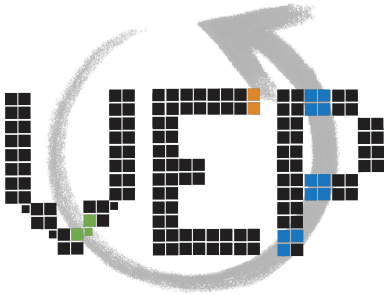
Virtual Execution Platform

INRIA Rennes

VEP 2.1 : Installation Guide

Release v2.1



Release Date: 26 September 2013

Project co-funded by the European Commission within the Seventh Framework Programme

Contents

Chapter 1

Requirements

1.1 System Configuration

A classic VEP set-up is showed in 1.1

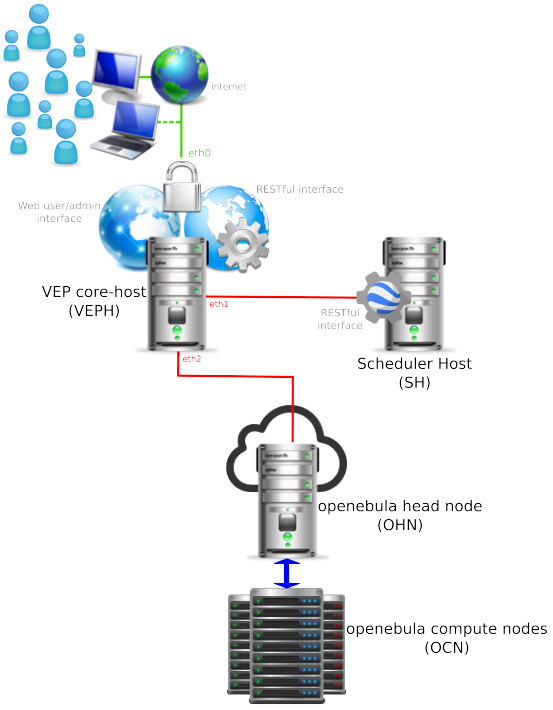


Figure 1.1: VEP Default Sytem Configuration

VEP is a Java Application. It was developed using openjdk-6, so it should be

executed using JAVA version 6. VEP is made to run up of a OpenNebula IaaS. The supported OpenNebula versions are 2.2.1 and 3.6, but we reccomend to use the 3.6. In the figure 1.1 the System Configuration is made of at least 4 nodes (the configuration can be also set-up to run everything in a single node).

The VEPH (VEP core-host) must have a reacheble ip-address for the users who want to connect to it. It has to be connected as well to the SH (Scheduler Host) and to the one head node (OHN) to deploy and control VMs. The best option is to use different NICs that allow to separate networks and connections to keep the system safer. The configuration that will be explained in the installation session refers to Fig 1.1, but the scheduler is running in the same node that the VEP-core application. On the external ip, VEPH offers a RESTful interface and a web-user interface where it can be configured. The SH offer as well a RESTful interface to communicate with VEP, but for the SH requires the GLASSFISH Application Server and a Mysql Server running. Let's make a list for each host of the required applications.

VEP Core Requirements

In the VEPH has to be installed:

- a JAVA RUN ENVIROMENT, the version 6 is the supported one.

Scheduler Requirements

The VEP scheduler is an external module that has to installed separatly and it runs indipendetly. The software requirements are:

- JAVA JDK 6
- GLASSFISH Application Server 3.0.1 installed
- Mysql Server running

Opennbula Requirements

The supported Opennebula versions are 2.2 and 3.6. It is possible to retrive and to download them from the opennbula website at `http://dev.opennebula.org/projects/opennebula/files`

Chapter 2

Installation

2.1 Installation steps

This is step-by-step installation guide of the VEP. In case of any problems, please refer to the documentation on the web site https://project.inria.fr/vep/ or write to the mailing list. In order to install the complete system, you need to set up three different parts:

- IaaS nodes (the head node of the ONE cluster and the compute nodes),
- scheduler node,
- vep core node.

The VEP archive can be downloaded from VEPDOWNLADURL. After you downloaded you can depackage it at <VEPFolder> .

Opennebula Installation

To install Opennebula please refer to the software proper installation guide. As the OHN and the OCN are runnning please keep the ip of the head node and the admin username password and id.

Scheduler Installation

In the <VEPFolder> folder there is scheduler.war. With it there is also dump-SQL file (dumpfilename.sql) to set-up the Database. Before start the scheduler service, it is necessary to configure the Mysql Server creating the user and the DB.

```
mysql -u root -p -e "create database schedulerEmpty"
mysql -u root -p schedulerEmpty -e "grant all privileges on \
schedulerEmpty.* TO 'scheduler'@'localhost' identified\
by 'password'"
mysql -u root -p -D schedulerEmpty< dumpfilename.sql
```

As stated in paragraph 1.1, Glassfish has to be installed and and running on the machine (you can download it from http://download.java.net/glassfish/3.0.1/release/glassfish-3.0.1-unix.sh and run the script after you have installed the JDK). If <GlasshFish>is the path where you installed the application server, to add the war you have to: The reccomended version of glassfish is 3.0.1 and it requires the JDFK 6.

```
$ nohup <GlasshFish>/glassfish/bin/startserv > /dev/null 2>&1 &
$ cd <GlassFish>/bin/
$ ./asadmin deploy <warPath>/scheduler.war
$ <GlasshFish>/glassfish/bin/stopserv
```

The Scheduler's Database is empty it will be filled after all the VEP-Configuration.
To check if the schedeluer is working properly you can type http://¡schedulerIP¿:¡schedulerPort¿
in your browser and you should get as answer "[]". The most common problem that you can have, it is that the glassfish server miss the mysql-connector library. The jar library is in the vep package you downloaded and you can add it to the glassfish library classpath.

VEP-Core Installation

Let's move with the component that currently needs most configuration, VEP-core. The software, as stated can be can be downloaded from VEPDOWNLADURL. Then you can decompress the file. There is two application that you have to execute to run properly VEP 2.0:

- vep2.0.jar, the VEP core application
- Move.jar , the Move Server

the Move Server has to run on the same host that VEP. It will download the image specified in the OVF and it will store it in the appropiate folder where VEP can use it to upload on the One Head Node.

As you start the Move Server typing

```
java -jar Move.jar
```

It will create a folder on your home directory with a configuration file inside and it exits. You have to set the configuration file:

```
cd ~/.moveServerVep
vi transferModuleVep.propreties
```

The file should look like

```
port=10556
user=admin
password=pass1234
temporary_memory=/tmp/
```

That is an example of a configuration file. You have to choose the port where the server will listen, the password and the username. Then you have to choose a temporary folder where you want to put the download images.

As you restart your Move Server the new configuration will be used. We suggest to run the Move Server using nohup

```
nohup java -jar Move.jar > /dev/null 2>&1 &
```

Now let's move to the vep-core application.VEP offers a secure connection and all its services are provided on https. At first, in order to create secure communication links with the clients and to perform mutual authentication, VEP REST server requires a signed server X.509 certificate contained in a Java keystore file (.jks). The task is straightforward if you have openvpn package installed in your system. We will further restrict acceptance of client certificates that are signed by trusted CAs only. Below are the steps for properly creating the server certificate in a java key store file, and configuring the global Java trust store to permit client certificates from trusted CAs to be accepted.

- If not already installed install openvpn package from your linux's package repository
- locate easy-rsa directory created when openvpn was installed

```
$ sudo updatedb
$ locate easy-rsa
```

Copy this easy-rsa folder into your home directory.
Change directory to the subfolder 2.0 inside easy-rsa directory:

```
$ cd easy-rsa
$ cd 2.0
```

Edit the vars file, put correct value for country, province, city, org, email parameters in it, an example vars file could look like this:

```
# easy-rsa parameter settings

# NOTE: If you installed from an RPM,
# don't edit this file in place in
# /usr/share/openvpn/easy-rsa --
# instead, you should copy the whole
# easy-rsa directory to another location
# (such as /etc/openvpn) so that your
# edits will not be wiped out by a future
# OpenVPN package upgrade.
```

```
# This variable should point to
# the top level of the easy-rsa
# tree.
export EASY_RSA="`pwd`"

#
# This variable should point to
# the requested executables
#
export OPENSSL="openssl"
export PKCS11TOOL="pkcs11-tool"
export GREP="grep"


# This variable should point to
# the openssl.cnf file included
# with easy-rsa.
export KEY_CONFIG=`$EASY_RSA/whichopensslcnf $EASY_RSA`

# Edit this variable to point to
# your soon-to-be-created key
# directory.
#
# WARNING: clean-all will do
# a rm -rf on this directory
# so make sure you define
# it correctly!
export KEY_DIR="$EASY_RSA/keys"

# Issue rm -rf warning
echo NOTE: If you run ./clean-all, I will be doing a rm -rf on $KEY_DIR

# PKCS11 fixes
export PKCS11_MODULE_PATH="dummy"
export PKCS11_PIN="dummy"
```

```
# Increase this to 2048 if you
# are paranoid.  This will slow
# down TLS negotiation performance
# as well as the one-time DH parms
# generation process.
export KEY_SIZE=1024

# In how many days should the root CA key expire?
export CA_EXPIRE=3650

# In how many days should certificates expire?
export KEY_EXPIRE=3650

# These are the default values for fields
# which will be placed in the certificate.
# Don't leave any of these fields blank.
export KEY_COUNTRY="FR"
export KEY_PROVINCE="BRETAGNE"
export KEY_CITY="Rennes"
export KEY_ORG="INRIA-Myriads"
export KEY_EMAIL="piyush.harsh@inria.fr"
```

Then, execute the following commands while in the 2.0 folder, it will create your own CA certificate, you should remember all the passwords you provide in the subsequent steps

```
$ source ./vars
$ ./clean-all
$ ./build-ca
```

next we will create the VEP REST server certificate using the just created CA for signing the certificate. Replace the ¡server-name¿ with the domain name of the host where VEP will run. DO provide a valid password when prompted, this password will be used later while creating the Java key store file.

```
$ ./build-key-server <server-name>
```

Change directory to keys inside the 2.0 folder, all generated certificates and private keys are stored there

```
$ cd keys
```

Next combine the server's private key and the certificate into a .pfx file, replace <server-name> with what you used in the step above

```
$ openssl pkcs12 -export -inkey <server-name>.key -in \
<server-name>.crt -out <server-name>.pfx -name default
```

Next create the REST server Java key store (.jks) file to be used with VEP software. It is important to use the same password for the key store as while creating the server certificate. The password for the key-store file is provided using the -destkeypass switch as shown below This will be the Keystore to specify in the vep configuration file.

```
$ keytool -importkeystore -srckeystore <server-name>.pfx \
-srcstoretype pkcs12 -destkeystore VEPRestKeyStore.jks \
-srcalias default -destalias <your-domain-name> -destkeypass \
<same-as-when-creating-the-server-key>
```

Import the CA certificate into your global Java trust store, this operation must be performed as root user. It is important to locate your default Java JRE if you have multiple JREs installed

```
$ java -version
```

Once you have determined the default JRE, locate the jre/lib/security folder in your system :

```
$ locate jre/lib/security
```

Change directory to the correct jre/lib/security corresponding to your default Java JRE, next as root perform

```
# keytool -import -alias <your-CA-name> -file <ca.crt> \
-keystore cacerts -storepass changeit
```

¡ca.crt¿ is the CA's certificate file that was generated by openvpn in the beginning and can be found in easy-rsa/2.0/keys folder. By default the cacerts trust-store has password changeit, if you have changed it then replace it with the changed password. Generating certificates for end-users using your generated CA

Now that you have generated the VEP REST server's Java key store (.jks) file and have properly configured your global Java trust-store to accept certificates generated by your CA, let us see how you can generate end-user certificates to distribute to your VEP clients.

change directory to easy-rsa/2.0 folder in your home directory create a client certificate for the user account ¡username¿ - when prompted for password, leave it blank

```
$ ./build-key <username>
```

This user will have an account with the VEP software with the exact same <username>. Change directory to keys as the newly created client certificates are stored there:

```
$ cd keys
```

Next convert the client's certificate into a .pfx file for a standard browser (Chrome, Firefox, etc.) import

```
$ openssl pkcs12 -export -out <username>.pfx \
-inkey <username>.key -in <username>.crt \
-certfile <ca.crt>
```

If the step above asks you to enter CA's private key's password, then provide the appropriate password that you might have used while creating the CA certificate (if password was left empty during the CA certificate creation step,

then just press the enter key). Distribute the ¡username¿.pfx certificate file to your end user, she must configure her browsers with this key before accessing the VEP's services.

As we downloaded and decompressed VEP (we suppose that is in <VEPpath>/vep2.0.jar), we can run it, just for the fist time, to let it to create all the configuratio files that it need using the option default.

```
java -jar  <VEPpath>/vep2.0.jar -d
```

VEP can be executed with many paramenters.The valid runtime options that can be provided are detailed next and can be displayed by using the –help option

```
usage: java -jar VEPController.jar [-d] [-h] [-l <arg>] [-p <arg>] [-t] [-v <arg>]
Contrail Virtual Execution Platform Controller Module
-d,--default        start with a default VEP properties file, you
                must change the defaults to the right values manually
-h,--help                       usage help
-l,--log-properties <arg>       path to the VEP logger properties file
-p,--vep-properties <arg>       path to the VEP properties file
-s,--supress-term-log           supress logger output to terminal
-t,--no-gui                     terminal only (no GUI) mode
-v,--log-level <arg>            log verbosity level, (0 = off, 1 = fatal, 2 =
                                   error, 3 = warn, 4 = info, 5 = debug, 6 =
                                   everything)
```

Now the VEP should have created the configuration file. A configuration folder with the configuration file should be under /.vep/vep.properties

```
#Author: VEP Team
#Thu Sep 05 17:37:52 CEST 2013
scheduler.port=8080
rest.restHTTPSPort=8183
webuser-interface.path=/home/fgaudenz/.vep//webuserInterface
user.timeout=600
rest.keystorepass=pass1234
```

```
cli.port=10555
caservice.uri=https\://one-test.contrail.rl.ac.uk\:8443/ca/delegateduser
pdp.endpoint=http\://146.48.96.75\:2000/contrailPDPwebApplication/contrailPDPsoap
caservice.storepass=changeme
mysql.ip=127.0.0.1
copyserver.admin=admin
mysql.dbname=vepdb2
vepdb.choice=sqlite
copyserver.port=10556
caservice.keystore=
mysql.user=vepuser
copyserver.ip=127.0.0.1
mysql.pass=contrail
sqlite.db=vep.db
veplog.size=1024
scheduler.url=http://127.0.0.1
pdp.use=false
copyserver.password=pass1234
webuser-interface.defaultHost=localhost
rest.keystore=/etc/openvpn/easy-rsa/2.0/keys/VEPRestKeyStore.jks
webuser-interface.port=8000
vep.scratch=/tmp/
veplog.file=vep.log
rest.restHTTPPort=10500
rest.keypass=pass1234
contrail.cluster=
mysql.port=3306
```

We suggest to use sqlite (vepdb.choice=sqlite), otherwise you have to install and configure the Mysql Server. The copyserver values refer to the Move Server. Rember to set up properly the keystore with the right path and password. The rest.keystore refers to the key store crated before. This shouldn't be accessible by the user who starts VEP: if he is not a sudouser you should copy in a reacheable folder and change the key store's permission.Rember to set up properly the keystore with the right path and password The file vep2.db is included in the archive you downloaded before, so you have just to put the right reference (sqlite.db=<here>). Also, remember to set the rest.restHTTPSPort

adn the scheduler key-values. Please use only full paths in the configuration file.

Now you can restart the VEP and check in the log that everything is running properly (you shouldn't have any ERROR message).

```
java -jar  <VEPpath>/vep2.0.jar
```

Now that VEP is running you can connect to the admin page and configure your VEP to work with your IaaS.

https://<vepip>:8183/admin/

As you type the admin username (admin) and password (pass1234) then you can click "Edit Configuration". As you are in the admin page it's time to set up the VEP database and the VEP cloud datacenter-layout.

Click on "Manage Datacenter". Then, you can add datacenter,cluster,rack and L2switch.You must enter your datacenter information using the VEP web admin interface (Fig 2.1). Providing correct information helps VEP honor the geographical placement restrictions (if any) of application's virtual machines. Once you have entered necessary information, press Submit button to store the data. Once you have added your datacenter information, add the Cluster (or Clusters if there are multiple) followed by the Rack (or Racks if there are multiple) information.

As you added Datacenter(s),rack(s) and cluster(s) you should be able to add the host. Before this operation it is necessary to allow VEP to connect to the cloud IaaS. Click on "cancel and go back" amd then on "Manage Cloud Parameters" and you will see the page to configure the Cloud connection (Fig 2.2). After it is necessary to add the openebula admin to the user management. Click on "Manage Account", there is already an admin user sotred, you have to change it as showed in Fig 2.3).

If the connection is set up properly when you will add hosts or network the fields will be automatically filled (Fig 2.4) . If not, you did something wrong.

As you configure the cloud parameters (connection, host(s) and netowrk) it is time to create the handlers. Remembers that a network handler has to be bound to a network and t is mandatory to have at least one vnethandler that is called "vin". As you finish your configuration, please syncronize the schedeluer running VEP as

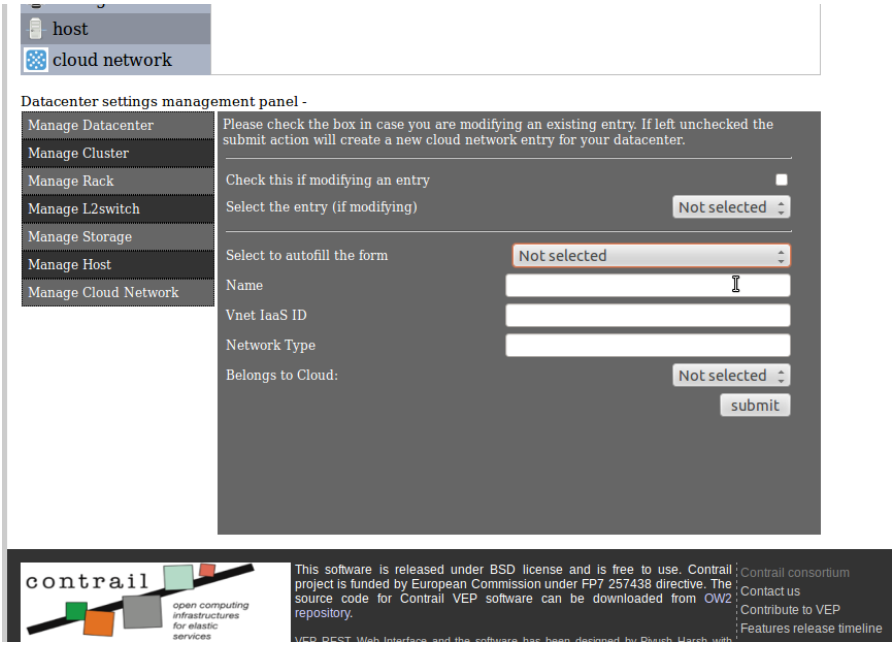


Figure 2.1: Manage Datacenter Page

```
java -jar  <VEPpath>/vep2.0.jar -u
```

Now VEP is ready to work! Enjoy the easiest-cloud-management-tool!

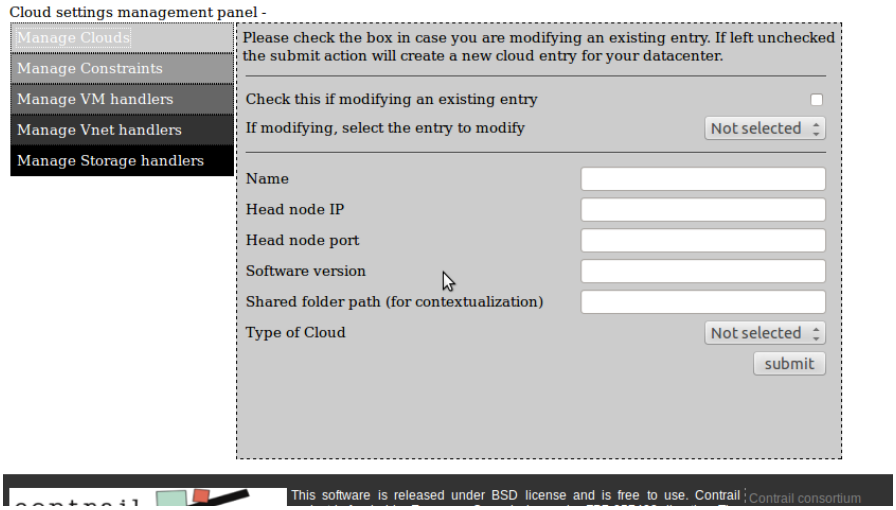


Figure 2.2: Manage Cloud Page

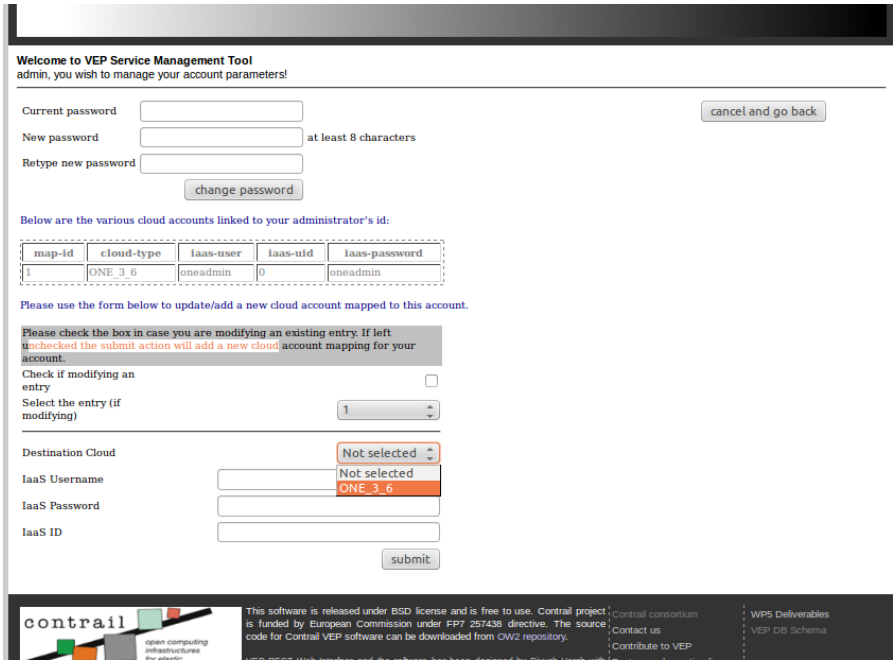


Figure 2.3: Manage Account - Admin ONE User

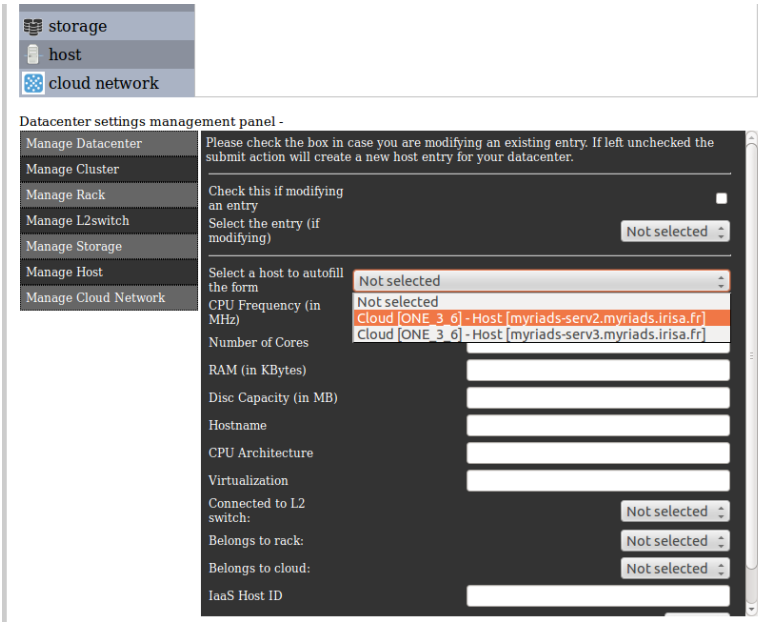


Figure 2.4: Manage Cloud Parameters Page - Host

Chapter 3

Running VEP

3.1 Script Section

If you followed exactly the guide now you have the VEP-core application, the Move Server and the scheduler running on the same host. In the archive you downloaded, there is two scripts that allow you to start and stop all the VEP's services easily.

There are:

- vep-start.sh
- vep-kill.sh

We have to edit the two file to set up the right path for the applications. With your favourite text editor open vep-start.sh

```
#!/bin/bash

vep=vep2.0.jar
moveServer=moveServer/Move.jar
scheduler= ~/glassfishv3/glassfish/bin/startserv

if [ $# -eq 1 ]
then
        if [ "$1" -eq "$1" >& /dev/null ]; then
```

```
                logLevel=$1
        else
                echo "Usage: `basename $0` {logLevel}"
                exit 0
        fi
else
        logLevel=6
fi

# Kill existing VEP and moveServer instances
pid=`ps x | grep '[j]ava -jar vep' | awk '{print $1}'`
if [ $pid ]; then
        kill $pid > /dev/null 2>&1
fi
pid=`ps x | grep '[j]ava -jar Move.jar' | awk '{print $1}'`
if [ $pid ]; then
        kill $pid > /dev/null 2>&1
fi

# Start VEP in background
nohup $scheduler > ~/vep2_0/log/scheduler.log 2>&1 &
nohup java -jar $moveServer > /dev/null 2>&1 &
nohup java -jar $vep -v logLevel -s > /dev/null 2>&1 &
```

You have to change the vep, moveServer and scheduler variable putting there the right paths.

Now edit the vep-kill.sh script as we did for vep-start.sh

```
 #!/bin/bash
scheduler=~/glassfishv3/glassfish/bin/stopserv
# Kill existing VEP instances
pid=`ps x | grep '[j]ava -jar vep' | awk '{print $1}'`
if [ $pid ]; then
        kill $pid > /dev/null 2>&1
fi
```

```
pid=‘ps x | grep ’[j]ava -jar Move.jar’ | awk ’{print $1}’‘
if [ $pid ]; then
        kill $pid > /dev/null 2>&1
fi
$scheduler
```