RED HAT®
CONSULTING

# Adding Hypervisors to RHEV-M

## PREPARED FOR - Volkswagen IT Group Cloud

Adrian Bradshaw

Version 0.0.2

## Table of Contents

## 1. History and Revisions

| Version | Date | Authors | Changes |
|---|---|---|---|
| 0.0.1 | 22/10/2015 | Adrian Bradshaw adrian@redhat.com | Initial Draft |
| 0.0.2 | 22/10/2015 | Adrian Bradshaw adrian@redhat.com | Added pre & post script settings |

# 2. Preface

## 2.1. Confidentiality, Copyright, and Disclaimer



## 2.2. About This Document

This document describes the solution to a request from Volkswagen, for a simple and repeatable way to add new Hypervisors into RHEV-M

## 2.3. Audience

The document is intended for those team members on site at Volkwagen, who will be responsible for the task of adding Hypervisors to RHEV-M

## 2.4. Additional Background and Related Documents

Numerous other documents have also been provided by Red Hat Consulting, explaining tasks such as installation, backup etc

## 2.5. 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. Introduction

One of the requirements that Volkswagen tasked Red Hat to provide was a simple and repeatable way to add or re-add hypervisors into RHEV-M. As RHEV provides a rich API, this was certainly possible and a python script was developed to do the majority of the work. *The full script can be found in the Appendix*

The process of adding a new hypervisor consists of three phases

1. Pre-requisites
2. Running the add script
3. Post script tasks

## 3.1. Pre-requisites

## 3.2. Adjustments of basic files before setting the hypervisor up

There are a couple of files that need to be changed before we start, one will facilitate better performance for the VMs under heavy load (mon.conf) and one will exclude the local disks from multipath.

*The first one (mom.conf) will be not need to be changed in future versions of RHEV as it will be the default*

*Listing 1. /etc/vdsm/mom.conf*

```
# ### DO NOT REMOVE THIS COMMENT -- MOM Configuration for VDSM ###

[main]
# The wake up frequency of the main daemon (in seconds)
main-loop-interval: 15

# The data collection interval for host statistics (in seconds)
host-monitor-interval: 15

# The data collection interval for guest statistics (in seconds)
guest-monitor-interval: 15

# The wake up frequency of the guest manager (in seconds).  The guest manager
# sets up monitoring and control for newly-created guests and cleans up after
# deleted guests.
guest-manager-interval: 15

# The interface MOM using to discover active guests and collect guest memory
# statistics. There're two choices for it: libvirt or vdsm.
hypervisor-interface: VDSM

# The wake up frequency of the policy engine (in seconds).  During each
# interval the policy engine evaluates the policy and passes the results
# to each enabled controller plugin.
policy-engine-interval: 30

# A comma-separated list of Controller plugins to enable
#controllers: Balloon, KSM, CpuTune
controllers: Balloon, KSM

# Sets the maximum number of statistic samples to keep for the purpose of
# calculating moving averages.
sample-history-length: 10

# Set this to an existing, writable directory to enable plotting.  For each
# invocation of the program a subdirectory momplot-NNN will be created where NNN
# is a sequence number.  Within that directory, tab-delimited data files will be
# created and updated with all data generated by the configured Collectors.
```

```
plot-dir:

# Activate the RPC server on the designated port (-1 to disable).  RPC is
# disabled by default until authentication is added to the protocol.
rpc-port: -1

# At startup, load a policy from the given directory.  If empty, no policy is loaded
policy-dir: /etc/vdsm/mom.d

[logging]
# Set the destination for program log messages.  This can be either 'stdio' or
# a filename.  When the log goes to a file, log rotation will be done
# automatically.
log: /var/log/vdsm/mom.log

# Set the logging verbosity level.  The following levels are supported:
# 5 or debug:   Debugging messages
# 4 or info:    Detailed messages concerning normal program operation
# 3 or warn:    Warning messages (program operation may be impacted)
# 2 or error:   Errors that severely impact program operation
# 1 or critical:  Emergency conditions
# This option can be specified by number or name.
verbosity: info

## The following two variables are used only when logging is directed to a file.
# Set the maximum size of a log file (in bytes) before it is rotated.
max-bytes: 2097152
# Set the maximum number of rotated logs to retain.
backup-count: 5

[host]
# A comma-separated list of Collector plugins to use for Host data collection.
collectors: HostMemory, HostKSM, HostCpu

[guest]
# A comma-separated list of Collector plugins to use for Guest data collection.
collectors: GuestQemuProc, GuestMemoryOptional, GuestBalloon, GuestCpuTune
```

*Listing 2. multipath.conf*

```
...
# defaults {
    find_multipaths yes
    user_friendly_names yes
}

devices {
         device {
                         vendor "EMC"
                         product "Invista"
                         product_blacklist "LUNZ"
                         path_grouping_policy "multibus"
                         path_checker "tur"
                         features "0"
                         hardware_handler "0"
                         prio "const"
                         rr_weight "uniform"
                         no_path_retry "5"
         }
}

blacklist {
    device {
         vendor FTS
         product        *
    }
}
```

Additionally in the B2X zone the /etc/hosts has to be extended so it contains every host in the cluster. In B2X this is necessary due to the lack of DNS. If one host can not resolve the other hosts by name, migration will not work.

## 3.3. Running the add script

The script is designed to run in three modes

1. Interactive Mode - In this mode, triggered by running the script with no parameters, you will be prompted to enter all the details required
2. DNS Mode - In this mode, trggered by providing just one parameter (the FQDN of the VM) you will still be prompted for each piece of information, but DNS lookups will be used to attempt to provide sensible defaults that can be accepted by simply hittin enter. The script is also Zone aware, again based on the hostname, it will determin if the VM is destined for the **B2X** Zone or the **Intranet** Zone. Subnet masks and gateways will be determined based on this information and provided as defaults, which can be overriden
3. Script mode - This mode in non-interactive, and so all the information must be provided by providing three parameters at run time - **This has not been coded yet**

### 3.3.1. Interactive Mode

In interactive mode, entered by providing zero parameters, you will be required to interactively provide all the information to the script. Some limited error checking, such as checking the format of an IP adddress, is provided but please take care when entering information to the script. You will be given the chance to verify the details before proceeding

*Listing 3. Interactive Mode Example, Intranet Zone*

```
[upxbyx2@lxf02p90 rhev]$ ./hv-new.py
Interactive mode
Please enter the address of the HV as FQDN: lxf102s001.wob.sec.vw.vwg
Please enter the IP for power management: lxf102s001.wob.sec.vw.vwg
Please enter the IP for power management: 10.208.24.66
Please enter the IP address of the first interface: 10.208.24.54
Please enter the mask of the first interface: 255.255.254.0
Please enter the gateway of the first interface: 10.208.24.1
Please enter the IP address of the move NIC: 10.116.100.146
Please enter the mask of the move NIC: 255.255.252.0
Please enter the bonding mode the move NIC (1 or 4): 4

Summary
-------
Using these settings ...
Zone:           intranet
Name:           lxf102s001
FQDN:           lxf102s001.wob.sec.vw.vwg
Host Address:   10.208.24.54/255.255.254.0
Gateway:        10.208.24.1
Fence Address:  10.208.24.66
Move IP:        10.116.100.146/255.255.252.0
Bonding Mode:   4

RHEV-M          https://rhevm-i01.wob.sec.vw.vwg:443/api
Certificate     ./ovirt-intranet-ca.crt

Press any key to continue, or ctrl-c to abort

Connected to Red Hat Enterprise Virtualization Manager successfully!
* Host was added successfully
* Waiting for host to install and reach the Up status
* Host is up
* Setting Host to maintenance
* Host is in maintenance mode
* Configuring the bonds and vlans
* waiting for labeled vlans to attach to the bonds
* Setting move vlan IP address
Finished - HV can now be actiavted when you are ready
```

### 3.3.2. DNS Mode

Below is an example of the script being run in **DNS Mode**. One parameter was passed (lxf102s001.wob.sec.vwg.vw) and then the user was prompted for each piece of information with sensible defaults shown inside square brackets As mentioned above, the script will determine which zone the VM should be in, and contact the correct RHEV-M instance. If the required certificate is not present, it will be downloaded.

*Listing 4. DNS Mode Example, Intranet Zone*

```
[upxbyx2@lxf02p90 rhev]$ ./hv-new.py  lxf102s001.wob.sec.vw.vwg
DNS mode
Please enter the IP for power management [10.208.24.66]:
Please enter the IP address of the first interface [10.208.24.54]:
Please enter the mask of the first interface [255.255.254.0]:
Please enter the gateway of the first interface [10.208.24.1]:
Please enter the IP address of the move NIC [10.116.100.146]:
Please enter the mask of the move NIC [255.255.252.0]:
Please enter bonding mode (1=Active/Passive 4=LACP) [4]: 1

Summary
-------
Using these settings ...
Zone:           intranet
Name:           lxf102s001
FQDN:           lxf102s001.wob.sec.vw.vwg
Host Address:   10.208.24.54/255.255.254.0
Gateway:        10.208.24.1
Fence Address:  10.208.24.66
Move IP:        10.116.100.146/255.255.252.0
Bonding Mode:   1

Missing certificate, attempting to get from the server
RHEV-M          https://rhevm-i01.wob.sec.vw.vwg:443/api
Certificate     ./ovirt-intranet-ca.crt

Press any key to continue, or ctrl-c to abort

Connected to Red Hat Enterprise Virtualization Manager successfully!
* Host was added successfully
* Waiting for host to install and reach the Up status
* Host is up
* Setting Host to maintenance
* Host is in maintenance mode
* Configuring the bonds and vlans
* waiting for labeled vlans to attach to the bonds
* Setting move vlan IP address
Finished - HV can now be activated when you are ready
```

## 3.4. Post script tasks

### 3.4.1. Migration Bandwidth

The defailt setting for migration bandwidth is 32 MiB/s bandwidth which is equal to 256 Mb/s. For most workloads this is perfectly fine. However, in the VW setup we have some **large** VMs (VMs with 1TB of RAM). VMs of this size will necesitate additional configuration. As the migration network is on bond2, which has 10Gbps bonded interfaces, we should increase the migration bandwidth to facilitate the migration of this VM. We should change this to 120 MiB/s.

1. Temporarily disable the power management of the hosts.
2. Add the below parameter in the [vars] section of /etc/vdsm/vdsm.conf

```
[vars]
...
migration_max_bandwidth = 120
```

3. Restart the vdsmd service

```
/etc/init.d/vdsmd restart
```

4. Re-enable power management for the hosts

## 4. Appendix 1

*Listing 5. The source code of the* **hv.new.py** *script*

```
#!/usr/bin/python
#
# Script to add new hypervisors into RHEV-M
#
# Version: 1.2.2
#
# adrian@redhat.com
#
# ToDo: Authentication - LDAP
# ToDo: Much more validation ;-)

import sys
import time
import os
import socket
import urllib2
from ovirtsdk.api import API
from ovirtsdk.xml import params

def usage():
  #The wrong number of parametes was entered or --help requested
  print
  print 'Usage: The HV Add script can run in three modes'
  print
  print 'Mode 1 - DNS Mode'
  print '-----------------'
  print 'First mode requires only one parameter - the short hostname'
  print 'All other information is resolved via DNS lookups'
  print 'If any information cannot be resolved, the process will abort'
  print
  print 'Mode 2 - Parameter Mode'
  print '------------------------'
  print 'In this mode, all information is sent to the script via the use of'
  print 'three parameters:'
  print '          ha-add.py <short name> <ilom address> <move address>'
  print
  print 'For example '
  print '          ha-add.py lxf101s001 10.208.24.42 10.116.100.143'
  print
  print 'Mode 3 - interactive'
  print '---------------------'
  print 'If no paramaters are entered at all, it will fallback into interactive'
  print 'mode, prompting for each piece of information required'
  print

def summary():
  print
  print 'Summary'
  print '-------'
  print 'Using these settings ...'
  print('Zone: \t\t%s' % ZONE)
  print('Name: \t\t%s' % HOST_NAME)
  print('FQDN: \t\t%s' % HOST_ADDRESS)
  print('Host Address: \t%s/%s' % (HOST_IP,HOST_MASK))
  print('Gateway: \t%s' % HOST_GATEWAY)
  print('Fence Address: \t%s' % PM_ADDRESS)
  print('Move IP: \t%s/%s' % (HOST_MOVE_IP, HOST_MOVE_MASK))
  print('Bonding Mode: \t%s' % MODE)
  print

def validIP(address):
    parts = address.split(".")
    if len(parts) != 4:
        return False
    for item in parts:
```

```
        if not 0 <= int(item) <= 255:
            return False
    return True

def validFQDN(address):
    # simplistic FQDN check (tests for a name with >3 . in it
    parts = address.split(".")
    if len(parts) <= 3:
        return False
    return True

def set_zone(address):
    VAL = address[:6]
    if VAL == "lxf101" or VAL == "lxf102":
      return "intranet"
    else:
      return "b2x"

#sys.exit()
# Main script starts here - Get the total number of args passed to the script
total = len(sys.argv)

VERSION = params.Version(major='3', minor='5')
PM_ADDRESS=""
HOST_NAME=""
HOST_ADDRESS=""
HOST_IP=""
HOST_MASK=""
HOST_GATEWAY=""
HOST_MOVE_IP=""
HOST_MOVE_MASK=""
MODE=""
DOMAIN=""
ZONE=""
PASSWORD=""

if total  == 1:
  #Go interactive and ask for each entry
  print 'Interactive mode'

  # host address is for the address field in the host definition
  while not validFQDN(HOST_ADDRESS):
    HOST_ADDRESS  = raw_input("Please enter the address of the HV as FQDN: ")

  HOST_NAME, DOMAIN = HOST_ADDRESS.split('.', 1)

  while not validIP(PM_ADDRESS):
    # Power management address is the ILOM address, for fencing
    PM_ADDRESS = raw_input("Please enter the IP for power management: ")

  # prompt for some more details
  while not validIP(HOST_IP):
    HOST_IP = raw_input("Please enter the IP address of the first interface: ")

  # we can use the same logic to test mask is ok
  while not validIP(HOST_MASK):
    HOST_MASK = raw_input("Please enter the mask of the first interface: ")

  while not validIP(HOST_GATEWAY):
    HOST_GATEWAY= raw_input("Please enter the gateway of the first interface: ")

  while not validIP(HOST_MOVE_IP):
    HOST_MOVE_IP = raw_input("Please enter the IP address of the move NIC: ")

  while not validIP(HOST_MOVE_MASK):
    HOST_MOVE_MASK = raw_input("Please enter the mask of the move NIC: ")

  #get the bonding mode
  MODE = raw_input("Please enter the bonding mode the move NIC (1 or 4): ")
```

```
  #determine the zone based on the hostname
  ZONE = set_zone(HOST_ADDRESS)

  summary()


elif total == 2:
  # Received 1 parameter - need to work out if its a request for help
  # or its a hostname and so we are in DNS mode
  if sys.argv[1] == "--help" or  sys.argv[1] == "-h":
          usage()
          sys.exit()
  else:
    #DNS mode - look up all info via DNS or fail
    print 'DNS mode'

    HOST_NAME, DOMAIN = sys.argv[1].split('.', 1)

    # assuming the masks dont for each zone
    ZONE = set_zone(HOST_NAME)
    if ZONE == "intranet":
      #intranet settings
      HOST_MASK     ="255.255.254.0"
      HOST_MOVE_MASK="255.255.252.0"
      HOST_GATEWAY  ="10.208.24.1"
      MOVE_ADDRESS      = HOST_NAME + "mo.wob.vw.vwg"
    else:
      #b2x settings
      HOST_MASK     ="255.255.254.0"
      HOST_MOVE_MASK="255.255.254.0"
      HOST_GATEWAY  ="10.252.100.4"
      MOVE_ADDRESS      = HOST_NAME + "mo.b2x.vwg"

    #print("move address: %s" % MOVE_ADDRESS)
    HOST_ADDRESS      = HOST_NAME + "." + DOMAIN
    LOM_ADDRESS       = HOST_NAME + "lom." + DOMAIN
    #HOST_IP           = socket.gethostbyname(HOST_ADDRESS)
    #PM_ADDRESS        = socket.gethostbyname(LOM_ADDRESS)
    #HOST_MOVE_IP      = socket.gethostbyname(MOVE_ADDRESS)
    MODE              = 4

    # still ask the questions, but show default answers based on DNS
    TEMP = socket.gethostbyname(LOM_ADDRESS)
    if TEMP:
        RESULT = raw_input("Please enter the IP for power management [" + str(TEMP) +"]: ")
    else:
        RESULT = raw_input("Please enter the IP for power management :")
    PM_ADDRESS = RESULT or TEMP

    # ip address of the first interface
    TEMP = socket.gethostbyname(HOST_ADDRESS)
    if TEMP:
            RESULT = raw_input("Please enter the IP address of the first interface [" + str(TEMP) +"]: ")
    else:
            RESULT = raw_input("Please enter the IP address of the first interface: ")
    HOST_IP = RESULT or TEMP

    # mask of the main interface
    TEMP = HOST_MASK
    if TEMP:
            RESULT = raw_input("Please enter the mask of the first interface [" + str(TEMP) +"]: ")
    else:
            RESULT = raw_input("Please enter the mask of the first interface :")
    HOST_MASK = RESULT or TEMP

    # ip gateway of the first interface
    TEMP = HOST_GATEWAY
    if TEMP:
            RESULT = raw_input("Please enter the gateway of the first interface [" + str(TEMP) +"]: ")
    else:
```

```
            RESULT = raw_input("Please enter the gateway of the first interface: ")
    HOST_GATEWAY = RESULT or TEMP

    TEMP = socket.gethostbyname(MOVE_ADDRESS)
    if TEMP:
            RESULT = raw_input("Please enter the IP address of the move NIC [" + str(TEMP) +"]: ")
    else:
            RESULT = raw_input("Please enter the IP address of the move NIC:")
    HOST_MOVE_IP = RESULT or TEMP

    # mask of the move interface
    TEMP = HOST_MOVE_MASK
    if TEMP:
            RESULT = raw_input("Please enter the mask of the move NIC [" + str(TEMP) +"]: ")
    else:
            RESULT = raw_input("Please enter the mask of the move NIC:")
    HOST_MOVE_MASK = RESULT or TEMP

    # bonding mode
    TEMP = MODE
    if TEMP:
            RESULT = raw_input("Please enter bonding mode (1=Active/Passive 4=LACP) [" + str(TEMP) +"]: ")
    else:
            RESULT = raw_input("Please enter bonding mode (1=Active/Passive 4=LACP: ")
    MODE = RESULT or TEMP

    summary()


elif total == 4:
  # All details are spcified as parameters on the command line
  print 'Script mode: Three parameters detected - not implememnted yet'
  sys.exit()
  print

else:
  # Wrong number of parameters - dispaly usage
  usage()

################################################

# set the parameters based on which zone
if ZONE == "intranet":
  URL =           'https://rhevm-i01.wob.sec.vw.vwg:443/api'
  USERNAME =      'admin@internal'
  PASSWORD =      'SYkxHmj_h2'
  INTRANET_CA =   './ovirt-intranet-ca.crt'
  #lets just check that the certificates are in place
  while not os.path.isfile(INTRANET_CA):
    print "Missing certificate, attempting to get from the server"
    CERT = urllib2.urlopen("https://rhevm-i01.wob.sec.vw.vwg/ca.crt")
    output = open(INTRANET_CA,'wb')
    output.write(CERT.read())
    output.close()
  CA = INTRANET_CA
  DC_NAME =       'default'
  CLUSTER_NAME =  'lxf117f118c002'
  #STORAGE_NAME =  'S001M_lxf117f118c001_VPLEX'
  ROOT_PASSWORD = 'RedHat1!'

else:
  URL =           'https://rhevm-b01.wob.sec.vw.vwg:443/api'
  USERNAME =      'admin@internal'
  PASSWORD =      '9LYOpJQX5y'
  B2X_CA =        './ovirt-b2x-ca.crt'
  #lets just check that the certificates are in place
  while not os.path.isfile(B2X_CA):
    print "Missing certificate, attempting to get from the server"
    CERT = urllib2.urlopen("https://rhevm-b01.wob.sec.vw.vwg/ca.crt")
    output = open(B2X_CA,'wb')
```

```
    output.write(CERT.read())
    output.close()
  CA =  B2X_CA
  DC_NAME =       'default'
  CLUSTER_NAME =  'lxf117f118c002'
  #STORAGE_NAME =  'S001M_lxf117f118c001_VPLEX'
  ROOT_PASSWORD = 'RedHat1!'

print('RHEV-M \t\t%s' % URL)
print('Certificate \t%s' %  CA)
print

# Give them the option to review before proceding
os.system('read -s -n 1 -p "Press any key to continue, or ctrl-c to abort"')
print
print

try:
        api = API(url=URL, username=USERNAME, password=PASSWORD, ca_file=CA)
        print "Connected to %s successfully!" % api.get_product_info().name

except Exception as err:
        print "Connection failed: %s" % err



sys.exit()
# this is where the main work is done
# ----------------------------------

try:
    pm = params.PowerManagement()
    pm.set_type('ipmilan')
    pm.set_enabled(True)
    pm.set_address(PM_ADDRESS)
    pm.set_username('fenceinguser')
    pm.set_password('8Y#6smB+z63q')
    pm.set_kdump_detection(True)

    if api.hosts.add(params.Host(name=HOST_NAME,
                     address=HOST_ADDRESS,
                     cluster=api.clusters.get(CLUSTER_NAME),
                     root_password=ROOT_PASSWORD,
                     power_management=pm)):
        print '* Host was added successfully'
        print '* Waiting for host to install and reach the Up status'
        while api.hosts.get(HOST_NAME).status.state != 'up':
            time.sleep(1)
        print "* Host is up"
    if api.hosts.get(HOST_NAME).deactivate():
        print '* Setting Host to maintenance'
        #print '* Waiting for host to reach maintenance status'
        while api.hosts.get(HOST_NAME).status.state != 'maintenance':
            time.sleep(1)
        print '* Host is in maintenance mode'

except Exception as e:
    print 'Failed to install Host:\n%s' % str(e)


# Configure the network
nic0 = params.HostNIC(name = 'enp8s0f0', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))
nic1 = params.HostNIC(name = 'enp8s0f1', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))

nic2 = params.HostNIC(name = 'ens3f0', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))
nic3 = params.HostNIC(name = 'ens4f0', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))
```

```
nic4 = params.HostNIC(name = 'ens3f1', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))
nic5 = params.HostNIC(name = 'ens4f1', network =  params.Network(), boot_protocol='none',  ip=params.IP(address=None, netmask=None,
gateway=None))

print "* Configuring the bonds and vlans"
# bond
bond = params.Bonding(
   slaves = params.Slaves(host_nic = [ nic0, nic1 ]),
            options = params.Options(
                        option = [
                          params.Option(name = 'miimon', value = '100'),
                          params.Option(name = 'mode', value = '1'),
                          params.Option(name = 'primary', value = 'enp8s0f0')
                        ]
                        )
                      )

# management network on top of the bond
managementNetwork = params.HostNIC(network = params.Network(name = 'rhevm'),
                      name = 'bond0',
                      boot_protocol = 'static',
                          ip = params.IP(
                          address = HOST_IP,
                          netmask = HOST_MASK,
                          gateway = HOST_GATEWAY),
                      override_configuration = True,
                      bonding = bond)

# now attempt the next bond
bond1 = params.Bonding(
   slaves = params.Slaves(host_nic = [ nic2, nic3 ]),
            options = params.Options(
                        option = [
                          params.Option(name = 'miimon', value = '100'),
                          params.Option(name = 'mode', value = MODE),
                          params.Option(name = 'primary', value = 'ens3f0')
                          ]
                        )
                      )

# management network on top of the bond
backendNetwork = params.HostNIC( #network = params.Network(name = 'vlan007'),
                      name = 'bond1',
                      boot_protocol = 'none',
                      override_configuration = True,
                      bonding = bond1)

# now attempt the last bond
bond2 = params.Bonding(
   slaves = params.Slaves(host_nic = [ nic4, nic5 ]),
            options = params.Options(
                        option = [
                          params.Option(name = 'miimon', value = '100'),
                          params.Option(name = 'mode', value = MODE),
                          params.Option(name = 'primary', value = 'ens3f1')
                          ]
                        )
                      )

# Management network on top of the bond
HeSyMoNetwork = params.HostNIC( #network = params.Network(name = 'vlan032'),
                      name = 'bond2',
                      boot_protocol = 'none',
                      override_configuration = True,
                      bonding = bond2)

# Now apply the rhevm network configuration
host = api.hosts.get(HOST_NAME)
host.nics.setupnetworks(params.Action(force = False, check_connectivity = True, host_nics = params.HostNics(host_nic = [
```

```
managementNetwork,backendNetwork,HeSyMoNetwork ] )))


# Add the labels to teh bonds, so vlans are assigned correctly
try:
        host.nics.get('bond1').labels.add(params.Label(id='BackEnd'))
        host.nics.get('bond2').labels.add(params.Label(id='HeSyMo'))

except Exception as err:
        print "Add label failed for %s" % err
        sys.exit(1)

# Wait for network to sync
print "* waiting for labeled vlans to attach to the bonds"
time.sleep(15)


# Set the IP address of the move network
print "* Setting move vlan IP address"
try:
        if ZONE == "b2x":
            nic = host.nics.get(name='bond2.33')
        else:
            nic = host.nics.get(name='bond2.32')

        nic.set_boot_protocol('static')
        nic.set_ip(params.IP(address=HOST_MOVE_IP, netmask=HOST_MOVE_MASK,gateway=None))
        nic.update()

except Exception as err:
        print "Setting move net IP address failed"
        sys.exit(1)

# Finally save all the settings
try:
        host.commitnetconfig()
        print "Finished - HV can now be actiavted when you are ready"
except:
        print "Problem saving the settings, you could click save on the general tab of the Host to fix"
```