АКЦИОНЕРНОЕ ОБЩЕСТВО
«СИСТЕМНЫЙ ОПЕРАТОР ЕДИНОЙ ЭНЕРГЕТИЧЕСКОЙ СИСТЕМЫ»

ИНСТРУКЦИЯ ПО УСТАНОВКЕ И НАСТРОЙКЕ

программного обеспечения формирования экспресс-протокола о достоверности и качестве данных системы мониторинга переходных режимов

Москва, 2022

СОДЕРЖАНИЕ

Перечень сокращений

Таблица 1. Перечень сокращений

Сокращение	Описание или расшифровка
API	Интерфейс программирования приложений (англ. Application programming interface) - набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах
PGP	preety good privacy, инструмент для шифрования
АС СИ СМПР	Автоматическая система сбора информации от регистраторов системы мониторинга переходных режимов
БД	База данных
ПО МФУК	Программное обеспечение мониторинга функционирования устройств и комплексов системы мониторинга переходных режимов
ПО ФЭП	Программное обеспечение формирования экспресс-протокола о достоверности и качестве данных системы мониторинга переходных режимов
НСИ	Нормативно-справочная информация
ОС	Операционная система
ПО	Программное обеспечение

1 Состав ФЭП для установки

Программное обеспечение ФЭП состоит из следующих элементов:

- системное ПО;
- Docker-образы компонентов;
- конфигурации.

1.1 Системное ПО

Системное ПО обеспечивает среду запуска приложения и состоит из следующих элементов:

- ОС Astra Linux;
- среда выполнения приложения Docker с возможностью запуска контейнеров приложений в виртуальной среде;
- СУБД Postgres Pro Ent.

1.2 Docker-образы компонентов

Docker-образы компонентов представляют собой основу для запуска виртуальной среды конкретного компонента в виде контейнера. Состав образов в рамках ФЭП следующий:

- fep-service - центральный серверный компонент обработки и хранения параметров качества данных, а также предоставляет REST API для пользовательского интерфейса;
- fep-ui - пользовательский интерфейс приложения;
- map-server – компонент, предоставляющий карту для отображения в интерфейсе на форме мониторинга.

1.3 Конфигурации

Конфигурации описывают связи между компонентами и необходимые им ресурсы. Поставляются в виде отдельного архива и подлежат редактированию при установке.

Состав файлов при установке без кластера (в docker) следующий:

- fep-service.config - конфигурационный файл настроек серверной части приложения;
- env-specific.json - конфигурационный файл веб-приложения;

– nginx.conf - конфигурационный файл nginx веб-приложения для настройки интеграции серверной части и веб-приложения, а также ssl-сертификатов.

2 Системные требования

Для функционирования системы требуются сервера (виртуальные или физические) в следующем составе:

1. Сервер приложений:

- 4х ядерный CPU с архитектурой x86-64;
- 16 Гб RAM;
- 500 Гб дискового пространства (один раздел).

2. Сервер БД:

- 4х ядерный CPU с архитектурой x86-64;
- 16 Гб RAM;
- 500 Гб дискового пространства (два раздела: 100 системный, 400 для БД (по умолчанию монтируется в /mnt/datastorage/fep-share-storage)).

3 Установка системного ПО

Необходимо установить следующее системное ПО, необходимое для функционирования системы:

– Astra Linux "Орел" версии 2.12 и выше;

– Docker версии не ниже 19.03.

– Postgres Pro Ent.

3.1 Установка Astra Linux

Установка описана в официальной документации, расположенной на сайте производителя - https://astralinux.ru/assets/docs/AstraLinuxCE_install_2-12.pdf.

3.2 Установка Docker

Установка docker для операционной системы Astra Linux ничем не отличается от установки на ОС Debian, описанной по адресу https://docs.docker.com/engine/install/debian/.

Для этого надо установить (если не установлено) ПО для интеграции с https-репозиториями для пакетного менеджера apt (все последующие команды - команды командной строки bash):

```
sudo apt-get update
```

```
 sudo apt-get install -y \
     apt-transport-https \
     ca-certificates \
     curl \
     gnupg-agent \
     software-properties-common
```

Установка docker на Astra Linux:

```
sudo apt install docker.io
```

Проверка корректной установки docker осуществляется запуском тестового контейнера:

```
sudo docker run hello-world
```

После его запуска будет отображено сообщение «hello world» в командной строке и контейнер завершит свою работу. Это значит, что docker установлен успешно.

Создание группы пользователей docker (если группа уже создана, то перейти к следующему шагу):

```
sudo groupadd docker
```

Добавить своего пользователя в группу docker:

```
sudo usermod -aG docker $USER
```

3.3 Добавление прокси для службы docker

Создайте каталог systemd для службы docker:

```
sudo mkdir -p /etc/systemd/system/docker.service.d
```

Создайте файл с именем /etc/systemd/system/docker.service.d/http-proxy.conf, который добавляет HTTP_PROXY переменную среды:

```
[Service]
Environment="HTTP_PROXY=http://proxy.example.com:80"
```

Если вы находитесь за прокси-сервером HTTPS, установите HTTPS_PROXY переменную среды:

```
[Service]
Environment="HTTPS_PROXY=https://proxy.example.com:443"
```

Можно установить несколько переменных среды; например, как HTTPS, так и HTTP-прокси;

```
[Service]
Environment="HTTP_PROXY=http://proxy.example.com:80"
Environment="HTTPS_PROXY=https://proxy.example.com:443"
```

Если у вас есть внутренние реестры Docker, с которыми вам нужно связаться без прокси, вы можете указать их через NO_PROXY переменную среды.

Переменная NO_PROXY указывает строку, содержащую разделенные запятыми значения для хостов, которые следует исключить из проксирования. Вот параметры, которые вы можете указать для исключения хостов:

1. Префикс IP-адреса (1.2.3.4).

2. Доменное имя или специальная метка DNS (*).

3. Доменное имя соответствует этому имени и всем поддоменам. Доменное имя с ведущим «.» соответствует только субдоменам. Например, учитывая домены foo.example.com и example.com:

   - example.com соответствует example.com и foo.example.com, и
   - .example.com соответствует только foo.example.com

4. Одна звездочка (*) указывает на то, что проксирование не должно выполняться.

5. Буквенные номера портов принимаются префиксами IP-адресов (1.2.3.4:80) и именами доменов (foo.example.com:80).

Пример конфигурации:

```
[Service]Environment="HTTP_PROXY=http://proxy.example.com:80"
Environment="HTTPS_PROXY=https://proxy.example.com:443"
Environment="NO_PROXY=localhost,127.0.0.1,docker-registry.example.com,.corp"
```

Сбросьте изменения и перезапустите Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Убедитесь, что конфигурация загружена и соответствует внесенным вами изменениям, например:

```
sudo systemctl show --property=Environment docker


Пример вывода:
Environment=HTTP_PROXY=http://proxy.example.com:80
HTTPS_PROXY=https://proxy.example.com:443 NO_PROXY=localhost,127.0.0.1,docker-
registry.example.com,.corp
```

4 Установка образов в docker

Далее необходимо загрузить образы приложения в репозиторий Docker:

```
sudo docker load -i fep-service.tar
sudo docker load -i fep-ui.tar
sudo docker load -i map-server.tar
```

5 Настройка СУБД

В рамках данной установки на второй узел устанавливается только СУБД Postgres Pro Ent, а на первый - приложение ФЭП, работающее в среде docker.

После установки СУБД необходимо выполнить следующие команды:

```
sudo mkdir -p /mnt/datastorage/fep-share-storage/postgres


docker run -d --restart unless-stopped  \
--name fep-postgres  \
--env POSTGRES_DB="postgres" \
--env POSTGRES_USER="postgres" \
--env POSTGRES_PASSWORD="postgres" \
--mount type=bind,source=/mnt/datastorage/fep-share-
storage/postgres,target=/var/lib/postgresql/data \
--publish 5432:5432 \
postgres:12.4
```

Для запуска загруженных образов приложений предварительно необходимо создать сеть внутри docker, через которую будут общаться компоненты приложения между собой и с внешней по отношению к docker сетью. Сеть с наименованием fep-network (далее будет использоваться в привязке к запускаемым контейнерам) создается следующей командой:

```
docker network create fep-network
```

Далее создаем контейнер для сервиса расчетов ФЭП с запуском в режиме daemon (-d) и опцией автозапуска (--restart unless-stopped), привязкой к fep-network, dns-именем fep-service внутри fep-network, а также монтируем конфигурацию приложения внутрь контейнера из файловой системы сервера по пути /mnt/datastorage/fep-share-storage/fep-service.config.

Перед созданием необходимо установить нужную timezone: Для этого необходимо изменить путь до файла нужной timezone на хосте (/usr/share/zoneinfo/**Europe/Moscow**) в аргументе

--mount type=bind,source=/usr/share/zoneinfo/**Europe/Moscow**,target=/etc/localtime \

```
sudo mkdir -p /mnt/datastorage/fep-share-storage/logs/fep-service


docker run -d --restart unless-stopped  \
--net fep-network \
--name fep-service  \
--env JAVA_OPTS="-Xmx2048m -XX:+UseG1GC -XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/mnt" \
--mount type=bind,source=/mnt/datastorage/fep-share-storage/fep-
service.config,target=/etc/opt/ch/application.properties,readonly \
--mount type=bind,source=/mnt/datastorage/fep-share-
storage/kerberos.keytab,target=/etc/opt/kerberos.keytab,readonly \
```

```
--mount type=bind,source=/mnt/datastorage/fep-share-storage/logs/fep-
service,target=/usr/local/tomcat/logs \
--mount type=bind,source=/usr/share/zoneinfo/Europe/Moscow,target=/etc/localtime
\
--publish 31005:8080 \
ap/fep-service:0.1
```

Создаем контейнер веб-сервера с пользовательским интерфейсом:

```
docker run -d --restart unless-stopped  \
--net fep-network \
--name fep-ui \
--mount type=bind,source=/mnt/datastorage/fep-share-storage/env-
specific.json,target=/usr/share/nginx/html/assets/env-specific.json,readonly \
--mount type=bind,source=/mnt/datastorage/fep-share-
storage/nginx.conf,target=/etc/nginx/nginx.conf,readonly \
--mount type=bind,source=/mnt/datastorage/fep-share-
storage/tls.key,target=/usr/share/nginx/ssl/cert.key,readonly \
--mount type=bind,source=/mnt/datastorage/fep-share-
storage/tls.crt,target=/usr/share/nginx/ssl/cert.pem,readonly \
--publish 31006:80 \
--publish 443:443 \
ap/fep-ui:0.1
```

Где привязываем также путь до ssl-сертификата с приватным ключом (/mnt/datastorage/fep-share-storage/tls.crt и mnt/datastorage/fep-share-storage/tls.key) и конфигурацию nginx, сконфигурированную под использование этих ключей, а также создающую reverse proxy на сервис, находящийся на порту 31005, через единый https-порт.

В файле nginx.conf необходимо задать dns-имя для веб-сервера - присвоить ключу server_name значение dns-имени.

Генерация kerberos.keytab и ssl-ключей описана в соответствующих подразделах, после чего они должны быть скопированы в директорию /mnt/datastorage/fep-share-storage сервера.

Аналогично создаем контейнер мап-сервера с сервисом карты:

```
docker run -d --restart unless-stopped  \
--net fep-network \
--name map-server \
--publish 31010:8080 \
ap/map-server:1.0
```

Описание всех параметров fep-service.config и env-specific.json смотрите в разделе Конфигурационные файлы приложения. После настройки всех параметров необходимо перезапустить контейнеры (проще всего - sudo systemctl restart docker.service).

6 Конфигурационные файлы приложения

Все файлы конфигураций (а также файлы с ключами для ssl и kerberos) самого приложения должны храниться на сервере в директории /mnt/datastorage/fep-share-storage.

В ФЭП в части настроек приложения присутствуют следующие файлы:

- fep-service.config - конфигурационный файл настроек серверной части приложения;
- env-specific.json - конфигурационный файл веб-приложения;
- kerberos.keytab - перманентный kerberos-токен для сервиса с SPN для веб-сервера;
- tls.key/tls.crt - закрытый и открытый ключи ssl-сертификата.
- nginx.conf - конфигурационный файл nginx

6.1 env-specific.json

В файле env-specific.json требуется указать следующие настройки:

(конструкции в виде "{{ FEP_SERVER_DNS_NAME }}" являются переменными и подлежат замене (вместе со скобками {{}}))

```
    {
 "backendhost": "https://{{ FEP_SERVER_DNS_NAME }}/backend",
 "mapurl": "https://{{ FEP_SERVER_DNS_NAME }}/map/map-
server/images/osm_tiles/{z}/{x}/{y}.png",
 "oauth2": {
    "endpoint": "https://{{ FEP_SERVER_DNS_NAME }}/backend/oauth",
    "clientId": "fepjwtclientid",
    "clientSecret": "fepjwtclientkey",
    "redirectUri": "https://{{ FEP_SERVER_DNS_NAME }}/monitoring",
    "authType": "ldap"
 }
}
```

Где:

"backendhost" - внешний адрес к backend REST API. Необходимо явное указание внешнего доступа, т.к. он в последующем отправляется пользователю как ресурс клиентского приложения.

"mapurl" - адрес сервера карт. Используется для отображения географической карты на форме мониторинга.

"oauth2:endpoint" - адрес к API выдачи токенов для аутентификации.

"oauth2:redirectUri" - адрес куда будет перенаправлен пользователь после аутентификации.

"oauth2:authType" - тип аутентификации.

6.2 fep-service.config

В файле fep-service.config задаются следующие настройки доступа к АС СИ СМПР и МФУК, почте, а также авторизации:

- (конструкции в виде "{{ SMPR_URL }}" являются переменными и подлежат замене (вместе со скобками {{}}));
- (конструкции в виде "{{ SMPR_BACKEND_PORT | default(4040) }}" являются переменными и подлежат замене (вместе со скобками {{}}) на значение внутри конструкци "default()");
- (конструкции в виде "(IF MFUK = ...)" являются переменными и подлежат замене (вместе со скобками ()) на одно из IF на значение после "=").

Если в config присутствуют русские символы, то необходимо воспользоваться инструментом **"native2ascii"** для перевевода символов в ascii, например: строка CN=username, OU=*Организация*, будет CN=username, OU=*\u041e\u0440\u0433\u0430\u043d\u0438\u0437\u0430\u0446\u0438\u044f*.

Пароль для {{ FEP_SERVICE_USER_PASSWORD }} не должен содержать спец. символы **"!*'();:@&=+$,/?#[]"**.

Файл конфигурации чувствителен к пробелам в конце строки.

```
data-quality-configuration.base-url=http://{{ SMPR_SERVER_HOST }}:{{
SMPR_BACKEND_PORT | default(4040) }}/backend
data-quality-configuration.default-admin-user={{ FEP_SERVICE_USER_USERNAME }}
data-quality-configuration.jwt.redirect-uri=https://{{ FEP_APP_SERVER_HOST
}}/monitoring
data-quality-configuration.kerberos.service-principal-name=HTTP/{{
FEP_APP_SERVER_HOST }}@{{ FEP_APP_SERVER_DOMAIN }}
data-quality-configuration.ldap.base={{ LDAP_BASE_PATH }}
data-quality-configuration.ldap.domain={{ LDAP_DOMAIN }}
data-quality-configuration.ldap.password={{ FEP_SERVICE_USER_PASSWORD }}
data-quality-configuration.ldap.url=ldap://{{ LDAP_SERVER_HOST }}:{{
LDAP_SERVER_PORT | default(3268) }}
data-quality-configuration.ldap.userDn={{ LDAP_USER_DN_FULL_PATH }}
data-quality-configuration.mail.sender={{ EMAIL_FOR_SENDLER }}
data-quality-configuration.username={{ SMPR_USERNAME }}
data-quality-configuration.password={{ SMPR_PASSWORD }}
data-source.url=jdbc:postgresql://{{ FEP_DB_SERVER_HOST }}/postgres
rest-topology-provider.auth.token-endpoint=(IF MFUK   = https://{{
MFUK_SERVER_HOST }}/backend/oauth/ldap/token )
#                           (IF FEP      = https://{{ GRAND_FEP_APP_SERVER_HOST
}}/backend/oauth/ldap/token )
rest-topology-provider.base-url=(IF MFUK   = https://{{ MFUK_SERVER_HOST
}}/backend )
#                   (IF FEP    = https://{{ GRAND_FEP_APP_SERVER_HOST
}}/backend/feap )
spring.mail.host={{ SMTP_HOST }}
# Примечание: Для {{ SMTP_HOST }} допускается только dns-имя
spring.mail.port={{ SMTP_PORT | default(25) }}
spring.mail.username={{ SMTP_USER }}
```

```
spring.mail.password={{ SMTP_USER_PASSWORD }}
spring.mail.properties.mail.smtp.auth.mechanisms={{ SMTP_AUTH_MECHANISMS |
default(NTLM) }}
spring.mail.properties.mail.smtp.auth.ntlm.domain={{
SMTP_DOMAIN_FOR_NTLM_AUTH_MECHANISM }}
thrift-telemetry-value-provider.host={{ SMPR_SERVER_HOST }}

#--- НЕ редактируемые  ---#
data-quality-configuration.jwt.backend-client-id=fepbackendclientid
data-quality-configuration.jwt.backend-client-key={noop}
data-quality-configuration.kerberos.keytab-location=/etc/opt/kerberos.keytab
data-source.driverClassName=org.postgresql.Driver
data-source.password=postgres
data-source.username=postgres
rest-topology-provider.fetch-topology-uri=/topology/all
rest-topology-provider.update-topology-uri=/config
rest-topology-provider.auth.client-id=fepbackendclientid
rest-topology-provider.auth.secret-id=fepbackendclientkey
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.profiles.default=secure
thrift-telemetry-value-provider.port=7777
telemetry-value-cache.storage-duration-minutes=120
data-quality-configuration.snmp.trap-address={{ SERVER_HOST }}
data-quality-configuration.snmp.trap-port= {{ SERVER_PORT }}
data-quality-configuration.snmp.auth.v2.community-name=public
data-quality-configuration.snmp.auth.v2.security-name=SOCDUro
spring.mail.properties.mail.smtp.connectiontimeout=15000
spring.mail.properties.mail.smtp.timeout=15000
spring.mail.properties.mail.smtp.writetimeout=15000
#--- НЕ редактируемые  ---#

#--- Откладка  ---#
#logging.level.root=DEBUG
```

Где:

- data-quality-configuration.base-url - базовый url REST API backend АС СИ СМПР для доступа к настройкам адаптера качества данных;
- data-quality-configuration.username - имя пользователя;
- data-quality-configuration.password – пароль;
- rest-topology-provider.base-url - базовый url REST API для доступа к НСИ;
- rest-topology-provider.auth.token-endpoint - базовый url REST API для получения токена аутентификаций;
- rest-topology-provider.auth.client-id - имя пользователя (если нет прямого доступа до ПО МФУК - учётные данные клиента (имя) приложения вышестоящего ПО ФЭП);
- rest-topology-provider.auth.secret-id – пароль (если нет прямого доступа до ПО МФУК - учётные данные клиента (пароль) приложения вышестоящего ПО ФЭП);
- thrift-telemetry-value-provider.host - адрес thrift api АС СИ СМПР;
- thrift-telemetry-value-provider.port - порт thrift api АС СИ СМПР;
- thrift-telemetry-value-provider.username - имя пользователя;
- thrift-telemetry-value-provider.password – пароль;

- data-quality-configuration.jwt.backend-client-id=fepbackendclientid - учётные данные клиента - имя;
- data-quality-configuration.jwt.backend-client-key={noop}fepbackendclientkey - учётные данные клиента - пароль;
- data-quality-configuration.ldap.userDn=polygon_pdc - Username преднастроенного администратора в системе для проверки присутствия пользователей в AD. Соответствует sAMAccountName из AD;
- data-quality-configuration.ldap.password=123123 - Пароль преднастроенного администратора
- data-quality-configuration.default-admin-user=polygon - Username пользователя для логина в приложении. Должен быть в AD. Может быть таким же как и CN в ...ldap.userDn
- data-quality-configuration.ldap.domain=example.com - Домен AD
- data-quality-configuration.ldap.base=DC=example,DC=com - Путь для поиска пользователей
- data-quality-configuration.ldap.searchFilter=(sAMAccountName={0}) - фильтр поиска (необязательное)
- data-quality-configuration.kerberos.service-principal-name - наименование SPN для сервиса (пользователя, под токеном которого будет работать сервис) в kerberos в нотации "HTTP/<dns-имя сервера>@<домен AD>;
- data-quality-configuration.jwt.redirect-uri - адрес перенаправления после автоматической авторизации по kerberos вида https://<dns-имя сервера>/monitoring;
- data-quality-configuration.mail.sender - почтовый ящик отправителя рассылаемых отчетов;
- spring.mail.host - хост почтового сервера;
- spring.mail.port - порт почтового сервера;
- spring.mail.username - пользователь, под которым происходит авторизация и отправка почты;
- spring.mail.password - пароль пользователя;
- spring.mail.properties.mail.smtp.auth.mechanisms - используемые механизмы авторизации. NTLM - основной;
- spring.mail.properties.mail.smtp.auth.ntlm.domain - NTLM-домен.

6.3 kerberos.keytab

Для аутентификации клиентов посредством схемы Negotiate для HTTP с поддержкой Kerberos необходимо выпустить ketyab файл.

Конструкции в виде "{{ SERVER_DNS_NAME }}" являются переменными и подлежат замене (вместе со скобками {{}}):

- если имя переменной указано как CamelCase - значение регистрозависимое;
- если имя переменной указано как UPPERCASE - значение должно быть в верхнем регистре;
- если имя переменной указано как lower_case - значение должно быть в нижнем регистре.

Открыть cmd от имени администратора AD. Сгенерировать kerberos.keytab файл для заданного spn:

```
ktpass -princ HTTP/{{ fep_server_dns_name_with_domain }}@{{ DOMAIN }} -pass {{
password }} -mapuser {{ domain }}\{{ Username }} -crypto ALL -ptype
KRB5_NT_PRINCIPAL -out kerberos.keytab
```

Полученный keytab-файл нужно перенести на сервер приложений для подключения к configuration-manager по следующему пути:

```
/mnt/datastorage/fep-share-storage/kerberos.keytab
```

Изменить следующие настройки в конфигурационном файле fep-service.config:

```
data-quality-configuration.kerberos.service-principal-name=HTTP/{{
fep_server_dns_name_with_domain            }}@{{            DOMAIN            }}
data-quality-configuration.kerberos.keytab-location=/etc/opt/kerberos.keytab
data-quality-configuration.jwt.redirect-uri=https://{{
fep_server_dns_name_with_domain }}/monitoring
```

Где:

- data-quality-configuration.kerberos.service-principal-name - назначенный SPN. Обычно имеет вид HTTP/{{ fep_server_dns_name_with_domain }}@{{ DOMAIN }};
- data-quality-configuration.kerberos.keytab-location - полный путь до .keytab файла внутри контейнера (/etc/opt/kerberos.keytab);
- data-quality-configuration.jwt.redirect-uri - адрес на который вернёт backend после успешной аутентификации.

Изменить следующие настройки (не стирая предыдущие для oauth2) в конфигурационном файле env-specific.json:

```
  "oauth2": {
     ...
     "redirectUri": "https://{{ fep_server_dns_name_with_domain}}/monitoring",
     "authType": "kerberos"}
```

Где:

- "oauth2:authType" - тип аутентификации;
- "oauth2:redirectUri" - адрес, на который вернёт backend после успешной аутентификации.

Далее необходимо перезагрузить контейнеры fep-service и fep-ui.

6.4 nginx.conf

В файле nginx.conf конструкции в виде "{{ SERVER_DNS_NAME }}" являются переменными и подлежат замене.

```
#user  nobody;
worker_processes  1;

#error_log  /dev/stderr;
#error_log  /dev/stderr  notice;
#error_log  /dev/stderr  info;
#error_log  /dev/stderr  debug;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  /dev/stdout  main;
    access_log  off;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  300;

    gzip  on;
    gzip_types      text/plain application/xml application/json
application/javascript;

    server {
       listen       80;
       server_name {{ SERVER_DNS_NAME }};
       return 301 https://$server_name$request_uri;
    }

    server {
        listen       443 ssl;
        server_name  {{ SERVER_DNS_NAME }};
        index index.html;
        root /usr/share/nginx/html;

        ssl_certificate      /usr/share/nginx/ssl/cert.pem;
```

```
        ssl_certificate_key  /usr/share/nginx/ssl/cert.key;

        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout  5m;

        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        client_max_body_size 100m;

        location / {
            try_files $uri /index.html;
        }

        proxy_read_timeout 300;

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /usr/share/nginx/html;
        }

        location /backend/ {
            proxy_pass http://fep-service:8080/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

    location /map/ {
            proxy_pass http://map-server:8080/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

    }

}
```

6.5 tls.key и tls.crt

Для работы протокола https веб-сервер должен быть настроен на использование приватного и публичного ssl-ключей.

Для этого предварительно необходимо сгенерировать ssl-сертификат.

Если сертификат сгенерирован в формате .pfx необходимо получить из сертификата файлы tls.key (приватный ключ) и tls.crt (открытый сертификат) через openssl, для этого необходимо выполнить следующие команды:

```
openssl pkcs12 -in SERVER_DNS_NAME.pfx -nocerts -nodes -out tls.key
openssl pkcs12 -in tls.pfx -clcerts -nokeys -out tls.crt
```

Данные файлы следует поместить в директорию mnt/datastorage/fep-share-storage сервера.