IT LAB

Инструкция по пуску

Сервис сквозной аналитики Мерлин

Сервер мониторинга

Версия документа: 2.2

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

В настоящей инструкции описывается процедуры установки, настройки и запуска компонента «Сервер мониторинга» Сервиса сквозной аналитики Мерлин, в дальнейшем, для краткости, называемого «система» или «система Мерлин».

Настоящая инструкция предназначена для специалиста, выполняющего изложенные в ней процедуры. Требования к уровню его подготовки изложены в разделе 3.1 настоящей инструкции. В инструкции предполагается, что она будет выполняться последовательно в порядке следования текста.

1.1 Список изменений

Версия документа	Дата	Изменение

2 СПИСОК СОКРАЩЕНИЙ

Сокращение	Расшифровка
СМ	Сервер мониторинга
ОС	Операционная система
ПО	Программное обеспечение
БД	База данных

3 НЕОБХОДИМЫЕ УСЛОВИЯ

3.1 Требования к квалификации специалиста

Все операции по установке, описанные в данной инструкции, должны выполняться системным администратором, имеющим соответствующую квалификацию по установке серверных систем.

Системный администратор должен отвечать следующим квалификационным требованиям:

- знание основ работы с ОС Linux;
- знание основ работы в СУБД PostgreSQL;
- знание основ работы с ActiveMQ;
- навыки управления ОС и утилитами в режиме командной строки;
- понимание средств управления защитой данных в операционных системах;
- знание методов и инструментария для решения задач системного администрирования.

3.2 Комплект поставки

В комплект поставки входят следующие файлы:

- `tracker-app-X.X.X.X.jar` – дистрибутив сервера мониторинга.
- `sms-server-0.1.8.jar` – ПО СМС-сервера.
- `zabbix_templates.zip` – скрипты для настройки Zabbix.
- `MERLIT-GL-RU-02.00.00.dSTP.01.00.pdf` – Сервис сквозной аналитики Мерлин. Сервер мониторинга. Инструкция по пуску (Настоящий документ).

3.3 Требуемое оборудование

Состав оборудования зависит от варианта развертывания системы.

4 ПУСК СИСТЕМЫ

4.1 Перед пуском

Прежде чем устанавливать и запускать сервер мониторинга, необходимо убедиться, что выполнены следующие действия и требования:

1) Подготовлены клиентские приложения, в которые будет встраиваться библиотека системы Мерлин. Как минимум, такие приложения должны быть запланированы к производству, и должны быть известны: для Android-приложения – имя пакета (package name), для iOS-приложения – bundle identifier.
2) Если планируется внедрение iOS-приложения, то в кабинете разработчика Apple должен быть создан ключ для отправки сообщений APNs. Этот ключ можно скачать из кабинета разработчика только один раз (!), поэтому его лучше сохранить в надежном месте.
3) Для приложений должен быть создан проект в Firebase console (https://console.firebase.google.com). В нем должны быть зарегистрированы package name для Android-приложения и bundle identifier для приложения iOS. Кроме того, необходимо на вкладке Cloud messaging загрузить ключ для отправки APNs сообщений. В этом случае все push-сообщения можно будет доставлять через Firebase.
4) Если необходимо использовать геокодирование (преобразование данных широты и долготы в адрес местонахождения клиента), то необходимо получить access token для сервиса Google Maps. Система будет работоспособна и без этой функции.

4.2 Базовый вариант развертывания

В базовом варианте серверный комплекс разворачивается на одном сервере и включает с себя базу данных PostgreSQL и сервер приложений в виде исполняемого jar файла. Сервер приложений содержит встроенный HTTP-сервер и встроенный брокер ActiveMQ.

Внешний доступ на сервер должен осуществляться через отдельный внешний сетевой интерфейс, закрытый nginx-фронтэндом, который обеспечивает шифрование клиентского трафика.

Служебный доступ к комплексу (настройка, администрирование, мониторинг Zabbix и т.п.) должны осуществляться через доверенную локальную сеть.

4.2.1 Установка Ubuntu

Необходимо установить Ubuntu 22.04. Дистрибутив и инструкцию по установке можно найти по ссылке:

https://help.ubuntu.com/community/Installation

4.2.2 Установка Java

Установка OpenJDK 8 выполняется командой `apt-get`:

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

4.2.3 PostgreSQL

4.2.3.1 Установка

Установку PostgreSQL и расширения `postgis` можно выполнить, используя репозиторий Ubuntu, командой `apt-get`:

```
sudo apt-get install postgresql postgresql-postgis
```

4.2.3.2 Создание баз данных и пользователя admin

Создайте две базы данных, основная (`merlin`) и хранилище сообщений ActiveMQ (amq):

```
sudo su postgres
createdb merlin
createdb amq
```

Запустите SQL консоль и создайте пользователя admin:

```
psql
create user admin superuser encrypted password 'admin';
\q
```

4.2.4 ActiveMQ

Данный шаг можно пропустить, если планируется использование встроенного брокера ActiveMQ в базовом варианте развертывания.

4.2.4.1 Установка

Загрузите архив версии 5.15.15 с официального сайта и распакуйте его:

```
cd /opt
… Download apache-activemq-5.15.15-bin.tar.gz
sudo tar xzf apache-activemq-5.15.15-bin.tar.gz
sudo mv apache-activemq-5.15.15 activemq
```

Создайте пользователя, от имени которого будет запускаться ActiveMQ:

```
sudo addgroup --quiet --system activemq
sudo adduser --quiet --system --ingroup activemq --no-create-home --
disabled-password activemq
sudo usermod -c "ActiveMQ Broker" -d /opt/activemq -g activemq
activemq
```

Скачайте JDBC-драйвер для postgres:

```
cd /opt/activemq/lib
sudo wget https://jdbc.postgresql.org/download/postgresql-42.2.4.jar
```

Установите права доступа к файлам ActiveMQ:

```
sudo chown -R activemq:activemq /opt/activemq
sudo chmod u=rwx,g=rxs,o=r /opt/activemq
```

4.2.4.2 Конфигурация

В файле `/opt/activemq/conf/activemq.xml` замените настройки сохранения сообщений по умолчанию на использование JDBC:

```
<persistenceAdapter>
  <jdbcPersistenceAdapter dataSource="#postgres"
lockKeepAlivePeriod="3000" createTablesOnStartup="true">
    <locker>
      <lease-database-locker lockAcquireSleepInterval="10000"
queryTimeout="8"/>
    </locker>
  </jdbcPersistenceAdapter>
</persistenceAdapter>
```

Добавьте настройки data source:

```
<bean id="postgres" class="org.postgresql.ds.PGPoolingDataSource">
  <property name="serverName" value="localhost"/>
  <property name="databaseName" value="amq"/>
  <property name="portNumber" value="5432"/>
  <property name="user" value="admin"/>
  <property name="password" value="admin"/>
  <property name="dataSourceName" value="postgres"/>
  <property name="initialConnections" value="1"/>
  <property name="maxConnections" value="10"/>
</bean>
```

После первого запуска ActiveMQ следует сбросить настройку, отвечающую за создание таблиц при запуске:

```
createTablesOnStartup="false"
```

4.2.4.3 Автоматический запуск

Создайте файл `/etc/systemd/system/activemq.service` со следующим содержимым:

```
[Unit]
Description=Apache ActiveMQ
After=network.target postgres.target

[Service]
Type=forking
WorkingDirectory=/opt/activemq/bin
ExecStart=/opt/activemq/bin/activemq start
ExecStop=/opt/activemq/bin/activemq stop
Restart=on-abort
User=activemq
Group=activemq

[Install]
WantedBy=multi-user.target
```

После чего обновите конфигурацию `systemd`, разрешите данный сервис для автозапуска и запустите сервис:

```
sudo systemctl daemon-reload
sudo systemctl enable activemq
sudo systemctl start activemq
```

4.2.5 Сервер мониторинга системы Мерлин

4.2.5.1 Установка

Создайте директорию `/opt/merlin`

```
sudo mkdir -p /opt/merlin
```

Скопируйте файл `tracker-app-X.X.X.X.jar` из комплекта поставки в директорию `/opt/merlin`.

Создайте пользователя, от имени которого будет запускаться СМ:

```
sudo addgroup --quiet --system merlin
sudo adduser --quiet --system --ingroup merlin --no-create-home --
disabled-password merlin
sudo usermod -c "Merlin Tracker" -d /opt/merlin -g merlin merlin
sudo chown -R merlin:merlin /opt/merlin
sudo chmod u=rwx,g=rxs,o=/opt/merlin
```

4.2.5.2 Конфигурация

Значения по умолчанию всех настраиваемых параметров содержатся в JAR архиве `tracker-app-X.X.X.X.jar/application.properties`.

Для переопределения настроек нужно создать файл `application.properties` в той же директории, где находится исполняемый JAR архив `tracker-app.X.X.X.X.jar`.

4.2.5.2.1 Веб-сервер APM

`gui.enabled = true` – Установка значения `false` позволяет отключить веб-сервер APM.

`gui.server.port = 8080` – Порт, на котором будет запускаться веб-сервер APM.

`gui.server.servlet.session.timeout = 30m` – Таймаут сессии пользователей. Можно указывать единицы измерения ms, s, m, h, d. По умолчанию используются секунды.

4.2.5.2.2 Веб-сервер обработки запросов от устройств

`device-api.enabled = true` – Установка значения `false` позволяет отключить веб-сервер обработки запросов.

`device-api.server.port = 8081` – Порт, на котором будет запускаться веб-сервер обработки запросов.

`device-api.jwt.secret = secret` – Секретный ключ. Используется при аутентификации устройств.

4.2.5.2.3 Подключение к базе данных

`primary.datasource.url = jdbc:postgresql://localhost/merlin` – Строка подключения к БД в формате `jdbc:postgresql://<host>:<port>/<database>`, где:

- `<host>` – имя хоста, на котором установлена БД PostgreSQL,
- `<port>` – порт для подключения к БД PostgreSQL,
- `<database>` – имя БД PostgreSQL.

`primary.datasource.username = admin` – Имя пользователя БД.

`primary.datasource.password = admin` – Пароль пользователя БД.

4.2.5.2.4 Встроенный брокер Active MQ

`jms-broker.enabled = true` – Установка значения `false` позволяет отключить встроенный JMS брокер. В этом случае должен использоваться внешний брокер.

`jms-broker.bind = vm://localhost` – URI брокера.

`jms-broker.create-tables-on-startup = true` – Создание таблиц при запуске. Эту настройку следует отключить после первого запуска приложения.

`jms-broker.datasource.url = jdbc:postgresql://localhost/amq` – Строка подключения к БД в формате `jdbc:postgresql://<host>:<port>/<database>`, где:

`<host>` – имя хоста, на котором установлена БД PostgreSQL,

`<port>` – порт для подключения к БД PostgreSQL,

`<database>` – имя БД PostgreSQL.

`jms-broker.datasource.username = admin` – Имя пользователя БД.

`jms-broker.datasource.password = admin` – Пароль пользователя БД.

4.2.5.2.5 Клиент ActiveMQ

`spring.activemq.broker-url = vm://localhost` – URI брокера.

4.2.5.2.6 Почтовый сервер

`tracker.mail.sender=` – E-mail отправителя сообщений.

`tracker.mail.smtp=10.1.0.10` – Адрес почтового сервера.

4.2.5.2.7 СМС-сервер

`smsserver.baseUrl =` – Адрес СМС-сервера (см.4.1.6).

4.2.5.2.8 Push уведомления

`push.url=https://fcm.googleapis.com/fcm/send` – Адрес веб-сервиса Firebase Cloud Messaging.

`push.api-key=` – Ключ для аутентификации в сервисе Firebase Cloud Messaging. Его можно найти в консоли Firebase, на вкладке Cloud messaging настроек приложения.

`push.duration=24h` – Таймаут отсутствия запросов с устройства, по истечении которого отправляется push уведомление (`h` – hours, `m` – minutes).

`push.cron-scheduler=* * 1 * * *` – Расписание проверок на необходимость отправки уведомлений.

`push.apns.certificate =` – Путь к сертификату APNs *.p8. Его обычно скачивают в кабинете разработчика Apple.

`push.apns.key-id =` – Идентификатор ключа APNs (с включенными пуш уведомлениями)

`push.apns.team-id =` – Team ID приложения, в которое встроена библиотека Merlin. Можно найти в кабинете разработчика Apple.

`push.apns.topic =` – bundle identifier приложения, в которое встроена библиотека Merlin и которое будет получать Push уведомления.

4.2.5.2.9 Отчистка старых данных

`stale-data-cleanup.cron = * * 2 * * *` – Расписание проверок на наличие старых данных.

`stale-data-cleanup.batch-size = 1000` – Количество запросов от устройств, удаляемых в одной транзакции.

4.2.5.2.10 Http-proxy для исходящих запросов

`proxy.host =` – Адрес прокси-сервера.

`proxy.user =` – Имя пользователя.

`proxy.password =` – Пароль.

4.2.5.3 Геокодирование адреса

`google.geocoding.token =` – token для google map API

`google.geocoding.cache-duration = 300000` – время жизни кэша, полученного адреса (что бы не обращаться в гугл для получения адреса повторно)

`google.geocoding.locale = ru` – язык полученного адреса

4.2.5.4 Лицензия

`license.filename =` – путь к файлу лицензии

`license.keyring =` – путь к ключу лицензии

4.2.5.5 Автоматический запуск

Создайте файл `/etc/systemd/system/merlin.service` со следующим содержимым:

```
[Unit]
Description=Merlin Tracker
Requires=postgresql
After=postgresql

[Service]
User=merlin
ExecStart=/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -jar
/opt/merlin/tracker-app-X.X.X.X.jar
WorkingDirectory=/opt/merlin
SuccessExitStatus=143
StandardOutput=null
```

```
[Install]
WantedBy=multi-user.target
```

После чего обновите конфигурацию `systemd`, разрешите данный сервис для автозапуска и запустите сервис:

```
sudo systemctl daemon-reload
sudo systemctl enable merlin
sudo systemctl start merlin
```

4.2.6 Установка СМС-сервера

Установка СМС-сервера необходима, если планируется использование СМС рассылки уведомлений о выделенных сигналах.

Для запуска СМС-сервера поместите `sms-server-X.X.X.jar` из комплекта поставки в одну из директорий на компьютере, например, `/home/$user/sms`

Создайте файл `/etc/systemd/system/sms.service` со следующим содержимым:

```
[Unit]
Description=sms

[Service]
User=$user

# Необходимо использовать абсолютный путь к java и файл sms-server-
X.X.X.jar
ExecStart=/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java -jar
/home/$user/sms/sms-server-x.x.x.jar
WorkingDirectory=/home/$user/sms
SuccessExitStatus=143
StandardOutput=null

[Install]
WantedBy=multi-user.target
```

где `/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java` поменяйте на путь к java на вашем компьютере, а `/home/$user/sms/sms-server-x.x.x.jar` на абсолютный путь, где находится sms-server.

Выполните следующие команды:

```
sudo systemctl daemon-reload
sudo systemctl enable sms
sudo systemctl start sms
sudo systemctl status sms
```

4.2.7 nginx

Установите nginx из репозитория Ubuntu

```
sudo apt install nginx
```

В файле `/etc/nginx/nginx.conf` установите параметры производительности:

```
worker_rlimit_nofile 128000;

events {
    worker_connections 4000;
    use epoll;
    multi_accept on;
}
```

В файле `/etc/nginx/nginx.conf` установите параметры работы SSL:

```
http {
    …
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 5m;
    ssl_prefer_server_ciphers on;
    ssl_stapling on;
    # Указать доверенный DNS или 8.8.8.8
    resolver <IP address of DNS server>;
    …
}
```

В директории `/etc/nginx` создайте файл конфигурации `common_params` следующего содержания:

```
keepalive_timeout 30;
keepalive_requests 1000;
reset_timedout_connection on;
client_body_timeout 10;
send_timeout 2;

open_file_cache max=20000 inactive=20s;
open_file_cache_valid 30s;
open_file_cache_min_uses 2;
open_file_cache_errors on;

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;
#proxy_set_header X-Frame-Options SAMEORIGIN;

proxy_read_timeout 300;
proxy_connect_timeout 300;
proxy_buffering on;
proxy_buffer_size 64k;
proxy_buffers 8 256k;
proxy_busy_buffers_size 256k;
proxy_temp_file_write_size 10m;

gzip on;
gzip_proxied any;
gzip_static on;
```

```
gzip_types text/plain text/xml text/javascript
application/javascript application/x-javascript text/css;
gzip_min_length 10240;
```

В директории `/etc/nginx/sites-available` создайте файл конфигурации виртуального хоста merlin-proxy следующего содержания:

```
upstream merlin_backend {
    # Указывается порт из параметра gui.server.port
    # Порт, на котором будет запускаться веб-сервер АРМ.
    server 127.0.0.1:8083;
}
upstream api_backend {
    # Указывается порт из параметра device-api.server.port
    # Порт, на котором будет запускаться веб-сервер обработки
запросов.
    server 127.0.0.1:8080;
}

server {
    listen 80;

    server_name gs.domain.ru;

    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;

    root /var/www/html;

    server_name gs.domain.ru;

    include common_params;

    ssl_certificate certificate_bundled.crt;
    ssl_certificate_key privatekey.key;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers "HIGH:!RC4:!aNULL:!MD5:!kEDH";
    add_header Strict-Transport-Security 'max-age=604800';

    client_max_body_size 1m;

    # ht(passwd|access)
    location ~* /\.ht  { deny all; }

    # repositories
    location ~* /\.(svn|hg|git) { deny all; }

    # use the file system to access files outside the site (cache)
    location ~* /\.\./ { deny all; }

    #location ~* \.(jpg|jpeg|png|pdf|eot|svg|ttf|woff) {
```

```
    #    expires 1d;
    #}
    #location ~* \.(css|js) {
    #    expires 3h;
    #}

    location ^~ /api/ {
        # А тут не надо закрывающего слэша!
        proxy_pass http://api_backend;
    }

    location / {
        # ВНИМАТЕЛЬНО! Здесь нужен закрывающий слэш!
        proxy_pass http://merlin_backend/;
    }
}
```

Уточните в этом файле имя сервера (в директиве `server_name`), путь к файлу сертификата (в директиве `ssl_certificate`), путь к файлу приватного ключа сертификата (в директиве `ssl_certificate_key`).

Создайте символическую ссылку на виртуальный хост в директории `/etc/nginx/sites-enabled`:

```
$ cd /etc/nginx/sites-enabled
$ sudo ln -s /etc/nginx/sites-available/merlin-proxy merlin-proxy
```

Перезапустите nginx:

```
$ sudo systemctl restart nginx.service
```

4.2.8 Настройка синхронизации времени

В файле `/etc/systemd/timesyncd.conf` установить адрес сервера синхронизации времени `ru.pool.ntp.org`.

4.3 Высокодоступный вариант развертывания

В случае необходимости обслуживания высокой нагрузки в непрерывном режиме, может быть реализован высокодоступный вариант развертывания системы.

В данной конфигурации для СМ в DNS должно быть записано несколько внешних IP-адресов под одним именем. Это обеспечит первоначальное распределение нагрузки за счет циклического разрешения доменных имен (`round-robin DNS`).

Для развертывания системы потребуется, как минимум, два сервера в случае настройки кластера с асинхронной репликацией данных и, как минимум, три сервера в случае настройки кластера с синхронной репликацией. У каждого сервера обязательно должны быть заданы уникальные `hostname`. Для автоматического переключения с основной базы данных на резервную базу данных используется `stolon + consul`.

Установка Ubuntu и Java на каждом сервере производится аналогично базовому варианту развертывания.

4.3.1 Установка Consul

На каждом сервере нужно установить `consul`:

```
sudo apt install consul

cat <<EOF |sudo tee /etc/consul.d/20-agent.json >/dev/null
{
  "server": true,
  "datacenter": "merlin",
  "bootstrap_expect": 3,
  "data_dir": "/var/lib/consul",
  "enable_syslog": true
}
EOF

sudo systemctl enable consul
sudo systemctl restart consul
```

После чего на одной из нод нужно присоединить соседние ноды к кластеру:

```
consul join ip.address1 ip.address2
```

где `ip.addressN` – IP адреса соседних нод. Команду достаточно выполнить на одной из нод.

4.3.2 Установка PostgreSQL

На каждом сервере нужно установить PostgreSQL с расширениями PostGIS. Необходимо выключить стандартный сервис postgres!

```
sudo apt install -y postgresql postgresql-postgis

sudo systemctl stop postgresql
sudo systemctl disable postgresql
```

4.3.3 Установка Stolon

На каждом сервере нужно установить `stolon`. Как вариант, сборку можно провести на одном сервере и скопировать бинарники на остальные серверы. Сборка выполняется около 1.5 минут.

```
sudo apt install -y golang git make
cd /tmp
git clone https://github.com/sorintlab/stolon.git
cd stolon
make all

# Установка исполняемых файлов
sudo cp bin/* /usr/bin

sudo touch /etc/default/stolon-proxy

cat <<EOF |sudo tee /etc/systemd/system/stolon-proxy.service
>/dev/null
[Unit]
```

```
Description=Stolon Proxy
After=network.target

[Service]
Type=simple
Environment=GOMAXPROCS=2
EnvironmentFile=/etc/default/stolon-proxy
ExecStart=/usr/bin/stolon-proxy --cluster-name merlin --listen-
address 127.0.0.1 --port 25432 --store-backend consul
ExecReload=/bin/kill -HUP $MAINPID
User=postgres
Group=postgres
Restart=on-failure
RestartSec=10
LimitNOFILE=infinity

[Install]
WantedBy=multi-user.target
EOF


sudo touch /etc/default/stolon-sentinel

cat <<EOF |sudo tee /etc/systemd/system/stolon-sentinel.service
>/dev/null
[Unit]
Description=Stolon Sentinel
After=network.target

[Service]
Type=simple
Environment=GOMAXPROCS=2
EnvironmentFile=/etc/default/stolon-sentinel
ExecStart=/usr/bin/stolon-sentinel --cluster-name merlin --store-
backend consul
ExecReload=/bin/kill -HUP $MAINPID
User=postgres
Group=postgres
Restart=on-failure
RestartSec=10
LimitNOFILE=infinity

[Install]
WantedBy=multi-user.target
EOF

echo
'PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/usr/lib/postgresql/14/bin"' | sudo tee
/etc/default/stolon-keeper > /dev/null

cat <<EOF |sudo tee /etc/systemd/system/stolon-keeper.service
>/dev/null
```

```
[Unit]
Description=Stolon Keeper
After=network.target

[Service]
Type=simple
Environment=GOMAXPROCS=2
EnvironmentFile=/etc/default/stolon-keeper
ExecStart=/usr/bin/stolon-keeper --data-dir /opt/postgres --uid
`hostname|sed 's/-//g'` --cluster-name merlin --pg-su-passwordfile
/etc/stolon/pgsupasswd --pg-repl-username repluser --pg-repl-
passwordfile /etc/stolon/pgreplpasswd --store-backend consul --pg-
listen-address `hostname -i`
ExecReload=/bin/kill -HUP $MAINPID
User=postgres
Group=postgres
Restart=on-failure
RestartSec=10
LimitNOFILE=infinity

[Install]
WantedBy=multi-user.target
EOF

sudo mkdir /opt/postgres
sudo chown -R postgres:postgres /opt/postgres

sudo mkdir /etc/stolon
# Здесь нужно выбрать хорошие пароли вместо единицы!
echo "1" | sudo tee /etc/stolon/pgsupasswd > /dev/null
echo "1" | sudo tee /etc/stolon/pgreplpasswd < /dev/null
sudo chown postgres:postgres /etc/stolon/*
sudo chmod 600 /etc/stolon/*


systemctl enable stolon-proxy
systemctl enable stolon-keeper
systemctl enable stolon-sentinel

systemctl start stolon-proxy
systemctl start stolon-sentinel
systemctl start stolon-keeper
```

Инициализация кластера выполняется на одной из нод один раз:

```
stolonctl init --cluster-name merlin --store-backend consul
```

После успешной инициализации кластура нужно включить и запустить сервисы stolon в нужном порядке:

```
sudo systemctl enable stolon-sentinel
sudo systemctl enable stolon-keeper
sudo systemctl enable stolon-proxy
```

```
sudo systemctl start stolon-sentinel
sudo systemctl start stolon-keeper
sudo systemctl start stolon-proxy
```

После завершения настройки подключение к БД происходит на порт 127.0.0.1:25432. Если требуется вынести сервер приложений на отдельный сервер – на нём нужно установить `consul`, присоединить его к кластеру `consul` и установить только `stolon-proxy`:

```
sudo apt -y install consul

cat <<EOF |sudo tee /etc/consul.d/20-agent.json </dev/null
{
  "server": true,
  "datacenter": "merlin",
  "bootstrap_expect": 3,
  "data_dir": "/var/lib/consul",
  "enable_syslog": true
}
EOF

sudo systemctl enable consul
sudo systemctl restart consul

consul join list.ip.addresses.of.all.nodes

sudo touch /etc/default/stolon-proxy

cat <<EOF |sudo tee /etc/systemd/system/stolon-proxy.service
</dev/null
[Unit]
Description=Stolon Proxy
After=network.target

[Service]
Type=simple
Environment=GOMAXPROCS=2
EnvironmentFile=/etc/default/stolon-proxy
ExecStart=/usr/bin/stolon-proxy --cluster-name merlin --listen-
address 127.0.0.1 --port 25432 --store-backend consul
ExecReload=/bin/kill -HUP $MAINPID
User=consul
Group=consul
Restart=on-failure
RestartSec=10
LimitNOFILE=infinity

[Install]
WantedBy=multi-user.target
EOF

sudo systemctl enable stolon-proxy
sudo systemctl start stolon-proxy
```

4.3.4 Синхронная репликация

Для синхронной репликации требуется не менее трех нод: мастер и две реплики. Фиксация транзакций в таком случае происходит только при записи данных в `master` и хотя бы один `standby`. Это снижает риск потери данных.

По умолчанию, `Stolon` использует асинхронную репликацию. Можно переключиться на синхронную прямо во время работы кластера:

```
stolonctl --store-backend consul --cluster-name=mycluster \
 config patch '{ "synchronous_replication" : true }'
```

4.3.5 ActiveMQ

Установка и настройка ActiveMQ на каждой ноде аналогична базовому варианту развертывания. Нужно только заменить порт для подключения к БД на порт `stolon-proxy` и присвоить каждому брокеру уникальное имя (например, `merlin1`, `merlin2`, ...):

```
<broker xmlns="http://activemq.apache.org/schema/core"
brokerName="merlinN" dataDirectory="${activemq.data}">
…
</broker>
…
<bean id="postgres" class="org.postgresql.ds.PGPoolingDataSource">
  …
  <property name="portNumber" value="25432"/>
  …
</bean>
```

Примечание. Для каждого ActiveMQ должны быть созданы пользователь и БД из 4.1.3.2. Для подключения к PSQL используйте команду:
`psql -h localhost -U postgres -p 25432`
Пароль: 1 (точнее, тот хороший пароль: который вы выбрали и записали в файл `/etc/stolon/pgsupasswd`)..

4.3.6 Сервер мониторинга системы Мерлин

По сравнению с базовым вариантом развертывания, нужно указать порт `stolon-proxy` в настройках подключения к базе данных:

```
primary.datasource.url = jdbc:postgresql://localhost:25432/merlin
```

А также перечислить IP-адреса всех брокеров в URI подключения к Active MQ:

```
spring.activemq.broker-url =
failover:(tcp://ip1:61616,tcp://ip2:61616,tcp://ip3:61616)
```

Использовать встроенный брокер в этом варианте развертывания системы нельзя.

В `application.properties` сервера поставить:

```
jms-broker.enabled = false
```

4.3.7 Настройка nginx для работы с кластером

Для всех экземпляров `nginx` на узлах кластера `GameServer` должен быть установлен один и тот же параметр `server_name`, один и тот же SSL сертификат.

IP-адреса всех внешних интерфейсов узлов кластера `GameServer` должны быть добавлены в DNS зону под одним именем. На DNS сервере должна быть включена опция `round robin` перемешивания ответов. Рекомендуется также использовать как можно большие значения TTL, чтобы минимизировать излишние переключения клиентов между нодами.

4.4 Дамп базы данных по расписанию

Для настройки резервного копирования базы данных (например, ежедневно в 04:00:00) добавьте следующую строку в `/etc/chrontab`:

```
0 3 * * * pg_dump -d merlin -Fc > ~/backup/merlin_$(date
+%Y_%m_%d).dump
```