Backup and Restore of RHEV-M

PREPARED FOR - Volkswagen IT Group Cloud

Adrian Bradshaw

Version 1.3.5

Table of Contents

1. History and Revisions

Version	Date	Authors	Changes
0.1		Adrian Bradshaw adrian@redhat.com	Initial Draft
0.8		Adrian Bradshaw adrian@redhat.com	added RHEV-M restore
1.0		Eike Holtz (Ben Haubeck)	added VM restore incl. FS
1.1		Ben Haubeck bhaubeck@redhat.com	minor changes
1.2		Ben Haubeck bhaubeck@redhat.com	updated LDAP-Backup
1.3.0	23.10.2015	Adrian Bradshaw adrian@redhat.com	Updates & LDAP packages added
1.3.5	29.10.2015	Adrian Bradshaw adrian@redhat.com	Minor corrections & addition LDAP step removed

2. Preface

2.1. Confidentiality, Copyright, and Disclaimer

2.2. Additional Background and Related Documents

This document also references additional information that can be found on Red Hat's documentation site at https://access.redhat.com/knowledge/docs/ and specifically at https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.5/

Documents specific to products covered in this solution include the following Guides

- RHEV 3.5 Installation Guide
- RHEV 3.5 Administration Guide
- RHEV 3.5 User Guide
- RHEV 3.5 Technical Guide

Additional information can be found on the RHEV upstream projects website (oVirt):

http://www.ovirt.org/Documentation

2.3. Terminology

Some of the acronyms using in this document are included in the table below

Table 1. Terminology Table

Term	Definition
RHEV	Red Hat Enterprise Virtualisation
RHEV-M	Red Hat Enterprise Virtualisation Manager
RHEL-H	Red Hat Enterprise Linux Hypervisor

3. RHEV Backup & Restore

3.1. Summary

This document describes the the backup and restore procedures for RHEV-M.

3.2. Background

RHEV-M contains a fully featured backup tool, which not only backs up the database(s) required but also all of the individual configuration files and certificates that are needed in order to do a complete restore from a fresh install of the operating system. This has been scheduled to run automatically at regular intervals, using cron.

There are two types of backup and restore that are possible. The first one (described above) uses the built in engine-backup tool. This will be covered first.

The second type of backup, that VW specifically requested to have available, was a backup of the “VM shells”, meaning that we should backup all the data about the VM (Memory, CPU, NICs, Disks) etc but not the data in those VMs, as this will be backed up via TPM. The idea behind this is that it is possible, if a VM is lost or accidentally deleted/corrupted, to recover the bare VM shell and restore the VM to full functionality by a combination of “Relax & Recover” and a TSM restore. This will be covered second.

4. Full Backup using engine-backup

As mentioned above, Red Hat Enterprise Virtualisation includes a fully featured backup tool, that not only backup up the required database (or databases, if DWH is also installed) but also takes care of including all config files and certificates as well.

The syntax of the tool is as follows

```
# engine-backup -h
engine-backup: backup and restore ovirt-engine environment
USAGE:
    /usr/bin/engine-backup [--mode=MODE] [--scope=SCOPE] [--file=FILE] [--log=FILE]
 MODE is one of the following:
    backup                          backup system into FILE
    restore                         restore system from FILE
 SCOPE is one of the following:
    all                             complete backup/restore (default)
    files                           files only
    db                              engine database only
    dwhdb                           dwh database only
    reportsdb                       reports database only
 --file=FILE                        file to use during backup or restore
 --log=FILE                         log file to use
...
```

An example command for doing a full backup (scope=all) is below

```
# engine-backup --mode=backup --scope=all --file=/path/to/backup/file --log=/path/to/log
```

4.1. Cron Jobs

Currently there are two cron jobs that initiate a full backup once a day and once an hour, keeping the previous 14 versions of each.

The cron jobs are below and the actual script is in the appendix.

Listing 1. DAILY

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
0 3 * * * root /usr/bin/rhevm-backup.sh daily 14 /appl/rhevm/pgsql/engine-backup >/dev/null
```

Listing 2. HOURLY

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
57 * * * * root /usr/bin/rhevm-backup.sh  hourly 14 /appl/rhevm/pgsql/engine-backup >/dev/null
```

5. Full Restore using engine-backup

The restore process involves more steps, but is quite straightforward so long as you follow the process carefully. The backup utility will restore the three databases and all the required configuration files and certificates. The process should take about ten or fifteen minutes from a freshly installed OS

5.1. Overview of Restore Process

To restore a RHEV-M from scratch, start off with a fresh, fully updated RHEL 6.6 VM and copy the required RHEV-M backup file to it. (This could be automated if we use the cold-backup VM as the destination)

- Change the hostname / IP address
    - In b2x only, populate the hosts file with the details of all HVs (as there is no DNS in this zone)
- Install RHEV and the Data Warehouse Components
- Initiate the postgres database
- Switch to the postgres user to create three users and three databases
- Switch back to root and edit the postgres access list file and restart postgres
- Run the engine-back utility in restore mode
- Small postres update
- Run engine-setup
- Additional steps to restore customized settings (LDAP)

5.2. Restore Process Detailed

1. Change the hostname / IP address / (and hosts file if b2x) Make sure the original RHEV-M is either down or has its network interfaces shutdown via the VMware UI
    - Change the frontend & backend IP address
    - Change the hostanme
    - If its B2X (so no DNS) be sure to update the hosts file to contain an entry for all HVs
        - (B2X) Make sure there is also an entry for the LDAP server (10.252.52.34   uxldapclb.wob.sec.vw.vwg)
2. Install RHEV and the Data Warehouse (reporting) Components

    ```
    # yum install -y rhevm rhevm-reports-setup rhevm-dwh-setup ovirt-engine-extension-aaa-ldap unboundid-ldapsdk
    ```

3. Initiate the postgres database

    ```
    # service postgresql initdb
    # service postgresql start
    # chkconfig postgresql on
    ```

4. Switch to the postgres user to create three users and three databases

```
# su postgres
bash-4.1$ psql
Type "help" for help.

postgres=# create role engine with login encrypted password 'redhat01';
CREATE ROLE
postgres=# create role ovirt_engine_reports with login encrypted password 'redhat01';
CREATE ROLE
postgres=# create role ovirt_engine_history with login encrypted password 'redhat01';
CREATE ROLE
postgres=# create database engine owner engine template template0 encoding 'UTF8' lc_collate 'en_US.UTF-8' lc_ctype 'en_US.UTF-8';
CREATE DATABASE
postgres=# create database ovirt_engine_history owner ovirt_engine_history template template0 encoding 'UTF8' lc_collate 'en_US.UTF-8'
lc_ctype 'en_US.UTF-8';
CREATE DATABASE
postgres=# create database ovirt_engine_reports owner ovirt_engine_reports template template0 encoding 'UTF8' lc_collate 'en_US.UTF-8'
lc_ctype 'en_US.UTF-8';
CREATE DATABASE
postgres=# \q
bash-4.1$ exit
#
```

4. Switch back to root and edit the postgres access list file Edit the end of the /var/lib/pgsql/data/pg_hba.conf file, FROM this

```
...
# TYPE  DATABASE    USER        CIDR-ADDRESS        METHOD

# "local" is for Unix domain socket connections only
local   all         all                             ident
# IPv4 local connections:
host    all         all         127.0.0.1/32        ident
# IPv6 local connections:
host    all         all         ::1/128             ident
```

TO this (be sure to comment out the two existing lines)

```
...
# TYPE  DATABASE    USER        CIDR-ADDRESS        METHOD

# "local" is for Unix domain socket connections only
local   all         all                             ident
# IPv4 local connections:
#host   all         all         127.0.0.1/32        ident
host    engine  engine  0.0.0.0/0  md5
host    ovirt_engine_history    ovirt_engine_history    0.0.0.0/0  md5
host    ovirt_engine_reports    ovirt_engine_reports    0.0.0.0/0  md5
# IPv6 local connections:
#host   all         all         ::1/128             ident
host    engine  engine  ::0/0   md5
host    ovirt_engine_history    ovirt_engine_history    ::0/0     md5
host    ovirt_engine_reports    ovirt_engine_reports    ::0/0     md5
```

and restart the service

```
# service postgresql restart
```

5. Run the engine-backup utility in restore mode

```
 engine-backup --mode=restore  --file=/path/to/backup.file --log=/path/to/logfile --change-db-credentials --db-host=localhost --db
-name=engine --db-user=engine --db-password=redhat01 --change-reports-db-credentials --reports-db-host=localhost --reports-db
-name=ovirt_engine_reports --reports-db-user=ovirt_engine_reports --reports-db-password=redhat01 --change-dwh-db-credentials --dwh-db
-host=localhost --dwh-db-name=ovirt_engine_history --dwh-db-user=ovirt_engine_history --dwh-db-password=redhat01

Preparing to restore:
- Setting credentials for Engine database 'engine'
- Setting credentials for DWH database 'ovirt_engine_history'
- Setting credentials for Reports database 'ovirt_engine_reports'
- Unpacking file '/tmp/hourly-2015-07-30_08-57-01.backup'
Restoring:
- Files
- Engine database 'engine'
- DWH database 'ovirt_engine_history'
- Reports database 'ovirt_engine_reports'
Rewriting /etc/ovirt-engine/engine.conf.d/10-setup-database.conf
Rewriting /etc/ovirt-engine-dwh/ovirt-engine-dwhd.conf.d/10-setup-database.conf
Rewriting /var/lib/ovirt-engine-reports/build-conf/master.properties
You should now run engine-setup.
Done.
```

NOTE

As dwhd was running during automated **backup**, 'engine-setup' would fail with the following error and exit:

```
[ ERROR ] dwhd is currently running. Its hostname is hostname. Please stop it before running Setup.
[ ERROR ] Failed to execute stage 'Transaction setup': dwhd is currently running
```

To pre-empt this, connect to the engine db and override a value within

```
# su postgres
bash-4.1$ psql
Type "help" for help.

\connect engine
UPDATE dwh_history_timekeeping SET var_value=0 WHERE var_name ='DwhCurrentlyRunning';
\q
exit
```

Now 'engine-setup' will not run into this issue

For more information about this see http://www.ovirt.org/Ovirt-engine-backup#DWH_up_during_backup

6. Run engine-setup

```
# engine-setup
[ INFO  ] Stage: Initializing
[ INFO  ] Stage: Environment setup
        Configuration files: ['/etc/ovirt-engine-setup.conf.d/10-packaging-dwh.conf', '/etc/ovirt-engine-setup.conf.d/10-packaging-
wsp.conf', '/etc/ovirt-engine-setup.conf.d/10-packaging.conf', '/etc/ovirt-engine-setup.conf.d/20-packaging-rhevm-reports.conf',
'/etc/ovirt-engine-setup.conf.d/20-setup-ovirt-post.conf']
        Log file: /var/log/ovirt-engine/setup/ovirt-engine-setup-20150730135105-vdms4c.log
        Version: otopi-1.3.2 (otopi-1.3.2-1.el6ev)
[ INFO  ] Stage: Environment packages setup
[ INFO  ] Stage: Programs detection
[ INFO  ] Stage: Environment setup
[ INFO  ] Stage: Environment customization

        Welcome to the RHEV 3.5 setup/upgrade.
        Please read the RHEV 3.5 install guide
        https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.5/html/Installation_Guide/index.html.
        Please refer to the RHEV Upgrade Helper application
        https://access.redhat.com/labs/rhevupgradehelper/ which will guide you in the upgrading process.
        Would you like to proceed? (Yes, No) [Yes]:Yes

        --== PRODUCT OPTIONS ==--


        --== PACKAGES ==--

[ INFO  ] Checking for product updates...
[ INFO  ] No product updates found

        --== ALL IN ONE CONFIGURATION ==--

        --== NETWORK CONFIGURATION ==--

        Setup can automatically configure the firewall on this system.
        Note: automatic configuration of the firewall may overwrite current settings.
        Do you want Setup to configure the firewall? (Yes, No) [Yes]: no

        --== DATABASE CONFIGURATION ==--
        The detected DWH database size is 144 MB.
        Setup can backup the existing database. The time and space required for the database backup depend on its size. This process takes
time, and in some cases (for instance, when the size is few GBs) may take several hours to complete.
        If you choose to not back up the database, and Setup later fails for some reason, it will not be able to restore the database and
all DWH data will be lost.
        Would you like to backup the existing database before upgrading it? (Yes, No) [Yes]: No
[WARNING] Are you sure you do not want to backup the DWH database?
        A positive reply makes sense only if you do not need the data in DWH, or have some other, external means to restore it to a
working state.
        Are you sure you do not want to backup the DWH database?(Yes, No) [No]: Yes
[WARNING] DWH Database will not be backed up. Rollback in case of failure will not be possible.

        --== OVIRT ENGINE CONFIGURATION ==--

        Skipping storing options as database already prepared

        --== PKI CONFIGURATION ==--

        --== APACHE CONFIGURATION ==-

        --== SYSTEM CONFIGURATION ==--

        --== MISC CONFIGURATION ==--

        --== END OF CONFIGURATION ==--

[ INFO  ] Stage: Setup validation
[ INFO  ] Cleaning stale zombie tasks and commands

        --== CONFIGURATION PREVIEW ==--

        Firewall manager                        : iptables
        Update Firewall                         : True
        Host FQDN                               : rhevm-i01.wob.sec.vw.vwg
        Engine database name                    : engine
        Engine database secured connection      : False
        Engine database host                    : localhost
        Engine database user name               : engine
        Engine database host name validation    : False
        Engine database port                    : 5432
        Engine installation                     : True
        DWH installation                        : True
        DWH database name                       : ovirt_engine_history
        DWH database secured connection         : False
        DWH database host                       : localhost
        DWH database user name                  : ovirt_engine_history
        DWH database host name validation       : False
        Backup DWH database                     : False
        DWH database port                       : 5432
        Reports installation                    : True
        Reports database name                 : ovirt_engine_reports
        Reports database secured connection   : False
        Reports database host                 : localhost
        Reports database user name            : ovirt_engine_reports
        Reports database host name validation : False
        Reports database port                 : 5432
        Engine Host FQDN                      : rhevm-i01.wob.sec.vw.vwg
        Configure WebSocket Proxy             : True

        Please confirm installation settings (OK, Cancel) [OK]:<enter>



[ INFO  ] Cleaning async tasks and compensations
[ INFO  ] Checking the Engine database consistency
[ INFO  ] Stage: Transaction setup
[ INFO  ] Stopping dwh service
[ INFO  ] Stopping reports service
[ INFO  ] Stopping engine service
[ INFO  ] Stopping ovirt-fence-kdump-listener service
[ INFO  ] Stopping websocket-proxy service
[ INFO  ] Stage: Misc configuration
[ INFO  ] Stage: Package installation
[ INFO  ] Stage: Misc configuration
[ INFO  ] Backing up database localhost:engine to '/var/lib/ovirt-engine/backups/engine-20150730140428.zJ8rSy.dump'.
[ INFO  ] Creating/refreshing Engine database schema
[ INFO  ] Creating/refreshing DWH database schema
[ INFO  ] Regenerating Jasper's build configuration files
[ INFO  ] Exporting data out of Jasper
[ INFO  ] Backing up database localhost:ovirt_engine_reports to '/var/lib/ovirt-engine-reports/backups/reports-
20150730140801.UHxGXE.dump'.
[ INFO  ] Deploying Jasper
[ INFO  ] Importing data into Jasper
[ INFO  ] Configuring Jasper Java resources
[ INFO  ] Configuring Jasper Database resources
[ INFO  ] Customizing Jasper
[ INFO  ] Customizing Jasper metadata
[ INFO  ] Customizing Jasper Pro Parts
[ INFO  ] Configuring WebSocket Proxy
[ INFO  ] Generating post install configuration file '/etc/ovirt-engine-setup.conf.d/20-setup-ovirt-post.conf'
[ INFO  ] Stage: Transaction commit
[ INFO  ] Stage: Closing up

        --== SUMMARY ==--

        SSH fingerprint: B2:90:AA:2B:7F:45:C7:02:3B:CC:29:E2:95:DE:FF:77
        Internal CA FB:E3:5C:4B:38:00:DA:F7:D1:98:09:14:58:09:82:06:34:F7:1D:36
        Web access is enabled at:
            http://rhevm-i01.wob.sec.vw.vwg:80/ovirt-engine
            https://rhevm-i01.wob.sec.vw.vwg:443/ovirt-engine

        --== END OF SUMMARY ==--

[ INFO  ] Starting engine service
[ INFO  ] Restarting httpd
[ INFO  ] Starting dwh service
[ INFO  ] Starting reports service
[ INFO  ] Stage: Clean up
        Log file is located at /var/log/ovirt-engine/setup/ovirt-engine-setup-20150730140402-r05fyu.log
[ INFO  ] Generating answer file '/var/lib/ovirt-engine/setup/answers/20150730141430-setup.conf'
[ INFO  ] Stage: Pre-termination
[ INFO  ] Stage: Termination
[ INFO  ] Execution of setup completed successfully
#
```

7. Additional steps

NOTE

For the RHEV-M in B2X you have to add these routes:

Routes for B2X

```
# route add -net 10.252.52.0 netmask 255.255.255.0 gw 10.252.72.4
# route add -net 10.252.100.0 netmask 255.255.255.0 gw 10.252.72.4
```

These routes should be made permaent

You should now be able to login to the restored version of RHEV-M

6. Exporting & Importing the VM Shells

The second way of backing up is using a combination of the python SDK and cURL to export the definitions of the VMs, such as memory, cpu, disk etc. These are exported to a number of XML files that can be used to redefine the empty VMs, on an individual basis. These empty VMs would then be recovered with “Relax & Recover” and TSM.

Two scripts were created, export_vms.py and import_vm.py. These are used together with a third file, that holds the credentials required for the API.

6.1. Export

The export_vms.py script exports all VMs into XML files that describe the VM, its NICs and Disks. It is not interactive as it is designed to be run, via a cron job, on a regular basis. The script removes all previous files when it runs, if access to older definitions is required then they can be retrieved via backup (TSM).

6.2. Import

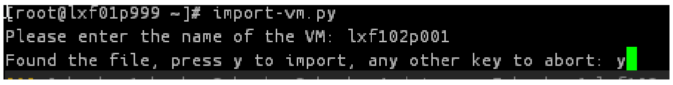
The import_vm.py is an interactive script that will prompt for a VM name, then check if it has the required XML files needed to restore it. If the VM doesn’t already exist and the required files are present, it prompts you to confirm that the you wish to proceed in recreating the empty VM shell.

Both scripts are included in the appendix section of this document.

7. Complete VM Recovery

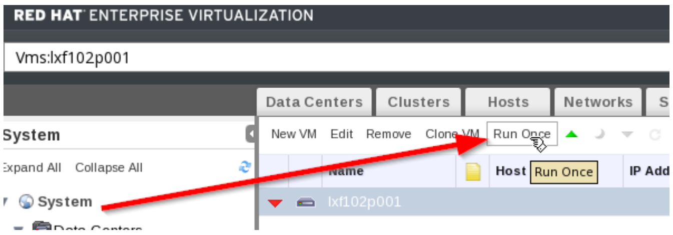
Note: The recovery is mainly done via an open source software called ReaR - Relax and Recover, which is basically a set of scripts that is building an ISO image per system and reuses the TSM backup. ReaR itself is currently (October 2015) not supported by Red Hat in RHEL 7.1.

1. Make sure, that VM is not existing anymore in the RHEV environment (check the RHEV - Manager)
2. Get the ReaR - ISO image
   a. recent ISO image for ReaR saved locally on every VM. If the VM is not existing anymore and / or the FS is not accessible anymore, the ISO image can be retrieved from TSM via the TSM client on a different node
   b. local path to the ReaR image: /opt/tivoli/tsm/rear/rear-<hostname>.iso
      i. Example: : /opt/tivoli/tsm/rear/rear-lxf102p001.iso
3. Copy the ReaR - ISO to the right location so it can be used by RHEV-M for booting the VM
   a. Intranet: lxf01p17:/appl/vwlinux-data/files/install/iso/images/rhev/4feb8cf0-4dd9-48e2-beb1-fa04bdb5096a/images/11111111-1111-1111-1111-111111111111/
   b. B2X: lxf117p099b:/appl/vwlinux-data/files/install/iso/images/rhev/6a6bb588-03ac-4bbb-aead-8c15b4bc0a9c/images/11111111-1111-1111-1111-111111111111/ (RHEV-M picks up all ISO files in this directory)
4. Files for the recovery of VM shell(s)
   a. Jump on the corresponding RHEV-M
      i. Intranet: rhevm-i01
      ii. B2X: rhevm-b01
   b. All VM shells definitions are stored on the RHEV-M in /appl/rhevm/pgsql/backups/vm-shells/ for every VM at least one file for the VM itself, one file per attached disk and one file per NIC Example:
      - lxf102p001-disk-0.xml
      - lxf102p001-disk-1.xml
      - lxf102p001-disk-2.xml
      - lxf102p001-nic-0.xml
      - lxf102p001.xml
5. Use import-vm.py (it is in $PATH) on the console of the corresponding RHEV-M
   a. Type in the name of the VM, that should be restored
   b. Confirm with “y”

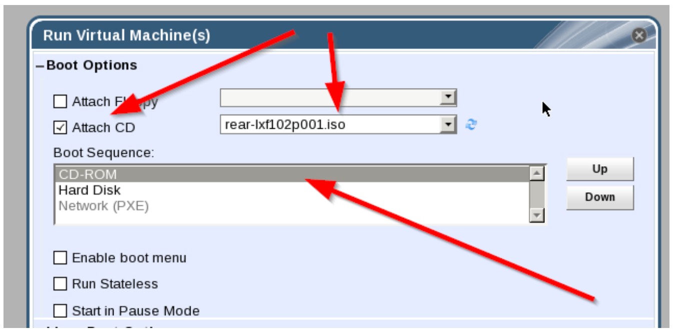


```
[root@lxf01p999 ~]# import-vm.py
Please enter the name of the VM: lxf102p001
Found the file, press y to import, any other key to abort: y
```

a. Every xml file has to be confirmed via ENTER
   1. Switch to the Webinterface of RHEV-M
b. After the complete process the VM is listed in the RHEV-M again and is turned off Turn it on via Run Once procedure



c. Mark the “attach CD” option, choose the corresponding ISO Image, move the CD-ROM device into the first position



- [ ] Attach Floppy
- [x] Attach CD: rear-lxf102p001.iso

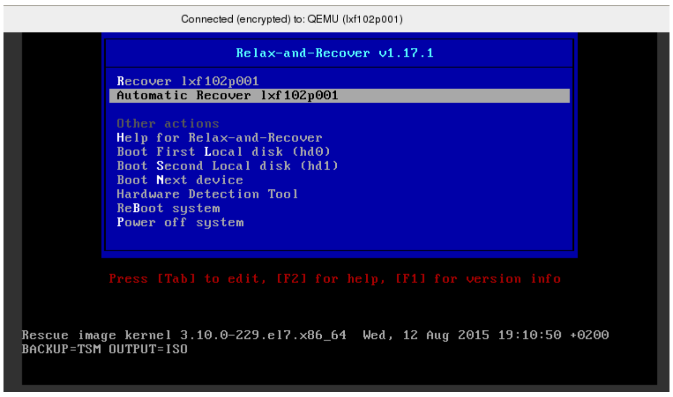
Boot Sequence:
CD-ROM
Hard Disk
Network (PXE)

- [ ] Enable boot menu
- [ ] Run Stateless
- [ ] Start in Pause Mode

d. Press “OK” to start the VM

e. Open the console for the VM

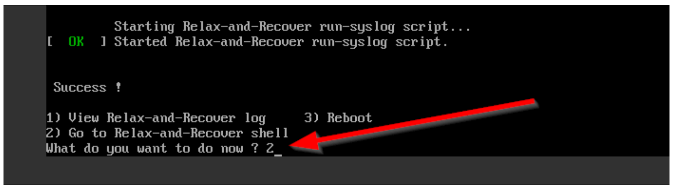
f. Choose “Automatic Recover <hostname>



```
Connected (encrypted) to: QEMU (lxf102p001)

                Relax-and-Recover v1.17.1

  Recover lxf102p001
  Automatic Recover lxf102p001

  Other actions
  Help for Relax-and-Recover
  Boot First Local disk (hd0)
  Boot Second Local disk (hd1)
  Boot Next device
  Hardware Detection Tool
  ReBoot system
  Power off system

  Press [Tab] to edit, [F2] for help, [F1] for version info

Rescue image kernel 3.10.0-229.el7.x86_64  Wed, 12 Aug 2015 19:10:50 +0200
BACKUP=TSM OUTPUT=ISO
```

g. ReaR starts and will ask some questions, that are all answered via the mostly fitting defaults if you do nothing within 30 sec.

8. After ReaR has finished choose “2” to enter the ReaR recovery shell, and hit enter to reach the prompt



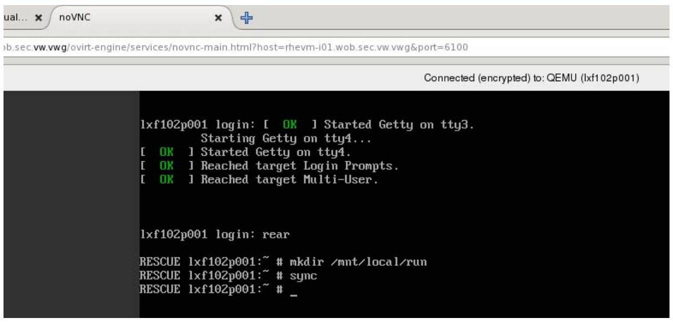
```
         Starting Relax-and-Recover run-syslog script...
[  OK  ] Started Relax-and-Recover run-syslog script.

 Success !

1) View Relax-and-Recover log      3) Reboot
2) Go to Relax-and-Recover shell
What do you want to do now ? 2_
```

a. Login with the user “rear”

b. Create the directory /mnt/local/run: **mkdir /mnt/local/run**

c. ensure that it is synced:

d. sync



```
b.sec.vw.vwg/ovirt-engine/services/novnc-main.html?host=rhevm-i01.wob.sec.vw.vwg&port=6100
```

Connected (encrypted) to: QEMU (lxf102p001)

```
lxf102p001 login: [  OK  ] Started Getty on tty3.
         Starting Getty on tty4...
[  OK  ] Started Getty on tty4.
[  OK  ] Reached target Login Prompts.
[  OK  ] Reached target Multi-User.


lxf102p001 login: rear

RESCUE lxf102p001:~ # mkdir /mnt/local/run
RESCUE lxf102p001:~ # sync
RESCUE lxf102p001:~ # _
```

7. Shutdown the restored machine and start it per standard start via RHEV-M
8. Check if machine is up and healthy and you’re done.

Notes

- The restored VM will probably got a new MAC. The configuration of the VM is adapted to the restored MAC and all interfaces are addresses are the same as before the restore process
- ReaR tries to recover all disks, that were attached to the VM. the virtual disks out of the storage domain(s) are complete emptied. The restore scripts try to reattach the direct LUNs, that were attached previously. If one of these LUNs can not be found, the creation of the VM will fail.
- If it succeed, the LUN is still filled with the data, that were written to it before the crash.
- The default is to recover all file systems - including the file systems for the application data - that machine that TSM is aware of. But you can choose on a per file system basis what ReaR should recover by not giving the default answer during the start of the ReaR process.
- The duration of the whole restore process depends mainly on the speed of the TSM server and on the amount of data.

8. Appendix 1 - Backup Cron Job script

Listing 3. rhevm-backup.sh

```
#!/bin/bash
#
# This script is to be used in combination with a cron job
#
# You supply it with a file prefix, based on when it will run
# (ie hourly daily weekly etc). The script first searches for
# files starting with the $PREFIX and deletes then, keeping
# only the newest $KEEP files.
# It then creates a backup with a timestamp in the filename


if [ $# -ne 3 ]; then
    echo "Usage: ./rhevm-backup <prefix> <number of files to keep>  <destination directory>"
    echo " "
    echo "example   ./rhevm-backup.sh hourly 6 /appl/rhevm/pgsql/engine-backup"
    echo
    exit 1
fi

PREFIX=$1
KEEP=$2
LOCATION=$3

if [ ! -d "${LOCATION}" ]; then
    echo "Destination dir doesnt exist"
    exit 1
fi

I=0

for file in $(ls -t ${LOCATION}/${PREFIX}* )
do
    ((I++))
    if [ "${I}" -le "${KEEP}" ]; then
    echo "keeping file - ${file} "
    else
    echo "removing file - ${file} "
    rm -f  ${file}
    fi
done

echo

# now do the backup

DATE=$(date +"%Y-%m-%d_%H-%M-%S")

# remove older log files
rm -f /tmp/hourly-*.log
rm -f /tmp/daily-*.log

/usr/bin/engine-backup --mode=backup --scope=all --file=${LOCATION}/${PREFIX}-${DATE}.backup --log=/tmp/${PREFIX}-${DATE}.log
```

9. Appendix 2 - Export VM Shells

Listing 4. export_vms.py

```
#!/usr/bin/python

import pycurl
import os
import sys
import lxml.etree
import shutil

from ovirtsdk.api import API
from ovirtsdk.xml import params

# Import Connection details from file
from ConfigParser import SafeConfigParser
parser = SafeConfigParser()
parser.read('connectionDetails')

URL = parser.get('connection', 'URL')
USER = parser.get('connection', 'USER')
PASSWORD = parser.get('connection', 'PASSWORD')
CA = parser.get('connection', 'CA')

config_dir = "configs"

# Prepare a cURL object to connect directly to the server, without
# the Python SDK, as this is required to get raw XML (or JSON):
curl = pycurl.Curl();

# Function to reset the state of the cURL object before each
# request:
def reset_curl():
    curl.reset()
    curl.setopt(pycurl.HTTPAUTH, pycurl.HTTPAUTH_BASIC)
    curl.setopt(pycurl.USERPWD, "%s:%s" % (USER, PASSWORD))
    curl.setopt(pycurl.CAINFO, CA)
    headers = [
        "Accept: application/xml",
        "Content-Type: application/xml",
    ]
    curl.setopt(pycurl.HTTPHEADER, headers)

# Function to strip the ID field from the disk, before importing
def remove_id(file_name):
    # Load and parse the file:
    document = lxml.etree.parse(file_name)
    root = document.getroot()

    # Remove the "id" and "href" attributes:
    if "id" in root.attrib:
        del root.attrib["id"]
    if "href" in root.attrib:
        del root.attrib["href"]

    # Write the modified document:
    content = lxml.etree.tostring(document)
    with open(file_name, "w") as fd:
        fd.write(content)

# remove old definitions
if not os.path.isdir(config_dir):
   print "The defined output directory seems to not be a directory"
   sys.exit(1)
shutil.rmtree(config_dir)
os.makedirs(config_dir)
```

```
# Connect to the server with the Python SDK:
api = API(
    url=URL,
    username=USER,
    password=PASSWORD,
    ca_file=CA
)

# Iterate through all VMs
vm_list=api.vms.list()
for vm in vm_list:
    vm_name = vm.name
    print "Exporting\t", vm_name

    # Lookup the VM id
    vm_id = vm.get_id()


    # Do a plain HTTP request in order to retrieve the XML
    # representation of the VM and save it to a file:
    vm_file_name = "%s/%s.xml" % (config_dir, vm_name)
    vm_url = "%s/vms/%s" % (URL, vm_id)
    with open(vm_file_name, "w") as vm_file:
        reset_curl()
        curl.setopt(pycurl.URL, vm_url)
        curl.setopt(pycurl.WRITEDATA, vm_file)
        curl.perform()

    # Lookup the disks of the VM, and extract their ids:
    disk_ids = []
    for disk in vm.disks.list():
        disk_id = disk.get_id()
        disk_ids.append(disk_id)

    # Do plain HTTP requests in order to retrieve the XML representation
    # of the disks and save them to files:
    disk_index = 0
    for disk_id in disk_ids:
        disk_file_name = "%s/%s-disk-%d.xml" % (config_dir, vm_name, disk_index)
        disk_url = "%s/vms/%s/disks/%s" % (URL, vm_id, disk_id)
        with open(disk_file_name, "w") as disk_file:
            curl.setopt(pycurl.URL, disk_url)
            curl.setopt(pycurl.WRITEDATA, disk_file)
            curl.perform()
        disk_index += 1

    # Lookup the network cards
    network_ids = []
    for nic in vm.nics.list():
        nic_id = nic.get_id()
        network_ids.append(nic_id)

    nic_index = 0
    for nic_id in network_ids:
        nic_file_name = "%s/%s-nic-%d.xml" % (config_dir, vm_name, nic_index)
        nic_url = "%s/vms/%s/nics/%s" % (URL, vm_id, nic_id)
        with open(nic_file_name, "w") as nic_file:
            reset_curl()
            curl.setopt(pycurl.URL, nic_url)
            curl.setopt(pycurl.WRITEDATA, nic_file)
            curl.perform()
        nic_index += 1


# Disconnect both the SDK and cURL:
api.disconnect()
curl.close()
```

10. Appendix 3

Listing 5. import_vms.py

```
#!/usr/bin/python

import pycurl
import os
import sys
import lxml.etree

from ovirtsdk.api import API
from ovirtsdk.xml import params

# Connection details:

from ConfigParser import SafeConfigParser
parser = SafeConfigParser()
parser.read('connectionDetails')

URL = parser.get('connection', 'URL')
USER = parser.get('connection', 'USER')
PASSWORD = parser.get('connection', 'PASSWORD')
CA = parser.get('connection', 'CA')

config_dir = "configs"

# Prepare a cURL object to connect directly to the server, without
# the Python SDK, as this is required to get raw XML (or JSON):
curl = pycurl.Curl();

# Function to reset the state of the cURL object before each
# request:
def reset_curl():
    curl.reset()
    curl.setopt(pycurl.HTTPAUTH, pycurl.HTTPAUTH_BASIC)
    curl.setopt(pycurl.USERPWD, "%s:%s" % (USER, PASSWORD))
    curl.setopt(pycurl.CAINFO, CA)
    headers = [
        "Accept: application/xml",
        "Content-Type: application/xml",
    ]
    curl.setopt(pycurl.HTTPHEADER, headers)

# Function to strip the ID field from the disk, before importing
def remove_id(file_name):
    # Load and parse the file:
    document = lxml.etree.parse(file_name)
    root = document.getroot()

    # Remove the "id" and "href" attributes:
    if "id" in root.attrib:
        del root.attrib["id"]
    if "href" in root.attrib:
        del root.attrib["href"]

    # Write the modified document:
    content = lxml.etree.tostring(document)
    with open(file_name, "w") as fd:
        fd.write(content)

# Function to add type to Direct Attached LUNs
def add_lun_storage_type(file_name):
    # Load and parse the file:
    document = lxml.etree.parse(file_name)
    root = document.getroot()

    # Find the LUN storage element, and add the type if needed:
    lun_storage = root.find("lun_storage")
    if lun_storage is not None:
        the_type = lun_storage.find("type")
        if the_type is None:
            the_type = lxml.etree.Element("type")
            the_type.text = "fcp"
            lun_storage.insert(0, the_type)

    # Write the modified document:
    content = lxml.etree.tostring(document)
    with open(file_name, "w") as fd:
        fd.write(content)

    # If the ID & href fields are left in the disk definition
    # files, it will try to find and attach existing disks
    # If you want it to create new disks, then we have to
    # remove these fields
    remove_id(file_name)

# Connect to the server with the Python SDK:
api = API(
    url=URL,
    username=USER,
    password=PASSWORD,
    ca_file=CA
)

# Get the name of the VM
vm_name = raw_input("Please enter the name of the VM: ")

file_path = config_dir + "/" + vm_name + ".xml"

# Test if the file exists, if not exit
if not os.path.exists(file_path):
    sys.exit("Could not find the files for this VM - %s" % vm_name)
else:
    response = raw_input("Found the file, press y to import, any other key to abort: ")
    if response is not "y":
        sys.exit("Aborting upon request")


# Load the XML definition from the file:
vm_file_name = "%s/%s.xml" % (config_dir, vm_name)
vm_file_content = None
with open(vm_file_name) as vm_file:
    vm_file_content = vm_file.read()

# Do a plain HTTP request in order to create the VM:
vms_url = "%s/vms" % URL
reset_curl()
curl.setopt(pycurl.URL, vms_url)
curl.setopt(pycurl.POSTFIELDS, vm_file_content)
curl.perform()

raw_input("Press Enter to re-create the disks...")

# Lookup the VM by name, in order to find the id:
vm = api.vms.get(name=vm_name)
vm_id = vm.get_id()

# Load the disk files, and for each one perform the HTTP
# request in order to create the disk:
disks_url = "%s/vms/%s/disks" % (URL, vm_id)
disk_index = 0
while True:
    disk_file_name = "%s/%s-disk-%d.xml" % (config_dir, vm_name, disk_index)
    print disk_file_name
    if not os.path.exists(disk_file_name):
        break
    disk_file_content = None
    # add the disk type (work around)
    add_lun_storage_type(disk_file_name)

    with open(disk_file_name) as disk_file:
        disk_file_content = disk_file.read()
    reset_curl()
    curl.setopt(pycurl.URL, disks_url)
    curl.setopt(pycurl.POSTFIELDS, disk_file_content)
    curl.perform()
    disk_index += 1

raw_input("Press Enter to re-create the nics...")

## Load the nic files
nic_url = "%s/vms/%s/nics" % (URL, vm_id)
nic_index = 0
while True:
    nic_file_name = "%s/%s-nic-%d.xml" % (config_dir, vm_name, nic_index)
    print nic_file_name
    if not os.path.exists(nic_file_name):
        break
    nic_file_content = None

    # we need to strip the ID from the nic before importing it
    remove_id(nic_file_name)

    with open(nic_file_name) as nic_file:
        nic_file_content = nic_file.read()
    reset_curl()
    curl.setopt(pycurl.URL, nic_url)
    curl.setopt(pycurl.POSTFIELDS, nic_file_content)
    curl.perform()
    nic_index += 1


# Disconnect both the SDK and cURL:
api.disconnect()
curl.close()
```