**LIVE MIGRATION**

Live Migration is the process of moving a running [virtual machine](http://www.webopedia.com/TERM/V/virtual_machine.html) (VM) from one physical server to another while not disrupting the availability of the virtual machine to users.

The goal of a VM live migration is to enable maintenance or upgrades to be performed on a VM without any of the virtual machine's users experiencing down-time during the migration. Live migrations are also known as seamless live migrations when there's no discernible downtime to the end user during the migration process.

**Requirement:**

1. Support hardware virtualization.
2. Use processors from the same manufacturer. For example, all AMD or all Intel.
3. Belong to either the same Active Directory domain, or to domains that trust each other.
4. Virtual machines must be configured to use virtual hard disks or virtual Fibre Channel disks (no physical disks).

**Overview:**

**1. Live migration setup occurs.**

During the live migration setup stage, the source server creates a connection with the destination server. This connection transfers the virtual machine configuration data to the destination server. A skeleton virtual machine is set up on the destination server and memory is allocated to the destination virtual machine.

**2. Memory pages are transferred from the source node to the destination node.**

In the second stage of a live migration, the memory assigned to the migrating virtual machine is copied over the network to the destination server. This memory is referred to as the “working set” of the migrating virtual machine. A page of memory is 4 KB.

**3. Modified pages are transferred.**

The third stage of a live migration is a memory copy process that duplicates the remaining modified memory pages for “test virtual machine” to the destination server. The source server transfers the CPU and device state of the virtual machine to the destination server.

**4. The storage handle is moved from the source server to the destination server.**

During the fourth stage of a live migration, control of the storage associated with “test virtual machine,” such as any virtual hard disk files or physical storage attached through a virtual Fibre Channel adapter, is transferred to the destination server.

**5. The virtual machine is brought online on the destination server.**

In the fifth stage of a live migration, the destination server now has the up-to-date working set for “test virtual machine,” as well as access to any storage used by “test virtual machine.” At this point “test virtual machine” is resumed.

**6. Network cleanup occurs.**

In the final stage of a live migration, the migrated virtual machine is running on the destination server. At this point, a message is sent to the network switch. This message causes the network switch to obtain the new the MAC addresses of the migrated virtual machine so that network traffic to and from “test virtual machine” can use the correct switch port.

**The following variables may affect live migration speed:**

* The number of modified pages on the virtual machine to be migrated—the larger the number of modified pages, the longer the virtual machine will remain in a migrating state.
* Available network bandwidth between source and destination servers.
* Hardware configuration of source and destination servers.
* Load on source and destination servers.
* Available bandwidth (network or Fibre Channel) between servers and shared storage.

**STEPS :**

**1.Creating an NFS Pool Storage:**

NFS pools are storage resources provided by OVP hosts, used by virtual machines for storage

purposes.

* 1. **Setup NFS Pool directory.**

Create the pool directory.

# mkdir -p /export/x86-64-kvm-guest-pool

Edit /etc/exports to add the corresponding export line.

# cat /etc/exports

/export/x86-64-kvm-guest-pool \*(rw,no\_subtree\_check,insecure,no\_root\_squash)

Tell the NFS server to reload the exports configuration file.

# exportfs –a

**1.2 Connect to the QEMU hypervisor.**

# virsh connect qemu:///system

**1.3 Load the configuration file for the NFS pool.**

Before loading the configuration file, it is recommended that you verify that the POOL\_HOST

variable contains a fully qualified name and not a local name such as localhost. Using fully

qualified names allows virtual machines to find the required storage resources, even when they

migrate across different OVP hosts.

# cat *xmlDir*/x86-64-kvm-guest-pool.xml

<pool type="netfs">

<name>x86-64-kvm-guest-pool</name>

<source>

<host name=**'HOST\_NAME'**/>

<dir path='/export/x86-64-kvm-guest-pool/'/>

</source>

<target>

<path>/export/images/</path>

</target>

</pool>

# virsh pool-define *xmlDir*/x86-64-kvm-guest-pool.xml

NOTE: Generating all sample files with default values.

# libvirt-xml-examples

# ls

x86-64-kvm-guest-glusterfs-qcow2.xml x86-64-kvm-guest-local-qcow2.xml

x86-64-kvm-guest-nfs-qcow2.xml x86-64-kvm-guest-pool

x86-64-kvm-guest-glusterfs-raw.xml x86-64-kvm-guest-local-raw.xml

x86-64-kvm-guest-nfs-raw.xml x86-64-kvm-guest-pool.xml

**1.4 Start the storage pool**

# virsh pool-start x86-64-kvm-guest-pool

Note: **virsh** is a command line interface tool for managing guest virtual machines and the hypervisor. The **virsh** command-line tool is built on the **libvirt** management API and operates as an alternative to the **qemu-kvm** command and the graphical **virt-manager** application.

**1.5 Create the storage volume on the x86-64-kvm-guest-pool storage pool.**

# virsh vol-create-as x86-64-kvm-guest-pool x86-64-kvm-guest-vda.raw 10G --format raw

**2. Launching a virtual machine on source server.**

**2.1 Create a Linux Bridge for network connectivity to VM**

brctl addbr <BRIDGE\_NAME>

ifconfig <INTERFACE\_NAME> 0.0.0.0 promisc up

brctl addif <BRIDGE\_NAME> <INTERFACE\_NAME>

ifconfig <BRIDGE\_NAME> <BRIDGE\_IP> up

**2.2 Edit virtual machine configuration file.**

# cp x86-64-kvm-guest-nfs-raw.xml <NAME\_FOR\_CONF\_FILE>.xml

#vim <NAME\_FOR\_CONF\_FILE>.xml

<domain type='kvm'>

<name>**L2\_VM2**</name> -🡪name for VM

………

<cpu mode='custom' match='exact'>

<model fallback='allow'>Conroe</model>

<topology sockets='**1**' cores='**3**' threads='**1**'/> -🡪 assign vcpus to VM

</cpu>

……..

<devices>

<emulator>/usr/bin/kvm</emulator>

<disk type='file' device='cdrom'>

<driver name='qemu' type='raw'/>

<source file='**….\*iso**' startupPolicy='optional'> -🡪 mention ISO file

</source>

<target dev='hdc' bus='ide'/>

<readonly/>

<serial></serial>

<boot order='1'/>

<alias name='ide0-1-0'/>

<address type='drive' controller='0' bus='1' target='0' unit='0'/>

</disk>

……..

<disk type='volume' device='disk' snapshot='no'>

<driver name='qemu' type='raw' cache='none' io='threads'/>

<source pool='x86-64-kvm-guest-pool' volume='**\*\*.raw'**/> --.assign here storage created in section 1.5

<target dev='vda' bus='virtio'/>

<serial>VDA</serial>

<boot order='2'/>

<alias name='virtio-disk0'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>

</disk>

……..

<interface type='bridge'>

<mac address='fc:e7:5a:74:1e:3e'/>

<source bridge='**br0**'/> -🡪 assign name for bridge created in section 2.1

<target dev='**vnet0**'/> -🡪 unique for every virtual machine

<model type='virtio'/>

<link state='up'/>

<boot order='3'/>

<alias name='net0'/>

<address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>

</interface>

**2.3 Start virtual machine.**

**2.3.1 Connect to qemu hypervisor.**

# virsh connect qemu:///system

**2.32 Load the virtual machine configuration file.**

#virsh define xmlDir/x86-64-kvm-guest-nfs-raw.xml -🡪 created in section 2.2

**2.3.3 Start VM.**

This is the first boot of the virtual machine to load the installer's image. Install guest image.

#virsh start VM\_NAME

Verify if virtaul machine runing.

#virsh list

Id Name State

----------------------------------------------------

8 x86-64-kvm-guest running

Can Access GUI of Virtual machine using vnc session :

<input type='mouse' bus='ps2'/>

<graphics type='vnc' port='5900' autoport='yes' listen='0' keymap='en-us'> -🡪 vnc port 5900.

**2.4.4** After succesfull instaalltion of OS. Edit virtual machine configuration file.

<devices>

<emulator>/usr/bin/kvm</emulator>

<disk type='file' device='cdrom'>

<driver name='qemu' type='raw'/>

<source file='**….\*iso**' startupPolicy='optional'> -🡪 remove ISO file name .

</source>

<target dev='hdc' bus='ide'/>

<readonly/>

<serial></serial>

<boot order='1'/>

<alias name='ide0-1-0'/>

Follow step mentioned in section 2.3 to start virtual machine.

**3. Migrating Virtual Machine.**

This exercise assumes that the virtual machine **x86-64-kvm-guest** is running on the OVP node

**SOURCE\_HOST**, and is to be migrated, live, to the OVP node **TARGET\_HOST**. Before proceding with this section verify if

Both machine are accessible to each other.

3.1 Create bridge with same name on **TARGET\_HOST** as created in section 2.1 .

3.2 Enable the shared NFS pool on the TARGET\_HOST.

# scp xmlDir/x86-64-kvm-guest-pool.xml root@TARGET\_HOST\_IP:xmlDir.

Follow the section section 1.3 and 1.4 on Target\_HOST.

3.3 On the SOURCE\_HOST start the migration.

# virsh migrate

--live \

--p2p \ // interface name used for migration

--verbose \

--x86-64-kvm-guest \

qemu+tcp://TARGET\_HOST/system

Verfiy if the migration was suucessful. Run command on TARGET\_HOST.

#virsh list

Id Name State

----------------------------------------------------

8 x86-64-kvm-guest running