically)
34 ssl-ca-bundle-1:
35 ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
36 # New file
37 ssl-ca-bundle-2:
38 ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

6. Примените изменения с помощью команды:

```
1 cd <installation_folder>
2 # В зависимости от того, какие файлы (values-override.yaml или
3 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
```

sh

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:

```
1 minio:
2   enabled: false
```

yaml

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

```
1 avatars-api:
2   config:
3     AVATARS_AWS_BUCKET_NAME: "bucket2"
4   general:
5     config:
6     AWS_CONNECTION_STRING: "http://minio-external:9000" # minio-
7     external could be either IP or DNS
8     TMS_BUCKET_NAME: "bucket1"
9   secrets:
10  AWS_ACCESS_KEY: "YourAccessKey"
    AWS_SECRET_KEY: "YourSecretKey"
```

yaml

6. Примените изменения с помощью команды:

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

sh

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/` :

```
1 # Номер файла задан для примера, можно использовать и
2 другое наименование, например ca-bundle-minio.crt
3 # Главное, чтобы нужные путь/название файла указывались в
  шаге 3 следующей секции
  testit_unichart/files/ssl/ca-bundle-2.crt
```

sh

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:

```
1 # testit_unichart/values-ssl.yaml или testit_unichart/values- yml
2 override.yaml
3 general:
4 config:
5   AWS_CONNECTION_STRING: "https://minio-external:9000"
6   APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
7 volumes:
8   ## Use to enable custom certificates
9   ssl-ca-bundle-1:
10    fromConfigMap: ssl-ca-bundle-1
11  mounts:
12    mountPath: /app/certs/ca-bundle-1.crt
13    subPath: ca-bundle-1.crt
14  # Mount new file
15  ssl-ca-bundle-2:
16    fromConfigMap: ssl-ca-bundle-2
17  mounts:
18    mountPath: /app/certs/ca-bundle-2.crt
19    subPath: ca-bundle-2.crt
20  configFile:
21  ## Use to enable custom certificates
22  # Already have been there (hypothetically)
23  ssl-ca-bundle-1:
24    ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
25  # New file
26  ssl-ca-bundle-2:
27    ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

4. Примените изменения с помощью команды:

```
1 cd <installation_folder> sh
2 # В зависимости от того, какие файлы (values-override.yaml или
3 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
```

Подключение Redis в Kubernetes

Проверьте совместимость версии БД

Перед началом работы убедитесь, что версия внешней базы данных Redis совпадает с версией, указанной в `.Values.auth-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:

```
1 | appendonly yes | sh
```

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

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

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

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

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

```
1 | auth-cache: | yml
2 | enabled: false
```

Затем добавьте

```
1 auth:
2 envFromSecret:
3 AUTH_CACHE_CONNECTION_STRING:
4 secretName: general-secret
5 secretKey: AUTH_CACHE_CONNECTION_STRING
```

yml

5. Примените изменения с помощью команды:

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

sh

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/` :

```
1 # Номер файла задан для примера, можно использовать и
2 другое наименование, например ca-bundle-redis.crt
3 # Главное, чтобы нужные путь/название файла указывались в
  шаге 3 следующей секции
  testit_unichart/files/ssl/ca-bundle-2.crt
```

sh

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:

```
1  redis-server \  
2  --appendonly yes \  
3  --tls-port 6379 \  
4  --port 0 \  
5  --tls-cert-file /tls/redis.crt \  
6  --tls-key-file /tls/redis.key \  
7  --tls-ca-cert-file /tls/ca.crt \  
8  --tls-auth-clients no
```

sh

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

```
1 # testit_unichart/values-ssl.yaml или testit_unichart/values- yml
2 override.yaml
3 general:
4 config:
5 AUTH_CACHE_CONNECTION_STRING: "redis-external" # IP or DNS of
6 outside Redis server
7 volumes:
8 ## Use to enable custom certificates
9 ssl-ca-bundle-1:
10 fromConfigMap: ssl-ca-bundle-1
11 mounts:
12 mountPath: /app/certs/ca-bundle-1.crt
13 subPath: ca-bundle-1.crt
14 # Mount new file
15 ssl-ca-bundle-2:
16 fromConfigMap: ssl-ca-bundle-2
17 mounts:
18 mountPath: /app/certs/ca-bundle-2.crt
19 subPath: ca-bundle-2.crt
20 configFile:
21 ## Use to enable custom certificates
22 # Already have been there (hypothetically)
23 ssl-ca-bundle-1:
24 ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
25 # New file
    ssl-ca-bundle-2:
    ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

5. Примените изменения с помощью команды:

```
1 cd <installation_folder> sh
2 # В зависимости от того, какие файлы (values-override.yaml или
3 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
```

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:

```
1 | max-series-per-database = 0 | sh
2 | max-values-per-tag = 0
```

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

```
1 | influxdb: | yml
2 | 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:

```
1 | general: yml
2 | config:
3 | INFLUX_CONNECTION_STRING: "http://influxdb-external:8086" #
4 | influxdb-external could be either IP or DNSservice
5 | INFLUX_AUTH_ENABLED: "true"
6 | secrets:
7 | INFLUX_USERNAME: "influx" # your influx username
   | INFLUX_PASSWORD: "influxpass" # your influx password
```

- Если авторизация не настроена, используйте следующую конфигурацию:

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

5. Примените изменения с помощью команды:

```
1 | cd <installation_folder> sh
2 | 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/` :

```
1 | # Номер файла задан для примера, можно использовать и sh
2 | другое наименование, например ca-bundle-influxdb.crt
3 | # 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:

```
1 # testit_unichart/values-ssl.yaml или testit_unichart/values- yml
2 override.yaml
3 general:
4 config:
5 INFLUX_CONNECTION_STRING: "https://influxdb:8086"
6 APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
7 volumes:
8 ## Use to enable custom certificates
9 ssl-ca-bundle-1:
10 fromConfigMap: ssl-ca-bundle-1
11 mounts:
12 mountPath: /app/certs/ca-bundle-1.crt
13 subPath: ca-bundle-1.crt
14 # Mount new file
15 ssl-ca-bundle-2:
16 fromConfigMap: ssl-ca-bundle-2
17 mounts:
18 mountPath: /app/certs/ca-bundle-2.crt
19 subPath: ca-bundle-2.crt
20 configFile:
21 ## Use to enable custom certificates
22 # Already have been there (hypothetically)
23 ssl-ca-bundle-1:
24 ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
25 # New file
26 ssl-ca-bundle-2:
27 ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

4. Примените изменения с помощью команды:

```
1 cd <installation_folder> sh
2 # В зависимости от того, какие файлы (values-override.yaml или
3 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
```

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`):

```
1 yum install postgresql-contrib
2 psql -U postgres
3 create database testitdb;
4 create user tester with encrypted password 'tester';
5 grant all privileges on database testitdb to tester;
6 \connect testitdb;
7 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
8
9 create database authdb;
10 grant all privileges on database authdb to tester;
11 \connect authdb;
12 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
13
14 create database avatarsdb;
15 grant all privileges on database avatarsdb to tester;
16 \connect avatarsdb;
17 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
18
19 create database backgrounddb;
20 grant all privileges on database backgrounddb to tester;
21 \connect backgrounddb;
22 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
23
24 create database licensedb;
25 grant all privileges on database licensedb to tester;
26 \connect licensedb;
27 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
28
29 create database globalsearchdb;
30 grant all privileges on database globalsearchdb to tester;
31 \connect globalsearchdb;
32 CREATE EXTENSION if not exists "uuid-oss" SCHEMA public;
33 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`

```
1 # Пример для license-service
2 license-service:
3   config:
4     LICENSE_DB_CONNECTION_STRING: >
5     Host=my-external-postgres-host;
6     Port=5432;
7     Database=licensedb;
8     Username=my-username;
9     Password=my-password;
10    Command Timeout=30;
11    Target Session Attributes=any;
```

yml

4. Примените изменения с помощью команды:

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

sh

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/` :

```
1 # Номер файла задан для примера, можно использовать и
2 другое наименование, например ca-bundle-postgres.crt
3 # Главное, чтобы нужные путь/название файла указывались в
  шаге 3 следующей секции
  testit_unichart/files/ssl/ca-bundle-2.crt
```

sh

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:

```
1 | postgres -c 'max_connections=300' -c 'shared_buffers=256MB' -c sh
2 | 'ssl=on' \
3 | -c 'ssl_ca_file=/var/lib/postgresql/tls/ca.crt' \
4 | -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:

```
1 # testit_unichart/values-ssl.yaml или testit_unichart/values- yml
2 override.yaml
3 general:
4 config:
5 APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
6 volumes:
7 ## Use to enable custom certificates
8 ssl-ca-bundle-1:
9 fromConfigMap: ssl-ca-bundle-1
10 mounts:
11 mountPath: /app/certs/ca-bundle-1.crt
12 subPath: ca-bundle-1.crt
13 # Mount new file
14 ssl-ca-bundle-2:
15 fromConfigMap: ssl-ca-bundle-2
16 mounts:
17 mountPath: /app/certs/ca-bundle-2.crt
18 subPath: ca-bundle-2.crt
19 configFile:
20 ## Use to enable custom certificates
21 # Already have been there (hypothetically)
22 ssl-ca-bundle-1:
23 ca-bundle-1.crt: "files/ssl/ca-bundle-1.crt"
24 # New file
25 ssl-ca-bundle-2:
26 ca-bundle-2.crt: "files/ssl/ca-bundle-2.crt"
```

5. Примените изменения с помощью команды:

```
1 cd <installation_folder> sh
2 # В зависимости от того, какие файлы (values-override.yaml или
3 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
```

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 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:

```
1  gotenberg-api:
2  enabled: true
3  excludeGeneralConfig: true
4  excludeGeneralVolumes: true
5  image:
6  repository: "gotenberg/gotenberg"
7  tag: "8.21"
8  pullPolicy: "IfNotPresent"
9  resources:
10 requests:
11 memory: "128Mi"
12 cpu: "250m"
13 limits:
14 memory: "500Mi"
15 cpu: "500m"
16 args:
17 - "gotenberg"
18 - "--api-port-from-env=API_PORT"
19 config:
20 API_PORT: "8080"
21 containerPort: 8080
22 service:
23 port: 8080
```

- Для внешнего подключения используйте конфигурацию:

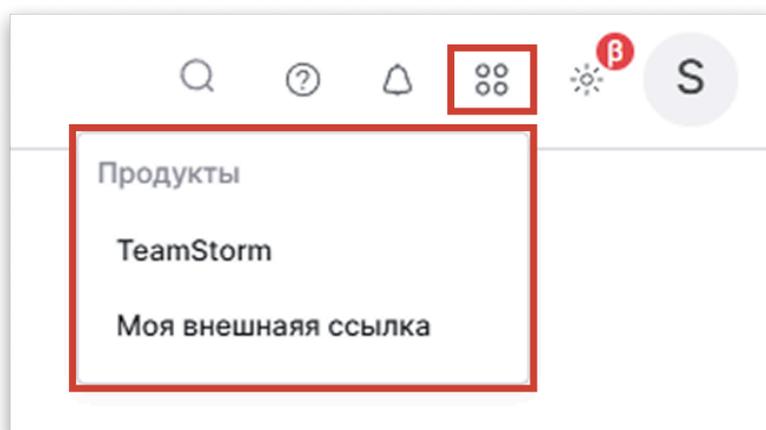
```
1  webapi:
2  enabled: true
3  image:
4  repository: "registry.testit.software/testit/testit"
5  resources:
6  requests:
7  memory: "512Mi"
8  cpu: "50m"
9  limits:
10 memory: "161"
11 cpu: "1"
12 lifecycle:
13 default: true
14 config:
15 AWS_FILE_EXPIRATION_IN_DAYS: "7"
16 Dotnet__Dll: "WebApi.dll"
17 Tms__DbTimeoutInSeconds: "608"
18 ASPNETCORE_ENVIRONMENT: "Production"
19 GotenbergApiUrl: "<YOUR_GOTENBERG_HOST>"
20 INSECURE_RENOTES: ""
21 USE_PKCE: "true"
```

Смотрите также:

- [Официальная документация Gotenberg](#)

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:

```
1  [                                                                                               json
2  { "name": "TeamStorm", "url": "https://work.teamstorm.io/" },
3  { "name": "Моя внешняя ссылка", "url":
4  "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:

```
1 | [                                                                                               json
2 |   { "name": "TeamStorm", "url": "https://work.teamstorm.io/" },
3 |   { "name": "Моя внешняя ссылка", "url":
4 |     "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:

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

6. Выполните перезапуск развертывания фронтенда с помощью команды:

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

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:

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

sh

Перезапуск системы в Kubernetes

- Для полного перезапуска системы используйте команду:

1

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

sh

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:

```
1 resources:
2 requests: # Минимальные выделенные ресурсы
3 memory: "3Gi"
4 cpu: "3"
5 limits: # Лимит выделенных ресурсов
6 memory: "5Gi"
7 cpu: "10"
```

yaml

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

```
1 # Взят компонент postgres для примера, нерелевантные для смены yml
2 ресурсов поля удалены из примера для прозрачности
3 postgres:
4 resources:
5 requests:
6 memory: "1Gi"
7 cpu: "1"
8 limits:
9 memory: "3Gi"
10 cpu: "3"
11 statefulset:
12 storage:
13 spec:
14 resources:
15 requests:
    storage: "5Gi"
```

2. Примените изменения с помощью набора команд:

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

Замена рабочего узла (ноды) в Kubernetes

Подготовка ноды

1. Получите имя текущей ноды с помощью команды:

```
1 | kubectl get nodes | sh
```

2. Сохраните название текущей ноды с помощью команды:

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

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:

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

Перенос рабочих нагрузок на новый узел

1. Остановите активные рабочие нагрузки с помощью команды:

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

2. Переустановите приложение с помощью команды:

```
1 cd <installation_folder> sh
2 export NAMESPACE=<my-namespace>
3 helm upgrade --install -f testit_unichart/values-override.yaml
4 -n $NAMESPACE --create-namespace testit testit_unichart/ --wait
5 --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:

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

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` :

```

1  auth-cache:
2  # Enable internal SSL for redis
3  args:
4  - "redis-server"
5  - "--appendonly"
6  - "yes"
7  - "--tls-port 6379"
8  - "--port 0"
9  - "--tls-cert-file /tls/redis.crt"
10 - "--tls-key-file /tls/redis.key"
11 - "--tls-ca-cert-file /tls/ca.crt"
12 - "--tls-auth-clients no"
13 configFile:
14 auth-cache-ssl-files:
15 redis.crt: "files/ssl/auth-cache.crt"
16 redis.key: "files/ssl/auth-cache.key"
17 ca.crt: "files/ssl/ca.crt"
18 volumes:
19 ssl:
20 fromConfigMap: auth-cache-ssl-files
21 mounts:
22 auth-cache:
23 mountPath: /tls

```

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

```

1  general:
2  config:
3  Redis__ConnectionString: "auth-cache:6379,ssl=true"
4  AUTH_CACHE_CONNECTION_STRING: "auth-cache:6379,ssl=true"
5  APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
6  volumes:
7  ## Use to enable internal SSL
8  ssl-ca-internal:
9  fromConfigMap: ssl-ca-internal
10 mounts:
11 mountPath: /app/certs/ca.crt
12 subPath: ca.crt
13 configFile:
14 ## Use to enable internal SSL
15 ssl-ca-internal:
16 ca.crt: "files/ssl/ca.crt"

```

5. Примените изменения с помощью команды:

```
1 cd <installation_folder>
2 export NAMESPACE=<my-namespace>
3 helm upgrade --install -f testit_unichart/values-override.yaml
4 -n $NAMESPACE --create-namespace testit testit_unichart/ --wait
5 --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` :

```
1 listeners.ssl.default = 5671
2 ssl_options.cacertfile = /etc/certs/server_ca.pem
3 ssl_options.certfile = /etc/certs/server_cert.pem
4 ssl_options.keyfile = /etc/certs/server_key.pem
5 ssl_options.verify = verify_none
6 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:

```
1  sslEnabled: true
2  general:
3  config:
4  APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION: "/app/certs"
5  volumes:
6  ## Use to enable custom certificates
7  ssl-ca-bundle:
8  fromConfigMap: ssl-ca-bundle
9  mounts:
10 mountPath: /app/certs/ca-bundle.crt
11 subPath: ca-bundle.crt
12 configFile:
13 ## Use to enable custom certificates
14 ssl-ca-bundle:
15 ca-bundle.crt: "files/ssl/ca-bundle.crt"
                                                                    yml
```

3. Примените изменения с помощью команды:

```
1 cd <installation_folder>
2 export NAMESPACE=<my-namespace>
3 helm upgrade --install -f testit_unichart/values-override.yaml
4 -n $NAMESPACE --create-namespace testit testit_unichart/ --wait
5 --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:

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

3. Перейдите на новый кластер с помощью команды:

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

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

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:

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

- Чтобы перезапустить все поды приложения, используйте команду:

```
1 kubectl -n <namespace> get deployments # выводит имена всех sh
2 deployment
3 kubectl -n <namespace> scale deployment <name> --replicas=0 #
  выключает все поды deployment-a
  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:

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

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` :

```
1  webapi:
2  configJson:
3  webapi-appsettings-json:
4  appsettings.json: # Name of the file in the configmap
5  # JSON contents:
6  AWS_ACCESS_KEY: "myAWSAccessKey"
7  AWS_SECRET_KEY: "myAWSSecretKey"
8  # Other variables you want to put in a file
9  # ---
10 Serilog:
11 System:
12 MinimumLevel: "Information" # Verbose, Debug, Information,
13 Warning, Error
14 # If you need the app to write to file
15 SystemAll:
16 MinimumLevel: "Information" # Verbose, Debug, Information,
17 Warning, Error
18 WriteTo:
19 - Args:
20 path: "/app/log/system.log" # SystemAll path logs
21 rollingInterval: "Day" # Rolling interval
22 retainedFileCountLimit: "7" # Retention period
23 UserAction:
24 MinimumLevel: "Information" # Verbose, Debug, Information,
25 Warning, Error
26 UserActionAll:
27 MinimumLevel: "Information" # Verbose, Debug, Information,
28 Warning, Error
29 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:

```
1 # values.yaml или values-override.yaml
2 webapi:
3 volumes:
4 appsettings-json:
5 fromConfigMap: webapi-appsettings-json
6 mounts:
7 webapi:
8 mountPath: /app/customs/appsettings.json
9 subPath: appsettings.json
10
11 general:
12 config:
13 APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
14   "/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:

```
1 ## пример для Hangfire__DbConnectionString
2 webapi:
3 configJson:
4 webapi-appsettings-json:
5 appsettings.json:
6 Handfire:
7 DbConnectionString:
8   "Host=postgres;Port=5432;Database=backgrounddb;Username=backgrounddbo
9   Pool Size=130;Command Timeout=30"
```

Обновление Test IT 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.

Обновление до версии 5.4 содержит длительные процедуры миграции

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. Основные рекомендации:

- Во время миграции работа в системе недоступна.
- Отслеживание миграции проводится 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.

- По окончании миграций в логах отобразится сообщение: `Application started` .
- При ошибке и прекращения работы контейнеров, необходимо повторное обновление системы.
- При возникновении проблем, связанных с миграцией, обратитесь в техническую поддержку: (support@yoonion.ru)

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

- Обновление онлайн
- Автономное обновление
- Несовместимость CPU V1 с версией MiniO в Test IT 5.1 и выше
- Переход на Alpine в версии 4.6
- Подготовка к обновлению до версии 4.5
 - Перед началом работы
 - Объединение контейнеров MiniO и Postgres
- Подготовка к обновлению до версии 4.2
 - Перед началом работы
 - Миграция бакетов MiniO
 - Миграция СУБД Postgres
- Подготовка к обновлению до версии 4.0.1
- Подготовка к обновлению до версии 4.0

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:

```
1 | docker compose -f docker-compose.yml --project-name testit up - sh  
  | 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:

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

- Для обновления **до версии 5.1** и предыдущих версий:

```
1 | docker load -i images.tar.gz | sh
2 | 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 `-cpu1`. В результате должно получиться: `minio/minio:RELEASE.2024-07-16T23-46-41Z-cpu1`.
- При автономном обновлении перенесите образ в офлайн-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: 03.07.2025, 14:04:51

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:

```
1 x-tms-vars: &tms-vars
2 ASPNETCORE_ENVIRONMENT: "${ASPNETCORE_ENVIRONMENT:-Production}"
3 APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
4 "${APP_CONFIG_FILEPATH:-}"
5 DOTNET_ENVIRONMENT: "${ASPNETCORE_ENVIRONMENT:-Production}"
6 INSECURE_REMOTES: "${INSECURE_REMOTES:-}"
7 Serilog__System__MinimumLevel: "${API_LOG_LEVEL}"
8 SYSTEM_NAME: "${SYSTEM_NAME:-testit}"
9 TMS_BUCKET_NAME: "${TMS_BUCKET_NAME}"
10 USE_PKCE: "${USE_PKCE}"
    AWS_CREATE_BUCKET_IF_REQUIRED: "${AWS_CREATE_BUCKET_IF_REQUIRED:-
    true}"
```

Также переменные переиспользуются в сервисах следующим образом:

```
1 ldapwebapi:
2 image:
3 "${TMS_DOCKER_REGISTRY}/ldapwebapi:${TMS_CONTAINER_VERSION}"
4 <<: [*tms-service-defaults, *tms-ca-certs-volume]
5 environment:
6 <<: [*tms-vars]
```

Обновление Test IT в Kubernetes

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.

Обновление до версии 5.4 содержит длительные процедуры миграции

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. Основные рекомендации:

- Во время миграции производительность системы снижается в связи с нагрузкой на базу данных.
- Отслеживание миграции проводится 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');`
- 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.

- По окончании миграций в логах отобразится сообщение: `Application started` .
- При ошибке и прекращения работы контейнеров, необходимо повторное обновление системы.
- При возникновении проблем, связанных с миграцией, обратитесь в техническую поддержку: (support@yoonion.ru)

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

- 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)

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:

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

sh

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:

```

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

```

sh

3. Удалите pvc RabbitMQ.

```

1 kubectl -n $NAMESPACE delete pvc rabbitmq-pv-claim-rabbitmq-0

```

text

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:

```

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

```

sh

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: 03.07.2025, 14:04:51

Обновление старых версий Test IT в Kubernetes

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:

```
1 | scripts/ | sh
2 | k8s_minio_migrate.sh
3 | jobs/
4 | minio/
5 | migrate/
6 | job.yaml
7 | configmap.yaml
8 | pvc.yaml
```

2. Добавьте разрешение на использование скрипта:

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

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

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

4. Запустите скрипт, передав ему соответствующее пространство имен:

```
1 | ./scripts/k8s_minio_migrate.sh <namespace> | sh
2 | # Пример: ./scripts/k8s_minio_migrate.sh my-testit-namespace
3 | # Для более развернутых логов и очистки временных файлов,
4 | создаваемых при выполнении скрипта, его можно запустить с
5 | флагами -d и -c соответственно.
6 | # ./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:

```
1 | Migration successful! | sh
```

В случае ошибки миграции логи сохранятся в текущую директорию:

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

```
1 | scripts/ | sh
2 | k8s_postgres_migrate.sh
3 | jobs/
4 | postgres/
5 | import/
6 | job.yaml
7 | configmap.yaml
8 | pvc.yaml
9 | export/
10 | job.yaml
11 | configmap.yaml
```

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

```
1 | export DUMP_VOLUME_SIZE_GB=50 | sh
2 | # 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:

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

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

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

5. Запустите скрипт, передав ему соответствующее пространство имен:

```
1 | ./scripts/k8s_postgres_migrate.sh <namespace> | sh
2 | # Пример: ./scripts/k8s_postgres_migrate.sh my-testit-namespace
3 | # Для более развернутых логов и очистки временных файлов,
4 | создаваемых при выполнении скрипта, его можно запустить с
5 | флагами -d и -c соответственно.
6 | # ./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:

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

В случае ошибки миграции отобразится строка:

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

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

Обновление

Важно

При онлайн-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:

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

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

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

yaml

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

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

sh

Обновлено: 20.06.2025, 18:27:06

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:

```
1 # minio_vars sh
2 export TMS_BUCKET_NAME=testit
3 export AVATARS_AWS_BUCKET_NAME=avatars
4
5 # docker_vars
6 export COMPOSE_FILE_PATH=/absolute/path/to/testit/docker-
7 compose.yml
8 export COMPOSE_PROJECT_PREFIX=prod
   export COMPOSE_NETWORK_NAME=123123_yoonion_network_local
```

2. Выполните резервное копирование docker-версии Test IT:

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

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

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

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

```
1 # testit_unichart/values.yaml
2 frontend:
3   enabled: true
4   image:
5     repository: "docker.testit.ru/tms-frontend/tms-frontend"
6     tag: v5.12.0 #<--env.TMS_FRONTEND_CONTAINER_VERSION
7   config:
8     NGINX_JSON_LOGGING: "true"
9     SSL_STRONG_CIPHERS: "false"
10  webapi:
11    enabled: true
12    image:
13      repository: "registry.testit.software/testit/testit"
14      tag: v5.10.1 #<--env.TMS_CONTAINER_VERSION
15    config:
16      USE_PKCE: "true"
17      INSECURE_REMOTES: ""
18      SYNC_RESULT_LINKS_EVERY_SEC: "120"
19      TEST_RESULT_LINK_REQUEST_LIFETIME_SEC: "600"
20      THREAD_PER_PROCESSOR: "10"
21      SMTP_ENABLE: "false"
22      SMTP_FROM: ""
23      SMTP_HOST: ""
24      SMTP_PORT: ""
25      SMTP_LOGIN: ""
26      SMTP_DOMAIN: ""
27      SMTP_PASSWORD: ""
28      SMTP_CONNECTION: ""
29      SMTP_AUTHENTICATION: ""
30      SMTP_SKIP_CERTIFICATE: "false"
31      SMTP_TLS_VERSION: "tls"
32      SMTP_LOG_ENABLE: "false"
33      PERF_MODULE_ENABLED: "false"
34  avatars-api:
35    enabled: true
36    image:
37      repository: "docker.testit.ru/avatars/avatars-api"
38      tag: v5.2.8 #<--env.AVATARS_CONTAINER_VERSION
39    config:
40      AVATARS_AWS_BUCKET_NAME: "avatars"
41  auth:
42    enabled: true
43    image:
44      repository: "docker.testit.ru/identity/authwebapi"
```

```
45 tag: v5.5.0 #<--env.AUTHWEBAPI_CONTAINER_VERSION
46 config:
47 ASPNETCORE_ACCESS_TOKEN_EXPIRATION_MINUTES: "120"
48 ASPNETCORE_REFRESH_TOKEN_EXPIRATION_MINUTES: "88000"
49 postgres:
50 config:
51 POSTGRES_PASSWORD: "F1rstL0g0N!"
52 POSTGRES_USER: "postgres"
53 POSTGRES_DB: "testitdb"
54 general:
55 config:
56 AWS_CONNECTION_STRING: "http://minio:9000"
57 AWS_CREATE_BUCKET_IF_REQUIRED: "true"
58 TMS_BUCKET_NAME: "testit"
59 COMPlus_EnableDiagnostics: "0"
60 INFLUX_CONNECTION_STRING: "http://influxdb:8086"
61 INFLUX_AUTH_ENABLED: "false"
62 RABBITMQ_DEFAULT_HOST: "rabbitmq"
63 RABBITMQ_SSL_ENABLED: "false"
64 RABBITMQ_DEFAULT_PORT: "5672"
65 RABBITMQ_DEFAULT_VHOST: "testitrabbit"
66 RABBITMQ_AUTH_MODE: "plain"
67 RABBITMQ_CLIENT_CERT_PASSPHRASE: ""
68 RABBITMQ_CLIENT_CERT_PATH: "/etc/rabbitmq/ssl/client/testit.pfx"
69 APPLICATION__CONFIGURATION__CUSTOMFILEPATH:
70 "/app/customs/appsettings.json"
71 FRONTEND_URL: "http://frontend:8080"
72 CHECK_DB_READY: "true"
73 secrets:
74 AWS_ACCESS_KEY: "testitAccessKey"
75 AWS_SECRET_KEY: "testitSecretKey"
76 INFLUX_USERNAME: "influx"
77 INFLUX_PASSWORD: "influxpass"
78 RABBITMQ_DEFAULT_USER: "testit"
79 RABBITMQ_DEFAULT_PASS: "password"
80 AUTH_CACHE_CONNECTION_STRING: "auth-cache"
81 TESTIT_DB_CONNECTION_STRING: >
82 Host=postgres;
83 Port=5432;
84 Database=testitdb;
85 Username=postgres;
86 Password=F1rstL0g0N!;
87 Command Timeout=30;
88 Target Session Attributes=any;
89 Pooling=true;
```

```
90 Maximum Pool Size=130;
91 BACKGROUND_DB_CONNECTION_STRING: >
92 Host=postgres;
93 Port=5432;
94 Database=backgrounddb;
95 Username=postgres;
96 Password=FirstL0g0N!;
97 Command Timeout=30;
98 Target Session Attributes=any;
99 Pooling=true;
100 Maximum Pool Size=130;
101 GLOBALSEARCH_DB_CONNECTION_STRING: >
102 Host=postgres;
103 Port=5432;
104 Database=globalsearchdb;
105 Username=postgres;
106 Password=FirstL0g0N!;
107 Command Timeout=30;
108 Target Session Attributes=any;
109 Pooling=true;
110 Maximum Pool Size=130;
111 AUTH_DB_CONNECTION_STRING: >
112 Host=postgres;
113 Port=5432;
114 Database=authdb;
115 Username=postgres;
116 Password=FirstL0g0N!;
117 Command Timeout=30;
118 Target Session Attributes=any;
119 Pooling=true;
120 Maximum Pool Size=130;
121 LICENSE_DB_CONNECTION_STRING: >
122 Host=postgres;
123 Port=5432;
124 Database=licensedb;
125 Username=postgres;
126 Password=FirstL0g0N!;
127 Command Timeout=30;
128 Target Session Attributes=any;
129 Pooling=true;
130 Maximum Pool Size=130;
131 AVATARS_DB_CONNECTION_STRING: >
132 Host=postgres;
133 Port=5432;
134 Database=avatarsdb;
```

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

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

```
1 # testit_unichart/values-override.yaml
2 # ...
3 general:
4 config:
5 # AUTH_CACHE_CONNECTION_STRING: "auth-cache:6379,ssl=true"
6 # INFLUX_CONNECTION_STRING: "https://influxdb:8086"
7 RABBITMQ_SSL_ENABLED: "true"
8 RABBITMQ_DEFAULT_PORT: "5671"
9 # APPLICATION__SECURITY__TRUSTEDCERTIFICATELOCATION:
10 "/etc/ssl/certs"
11 frontend:
12 replicaCount: 3
13 # resources: {}
    # ...
```

yaml

6. Установите Test IT Kubernetes.

7. Задайте необходимые значения переменных для проекта:

```
1 # Убедитесь, что вводите правильные значения переменных sh
2 (testit_unichart/values.yaml или testit_unichart/values-
3 override.yaml)
4 # minio_vars
5 export AWS_CONNECTION_STRING="http://minio:9000"
6 export AWS_ACCESS_KEY="testitAccessKey"
7 export AWS_SECRET_KEY="testitSecretKey"
8 export TESTIT_BUCKET="testit"
9 export AVATARS_BUCKET="avatars"
10
11 # postgres_vars
12 export POSTGRES_USER="postgres"
13 export POSTGRES_PASSWORD="F1rstL0g0N!"
   export POSTGRES_HOST="postgres"
   export POSTGRES_PORT="5432"
```

8. 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` :

```
1 # Задавать эти переменные не обязательно, скрипт может sh
2 подтянуть из уже запущенных внутренних postgres и minio
3 export NAMESPACE=<my-namespace>
4 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:

```
1 Restore process completed successfully! Your Test IT is running text
2 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.
```

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:

```
1 # minio_vars sh
2 export TMS_BUCKET_NAME=testit
3 export AVATARS_AWS_BUCKET_NAME=avatars
4
5 # docker_vars
6 export COMPOSE_FILE_PATH=/absolute/path/to/testit/docker-
7 compose.yml
8 export COMPOSE_PROJECT_PREFIX=prod
   export COMPOSE_NETWORK_NAME=123123_yoonion_network_local
```

2. Выполните резервное копирование **docker-версии** Test IT:

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

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

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

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

```
1 # testit_frontend/values.yaml
2 general:
3 image:
4 repository: "docker.testit.ru/testit" # <--
5 env.TMS_DOCKER_REGISTRY
6 tag: 4.4.0 #<--env.TMS_CONTAINER_VERSION
7 pullPolicy: IfNotPresent # политика скачивания docker-образов
8 (IfNotPresent, Always, Never)
9 config:
10 CWM_ENABLED: "false"
11 CWM_S3_BUCKET_SECRET_KEY: "secretKey"
12 WIKI_ENABLED: "false"
13 WIKI_S3_BUCKET_SECRET_KEY: ""
14
15 # testit_backend/values.yaml
16 general:
17 image:
18 repository: "docker.testit.ru/testit" # <--
19 env.TMS_DOCKER_REGISTRY
20 tag: 4.4.0 #<--env.TMS_CONTAINER_VERSION
21 pullPolicy: IfNotPresent # политика скачивания docker-образов
22 (IfNotPresent, Always, Never)
23 config:
24 AUTH_CACHE_CONNECTION_STRING: "auth-cache" # должно
25 соответствовать .authCache.name при стандартном запуске
26 FRONTEND_URL: "http://localhost"
27 RABBITMQ_DEFAULT_USER: "testit"
28 RABBITMQ_DEFAULT_PASS: "password"
29 RABBITMQ_DEFAULT_VHOST: "testitrabbit"
30 RABBITMQ_DEFAULT_HOST: "rabbitmq"
31 RABBITMQ_DEFAULT_PORT: "5672"
32 RABBITMQ_AUTH_MODE: "plain"
33 RABBITMQ_CLIENT_CERT_PATH:
34 "/etc/rabbitmq/ssl/client/testit.pfx"
35 RABBITMQ_CLIENT_CERT_PASSPHRASE:
36 RABBITMQ_SSL_ENABLED: "false"
37 USE_PKCE: "true"
38 INSECURE_REMOTES: ""
39 SYNC_RESULT_LINKS_EVERY_SEC: "120"
40 AWS_CONNECTION_STRING: "http://minio:9000" # должно
41 соответствовать .minio.name при стандартном запуске
42 AWS_ACCESS_KEY: "testitAccessKey"
43 AWS_SECRET_KEY: "testitSecretKey"
44 INFLUX_CONNECTION_STRING: "http://influxdb:8086" # должно
```

```
45 | соответствовать .influxdb.name при стандартном запуске
46 | INFLUX_AUTH_ENABLED: "false"
47 | INFLUX_USERNAME: "influx"
48 | INFLUX_PASSWORD: "influxpass"
49 | TEST_RESULT_LINK_REQUEST_LIFETIME_SEC: "600"
50 | DATABASE_TIMEOUT_SEC: "600"
51 | THREAD_PER_PROCESSOR: "10"
52 | TMS_BUCKET_NAME: "testit"
53 | AVATARS_AWS_BUCKET_NAME: "avatars"
54 | SMTP_ENABLE: "false"
55 | SMTP_FROM: ""
56 | SMTP_HOST: ""
57 | SMTP_PORT: ""
58 | SMTP_LOGIN: ""
59 | SMTP_DOMAIN: ""
60 | SMTP_PASSWORD: ""
61 | SMTP_CONNECTION: ""
62 | SMTP_AUTHENTICATION: ""
63 | SMTP_SKIP_CERTIFICATE: "false"
64 | SMTP_TLS_VERSION: "tls"
65 | SMTP_LOG_ENABLE: "false"
66 | PERF_MODULE_ENABLED: "false"
67 | POSTGRES_USER: "postgres" # данные для подключения к PostgreSQL
68 | и названия баз данных
69 | POSTGRES_PASSWORD: "F1rstL0g0N!"
70 | 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"
```

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

```
1 # testit_backend/values-override.yaml
2 # ...
3 general:
4 config:
5 # ...
6 # SMTP_LOG_ENABLE:
7 # PERF_MODULE_ENABLED:
8 POSTGRES_USER: "testituser"
9 POSTGRES_PASSWORD: "8wkj9l!0asdk"
10 # POSTGRES_DB:
11 # POSTGRES_AUTH_DB:
12 # POSTGRES_AVATARS_DB:
13 # POSTGRES_BACKGROUND_DB:
14 # POSTGRES_LICENSE_DB:
15 # ...
16 # ...
17
18 # testit_frontend/values-override.yaml
19 # ...
20 frontend:
21 replicaCount: 3
22 # resources: {}
23 # ...
```

6. Установите Test IT Kubernetes.

7. Задайте необходимые значения переменных для проекта:

```
1 # Убедитесь, что вводите правильные значения переменных
2 (testit_backend/values.yaml или testit_backend/values-
3 override.yaml)
4 # minio_vars
5 export AWS_CONNECTION_STRING="http://minio:9000"
6 export AWS_ACCESS_KEY="testitAccessKey"
7 export AWS_SECRET_KEY="testitSecretKey"
8 export TESTIT_BUCKET="testit"
9 export AVATARS_BUCKET="avatars"
10
11 # postgres_vars
12 export POSTGRES_USER="postgres"
13 export POSTGRES_PASSWORD="F1rstL0g0N!"
   export POSTGRES_HOST="postgres"
   export POSTGRES_PORT="5432"
```

8. 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` :

```
1 # Задавать эти переменные не обязательно, скрипт может sh
2 подтянуть из уже запущенных внутренних postgres и minio
3 export NAMESPACE=<my-namespace>
4 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:

```
1 Restore process completed successfully! Your Test IT is running text
2 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: 08.04.2025, 12:42:56

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.

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:

```
1 # testit_backend/values-override.yaml
2 license:
3   DiskSpaceCheckInterval: "01:00:00"
4   DiskSpacePercentageTheshold: 10
```

.yaml

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.

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:

```
1  chmod +x scripts/backup.sh
2  scripts/backup.sh docker-compose.yml testit
3  # Или
4  scripts/backup.sh docker-compose.yml testit --notar
5  # 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
```

sh

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:

```
1  chmod +x scripts/restore.sh                                     sh
2  scripts/restore.sh docker-compose.yml testit
3  backup_21_05_2019.tar
4  # Или
5  scripts/restore.sh docker-compose.yml testit backup_21_05_2019
   --notar
   # Запуск с флагом '--notar' работает 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/`:

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

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

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

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

```
1 # Задайте переменные для доступа на PostgreSQL и Minio: sh
2 # Значения для переменных располагаются в
3 testit_unichart/values.yaml:general:config или
4 testit_unichart/values-override.yaml:general:config
5 export POSTGRES_HOST="postgres"
6 export POSTGRES_PORT="5432"
7 export POSTGRES_USER="postgres"
8 export POSTGRES_PASSWORD="F1rstL0g0N!"
9 export AWS_CONNECTION_STRING="http://minio:9000"
10 export AWS_ACCESS_KEY="testitAccessKey"
11 export AWS_SECRET_KEY="testitSecretKey"
12 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:

```
1 backups/ sh
2 {день}_{месяц}_{год}/
3 {сервис1}_backup.tar
4 {сервис2}_backup.tar
5 ...
6 # Например:
7 backups/
8 18_07_2023/
9 postgres_backup.tar
10 minio_backup.tar
11 ...
```

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:

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

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

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

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

```
1 # Задайте переменные для доступа на PostgreSQL и Minio:
2 # Значения для переменных располагаются в
3 testit_unichart/values.yaml:general:config или
4 testit_unichart/values-override.yaml:general:config
5 export POSTGRES_HOST="postgres"
6 export POSTGRES_PORT="5432"
7 export POSTGRES_USER="postgres"
8 export POSTGRES_PASSWORD="F1rstL0g0N!"
9 export AWS_CONNECTION_STRING="http://minio:9000"
10 export AWS_ACCESS_KEY="testitAccessKey"
11 export AWS_SECRET_KEY="testitSecretKey"
12 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.

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/`:

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

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

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

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

```

1 # testit_backend/values.yaml
2 postgres:
3 image:
4 name: <...>
5 tag: 14.8-bookworm
6
7 # /jobs/postgres/restore/job.yaml,
8 /jobs/postgres/backup/job.yaml
9 spec:
10 template:
11 spec:
12 containers:
13 - name: <...>
14 image: postgres:14.8-bookworm
15 #####
16 # testit_backend/values.yaml
17 minio:
18 image:
19 name: <...>
    tag: RELEASE.2023-05-04T21-44-30Z

```

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

```

1 # Задайте переменные для доступа на PostgreSQL и Minio:
2 # Значения для переменных располагаются в
3 testit_backend/values.yaml:general:config или
4 testit_backend/values-override.yaml:general:config
5 export POSTGRES_HOST="postgres"
6 export POSTGRES_PORT="5432"
7 export POSTGRES_USER="postgres"
8 export POSTGRES_PASSWORD="F1rstL0g0N!"
9 export AWS_CONNECTION_STRING="http://minio:9000"
10 export AWS_ACCESS_KEY="testitAccessKey"
11 export AWS_SECRET_KEY="testitSecretKey"
12 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:

```
1 backups/ sh
2 {день}_{месяц}_{год}/
3 {сервис1}_backup.tar
4 {сервис2}_backup.tar
5 ...
6 # Например:
7 backups/
8 18_07_2023/
9 postgres_backup.tar
10 minio_backup.tar
11 ...
```

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:

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

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

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

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

```
1 # Задайте переменные для доступа на PostgreSQL и Minio: sh
2 # Значения для переменных располагаются в
3 testit_backend/values.yaml:general:config или
4 testit_backend/values-override.yaml:general:config
5 export POSTGRES_HOST="postgres"
6 export POSTGRES_PORT="5432"
7 export POSTGRES_USER="postgres"
8 export POSTGRES_PASSWORD="F1rstL0g0N!"
9 export AWS_CONNECTION_STRING="http://minio:9000"
10 export AWS_ACCESS_KEY="testitAccessKey"
11 export AWS_SECRET_KEY="testitSecretKey"
12 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.

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:

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

sh

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`:

```
1 // пример содержимого лога пользовательских действий
2 {
3 //...
4 "Properties": {
5 //...
6 "LogType": "UserAction",
7 //...
8 },
9 //...
10 }
```

json

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:

```
1 - filebeat.sh
2 - filebeat.yml
3 - kibana.yml
4 - logstash.sh
5 - logstash.yml
6 - logstash.conf
7 - docker-compose.elk.yml
```

yml

Ниже представлено содержимое файлов.

filebeat.sh

```
1 #!/bin/bash
2
3 cat /tmp/filebeat.yml > /usr/share/filebeat/filebeat.yml
4
5 chown -R root /usr/share/filebeat/
6 chmod -R go-w /usr/share/filebeat
7
8 /usr/local/bin/docker-entrypoint -environment container
```

sh

filebeat.yml

```
1 filebeat.inputs:
2   - type: log
3   paths:
4     - /var/lib/docker/containers/*/*-json.log
5
6   output.logstash:
7     hosts: ["logstash:5044"]
```

yml

kibana.yml

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

yml

logstash.sh

```
1  #! /bin/bash
2
3  generate_put_data()
4  {
5  cat <<EOF
6  {
7  "policy": {
8  "phases": {
9  "hot": {
10 "min_age": "0ms",
11 "actions": {
12 "rollover": {
13 "max_age": "24h"
14 }
15 }
16 },
17 "delete": {
18 "min_age": "${EVENT_LOG_MAX_AGE:-30d}",
19 "actions": {
20 "delete": {}
21 }
22 }
23 }
24 }
25 }
26 EOF
27 }
28
29 curl --retry 8 --location \
30 --request PUT
31 "${ELASTICSEARCH_CONNECTION_STRING}/_ilm/policy/testit_logs_policy?
32 pretty" \
33 --header 'Content-Type: application/json' \
34 --data "$(generate_put_data)"
35
36 cat /tmp/logstash.yml > /usr/share/logstash/config/logstash.yml
37 cat /tmp/logstash.conf > /usr/share/logstash/pipeline/logstash.conf
38
39 /usr/local/bin/docker-entrypoint
```

1

http.host: "0.0.0.0"

yml

logstash.conf

```
1  input {
2  # Accept logs from filebeat, listening on port 5044
3  beats {
4  port => 5044
5  }
6  }
7
8  filter {
9  # Docker-level JSON logs parser
10 json {
11 source => "message"
12 tag_on_failure => ["json_parse_error"] # Error-handling for JSON
13 parsing
14 }
15 # Date format for Kibana
16 date {
17 match => ["time", "ISO8601"]
18 target => "@timestamp"
19 }
20 # Renaming fields for convenience
21 mutate {
22 rename => { "[log]" => "message" }
23 rename => { "[stream]" => "log_stream" }
24 }
25 #####---Optional Block---
26 #####
27 # #
28 # #
29 if [message] =~ /\^\{.*\}$/ { # Check the message field for JSON-
30 compatibility
31 json {
32 source => "message"
33 tag_on_failure => ["nested_json_parse_error"] # Error-handling for
34 JSON parsing
35 }
36 } else {
37 drop { } # Non-JSON logs are dropped (necessary for the filter
38 below)
39 }
40
41 if [Properties][LogType] != "UserAction" {
42 drop { } # Non-"aciton" logs are dropped (optional)
43 }
44 # #
```

```
45  # #
46  #####---Optional Block---
47  #####
48  }
49
50  output {
51  elasticsearch {
52  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

```
1  version: '3.8'
2
3  x-logging: &default-logging
4  driver: "json-file"
5  options:
6  max-size: "500m"
7  max-file: "2"
8
9  networks:
10 testit_monitoring_network:
11 name: "testit_monitoring_stack"
12 external: true
13
14 x-networks: &default-networks
15 - "testit_monitoring_network"
16
17 x-elk-service: &elk-service-defaults
18 logging: *default-logging
19 networks: *default-networks
20 restart: unless-stopped
21
22 services:
23 elasticsearch:
24 image: elasticsearch:8.8.1
25 <<: *elk-service-defaults
26 environment:
27 ES_JAVA_OPTS: "-Xms1024m -Xmx1024m"
28 bootstrap.memory_lock: "true"
29 cluster.name: "elasticsearch"
30 discovery.type: "single-node"
31 network.host: "0.0.0.0"
32 http.port: "9200"
33 xpack.security.enabled: "false"
34 xpack.security.http.ssl.enabled: "false"
35 ulimits:
36 memlock:
37 soft: -1
38 hard: -1
39 healthcheck:
40 test: ["CMD", "curl", "-f",
41 "http://localhost:9200/_cluster/health"]
42 interval: 30s
43 timeout: 10s
44 retries: 5
```

```
45 volumes:
46 - elastic-volume:/usr/share/elasticsearch/data
47
48 logstash:
49 image: logstash:8.8.1
50 <<: *elk-service-defaults
51 environment:
52 PATH_CONFIG: "/usr/share/logstash/pipeline/logstash.conf"
53 ELASTICSEARCH_CONNECTION_STRING: "http://elasticsearch:9200"
54 ELASTICSEARCH_INDEX: "testit"
55 ELASTICSEARCH_LOGS_INDEX: "testit_logs"
56 LS_JAVA_OPTS: "-Xmx512m -Xms512m"
57 EVENT_LOG_MAX_AGE: 30d
58 ELASTICSEARCH_SSL_ENABLED: "false"
59 entrypoint: ["/bin/bash", "-c"]
60 command: ["/tmp/launch.sh"]
61 healthcheck:
62 test: ["CMD", "curl", "-f", "http://localhost:9600/_node"]
63 interval: 30s
64 timeout: 10s
65 retries: 3
66 volumes:
67 - ${PWD}/logstash.conf:/tmp/logstash.conf:ro
68 - ${PWD}/logstash.yml:/tmp/logstash.yml:ro
69 - ${PWD}/logstash.sh:/tmp/launch.sh:ro
70 - /var/lib/docker/containers:/var/lib/docker/containers:ro
71 depends_on:
72 elasticsearch:
73 condition: service_healthy
74
75 kibana:
76 image: "kibana:8.8.1"
77 <<: *elk-service-defaults
78 ports:
79 - ${KIBANA_EXPOSE_PORT:-5601}:5601
80 environment:
81 SERVER_NAME: "localhost"
82 SERVER_HOST: "0.0.0.0"
83 ELASTICSEARCH_URL: "http://elasticsearch:9200"
84 ELASTICSEARCH_HOSTS: "http://elasticsearch:9200"
85 SERVER_SSL_ENABLED: "false"
86 healthcheck:
87 test: ["CMD", "curl", "-f", "http://localhost:5601/api/status"]
88 interval: 30s
89 timeout: 10s
```

```
90   retries: 3
91   volumes:
92   - ${PWD}/kibana.yml:/usr/share/kibana/config/kibana.yml:ro
93   depends_on:
94   elasticsearch:
95     condition: service_healthy
96
97   filebeat:
98     image: elastic/filebeat:8.8.1
99     user: root
100    <<: *elk-service-defaults
101    entrypoint: ["/bin/bash", "-c"]
102    command: ["/tmp/launch.sh"]
103    volumes:
104    - ${PWD}/filebeat.yml:/tmp/filebeat.yml:ro
105    - ${PWD}/filebeat.sh:/tmp/launch.sh:ro
106    - /var/lib/docker/containers:/var/lib/docker/containers:ro
107    healthcheck:
108      test: ["CMD", "filebeat", "test", "output"]
109      interval: 30s
110      timeout: 10s
111      retries: 3
112      depends_on:
113      logstash:
114      condition: service_healthy
115
116
117    volumes:
118    elastic-volume:
```

Команды для развертывания ELK стека:

```
1 # Создаем в локальной директории все файлы из списка выше sh
2
3 # Добавляем возможность запуска для скриптов
4 chmod +x logstash.sh filebeat.sh
5
6 # Создаем сеть docker
7 docker network create testit_monitoring_stack
8
9 # Запускаем стек
10 docker compose -f docker-compose.elk.yml -p testit-monitoring up -
11 d
12
13 # После завершения команды запуска переходите на
14 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

```
1 # Временная остановка проекта с возможностью перезапуска sh
2 контейнеров
3 docker compose -f docker-compose.elk.yml -p testit-monitoring down
4
5 # Полная остановка с удалением данных, содержащихся в томе
6 elastic-volume
7 docker compose -f docker-compose.elk.yml -p testit-monitoring down
8 --volumes
9
10 # Удаление сети docker
11 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:

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

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:

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

- Для создания резервной копии и восстановления из нее следующие команды:

```
1 | scripts/backup.sh "docker-compose.yml -f docker- | sh
2 | 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` :

```
1 | cd ~/testit | sh
2 | cp docker-compose.yml docker-compose.yml.bak # резервная копия
3 | 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`:

```
1  ## internal certificate path
2  SSL_CERTIFICATE=/etc/nginx/ssl/testit.crt
3  SSL_CERTIFICATE_KEY=/etc/nginx/ssl/testit.key
4  #REDIRECT_TO_HTTPS=true
```

text

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:

```
1  services:
2  frontend:
3  image: "${DOCKER_REGISTRY}/frontend:${CONTAINER_VERSION}"
4  ports:
5  - 80:80
6  - 443:443
7  ...
```

text

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:

```
1 certs=$(docker inspect testit_ssl-volume --format '{{
2 .Mountpoint }}')
3
4 chown 101 testit.crt testit.key
5
6 chgrp 0 testit.crt testit.key
7
8 cp -p testit.crt ${certs}/
9
cp -p testit.key ${certs}/
```

text

6. Примените изменения, выполнив команду:

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

Обновлено: 07.02.2024, 17:36:29

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:

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

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

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

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

```
1 | # testit_unichart/values-override.yaml | yml
2 | frontend:
3 |   ingress:
4 |     main:
5 |       ingressClassName: "nginx" # Set ingress class name
6 |       host: "my.hostname.com" # Set the desired hostname you own
7 |       path: /
8 |       pathType: Prefix
9 |       annotations:
10 |         nginx.ingress.kubernetes.io/ssl-redirect: "True"
11 |         nginx.ingress.kubernetes.io/proxy-connect-timeout: "300"
12 |         nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
13 |         nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
14 |         nginx.ingress.kubernetes.io/proxy-body-size: 100m
15 |         nginx.ingress.kubernetes.io/proxy-buffer-size: 50m
16 |         nginx.ingress.kubernetes.io/proxy-buffers-number: "4"
17 |       tls: true
18 |       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:

```
1 | cd <installation_folder> | sh
2 | 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*:

```
1 | # testit_unichart/values-override.yaml | yml
2 | frontend:
3 | ingress:
4 |   main:
5 |     ingressClassName: "nginx" # Set ingress class name
6 |     host: "my.hostname.com" # Set the desired hostname you own
7 |     path: /
8 |     pathType: Prefix
9 |     annotations:
10 |      nginx.ingress.kubernetes.io/ssl-redirect: "True"
11 |      nginx.ingress.kubernetes.io/proxy-connect-timeout: "300"
12 |      nginx.ingress.kubernetes.io/proxy-read-timeout: "3600"
13 |      nginx.ingress.kubernetes.io/proxy-send-timeout: "3600"
14 |      nginx.ingress.kubernetes.io/proxy-body-size: 100m
15 |      nginx.ingress.kubernetes.io/proxy-buffer-size: 50m
16 |      nginx.ingress.kubernetes.io/proxy-buffers-number: "4"
17 |      cert-manager.io/cluster-issuer: letsencrypt-production
18 |      cert-manager.io/common-name: my.hostname.com
19 |     tls: true
20 |     tlsSecretName: my-ssl-secret
```

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:

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

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:

```
1 | #!/bin/bash sh
2 | trusted_certs=$(docker inspect testit_trusted-certificates-volume
3 | --format '{{ .Mountpoint }}')
4 | Cert=$(find . -name "*.crt")
5 | Cert_count=$(find . -name "*.crt" | wc -l)
6 | echo "Найдено '${Cert_count}' файлов формата .crt"
7 | 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 по permissions. 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:

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

Чтобы удалить систему полностью (с потерей данных):

- Выполните команду:

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

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:

```
1 # Зафиксируйте наименование пространства имен. sh
2 export NAMESPACE=<my-namespace>
3 # Проверьте, имеются ли в пространстве имен установленные
4 чарты.
5 helm -n $NAMESPACE list
6
7 # Удалите чарты внешнего интерфейса (frontend) и внутреннего
8 интерфейса (backend).
9 helm -n $NAMESPACE uninstall testit
10
# Опционально: Дождитесь остановки подов.
kubectl -n $NAMESPACE get pods --watch
```