ДОКУМЕНТАЦИЯ «ФАКТОРА» 22.1 «VOYAGER»

Дата: 21.01.2022

СОДЕРЖАНИЕ

УСТАНОВКА И ОБНОВЛЕНИЕ В DOCKER

- Для сотрудников HFLabs
- Требования к установке «Фактор» и «Подсказок» в Docker
- Руководство по развертыванию docker-образов
- Обновление «Фактора» и «Подсказок» в Docker

РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ DOCKER-ОБРАЗОВ

ПОДГОТОВКА К УСТАНОВКЕ DOCKER-ОБРАЗОВ

Docker Engine

Проверьте, что установлен и запущен Docker Engine версии 19+

Проверить работоспособность можно командой:

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

Docker Compose

Проверьте, что установлен Docker Compose версии не ниже 1.26:

```
docker-compose --version
```

Сделайте symlink:

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Пользователь

> Если на сервер уже созданы группа и пользователь, под которыми работают приложения HFLabs, этот шаг можно пропустить.

Создайте пользователя и группу `factor/factor`:

```
useradd factor
```

НАСТРОЙКА КОНФИГУРАЦИИ И ЗАПУСК

Выделение дискового пространства

> Если в /opt есть 150+ Гб свободного пространства, переходите к пункту "Настройка конфигурации Docker".
>
> Для любого варианта в корне ("/") должно быть минимум 15 Гб свободного места.

Если на сервере нет раздела с достаточным количеством свободного места, то увеличьте размер раздела /opt.

Если на сервере есть раздел с достаточным количеством свободного пространства, следуйте инструкции (/data - пример такого раздела):

1. Создайте директории для Docker образов:

```
mkdir -p /data/hflabs-docker
mkdir -p /opt/hflabs-docker
```

2. Сделайте bind одной директории на другую:

```
echo "/data/hflabs-docker       /opt/hflabs-docker       bind  bind  0  0" >> /etc/fstab
mount -a
```

3. Перенесите директорию с Docker и сделайте symlink для Docker и его образов:

```
mv /var/lib/docker /opt/hflabs-docker/
mv /var/lib/containerd /opt/hflabs-docker/

ln -s /opt/hflabs-docker/docker /var/lib/docker
ln -s /opt/hflabs-docker/containerd /var/lib/containerd
```

Настройка конфигурации Docker

1. Загрузите архив конфигурационных настроек и распакуйте его в корень:

```
cd /opt
wget https://fs.hflabs.ru/<customer>/Factor/<PathToBuild>/hflabs-docker-compose-cfg.zip
unzip -o hflabs-docker-compose-cfg.zip -d /
```

2. Примените настройки для ОС:

```
sysctl --system
```

3. Произведите регистрацию сервисов:

```
systemctl daemon-reload
systemctl enable -f docker.service
```

УСТАНОВКА МОДУЛЯ «ПОДСКАЗКИ» В DOCKER

1. Настройте автозапуск службы «Подсказок»:

```
systemctl enable -f suggestions.service
```

2. Узнайте номер uid пользователя factor:

```
id -u factor
```

3. Запишите в `/opt/hflabs-docker/hflabs-compose.env` номер uid из шага 2 для «Подсказок» в SGT_UID:

```
SGT_UID=1000
```

4. Скачайте образ в директорию `/opt/hflabs-docker`:

```
cd /opt/hflabs-docker
wget https://fs.hflabs.ru/<customer>/Suggestions/build/<version>/suggestions-<version>-SNAPSHOT-docker.tar.gz
```

5. Создайте директории для справочников и конфигурационных файлов:

```
mkdir -p /opt/hflabs-docker/suggestions/configuration
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/geonames
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address/house
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/ip
```

6. Скачайте и распакуйте справочники:

```
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/nalog/fias/fias/20200602/fias-20200602.zip -P /opt/hflabs-docker/suggestions/dictionaries/address
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/geonames/20200629/geonames-20200629.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/geonames
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/iso3166/20190819/iso3166-20190819.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/house/20200626/house-20200626.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/house
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/ip/20200518/ip-20200518.zip -P /opt/hflabs-docker/suggestions/dictionaries/ip

unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/fias-20200602.zip ADDROB*.DBF HOUSE*.DBF SOCRBASE.DBF -d /opt/hflabs-docker/suggestions/dictionaries/address
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/geonames/geonames-20200629.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/geonames
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/iso3166/iso3166-20190819.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/house/house-20200626.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/house
unzip -o /opt/hflabs-docker/suggestions/dictionaries/ip/ip-20200518.zip -d /opt/hflabs-docker/suggestions/dictionaries/ip
```

7. Скопируйте лицензию (файл вида nnn_licence.sgt, предоставляет техническая поддержка HFLabs) в каталог `/opt/hflabs-docker/suggestions/configuration`
8. Настройте права пользователя на директории:

```
chown -R factor:factor /opt/hflabs-docker/suggestions
chmod -R 775 /opt/hflabs-docker/suggestions/dictionaries
```

9. Загрузите образ в докер:

```
cd /opt/hflabs-docker/
docker load -i suggestions-<version>-SNAPSHOT-docker.tar.gz
```

10. Запустите «Подсказки»:

```
systemctl start suggestions
```

УСТАНОВКА МОДУЛЯ «ФАКТОР» В DOCKER

1. Настройте автозапуск службы «Фактора»:

```
systemctl enable -f factor.service
```

2. Узнайте номер uid пользователя factor:

```
id -u factor
```

3. Запишите в `/opt/hflabs-docker/hflabs-compose.env` номер uid из шага 2 для «Фактора» в `FACTOR_UID`:

```
FACTOR_UID=1000
```

4. Загрузите образ «Фактора» и положите в директорию `/opt/hflabs-docker`:

```
wget https://fs.hflabs.ru/<customer>/Factor/<PathToBuild>/factor-<customer>-<version>-docker.tar.gz -P
/opt/hflabs-docker
```

5. Загрузите образ в докер:

```
cd /opt/hflabs-docker/
docker load -i factor-<customer>-<version>-docker.tar.gz
```

6. Создайте директорию для справочников и положите туда используемые справочники и контрольные суммы для них:

```
mkdir -p /opt/hflabs-docker/factor/dictionaries
```

7. Создайте директорию для настроек и положите туда конфигурационнные файлы вида: *.properties

```
mkdir -p /opt/hflabs-docker/factor/configuration
```

8. Настройте права пользователя на директорию:

```
chown -R factor:factor /opt/hflabs-docker/factor
```

9. Запустите «Фактор»:

```
systemctl start factor
```

ПОДГОТОВКА К УСТАНОВКЕ DOCKER-ОБРАЗОВ

Docker Engine

Проверьте, что установлен и запущен Docker Engine версии 19+

Проверить работоспособность можно командой:

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

Docker Compose

Проверьте, что установлен Docker Compose версии не ниже 1.26:

```
docker-compose --version
```

Сделайте symlink:

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Пользователь

> Если на сервер уже созданы группа и пользователь, под которыми работают приложения HFLabs, этот шаг можно пропустить.

Создайте пользователя и группу `factor/factor`:

```
useradd factor
```

НАСТРОЙКА КОНФИГУРАЦИИ

Выделение дискового пространства

Если в /opt есть 150+ Гб свободного пространства, переходите к пункту "Настройка конфигурации Docker".

Для любого варианта в корне ("/") должно быть минимум 15 Гб свободного места.

Если на сервере нет раздела с достаточным количеством свободного места, то увеличьте размер раздела /opt.

Если на сервере есть раздел с достаточным количеством свободного пространства, следуйте инструкции (/data - пример такого раздела):

1. Создайте директории для Docker образов:

```
mkdir -p /data/hflabs-docker
mkdir -p /opt/hflabs-docker
```

2. Сделайте bind одной директории на другую:

```
echo "/data/hflabs-docker       /opt/hflabs-docker       bind  bind  0  0" >> /etc/fstab
mount -a
```

3. Перенесите директорию с Docker и сделайте symlink для Docker и его образов:

```
mv /var/lib/docker /opt/hflabs-docker/
mv /var/lib/containerd /opt/hflabs-docker/

ln -s /opt/hflabs-docker/docker /var/lib/docker
ln -s /opt/hflabs-docker/containerd /var/lib/containerd
```

Настройка конфигурации Docker

1. Загрузите архив конфигурационных настроек и распакуйте его в корень:

```
cd /opt
wget https://fs.hflabs.ru/<customer>/Factor/<PathToBuild>/hflabs-docker-compose-cfg.zip
unzip -o hflabs-docker-compose-cfg.zip -d /
```

2. Примените настройки для ОС:

```
sysctl --system
```

3. Произведите регистрацию сервисов:

```
systemctl daemon-reload
systemctl enable -f docker.service
```

УСТАНОВКА МОДУЛЯ «ПОДСКАЗКИ» В DOCKER (20.5+)

1. Настройте автозапуск службы «Подсказок»:

```
systemctl enable -f suggestions.service
```

2. Узнайте номер uid пользователя factor:

```
id -u factor
```

3. Запишите в `/opt/hflabs-docker/hflabs-compose.env` номер uid из шага 2 для «Подсказок» в SGT_UID:

```
SGT_UID=1000
```

4. Скачайте образ в директорию `/opt/hflabs-docker`:

```
cd /opt/hflabs-docker
wget https://fs.hflabs.ru/<customer>/Suggestions/build/<version>/suggestions-<version>-SNAPSHOT-docker.tar.gz
```

5. Создайте директории для справочников и конфигурационных файлов:

```
mkdir -p /opt/hflabs-docker/suggestions/configuration
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/geonames
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/address/house
mkdir -p /opt/hflabs-docker/suggestions/dictionaries/ip
```

6. Скачайте и распакуйте справочники:

```
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/nalog/fias/fias/20200602/fias-20200602.zip -P /opt/hflabs-docker/suggestions/dictionaries/address
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/geonames/20200629/geonames-20200629.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/geonames
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/iso3166/20190819/iso3166-20190819.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/house/20200626/house-20200626.zip -P /opt/hflabs-docker/suggestions/dictionaries/address/house
wget http://maven.hflabs.ru/artifactory/ext-releases-public/ru/hflabs/suggestions/ip/20200518/ip-20200518.zip -P /opt/hflabs-docker/suggestions/dictionaries/ip

unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/fias-20200602.zip ADDROB*.DBF HOUSE*.DBF SOCRBASE.DBF -d /opt/hflabs-docker/suggestions/dictionaries/address
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/geonames/geonames-20200629.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/geonames
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/iso3166/iso3166-20190819.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/iso3166
unzip -o /opt/hflabs-docker/suggestions/dictionaries/address/house/house-20200626.zip -d /opt/hflabs-docker/suggestions/dictionaries/address/house
unzip -o /opt/hflabs-docker/suggestions/dictionaries/ip/ip-20200518.zip -d /opt/hflabs-docker/suggestions/dictionaries/ip
```

7. Скопируйте лицензию (файл вида nnn_licence.sgt, предоставляет техническая поддержка HFLabs) в каталог `/opt/hflabs-docker/suggestions/configuration`
8. Настройте права пользователя на директории:

```
chown -R factor:factor /opt/hflabs-docker/suggestions
chmod -R 775 /opt/hflabs-docker/suggestions/dictionaries
```

9. Загрузите образ в докер:

```
cd /opt/hflabs-docker/
docker load -i suggestions-<version>-SNAPSHOT-docker.tar.gz
```

10. Запустите «Подсказки»:

```
systemctl start suggestions
```

УСТАНОВКА МОДУЛЯ «ФАКТОР» В DOCKER (20.05+)

1. Настройте автозапуск службы «Фактора»:

```
systemctl enable -f factor.service
```

2. Узнайте номер uid пользователя factor:

```
id -u factor
```

3. Запишите в `/opt/hflabs-docker/hflabs-compose.env` номер uid из шага 2 для «Фактора» в `FACTOR_UID`:

```
FACTOR_UID=1000
```

4. Загрузите образ «Фактора» и положите в директорию `/opt/hflabs-docker`:

```
wget https://fs.hflabs.ru/<customer>/Factor/<PathToBuild>/factor-<customer>-<version>-docker.tar.gz -P
/opt/hflabs-docker
```

5. Загрузите образ в докер:

```
cd /opt/hflabs-docker/
docker load -i factor-<customer>-<version>-docker.tar.gz
```

6. Создайте директорию для справочников и положите туда используемые справочники и контрольные суммы для них:

```
mkdir -p /opt/hflabs-docker/factor/dictionaries
```

7. Создайте директорию для настроек и положите туда конфигурационнные файлы вида: *.properties

```
mkdir -p /opt/hflabs-docker/factor/configuration
```

8. Настройте права пользователя на директорию:

```
chown -R factor:factor /opt/hflabs-docker/factor
```

9. Запустите «Фактор»:

```
systemctl start factor
```

УСТАНОВКА «ФАКТОРА» НА SYSTEMD (20.02+)

Для разворачивания используются install и cli-скрипты.

- Руководство для HFLabs
- Настройка Linux для активной работы с дисками
- Настройка ОС
- Установка OpenJDK 11
- Установка контейнера приложений и настройка сервиса
- Настройка «Фактора»

НАСТРОЙКА LINUX ДЛЯ АКТИВНОЙ РАБОТЫ С ДИСКАМИ

Отключение времени модификации файлов

Наши приложения часто и многократно пишут и читают файлы, поэтому рекомендуется использовать файловую систему **xfs**.

Если используете **ext3** и **ext4**, нужно отключить дополнительные функции работы с метаданными файлов:

Для этого нужно изменить параметры монтирования диска, добавив следующие опции:

- **noatime** — полностью отключает запись времени доступа к файлу. Большинство программ не используют это поле.
- **data=ordered** — журналирует только изменения метаданных, но обновления данных сбрасываются на жесткий диск до совершения транзакции. Данные записываются не атомарно, но этот режим гарантирует, что после падения файлы не будут содержать блоки данных из устаревших файлов.

В итоге строка в */etc/fstab* должна выглядеть примерно следующим образом (**sdX** — устрой ство SSD)

/etc/fstab

```
# <fs>    <mountpoint> <type>  <opts>                  <dump/pass>
/dev/sdX  /opt         ext4   defaults,noatime,data=ordered,errors=remount-ro 0 2
```

Выключите IO Scheduller для SSD

Выполните команду и добавьте ее в скрипт автозапуска

```
echo noop > /sys/block/sdX/queue/scheduler
```

для каждого устройства (заменяя **sdX** на нужное имя)

НАСТРОЙКА ОС

Предварительные требования

1. Файловая система для «Фактора» должна быть смонтирована в `/opt`.
2. Отключены страницы THP (Transparent Huge Pages)

Установить доступ к сайту HFLabs

Доступ нужен для скачивания приложения. Уточните логин и пароль в службе поддержки и внесите их в файл ~/.netrc для пользователей root и factor.

Для пользователя root:

/root/.netrc

```
machine fs.hflabs.ru
  login ЛОГИН
  password ПАРОЛЬ
```

Для пользователя factor:

/data/.netrc

```
machine fs.hflabs.ru
  login ЛОГИН
  password ПАРОЛЬ
```

НАСТРОЙКА ПАРАМЕТРОВ ОС

С версии «Фактора» 20.01 настройки выполняются автоматически при разворачивании по install-скрипту.

Настройки нужны для учёта специфики работы «Фактора» с памятью, большим количеством одновременно запущенных процессов и открытых файлов.

sysctl

Параметр	Описание	Требуется	Что означает	Настройка
vm.overcommit_memory	Отвечает за стратегию overcommit.	2	Без overcommit. Отказ обработки запросов, запрашивающих память, размер которой превышает суммарный размер памяти пространства подкачки и ОЗУ в соответствии с *overcommit_ratio*.	vm.overcommit_memory=2
vm.overcommit_ratio	Имеет значение только при vm.overcommit_memory=2 Определяет процентную часть физической памяти.	100	Должно использоваться 100% физической памяти, чтобы память не уходила в swap.	vm.overcommit_ratio=100
vm.max_map_count	Максимальное число областей памяти, доступных процессу.	от 16777216 и более	Для процесса должно быть доступно 2^24 областей памяти.	vm.max_map_count = 16777216
vm.swappiness	Зависимость свободной оперативной памяти и swap.	10 и менее	Установка этого параметра в 10 позволит максимально использовать оперативную память (на 90%), без задействования swap-а	vm.swappiness = 10

systemd

Настройки используются в процессах «Фактора»:

- стандартизация данных;
- поиск дубликатов;
- построение и чтение справочников на основе MMAP технологии работы с файлами.

Параметр	Описание	Требуется	Что означает	Настройка

OOMScore Adjust	Право на остановку сервиса из-за нехватки памяти.	-1000	Запрет на остановку сервиса.	OOMScore Adjust=-1000
LimitNOFILE	Максимальное количество одновременно открытых дескрипторов.	от 65535 и более	Требуется разрешение на 65535 одновременно открытых дескрипторов.	LimitNOFILE=65535
LimitNPROC	Максимальное количество одновременно запущенных потоков.	от 8192 и более	Требуется разрешение на 8192 одновременно запущенных потоков.	LimitNPROC=8192
LimitFSIZE	Максимальный размер одного файла.	infinity	Файл может быть любого размера.	LimitFSIZE=infinity
LimitAS	Максимальный размер выделяемой памяти процессу.	infinity	Размер выделяемой памяти для процесса без ограничений.	LimitAS=infinity
TasksMax	Максимальное количество потоков доступное приложению	infinity	Сколько потоков может создать приложение прежде чем его остановит ОС (актуально для старых версий systemd, например, в SUSE Linux)	TasksMax=infinity

УСТАНОВКА OPENJDK 11

Установите OpenJDK

```
mkdir -p /usr/java/
cd /usr/java/

wget https://download.java.net/java/GA/jdk11/9/GPL/openjdk-11.0.2_linux-x64_bin.tar.gz
tar xzf openjdk-11.0.2_linux-x64_bin.tar.gz

alternatives --install /usr/bin/java java /usr/java/jdk-11.0.2/bin/java 2
alternatives --install /usr/bin/jar jar /usr/java/jdk-11.0.2/bin/jar 2
alternatives --install /usr/bin/javac javac /usr/java/jdk-11.0.2/bin/javac 2
alternatives --install /usr/bin/jstack jstack /usr/java/jdk-11.0.2/bin/jstack 2
alternatives --set java /usr/java/jdk-11.0.2/bin/java
alternatives --set jar /usr/java/jdk-11.0.2/bin/jar
alternatives --set javac /usr/java/jdk-11.0.2/bin/javac
alternatives --set jstack /usr/java/jdk-11.0.2/bin/jstack
```

Проверьте корректность установки:

```
java -version
```

УСТАНОВКА КОНТЕЙНЕРА ПРИЛОЖЕНИЙ И НАСТРОЙКА СЕРВИСА

ПОДГОТОВКА К УСТАНОВКЕ

Создание локальных переменных

Создайте локальную переменную *HFLABS_ARTEFACTS* — директория, куда выложены предоставленные HFLabs ресурсы, необходимые для установки.

```
# export HFLABS_ARTEFACTS=/opt/hflabs_dist && mkdir -p $HFLABS_ARTEFACTS
```

Скачивание артефактов

Поддержка предоставит архивы:

- factor-<customer>-<version>-wildfly-cli.zip
- factor-<customer>-<version>-wildfly-cli.zip.sha1
- factor-<customer>-<version>-wildfly-install.zip
- factor-<customer>-<version>-wildfly-install.zip.sha1
- factor-<customer>-<version>.war
- factor-<customer>-<version>.war.sha1
- wildfly-16.0.0.Final-vanilla.zip
- wildfly-16.0.0.Final-vanilla.zip.sha1

Скачайте их с веб-сервера и скопируйте на сервер с «Фактором» в директорию *HFLABS_ARTEFACTS*.

Установка контейнера приложений

1. Перейдите в *HFLABS_ARTEFACTS*.
2. Выполните скрипт установки из-под root:

   ```
   unzip $HFLABS_ARTEFACTS/factor-<customer>-<version>-wildfly-install.zip && sh $HFLABS_ARTEFACTS
   /install-wf.sh
   ```

   a. Скрипт
      i. проверит: наличие артефактов и контрольных сумм для них; корректность установки OpenJDK;
      ii. установит и зарегистрирует службу в systemd;
      iii. создаст группу и пользователя для службы;
      iv. выполнит проверочный запуск контейнера приложений.

3. Проверьте, что служба `factor` успешно установлена:

```
systemctl status factor
```

СТРУКТУРА КАТАЛОГОВ «ФАКТОРА»

Каталоги создаются автоматически при использовании cli и install скриптов

Требования к структуре каталогов

Файловая система должна быть смонтирована в `/opt`.

Стандартная структура каталогов для «Фактора»:

```
/opt/factor
|-appserver
|-blacklists
|-configuration
|-dedup
|-dictionaries
|-indexes
|-tmp
```

Где:

- `appserver` — контейнер приложений;
- `blacklists` — директория исходных справочников для ЧС;
- `configuration` — файлы конфигураций (`factor.conf, *.properties`);
- `dedup` — директория индексов дедупликации;
- `dictionaries` — директория исходных справочников для горячего обновления;
- `indexes` — директория внутренних индексов;
- `tmp` — временная директория.

Настройки службы находятся в `/etc/systemd/system/factor.service`

Настройки ОС находятся в `/etc/sysctl.d/50-hflabs.conf`

Настройки переменных запуска «Фактора» в `/opt/factor/configuration`:

- `factor.conf` — настройки одинаковые для всех серверов (используется по умолчанию)
- `factor-customer.conf` — настройки отличающиеся от factor.conf (используется опционально, если у заказчика несколько серверов в промышленной эксплуатации с разными настройками)

НАСТРОЙКА «ФАКТОРА»

Настройте конфигурационные файлы

Подложите в `/opt/factor/configuration` конфигурационные файлы вида `*.properties`, если функционал используется:

- `siebel-db.properties`
- `monitor-db.properties`

Настройте переменные для запуска

> Изменять в `/opt/factor/appserver/bin/standalone.conf` ничего не нужно.

Все переменные — в конфигурационном файле `/opt/factor/configuration/factor.conf` и `factor-customer.conf` (опционально).

По умолчанию:

- используется standalone-режим;
- порт 8080;
- GC-лог включен;
- для горячего обновления справочников используется директория `/opt/factor/dictionaries`;
- режим загрузки внешних справочников — асинхронный (ASYNC). *Как настроить другой режим загрузки внешних справочников?*

При необходимости измените настройки по умолчанию.

Подложите сборку «Фактора»

```
mv $HFLABS_ARTEFACTS/factor-<customer>-<VersionRevision>.war /opt/factor/appserver/standalone
/deployments
```

Подложите необходимые справочники

Справочники подкладываются вместе с контрольными суммами в директорию `/opt/factor/dictionaries`.

Выдайте права

```
chown -R factor:hflabs /opt/factor
```

Запустите приложение

Под пользователем root.

```
systemctl start factor
```

Проверьте корректность запуска

1. Через 15-20 минут после запуска откройте `wsdl` в браузере:

   ```
   http://[Ip_адрес_сервера_с_Фактором]:[порт_сервера_с_Фактором]/factor-service-<customer>/services/EssenceCleanService?wsdl
   ```

   Если открылся xml-файл, значит сервис стартовал корректно.
2. Проверьте логи в директории `/opt/factor/appserver/standalone/log`, в них не должно быть ошибок (`ERROR`).
3. Откройте в браузере ссылку:

   ```
   http://[Ip_адрес_сервера_с_Фактором]:[порт_сервера_с_Фактором]/factor-service-<customer>/
   ```

   Номер ревизии и версия установленного «Фактора» должны совпадать с указанными при поставке.

ИЗМЕНЕНИЕ НАСТРОЕК И ПОРЯДОК ИХ ПРИМЕНЕНИЯ

Страница видна только сотрудникам HFLabs.

Настройка

В разделе описано изменение настроек, которые объявлены через `PropertySourcesPlaceholderConfigurer` + `PreferredResourcesFactoryBean`.

Задать и перекрыть настройки можно в следующих местах:

1. JVM-переменные
2. ENV-переменные
3. ${jboss.home.dir}/some_name.properties
4. ${FACTOR_CONFIGURATION_PATH}/some_name.properties
5. customer/build-config/server/some_name.properties
6. XML

Чем выше номер в списке, тем выше приоритет. Например, пункт 1 перекроет значения настроек в пунктах 2-6.

Значение каждого параметра настройки может быть указано в одном или нескольких пунктах.

Пример: в XML указаны все параметры, в `${FACTOR_CONFIGURATION_PATH}/some_name.properties` задана настройка только двух параметров. Настройки по остальным вариантам отсутствуют.

При изменении настроек в любом из перечисленных пунктов нужен рестарт «Фактора».

Конечное значение настройки можно увидеть в `factor-stats-system.log`.

В примере ниже в `blacklist-service.xml` объявлена конфигурация `blacklistPropertyPlaceholderConfigurer`. Ее свойства можно изменить через:

- свойства бина `blacklistPropertyPlaceholderConfigurer` через элемент `<property name="properties">`.
- файл `blacklist.properties`. В файле указываются названия параметров и их значения.
- JVM и ENV переменные.

blacklist/config/blacklist-service.xml

```
<bean id="blacklistPropertyPlaceholderConfigurer" class="org.springframework.context.support.
PropertySourcesPlaceholderConfigurer"
    p:placeholderPrefix="$blacklist-service{"
    p:ignoreResourceNotFound="true">
  <property name="locations">
          <!-- Настройки из файла blacklist.properties можно менять извне -->
    <bean class="ru.hflabs.cleaner.util.spring.PreferredResourcesFactoryBean" c:relativePath="blacklist.
properties"/>
  </property>
  <property name="properties">
    <bean class="ru.hflabs.cleaner.util.spring.BeanUtils" factory-method="mergeProperties">
      <constructor-arg name="firstProperties" ref="deduplicationServiceProperties.engine"/>
      <constructor-arg name="secondProperties">
        <util:properties>
          <prop key="factor.blacklist.offline.cron"/>

          <prop key="factor.dedup.hasherMatchLimit">3000</prop>

          <prop key="factor.dedup.rebuild.threadsCount">-1</prop>
          <prop key="factor.dedup.rebuild.threadsPriority">-1</prop>
        </util:properties>
      </constructor-arg>
    </bean>
  </property>
</bean>
```

1. JVM-переменные

JVM-переменные — это параметры, настраиваемые для java-машины. Не рекомендуется использовать этот вариант настройки на стендах заказчика. Вариант может использоваться для локальных экспериментов или у заказчиков, у которых исторически настройки заданы через JVM-переменные.

Настройка для Windows

Настройка в файле standalone.conf.bat для Windows.

Настройка для Linux

Настройки переменных запуска «Фактора» в `/opt/factor/configuration`:

- `factor.conf` — настройки одинаковые для всех серверов (используется по умолчанию);
- `factor-customer.conf` — настройки отличающиеся от `factor.conf` (используется опционально, если у заказчика несколько серверов в промышленной эксплуатации с разными настройками).

standalone.conf.bat

```
rem # Extra JVM options
set "FACTOR_DEDUP_HASHER_MATCH_LIMIT=-Dfactor.dedup.hasherMatchLimit=10000"
set "JAVA_OPTS_EXTRA=%FACTOR_DEDUP_HASHER_MATCH_LIMIT%"

set "JAVA_OPTS=%JAVA_OPTS% %JAVA_OPTS_JPDA% %JAVA_OPTS_BINDING% %JAVA_OPTS_TIMEOUT% %
JAVA_OPTS_IO% %JAVA_OPTS_MEMORY% %JAVA_OPTS_CODECACHE% %JAVA_OPTS_GC% %
JAVA_OPTS_EXTRA%"
```

2. ENV-переменные

ENV-переменные — переменные среды окружения. Не рекомендуется использовать этот вариант настройки на стендах заказчика. Вариант может использоваться для локальных экспериментов или у заказчиков, у которых исторически настройки заданы через ENV-переменные.

Настройка для Windows

ENV-переменные настраиваются через пункт меню «Переменные среды».

Пример настройки переменной JAVA_HOME.

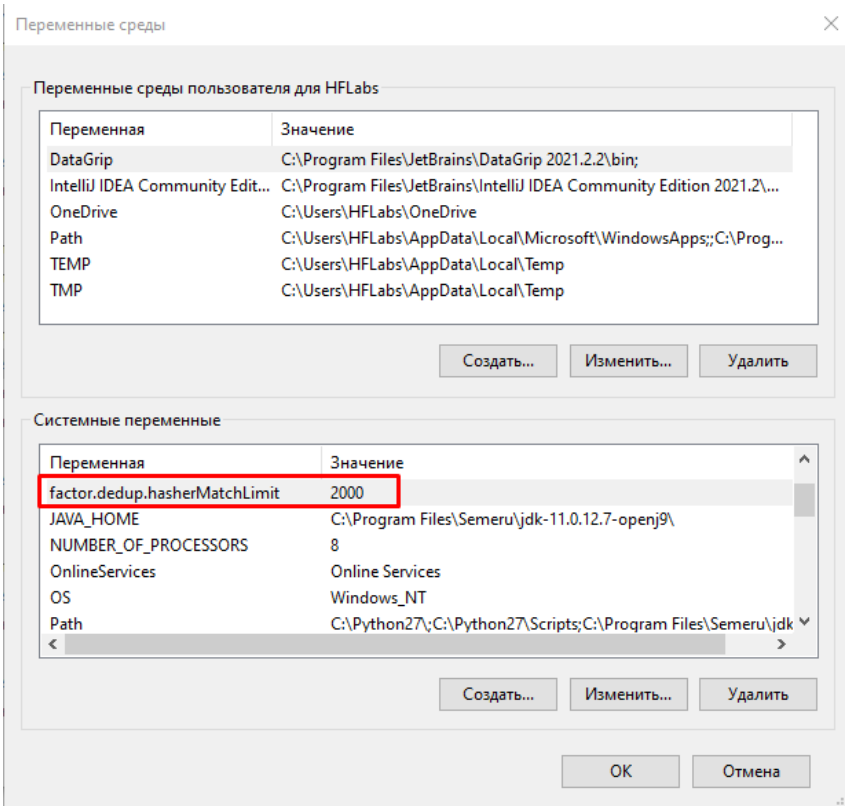
Настройка для Linux

Настройка зависит от используемого дистрибутива.

Пример настройки.

Настройка для Docker

ENV-переменные задаются при запуске контейнера через параметры `--env` , `-e`.



3. ${jboss.home.dir}/some_name.properties

`${jboss.home.dir}` — корневая директория Wildfly (WF_HOME). Не рекомендуется использовать этот вариант настройки на стендах заказчика. Вариант может использоваться для локальных экспериментов или у заказчиков, у которых исторически настройки лежат в корне WF.

/u01/factor/appserver/blacklist.properties

```
factor.dedup.online.restrict=50
# PND hasher & comparator require sequential iteration
factor.dedup.rebuild.threadsCount=1
# example
factor.dedup.hasherMatchLimit=8000
```

4. ${FACTOR_CONFIGURATION_PATH}/some_name.properties

`${FACTOR_CONFIGURATION_PATH}` — директория для хранения property-файлов. Рекомендуемый вариант настройки.

Путь к `FACTOR_CONFIGURATION_PATH` задаётся в файле `factor.conf`.

/u01/factor/configuration/factor.conf

```
FACTOR_ROOT_PATH=${FACTOR_ROOT_PATH:-/u01/factor}

FACTOR_CONFIGURATION_PATH=${FACTOR_CONFIGURATION_PATH:-${FACTOR_ROOT_PATH}/configuration}
```

/u01/factor/configuration/blacklist.properties

```
factor.dedup.online.restrict=50
# PND hasher & comparator require sequential iteration
factor.dedup.rebuild.threadsCount=1
# example
factor.dedup.hasherMatchLimit=5000
```

5. customer/build-config/server/some_name.properties

property-файл можно положить в директорию заказчика в репозитории проекта. Вариант используется для настроек, которые редко меняются и будут применяться к сборке «Фактора» заказчика независимо от настроек ОС и параметров сервера.

demo/build-config/server/blacklist.properties

```
factor.dedup.online.restrict=50
# PND hasher & comparator require sequential iteration
factor.dedup.rebuild.threadsCount=1
# example
factor.dedup.hasherMatchLimit=13000
```

6. XML

Значения параметров, указываемые в xml-файле в репозитории проекта. Вариант используется для настроек, которые будут применяться ко всем заказчикам по умолчанию.

blacklist/config/blacklist-service.xml

```
<bean id="blacklistPropertyPlaceholderConfigurer" class="org.springframework.context.support.
PropertySourcesPlaceholderConfigurer"
    p:placeholderPrefix="$blacklist-service{"
    p:ignoreResourceNotFound="true">
  <property name="locations">
        <!-- Настройки из файла blacklist.properties можно менять извне -->
    <bean class="ru.hflabs.cleaner.util.spring.PreferredResourcesFactoryBean" c:relativePath="blacklist.
properties"/>
  </property>
  <property name="properties">
    <bean class="ru.hflabs.cleaner.util.spring.BeanUtils" factory-method="mergeProperties">
      <constructor-arg name="firstProperties" ref="deduplicationServiceProperties.engine"/>
      <constructor-arg name="secondProperties">
        <util:properties>
                    <!-- Можем указать настройки по умолчанию -->
          <prop key="factor.blacklist.offline.cron"/>

          <prop key="factor.dedup.hasherMatchLimit">3000</prop>

          <prop key="factor.dedup.rebuild.threadsCount">-1</prop>
          <prop key="factor.dedup.rebuild.threadsPriority">-1</prop>
        </util:properties>
      </constructor-arg>
    </bean>
  </property>
</bean>
```

Пример

№	Способ настройки	Значение
1	JVM-переменная	set "FACTOR_DEDUP_HASHER_MATCH_LIMIT=-Dfactor.dedup.hasherMatchLimit=10000"
2	ENV-переменная	factor.dedup.hasherMatchLimit=2000
3	Корень Wildfly /u01/factor/appserver/blacklist.properties	factor.dedup.hasherMatchLimit=8000
4	FACTOR_CONFIGURATION_PATH /u01/factor/configuration	factor.dedup.hasherMatchLimit=5000
5	demo/build-config/server/blacklist.properties	factor.dedup.hasherMatchLimit=13000
6	blacklist-service.xml <prop key="factor.dedup.hasherMatchLimit">3000</prop>	<prop key="factor.dedup.hasherMatchLimit">3000</prop>

Результатом применения и приоритезации настроек будет значение factor.dedup.hasherMatchLimit=10000.

НАСТРОЙКИ ПЕРЕМЕННЫХ ЗАПУСКА «ФАКТОР»

- Добавление переменных
- Изменение порта
- Переключение в кластерный режим

ДОБАВЛЕНИЕ ПЕРЕМЕННЫХ

Используется Bash-синтаксис.

В разделе `# Extra settings`:

- указывается название переменной и её значения в виде: `НазваниеАргумента=ПрисвоенноеЗначение`
- в переменной FACTOR_JVM_EXTRA перечисляются все переменные, которые необходимо переопределить, в виде синтаксиса: `$НазваниеПеременной1 $НазваниеПеременной2 $НазваниеПеременной3`

Например

/opt/factor/configuration/factor.conf

```
# Extra settings
FACTOR_BL_PATH=-Dfactor.blacklistPath=/opt/blacklist
FACTOR_BL_PATH_1=-Dfactor.blacklist01=${factor.blacklistPath}
FACTOR_BL_PATH_2=-Dfactor.blacklist02=${factor.blacklistPath}

FACTOR_JVM_EXTRA=$FACTOR_BL_PATH $FACTOR_BL_PATH_1 $FACTOR_BL_PATH_2
```

ИЗМЕНЕНИЕ ПОРТА

В конфигурационном файле расскоментировать строку и добавить необходимое смещение порта.

Пример настройки, чтобы Фактор был доступен на порту 18080:

/opt/factor/configuration/factor.conf

```
# The offset of all used ports
WILDFLY_PORT_OFFSET=10000
```

ПЕРЕКЛЮЧЕНИЕ В КЛАСТЕРНЫЙ РЕЖИМ

Для *master* узла в конфигурационном файле сервиса установить режим domain:

/opt/factor/configuration/factor.conf

```
WILDFLY_MODE=domain
```

Для *slave* узла в конфигурационном файле сервиса установить режим *domain* и прописать *IP адрес master* узла

/opt/factor/configuration/factor.conf

```
WILDFLY_MODE=domain
WILDFLY_DOMAIN_ADDRESS=192.168.0.1
```

Запустить сервисы на всех узлах (сначала *master*, затем *slave*)

```
systemctl start factor
```

Управление кластером осуществляется через CLI или WEB (*http://MASTER_NODE_IP:9990*) интерфейсы *master* ноды. Параметры аутентификации:

- login – admin
- pswd – cdi

РЕЖИМЫ ЗАГРУЗКИ ВНЕШНИХ РЕСУРСОВ

Переменные окружения имеют приоритет, который работает и для загрузки ресурсов при старте «Фактора».

Справочник ищем в той директории, приоритет переменной которой выше.

Например, для ФИАС:

- приоритет директорий со справочником: `FACTOR_FIAS_CURRENT_PATH` → `FACTOR_INDEXES_PATH` → `${factor.userLibPath}/address2/resources/current`
- приоритет директорий для обновления справочников: `FACTOR_FIAS_SHARE_PATH` → `FACTOR_DICTIONARIES_PATH` → `${factor.userLibPath}/address2/resources/share`

Асинхронный (по умолчанию)

Работает с версии 21.2+

1. Проверяем директорию с локальным справочником / директорию с индексами:
   a. В директории есть подходящий справочник / индексы — загружаем.
   b. В директории нет подходящего справочника / индексов — загружаем доступный ресурс из сборки.
   c. В директории и в сборке нет подходящего справочника / индексов — проверяем директорию для обновлений справочников:
      i. В директории есть подходящий справочник — загружаем его.
2. Запускаем инит-мэппинги.
3. Старт приложения.

Синхронный (настраиваемый)

1. Проверяем директорию с локальным справочником / директорию с индексами:
   a. В директории нет подходящего справочника / индексов — загружаем доступный ресурс из сборки.
   b. В директории есть подходящий справочник / индексы — загружаем, если они актуальнее.
2. Проверяем директорию для обновлений справочников:

a. Если справочник был загружен в п.1 и в директории есть справочник актуальнее загруженного — обновляем его.
b. Если справочник не был загружен в п.1 и в директории есть подходящий справочник — загружаем его.

3. Запускаем инит-мэппинги.
4. Старт приложения.

Без загрузки ресурсов (настраиваемый)

Нет проверки доступных ресурсов при старте «Фактора».

Настройка режима загрузки

/opt/factor/configuration/factor.conf

```
# Check external resources mode (DISABLED|SYNC|ASYNC)
FACTOR_RESOURCES_CHECKMODE=SYNC
```