**Описание настроек компонент платформенного**

**и прикладного ПО**

**СОДЕРЖАНИЕ**

[Термины, определения, используемые сокращения и обозначения 3](#_Toc497892206)

[1 Основные компоненты системы 4](#_Toc497892207)

[2 Установка ПО 5](#_Toc497892208)

[2.1 Установка Memcached 5](#_Toc497892209)

[2.2 Установка сервера приложений Tomcat 5](#_Toc497892210)

[2.3 Система обмена сообщениями RabbitMQ 10](#_Toc497892211)

[2.4 Установка приложения wkhtmltox 11](#_Toc497892212)

[2.5 Установка сервера HAProxy 11](#_Toc497892213)

[2.6 Установка сервера PostgreSQL 13](#_Toc497892214)

[2.7 Установка сервера Cassandra 14](#_Toc497892215)

[3 Сборка приложений 16](#_Toc497892216)

[3.1 Подготовка системы 16](#_Toc497892217)

[3.2 Подготовка к сборке 17](#_Toc497892218)

[3.3 Сборка 18](#_Toc497892219)

[4 Список рисунков 20](#_Toc497892220)

Термины, определения, используемые сокращения и обозначения

| Термин | Определение |
| --- | --- |
| RPC | Remote procedure call (удаленный вызов процедур) |
| XML | eXtensible Markup Language (Расширяемый язык разметки) |
| XML-документ | Электронный документ, представленный XML-схемой |
| XML-схема (XML Schema) | Язык описания структуры документа. Предусматривает описание допустимой структуры документа и типов данных в значениях атрибутов и содержимом элементов |
| БД | База данных |
| ИС ЭП, ИС | Информационная система «Электронная Путевка» |
| ПО | Программное обеспечение |
| Сообщение | Структурированная информационная единица, передаваемая между объектами с целью инициирования некоторой деятельности (прием сообщения обычно трактуется как возникновение события) |
| Сценарий обмена электронными сообщениями | Структурированное описание (текстовое и / или в виде диаграмм) условий, состава и последовательности передачи электронных сообщений между участниками информационного обмена (прикладными процессами), отражающее особенности информационного обмена в той или иной ситуации |
| ЭП | Электронная путевка |
| Электронное сообщение | Управляющая информация, структурируемая и передаваемая в соответствии с порядком, определенным настоящим Протоколом. Может включать в себя один электронный документ |
| Электронный документ | Документ в электронном виде, подготовленный в соответствии с описанием форматов электронных форм документов, создаваемый и используемый для передачи информации между участниками обмена на прикладном уровне |
| СКЗИ | Средство криптографической защиты информации |

# Основные компоненты системы

Основными компонентами системы являются:

* Сервера приложений epCoreApplication (Web application):
* Сервер приложений Tomcat;
* Сервер memcached;
* Приложение wkhtmltox;
* Библиотека КриптоПро JCP 2.0;
* Сервера приложений epTransportApplication (Web application):
* Сервер приложений Tomcat;
* Система обмена сообщениями RabbitMQ;
* СУБД Postgressql;
* СУБД Cassandra;
* Балансировщик нагрузки HAProxy.

# Установка ПО

* 1. Установка Memcached

Сервер memcached устанавливается на каждом сервере приложений Tomcat приложения epCoreAppication, служит как единое место хранения сессий.

# yum –y install memcached

# systemctl enable memcached

# systemctl restart memcached

* 1. Установка сервера приложений Tomcat

Установка производится на всех серверах приложений epCoreAppication и epTransport.

* + 1. Пакеты ОС

Установить пакеты, необходимые для работы:

# yum -y install wget tar nano zip

* + 1. Установка Java

# yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel

Установка переменной JAVA\_HOME.

Добавить в конец файла ~/.bash\_profile:

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0/

Выполнить:

# source ~/.bash\_profile

Для проверки установки:

# java –version

* + 1. Установка сервера приложений Tomcat

Для каждого tomcat сервера, что для серверов приложения epCore, что для серверов приложения epTransport, настройки идентичные:

# cd ~

# wget http://apache-mirror.rbc.ru/pub/apache/tomcat/tomcat-9/v9.0.1/bin/apache-tomcat-9.0.1.tar.gz

# mkdir /opt

# tar xvf apache-tomcat-9\*tar.gz -C /opt/

# ln -s /opt/apache-tomcat-9.0.1 /opt/tomcat

# sudo chown -R `echo $USER` /opt/tomcat

Создать systemd unit для управления и автозапуска tomcat:

# nano /etc/systemd/system/tomcat.service

С содержимым:

[Unit]

Description=Apache Tomcat 9 Servlet Container

After=syslog.target network.target

[Service]

User=**username**

Type=forking

Environment=CATALINA\_PID=/opt/tomcat/apache-tomcat.pid

Environment=CATALINA\_HOME=/opt/tomcat

Environment=CATALINA\_BASE=/opt/tomcat

Environment=CATALINA\_OPTS=-Xmx11024m

ExecStart=/opt/tomcat/bin/startup.sh

ExecStop=/opt/tomcat/bin/shutdown.sh

Restart=on-failure

[Install]

WantedBy=multi-user.target

где **username** – имя пользователя, под которым должен работать tomcat.

Для настройки tomcat сервера, для того что бы отвечал с https прокси, необходимо в файле настроек server.xml вместо строки:

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

вставить замену на:

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="9443"

proxyPort="8443"

scheme="https" />

где **proxyport** - это порт https, который слушает прокси сервер.

Для настройки хранения сессий в memcached сервере необходимо на серверах приложений epCoreAppication:

1. Скачать дополнительные библиотеки и сохранить их папке lib tomcat сервера:

# wget https://repo1.maven.org/maven2/de/javakaffee/msm/memcached-session-manager/2.1.1/memcached-session-manager-2.1.1.jar -O /opt/tomcat/lib/memcached-session-manager-2.1.1.jar

# wget https://repo1.maven.org/maven2/de/javakaffee/msm/memcached-session-manager-tc8/2.1.1/memcached-session-manager-tc8-2.1.1.jar -O /opt/tomcat/lib/memcached-session-manager-tc8-2.1.1.jar

# wget https://repo1.maven.org/maven2/de/javakaffee/msm/msm-kryo-serializer/2.1.1/msm-kryo-serializer-2.1.1.jar -O /opt/tomcat/lib/msm-kryo-serializer-2.1.1.jar

# wget http://central.maven.org/maven2/de/javakaffee/kryo-serializers/0.41/kryo-serializers-0.41.jar -O /opt/tomcat/lib/kryo-serializers-0.41.jar

# wget http://central.maven.org/maven2/com/esotericsoftware/kryo/4.0.0/kryo-4.0.0.jar -O /opt/tomcat/lib/kryo-4.0.0.jar

# wget http://central.maven.org/maven2/org/ow2/asm/asm/5.0.4/asm-5.0.4.jar -O /opt/tomcat/lib/asm-5.0.4.jar

# wget http://central.maven.org/maven2/com/esotericsoftware/minlog/1.3.0/minlog-1.3.0.jar -O /opt/tomcat/lib/minlog-1.3.0.jar

# wget http://central.maven.org/maven2/org/objenesis/objenesis/2.2/objenesis-2.2.jar -O /opt/tomcat/lib/objenesis-2.2.jar

# wget http://central.maven.org/maven2/com/esotericsoftware/reflectasm/1.11.3/reflectasm-1.11.3.jar -O /opt/tomcat/lib/reflectasm-1.11.3.jar

# wget http://central.maven.org/maven2/net/spy/spymemcached/2.11.1/spymemcached-2.11.1.jar -O /opt/tomcat/lib/spymemcached-2.11.1.jar

1. Настроить conf/context.xml. Для этого необходимо добавить секцию перед закрывающим тэгом </Context>:

<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"

memcachedNodes="n1:**< node1\_ip\_address >**:11211,n2:**< node2\_ip\_address >**:11211"

sticky="false"

sessionBackupAsync="true"

storageKeyPrefix="static:tmctx"

requestUriIgnorePattern=".\*\.(ico|png|gif|jpg|css|js)$"

transcoderFactoryClass="de.javakaffee.web.msm.serializer.kryo.KryoTranscoderFactory"

/>

где в параметре memcachedNodes перечислены memcached сервера, n1, n2 – индексы.

Для настройки логирования, уменьшения занимаемого места логами необходимо отключить механизм ротации логов от tomcat и предоставить это действие утилите logrotate.

Для этого необходимо:

1. в файле настроек server.xml вместо строки:

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log" suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

вставить заменяющую:

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix="localhost\_access\_log" suffix=".log" rotatable="false"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

1. заменить все содержимое файла logging.properties на:

handlers = 1catalina.org.apache.juli.AsyncFileHandler, 2localhost.org.apache.juli.AsyncFileHandler, 3manager.org.apache.juli.AsyncFileHandler, java.util.logging.ConsoleHandler

.handlers = 1catalina.org.apache.juli.AsyncFileHandler, java.util.logging.ConsoleHandler

1catalina.org.apache.juli.AsyncFileHandler.level = FINE

1catalina.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs

1catalina.org.apache.juli.AsyncFileHandler.rotatable = false

1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina

2localhost.org.apache.juli.AsyncFileHandler.level = FINE

2localhost.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs

2localhost.org.apache.juli.AsyncFileHandler.rotatable = false

2localhost.org.apache.juli.AsyncFileHandler.prefix = localhost

3manager.org.apache.juli.AsyncFileHandler.level = FINE

3manager.org.apache.juli.AsyncFileHandler.directory = ${catalina.base}/logs

3manager.org.apache.juli.AsyncFileHandler.rotatable = false

3manager.org.apache.juli.AsyncFileHandler.prefix = manager

java.util.logging.ConsoleHandler.level = FINE

java.util.logging.ConsoleHandler.formatter = org.apache.juli.OneLineFormatter

org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level = INFO

org.apache.catalina.core.ContainerBase.[Catalina].[localhost].handlers = 2localhost.org.apache.juli.AsyncFileHandler

org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/manager].level = INFO

org.apache.catalina.core.ContainerBase.[Catalina].[localhost].[/manager].handlers = 3manager.org.apache.juli.AsyncFileHandler

1. создать файл настроек logrotate:

# nano /etc/logrotate.d/tomcat

с содержимым

/opt/tomcat/logs/\*.log

/opt/tomcat/logs/\*.out

/opt/tomcat/logs/\*.txt {

copytruncate

daily

rotate 30

compress

missingok

size 5M

}

Таким образом, ежедневно будет создаваться архив с журналом, который больше 5Mb, храниться будет 30 дней.

Перезапустить systemd, запустить tomcat и включить его автозапуск:

# systemctl daemon-reload

# systemctl enable tomcat

# systemctl start tomcat

* + 1. Установка КриптоПро JCP 2.0

Криптопровайдер устанавливается на каждом сервере приложений Tomcat приложения epCoreAppication. Необходимо скачать его с сайта <https://www.cryptopro.ru/products/csp/jcp/downloads>. (пример страницы для скачивания приведен на рисунке 1. Для доступа к скачиванию нужна учетная запись на сайте.

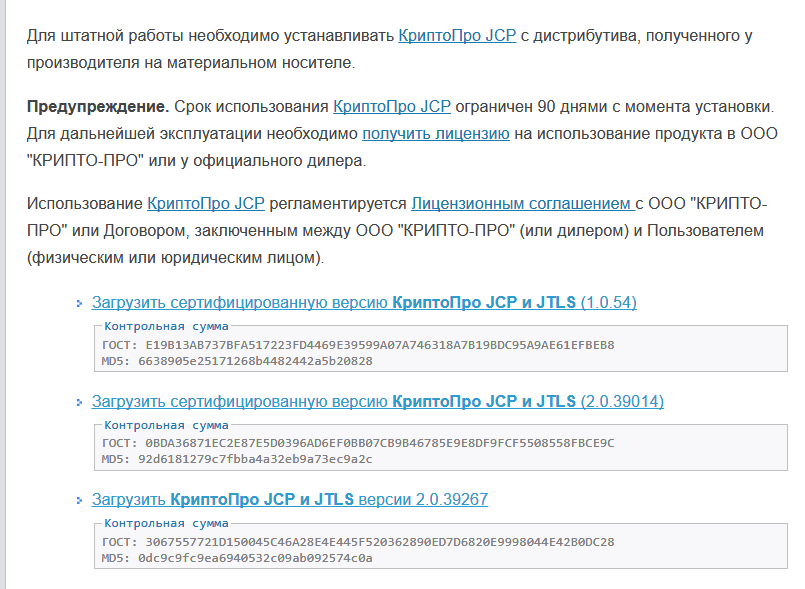


Рисунок 1 Пример страницы для скачивания дистрибутива КриптоПро JCP2.0

Скачивать сертифицированную версию КриптоПро JCP и JTLS версии 2.0

После скачивания и загрузки на сервер необходимо выполнить следующие команды:

# unzip jcp-2.0.\*.zip

# cd jcp-2.0\*

# ./setup\_console.sh /usr/lib/jvm/java-1.8.0/

Запустится мастер установки. В ходе установки, по всем запросам мастера установки, нажимать Enter.

# cp dependencies/commons-logging-\*.jar /usr/lib/jvm/java-1.8.0/jre/lib/ext/

# cp dependencies/xmlsec-\*.jar /usr/lib/jvm/java-1.8.0/jre/lib/ext/

* 1. Система обмена сообщениями RabbitMQ

Сервер RabbitMQ устанавливается на одном из серверов приложений Tomcat приложения epTransport. Для его работы необходим так же Erlang.

Создать файл:

# nano /etc/yum.repos.d/rabbitmq-erlang.repo

с содержимым:

[rabbitmq-erlang]

name=rabbitmq-erlang

baseurl=https://dl.bintray.com/rabbitmq/rpm/erlang/19/el/7

gpgcheck=0

repo\_gpgcheck=0

enabled=1

Обновить репозиторий и установить erlang:

# yum update

# yum install erlang

Установка Rabbit:

# wget https://github.com/rabbitmq/rabbitmq-server/releases/download/rabbitmq\_v3\_6\_10/rabbitmq-server-3.6.10-1.el7.noarch.rpm

# rpm --import https://www.rabbitmq.com/rabbitmq-release-signing-key.asc

# yum install rabbitmq-server-3.6.10-1.el7.noarch.rpm

Включение консоли управления и настройка RabbitMQ:

# rabbitmq-plugins enable rabbitmq\_management

Запустить rabbitmq и включить его автозапуск:

# systemctl start rabbitmq-server

# systemctl enable rabbitmq-server

Создать пользователя и пароль:

* пользователь: <имя пользователя основного приложения в сервере очередей, например, rabbit>;
* пароль: <пароль пользователя основного приложения в сервере очередей, например, password>:

# rabbitmqctl add\_user rabbit password

# rabbitmqctl set\_user\_tags rabbit administrator

# rabbitmqctl set\_permissions -p / rabbit ".\*" ".\*" ".\*"

Создать очередь:

# wget <http://127.0.0.1:15672/cli/rabbitmqadmin>

# chmod +x rabbitmqadmin

# ./rabbitmqadmin declare queue --vhost=/ name=ep.createVoucher.queue durable=true

# ./rabbitmqadmin declare queue --vhost=/ name=ep.createVoucherNumber.queue durable=true

# ./rabbitmqadmin declare queue --vhost=/ name=ep.getClassifiers.queue durable=true

# ./rabbitmqadmin declare queue --vhost=/ name=ep.getDictionary.queue durable=true

* 1. Установка приложения wkhtmltox

Приложение устанавливается на каждом сервере приложений Tomcat приложения epCoreAppication:

# yum install -y epel-release libpng libjpeg openssl icu libX11 libXext libXrender xorg-x11-fonts-Type1 xorg-x11-fonts-75dpi curl cabextract xorg-x11-font-utils fontconfig

# wget https://downloads.sourceforge.net/project/mscorefonts2/rpms/msttcore-fonts-installer-2.6-1.noarch.rpm

# rpm -i msttcore-fonts-installer\*rpm

# wget https://github.com/wkhtmltopdf/wkhtmltopdf/releases/download/0.12.4/wkhtmltox-0.12.4\_linux-generic-amd64.tar.xz

# unxz wkhtmltox-\*\_linux-generic-amd64.tar.xz

# tar -xvf wkhtmltox-\*\_linux-generic-amd64.tar

# chown -R root:root wkhtmltox

# cp -R wkhtmltox/\* /usr/

* 1. Установка сервера HAProxy

На отдельный сервер ставится балансировщик HAProxy, который будет проксировать исходя из URL на tomcat сервера с приложениями. Так же HAProxy используется как балансировщик между нодами tomcat серверов.

Схематично это отражено на рисунке 2:

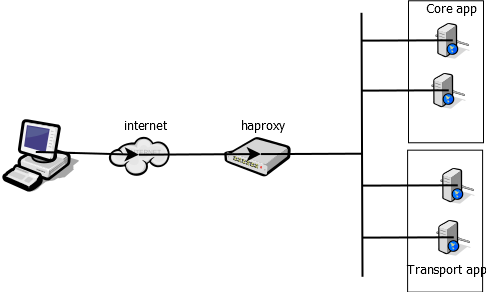


Рисунок 2 Схема установки сервера HAProxy

HAProxy проверят каждый tomcat сервер на доступность. При доступности всех серверов одного приложения haproxy пересылает запросы на tomcat сервера, выбирая их с использованием алгоритма round-robin. При недоступности одного из серверов, haproxy помечает его как недоступный и посылает запросы на другой.

# yum –y install haproxy

В файле конфигурации /etc/haproxy/haproxy.conf заменяется всё содержимое на:

global

log 127.0.0.1 local0

chroot /var/lib/haproxy

pidfile /var/run/haproxy.pid

maxconn 4000

user haproxy

group haproxy

daemon

stats socket /var/lib/haproxy/stats

defaults

mode http

log global

option httplog

option dontlognull

option http-server-close

option forwardfor except 127.0.0.0/8

option redispatch

retries 3

timeout http-request 10s

timeout queue 1m

timeout connect 10s

timeout client 1m

timeout server 1m

timeout http-keep-alive 10s

timeout check 10s

maxconn 3000

balance roundrobin

frontend http

bind \*:80

acl acl\_transport url\_beg -i /eptransportapplication

acl acl\_core url\_beg -i /epcoreapplication

use\_backend back\_transport if acl\_transport

use\_backend back\_core if acl\_core

backend back\_transport

cookie JSESSIONID prefix

appsession JSESSIONID len 52 timeout 90s request-learn prefix

option httpchk GET /epTransportApplication/ws/

http-check expect status 200

server www01 **< node1\_ip\_address\_transport\_appplication >**:8080 cookie n1 check

server www02 **< node2\_ip\_address\_transport\_appplication >**:8080 cookie n2 check

backend back\_core

cookie JSESSIONID prefix nocache

appsession JSESSIONID len 52 timeout 90s request-learn prefix

option httpchk GET /epCoreApplication/images/topHeader/topslider\_\_logo.png

http-check expect status 200

server www01 **< node1\_ip\_address\_core\_appplication >**:8080 cookie n1 check

server www02 **< node2\_ip\_address\_core\_appplication >**:8080 cookie n2 check

где node\*\_ip\_address\_\*\_application соответствующие IP адреса приложений

Включить автозапуск и запустить haproxy:

# systemctl enable haproxy

# systemctl restart haproxy

* 1. Установка сервера PostgreSQL

# yum –y install https://yum.postgresql.org/9.6/redhat/rhel-7-x86\_64/pgdg-redhat96-9.6-3.noarch.rpm

# yum -y install postgresql96-server postgresql96-contrib

# /usr/pgsql-9.6/bin/postgresql96-setup initdb

# systemctl enable postgresql-9.6

# systemctl start postgresql-9.6

Создать пользователя и базу (эти параметры понадобятся для дальнейшей сборки приложения):

* имя базы: **<имя БД основного приложения (PostgreSQL), например, tourismdb>**
* пользователь: **<имя пользователя приложения в БД основного приложения (PostgreSQL), например, tourismusr>**
* пароль: **<пароль пользователя основного приложения в сервере очередей, например, postgresql17>**

sudo -u postgres psql

CREATE DATABASE tourismdb WITH ENCODING 'UTF8' TEMPLATE=template0;

CREATE USER tourismusr WITH password 'postgresql17';

GRANT ALL privileges ON DATABASE tourismdb TO tourismusr;

для выхода команда \q

Для включения авторизации по паролю необходимо отредактировать pg\_hba.conf:

# nano /var/lib/pgsql/9.6/data/pg\_hba.conf

Найти строки:

local all all peer

host all all 127.0.0.1/32 ident

host all all ::1/128 ident

и заменить в них ident на md5

local all postgres peer

host all all 127.0.0.1/32 md5

host all all ::1/128 md5

Для настройки резервного копирования необходимо отредактировать pg\_hba.conf:

# nano /var/lib/pgsql/9.6/data/pg\_hba.conf

Найти строку:

#host replication postgres 127.0.0.1/32 ident

Раскомментировать и заменить в них ident на trust:

host replication postgres 127.0.0.1/32 trust

В файле postgresql.conf найти, раскомментировать и указать следующие параметры:

* wal\_level = archive;
* max\_wal\_senders = 1

# sed -i -e "s/^#wal\_level = minimal/wal\_level = archive/" /var/lib/pgsql/9.6/data/postgresql.conf

# sed -i -e "s/^#max\_wal\_senders = 0/max\_wal\_senders = 1/" /var/lib/pgsql/9.6/data/postgresql.conf

В целях увеличения максимального числа одновременных подключений к серверу БД, в файле postgresql.conf установить значение параметра **max\_connections** равным **250** (100 соединений для каждого сервера приложений кластера + 50 резервных соединений для возможности использования различных административных утилит, работающих напрямую с сервером БД), а также значение параметра **shared\_buffers** равным **960MB**.

Запуск сервера:

# systemctl start postgresql-9.6

* 1. Установка сервера Cassandra

Добавить вручную репозиторий DataStax. Для этого в файле:

# nano /etc/yum.repos.d/datastax.repo

указать:

[datastax]

name = DataStax Repo for Apache Cassandra

baseurl = http://rpm.datastax.com/community

enabled = 1

gpgcheck = 0

Установить Cassandra:

# yum -y install dsc30

Для создания кластера необходимо изменить настройки. Для этого на каждой ноде сервера Cassandra в файле /etc/cassandra/default.conf/cassandra.yaml указать:

authenticator: PasswordAuthenticator

…

seed\_provider:

- class\_name: org.apache.cassandra.locator.SimpleSeedProvider

parameters:

- seeds: "server\_ip,server\_ip\_2,…server\_ip\_n"

…

listen\_address: server\_ip

…

rpc\_address: server\_ip

…

endpoint\_snitch: GossipingPropertyFileSnitch

где server\_ip – ip адрес этого сервера. В параметре seeds перечислены через запятую все ip адреса нод кластера Cassandra. На всех нодах различающимися параметрами должны быть только listen\_address и rpc\_address.

Создать systemd unit для управления и автозапуска cassandra:

# nano /etc/systemd/system/cassandra.service

С содержимым:

[Unit]

Description=Cassandra

After=network.target

[Service]

PIDFile=/var/lib/cassandra/data/cassandra.pid

User=cassandra

Group=cassandra

ExecStart=/usr/sbin/cassandra -p /var/lib/cassandra/data/cassandra.pid

StandardOutput=journal

StandardError=journal

LimitNOFILE=100000

LimitMEMLOCK=infinity

LimitNPROC=32768

LimitAS=infinity

Restart=always

SuccessExitStatus=143

[Install]

WantedBy=multi-user.target

Запустить Cassandra и включить его автозапуск:

# systemctl enable cassandra

# systemctl start cassandra

# Сборка приложений

Сборка производится на отдельном компьютере.

* 1. Подготовка системы
     1. Установка Java

# yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-devel

Установка переменной JAVA\_HOME:

Добавить в конец файла ~/.bash\_profile

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0/

Выполнить

# source ~/.bash\_profile

Для проверки установки:

# java –version

* + 1. Установка Maven

# cd ~

# wget http://apache-mirror.rbc.ru/pub/apache/maven/maven-3/3.5.0/binaries/apache-maven-3.5.0-bin.tar.gz

# tar -xvzpf apache-maven-3.5.0-bin.tar.gz

# mkdir -p /opt/maven/3.5.0

# mv apache-maven-3.5.0/\* /opt/maven/3.5.0

Установка переменных окружения. Добавить в конец файла ~/.bash\_profile:

export M2\_HOME=/opt/maven/3.5.0

export M2=$M2\_HOME/bin

export PATH=$M2:$PATH

Выполнить:

# source ~/.bash\_profile

Для проверки установки:

# mvn -version

* 1. Подготовка к сборке

Для первого деплоя приложения epCoreApplication необходимо скопировать на Tomcat сервера epCoreApplication в директорию /opt из директории дистрибутива epCoreApp/src/main/resources файлы certStore и RST-ESIA.jks (если необходима интеграция с ЕСИА).

Скопировать файл параметров application-dev.yml из директории src/main/resources/config на каждый Tomcat сервер epCoreApplication в директорию /opt/tomcat/lib, указав имя этому файлу application-profile.yml. Так же на каждом Tomcat сервере epCoreApplication в конфигурационном файле /opt/tomcat/conf/catalina.properties добавить параметр spring.profiles.active=profile. Аналогично, необходимо проделать эти операции с файлом параметров для приложения epTransport – распространить application-dev.yml на Tomcat сервера и добавить параметр.

Необходимо указать параметры в файлах application-profile.yml для epCoreApplication и epTransport.

Параметры epCoreApplication:

* application-url url = "http://**<ip\_address\_haproxy>**/epCoreApplication";
* application-url public-url= "http://**<ip\_address\_haproxy>**/epCoreApplication";
* spring jpa properties hibernate search default indexBase= "/opt/apache-tomcat/webapps/epCoreApplication" # директория для хранения индексов поиска;
* epCore grpc port = 8030 # порт, на котором будет запущен gRPC сервис основного приложения;
* epCore ep-transport grpc host = "**< ip\_address\_epTransportApplication >**" # IP адрес сервера основного приложения;
* epCore ep-transport grpc port = 8040 # порт, на котором будет запущен gRPC сервис транспортного приложения;
* epCore datasource host = "**< ip\_address\_postgresql >**" #адрес сервера БД основного приложения (PostgreSQL);
* epCore datasource name = "tourismdb" #имя БД основного приложения (PostgreSQL);
* epCore datasource username = "tourismusr" #имя пользователя приложения в БД основного приложения (PostgreSQL);
* epCore datasource password = "postgresql17" #пароль пользователя приложения в БД основного приложения (PostgreSQL);
* spring rabbitmq addresses = "**< ip\_address\_rabbit >**" #адрес сервера очередей;
* spring rabbitmq username = "rabbit" #имя пользователя основного приложения в сервере очередей;
* spring rabbitmq password = "password" #пароль пользователя основного приложения в сервере очередей;
* spring mail host= "**< ip\_address\_smpt >**" #адрес почтового сервера;
* spring mail port = 25 #smtp порт. 25 - если не используется шифрование, 587 - с шифрованием;
* epCore mail from = "example@example.com" #адрес, от кого письмо;
* spring mail properties.mail smtp.auth = false # false/true значения. Необходима или нет авторизация на smtp сервере;
* spring mail username = "" #имя пользователя для авторизации на smtp. Не задействован, если mail\_auth – false;
* spring mail password = "" #пароль пользователя;
* spring mail properties.mail smtp.starttls.enable = false #false/true значения. true - если на почтовом сервере используется шифрование;
* spring mail properties.mail smtp.ssl.trust = "**< ip\_address\_smpt >**" #адрес почтового сервера, с доверенным сертификатом. Не задействован, если smtp.starttls.enable = false;
* epCore esia enabled = true #false/true значения. Использование интеграции с ЕСИА;
* epCore esia signer keystore-path = "/opt/RST-ESIA.jks" # путь до ключевого контейнера;
* epCore esia signer keystore-password = "RST-ESIA" # пароль ключевого контейнера;
* epCore esia signer key-alias = "RST-ESIA" #наименование ключевого контейнера.

Параметры epTransport:

* epTransport rabbitmq addresses = "**< ip\_address\_rabbit >**" #адрес сервера очередей;
* epTransport rabbitmq username = "rabbit" #имя пользователя основного приложения в сервере очередей;
* epTransport rabbitmq password = "password" #пароль пользователя основного приложения в сервере очередей;
* epTransport cassandra contact-points = "**< ip\_address\_cassandra >**" #адрес для клиентских подключений (rpc\_address) к БД Cassandra.
  1. Сборка

Необходимо собрать первым epTransportApp:

mvn -B -U exec:exec clean package –P dev -Dmaven.test.skip=true --file epTransportApp/pom.xml

Полученный после сборки epTransportApp/target/epTransportApplication-0.0.1-SNAPSHOT.war необходимо распространить на каждый сервер приложений Tomcat приложения epTransportApplication в директорию /opt/tomcat/webapps.

Для проверки необходимо открыть в браузере http://**< ip\_address\_haproxy >**/ epTransportApplication/ws/EPMessageExchangeWS?wsdl

epCoreAppication соберется только после успешного запуска epTransportApplication.

mvn -B -U exec:exec clean package –P dev --Dmaven.test.skip=true --Dtransporthost=**< ip\_address\_haproxy >** --Dtransportport=80 --file epCoreApp/pom.xml

где:

* transporthost – адрес сервера транспортного приложения (для получения wsdl);
* transportport – порт для обращения к транспортному приложению (для получения wsdl).

Полученный после сборки epCoreApp/target/epCoreApplication-0.0.1-SNAPSHOT.war необходимо распространить на каждый сервер приложений Tomcat приложения epCoreApplication в директорию /opt/tomcat/webapps.

Для проверки необходимо открыть в браузере http://**< ip\_address\_haproxy >**/ epCoreApplication.

# Список рисунков

[Рисунок 1 Пример страницы для скачивания дистрибутива КриптоПро JCP2.0 9](#_Toc497912532)

[Рисунок 2 Схема установки сервера HAProxy 11](#_Toc497912533)