About this paper

Virtual machine technology is a powerful capability that can reduce costs while improving utilization of resources. HPE is also applying virtualization to other aspects of the data center and uniting virtual and physical resources to create an environment suitable for deploying mission-critical applications.

HPE Serviceguard for Linux® is certified for deployment on Red Hat® and SUSE Linux Enterprise Server® (SLES) kernel-based virtual machine (KVM) guests. This white paper discusses the different ways in which a KVM guest can be integrated in a Serviceguard for Linux cluster. It describes how a Serviceguard for Linux cluster can be configured using KVM guests from a single host and multiple hosts, as well as a combination of KVM guests and physical machines, to provide high availability for applications.

This white paper provides details on recommended network and storage configurations for VMs used as Serviceguard cluster nodes. In addition, this paper recommends how to eliminate single point of failures and provides pointers to other useful, relevant documents as appropriate.

For the complete list of supported operating systems, certified network and storage configurations, and Red Hat hypervisor versions with the listed version of HPE Serviceguard for Linux release, please refer to the “HPE Serviceguard for Linux Certification Matrix” document at hpe.com/info/linux-serviceguard-docs.

Note

Except as noted in this technical white paper, all HPE Serviceguard configuration options documented in the “Managing HPE Serviceguard for Linux Manual” available at hpe.com/info/linux-serviceguard-docs are supported for Red Hat and SLES KVM guests, and all the documented requirements apply.

Introduction

KVM based VM's and RHEV-H Virtual machines are increasingly being deployed for server consolidation and flexibility. Virtual machine technology allows one physical server to simulate multiple servers, each concurrently running its own operating system (OS). In virtual machine technology, the virtualization layer also known as hypervisor¹ abstracts the physical resources so that each instance of an OS appears to have its own processor, memory, NIC etc., when in fact they are virtual instances. This abstraction allows you to replace numerous existing physical servers with just one or few, but at the cost of greater exposure to single points of failure.

Linux KVM provided by Red Hat Enterprise Linux (RHEL) and SLES as a full virtualization solution or by RHEV-H as thin hypervisors. KVM differs from other popular alternatives like Xen and VMware® in terms of operation, performance, and flexibility. KVM comes as a kernel module, with a set of user-space utilities to create and manage the virtual machines.

HPE Serviceguard for Linux software is designed to protect applications and services from planned and unplanned downtime. By packaging an application or service with its associated resources, and moving that package to other servers as needed, Serviceguard for Linux ensures 24x7 application availability. Packages can be moved automatically when Serviceguard detects a failure in a resource, or manually to perform system maintenance or upgrades. By monitoring the health of each server (node) within a cluster, Serviceguard for Linux can quickly respond to failures such as those that affect processes, memory, LAN media and adapters, disk, operating environments, and more.

HPE Serviceguard for Linux provides a significant level of protection. Specifically, it fails over an application when any of a large number of failures occurs, including:

- Application failure
- Failure of any of the components in the underlying network infrastructure that can cause failure of the application network
- Failure of storage
- An OS “hang” or failure of the virtual machine itself
- Failure of the physical machine

¹ Hypervisor often refers to a layer that resides directly on server hardware, but terms are not used consistently across the industry.
In addition, HPE Serviceguard for Linux provides a framework for integrating custom user-defined monitors, using the generic resource monitoring service.

HPE Serviceguard for Linux, combined with KVM (RHEL, SLES and RHEV-H) software solutions, can protect your applications, while also optimizing the cost, with no compromises on application availability and reliability.

### Terms and symbols

**Table 1. KVM terminology used in this document**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM</td>
<td>Kernel-based virtual machine</td>
</tr>
<tr>
<td>RHEL</td>
<td>Red Hat Enterprise Linux</td>
</tr>
<tr>
<td>SLES</td>
<td>SUSE Linux Enterprise Server</td>
</tr>
<tr>
<td>RHEV-H</td>
<td>Red Hat Enterprise Virtualization-hypervisor</td>
</tr>
<tr>
<td>RHEV-M</td>
<td>Red Hat Enterprise Virtualization-manager</td>
</tr>
<tr>
<td>KVM Host, host, hypervisor</td>
<td>Physical server on which KVM Hypervisor (RHEL, SLES or RHEV-H) is installed</td>
</tr>
<tr>
<td>KVM guest, guest, VM</td>
<td>KVM virtual machine carved out of the hypervisor</td>
</tr>
<tr>
<td>Physical machine</td>
<td>Physical server configured as a Serviceguard cluster node</td>
</tr>
<tr>
<td>Bridge</td>
<td>A device bound to a physical network interface on the host which enables any number of guests to connect to the local network on the host. It is mapped to a physical NIC which acts as a switch to KVM guests</td>
</tr>
<tr>
<td>Cluster, Serviceguard cluster</td>
<td>HPE Serviceguard for Linux cluster</td>
</tr>
<tr>
<td>FC</td>
<td>Fibre Channel storage</td>
</tr>
<tr>
<td>iSCSI</td>
<td>Internet Small Computer System Interface storage (IP based SCSI storage)</td>
</tr>
<tr>
<td>NPIV</td>
<td>N-Port ID Virtualization</td>
</tr>
<tr>
<td>vHBA</td>
<td>Virtual host bus adapter</td>
</tr>
<tr>
<td>NPIV enabled FC</td>
<td>Fibre Channel storage exposed using NPIV enabled vHBA</td>
</tr>
</tbody>
</table>

**Table 2. Symbols used in this document**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual machine guest which is a Serviceguard cluster node</td>
<td></td>
</tr>
</tbody>
</table>
Supported KVM host and guest OS

For the complete list of hypervisor and guest operating systems supported with HPE Serviceguard for Linux, refer to the “HPE Serviceguard for Linux Certification Matrix.”

Supported cluster deployment models with KVM guests

The supported HPE Serviceguard for Linux cluster deployment models when using KVM guests as cluster nodes are as follows:

**Cluster in a box**

In the “Cluster in a box” model, a cluster is formed with KVM guests, all of which are carved out of a single host. Though this configuration provides consolidation of resources, it is not an ideal configuration, as failure of the host will bring down all the nodes of the cluster. Hence, this configuration is not recommended.

![Cluster in a box diagram](image)

**Cluster across box (host shared)**

In the “Cluster across box (host shared)” model, a cluster is formed with multiple guests that are hosted on two or more hosts. The term “host shared” denotes that multiple VMs sharing a host can be configured in the same cluster. When designing such a cluster, one must ensure that the cluster nodes are distributed across the hosts in such a manner, that the failure of any one of the hosts will not result in more than half the cluster nodes going down, thereby creating a single point of failure (SPOF).

Serviceguard is installed on the guests and a cluster is formed with these VMs. Serviceguard provides high availability to the applications deployed as packages in the VMs, and fails over the applications to adoptive nodes in case of failures.

Let us consider the example (figure 2) of a four node cluster. In this case, the correct distribution would be to have two VMs each from Host1 and Host2 configured as Serviceguard cluster nodes, rather than having three VMs from Host1 and one from Host2. In the second case, the failure of Host1 with three VMs will bring down the entire cluster.

![Cluster across box diagram](image)

**Figure 1. Cluster with multiple KVM guests all from same host**

**Figure 2. Cluster with KVM guest nodes across multiple hosts**
**Cluster across box (host exclusive)**

In the “Cluster across box (host exclusive)” model, a cluster is formed with multiple guests, each hosted on a different host. The term “host exclusive” denotes that none of the cluster nodes shares a host as shown in figure 3. In other words, one host can have multiple guests, all of which can be part of different clusters, but no two guests from the same host can belong to the same cluster.

This model does not mandate the use of NPIV-enabled storage infrastructure when using FC devices as shared storage.

Serviceguard is installed on the guests and a cluster is formed with these VMs. Serviceguard provides high availability to the applications deployed as packages in the VMs and fails over the applications to adoptive nodes in case of failures.

The example in figure 3 shows two hosts—Host1 and Host2. Host1 has Guest1 and Guest3, and Host2 has Guest2 and Guest4 hosted on them. Here, two clusters are configured where Cluster1 is configured with Guest1 and Guest2 and Cluster2 is configured with Guest3 and Guest4. In both the clusters, none of the guests shares a host.

![Figure 3. Two clusters with multiple guests where none of them shares a host](image-url)
Hybrid cluster (host shared)
In the “Hybrid cluster (host shared)” model, a combination of one or more physical machines and VMs are used as nodes in a Serviceguard cluster. The VMs may be hosted on multiple hosts. The term “host shared” denotes that multiple VMs sharing a host can be configured in the same cluster. When designing such a cluster one must ensure that the VMs are distributed across the hosts in such a manner that the failure of any one of the hosts will not result in more than half the cluster nodes going down, thereby creating a single point of failure (SPOF).

Serviceguard is installed on the guests and physical machines, and a cluster is formed among them. Serviceguard provides high availability to the applications deployed as packages in the VMs and physical machines. In case of failures, Serviceguard fails over the application to other adoptive cluster nodes. The application can be failed over from a VM to a physical machine and vice versa.

This is a very powerful model where the application can primarily run on the physical machine and, in case of failures, can fail over to an adoptive VM. This enables users to take advantage of the performance of a physical machine, and at the same time allows for consolidation of standby resources.

The example in figure 4 of a four node cluster, a cluster is configured using two physical machines and two VMs. The package can be failed over between any of the cluster nodes. This example shows a correct distribution of VMs where failure of Host1 will not result in more than half of the nodes going down.

![Figure 4. Cluster with KVM guests and physical machine](image)

Hybrid cluster (host exclusive)
In the “Hybrid cluster (host exclusive)” model a combination of one or more physical machines and one or more VMs (all hosted on different hosts) are used as nodes in a Serviceguard cluster. The term “host exclusive” denotes that none of the cluster nodes shares a host as shown in figure 5.

This model does not mandate the use of NPIV-enabled storage infrastructure when using FC devices as shared storage. Serviceguard is installed on the guests and physical machines, and a cluster is formed among them. Serviceguard provides high availability to the applications deployed as packages in the VMs and physical machines. In case of failures, Serviceguard fails over the application to other adoptive cluster nodes. The application can be failed over from a VM to a physical machine and vice versa.

This is a very powerful model where the application can primarily run on the physical machine and, in case of failures, can fail over to an adoptive VM. This enables users to take advantage of the performance of a physical machine, and at the same time allows for consolidation of standby resources.
In the “Cluster with VM as a package” model, Serviceguard is deployed on the KVM hosts and the KVM guests are deployed as Serviceguard packages, and Serviceguard provides high availability for the VMs. Serviceguard starts, stops and monitor the VM that are deployed as packages. In the event of a failure (VM, Host failure, etc.) Serviceguard and will restart the VM on the adoptive Host cluster node. Compared to the VM as a node deployment in this model, Serviceguard monitors the VM and the Host for failures.

This model provides multiple benefits:

1. This model reduces the resources required for standby as there is no need for a dedicated standby VM that needs to be up and running all the time.
2. This model translates to lesser resources like IP addresses, etc. This also makes management easier as one needs to install and maintain only one OS image.
3. This model provides root disk monitoring.

In this deployment model, VM must have its VM image/root disk on a logical volume of a volume group carved out of a shared disk which is available on all the adoptive nodes (Host). Ensure that each VM image is placed on a single volume group, and must be exclusively used only for this purposes. This VM image/root disk must be configured in the package.

This model does not mandate the use of NPIV-enabled storage infrastructure when using FC devices as shared storage.

For more details about this deployment model, please refer to “HPE Serviceguard Toolkit for KVM on Linux User Guide.”
Summary of supported cluster deployment models with various storage configurations

An HPE Serviceguard for Linux cluster that includes virtual machines as cluster nodes have multiple deployment model. Table 3 provides a summary of the supported models with different storage configurations in a Serviceguard cluster. Please refer to the above sections in this document to find out more about each supported model.

Table 3. Snapshot of supported Serviceguard cluster deployment models with various storage configurations with RHEL, SLES and RHEV-H guests

<table>
<thead>
<tr>
<th>Supported cluster models</th>
<th>RHEL</th>
<th>SLES</th>
<th>RHEV-H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FC</td>
<td>NPIV enabled FC</td>
<td>ISCSI</td>
</tr>
<tr>
<td>Cluster in a box</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster across box (host shared)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster across box (host exclusive)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hybrid cluster (host shared)</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hybrid cluster (host exclusive)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster with VM as a package</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extended Distance Cluster (XDC)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Metrocluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Continentalcluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Supported KVM guest network interface device models

When configuring the guest network interfaces the “device model” used must be one of the Serviceguard supported device models as listed in “HPE Serviceguard for Linux Certification Matrix.” Ensure that all interfaces across the KVM guest cluster nodes have the same “device model.”

Configuring a KVM guest

Linux KVM available with RHEL, SLES as a full virtualization solution and with RHEV-H as a thin hypervisors. SLES and RHEL provides a tool, “virt-manager”, which is a very simple, easy-to-use, and intuitive GUI interface to create, manage, and administer virtual machines. A command line alternative, “virsh,” also gives a shell that can be used to create, manage, and administer virtual machines using a rich set of commands.

Similarly, RHEV-H provides a tool—Red Hat Enterprise Virtualization Management (RHEV-M) to create, manage, and administer virtual machine created on RHEV-H.

Here is a high-level overview of the steps required to setup a Serviceguard for Linux cluster using KVM guests.

Before creating KVM guests, ensure that the CPU, memory resources required by the KVM guests are available and if the virtualization support is enabled at the BIOS on all KVM hosts to be used in the cluster.

1. Make sure that the required KVM packages are installed on the hosts.
2. Make sure that the service “libvirtd” is running on the hosts where KVM guests are to be created.
3. Create the KVM guests and install the required operating system in the KVM guests. HPE Serviceguard for Linux does not have any restriction on where the boot image will be stored. You can use “virt-manager” for creating and configuring the KVM guests. For the list of supported guest OSs, refer to Supported KVM host and guest OS with Serviceguard for Linux.
4. Complete the network configuration as mentioned in the section Configure resilient network for Serviceguard cluster using KVM guests on all guests.

2 Hypervisor often refers to a layer that resides directly on server hardware, but terms are not used consistently across the industry.
5. Complete the storage configuration as mentioned in the section Configure shared storage for Serviceguard cluster on KVM guests on all guests.

**Configuring Serviceguard cluster on KVM guests**

Install HPE Serviceguard for Linux and its prerequisites on all the KVM guests that need to be in the cluster. For information about installing and updating Serviceguard, see the latest HPE Serviceguard for Linux Release Notes at [hpe.com/info/linux-serviceguard-docs](http://hpe.com/info/linux-serviceguard-docs).

Configure and creating HPE Serviceguard for Linux cluster and other required resources like packages, service, etc. For more information on how to configure, refer to Managing HPE Serviceguard for Linux at [hpe.com/info/linux-serviceguard-docs](http://hpe.com/info/linux-serviceguard-docs).

**Configuration requirements for a Serviceguard cluster with KVM guests**

**Configured resilient network**

HPE Serviceguard for Linux recommends having a highly resilient network configuration with redundant heartbeats and redundant data networks to avoid single point of failures. The following section describes how to achieve network redundancy using bridged network configuration (also known as physical device sharing).

There are multiple ways of configuring a resilient network using networking technologies like bridging and bonding, at different layers. Figure 7 shows a simple example of a resilient network configuration using bridging at host and bonding at KVM guests.

Figure 7 shows a typical network configuration where four KVM guest across two physical hosts are bridged to form an HPE Serviceguard for Linux cluster with redundant heartbeat networks.

![Figure 7. Typical network configurations for RHEL (KVM) host and guest](image)
### Table 4. Network configuration details

<table>
<thead>
<tr>
<th>Node name</th>
<th>Interface name</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM Guest SG_NODE1</td>
<td>bond0 (eth0 and eth1)</td>
<td>Assign heartbeat IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE2</td>
<td>bond0 (eth0 and eth1)</td>
<td>Assign heartbeat IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE3</td>
<td>bond0 (eth0 and eth1)</td>
<td>Assign heartbeat IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE4</td>
<td>bond0 (eth0 and eth1)</td>
<td>Assign heartbeat IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE1</td>
<td>eth2</td>
<td>Assign public IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE2</td>
<td>eth2</td>
<td>Assign public IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE3</td>
<td>eth2</td>
<td>Assign public IP</td>
</tr>
<tr>
<td>KVM Guest SG_NODE4</td>
<td>eth2</td>
<td>Assign public IP</td>
</tr>
<tr>
<td>KVM Host1 and KVM Host2</td>
<td>bridge0 (uses eth0 as slave)</td>
<td>Used as “Source device” for eth0 in all KVM guests</td>
</tr>
<tr>
<td>KVM Host1 and KVM Host2</td>
<td>bridge1 (uses eth1 as slave)</td>
<td>Used as “Source device” for eth1 in all KVM guests</td>
</tr>
<tr>
<td>KVM Host1 and KVM Host</td>
<td>bridge2 (uses eth2 as slave)</td>
<td>Used as “Source device” for eth2 in all KVM guests</td>
</tr>
<tr>
<td>VM Host1 and KVM Host2</td>
<td>eth0</td>
<td>Connected to heartbeat network switch</td>
</tr>
<tr>
<td>VM Host1 and KVM Host2</td>
<td>eth1</td>
<td>Connected to heartbeat network switch</td>
</tr>
<tr>
<td>KVM Host1 and KVM Host2</td>
<td>eth2</td>
<td>Connected to public network</td>
</tr>
<tr>
<td>KVM Host1 and KVM Host2</td>
<td>eth3</td>
<td>Can be used for host management (not mandatory)</td>
</tr>
</tbody>
</table>

The bridge network configuration must be performed in two parts:

- Network configuration on host
- Network configuration on guest

**Note**

The steps listed after this section are based on the respective OS documentation; for any updates or alternatives of these steps refer to the latest versions of

- Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide
- Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide
- SUSE Linux Enterprise Server 12 SP2 Virtualization Guide

Similarly, to configure a network for RHEV-H and their guests, use RHEV-M. For detailed steps to configure network, refer to the latest version of the “Administration Guide.”
**Network configuration on host**

This bridge can be created using the following steps:

1. Create a new network interface script file in "/etc/sysconfig/network-scripts/" directory with the name "ifcfg-<device_name>" where the <device_name> must match the value of the DEVICE parameter inside the file. Also disable NetworkManager for the interface to be bridged by add "NM_CONTROLLED=no" to the ifcfg-* network script being used for the bridge. Its contents are as follows:

   DEVICE=bridge0
   TYPE=Bridge
   ONBOOT=yes
   DELAY=0
   NM_CONTROLLED=no

   **Note**
   The line TYPE=Bridge is case sensitive. It must have uppercase “B” and lowercase “ridge”

2. Add the physical interface to the bridge by modifying the network interface script of the given physical interface. Edit the file and add a line BRIDGE=bridge0, so that the contents of the configuration file look like the following example:

   DEVICE=eth0
   BRIDGE=bridge0
   BOOTPROTO=None
   HWADDR=00:19:b9:7e:c8:63
   ONBOOT=yes
   TYPE=Ethernet
   IPV6INIT=no
   NM_CONTROLLED=no

   **Note**
   Repeat steps 1 and 2 for every interface that requires to be bridged.

3. Configure IP tables to allow all traffic to be forwarded across the bridge

   # iptables -I FORWARD -m physdev --physdev-is-bridged -j ACCEPT
   # service iptables save
   # service iptables restart

   Alternatively, prevent bridged traffic from being processed by iptables rules by appending these lines in the file

   `/etc/sysctl.conf`:
   
   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0

   Reload the kernel parameter configured with sysctl by executing this command:

   # sysctl –p /etc/sysctl.conf
4. Restart the network services to bring all the network configuration changes into effect.
   
   ```
   # service network restart
   ```

5. Verify that your eth0 was added to the bridge0 using the brctl show command.
   
   ```
   # brctl show
   ```
   The output must look similar to the following:

<table>
<thead>
<tr>
<th>bridge name</th>
<th>bridge Id</th>
<th>STP enabled interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>virbr0</td>
<td>8000.000000000000</td>
<td>yes</td>
</tr>
<tr>
<td>bridge0</td>
<td>8000.0019b97ec863</td>
<td>yes eth0</td>
</tr>
</tbody>
</table>

   “bridge0” is now available through virt-manager and libvirt. Guests can now connect to this device for full network access. This can be also viewed through virt-manager as shown here:

   Click on Edit ➔ Connection Details ➔ Network Interfaces. You can see the list of bridge interface created in figure 8.

   ![virt-manager window of all network interfaces on guest](image)

**Network configuration on guest**

Network access is provided to the guest VMs using the underlying bridged network interfaces configured in the host as shown in the previous steps. To achieve this, add the required virtual interfaces to the guest and configure the appropriate “Source device” and “Device model” for each of them.

Step 1: Open virt-manager by executing the command “virt-manager.”

Step 2: From the list of VM select the VM to which the interfaces are to be added and click to open its window.

Step 3: Go to “Details” view, and click “Add Hardware.” As shown in figure 9.
Step 4: Click the “Network” tab, choose the “Host device” and “Device model” and click “Finish”. “Host device” should be the appropriate bridge interface, which has been created on the host as explained under Network configuration on host section.

Step 5: Repeat the steps 1–4 for all the required network interfaces on the guest.

Configure shared storage

HPE Serviceguard for Linux is a shared storage HA cluster that requires all applications deployed as Serviceguard packages to have their data on shared storage so as to enable access from all cluster nodes. In shared storage clustering environments, data integrity is of utmost importance and to ensure that, during all failover scenarios Serviceguard uses SCSI-3 Persistent Reservation. For more information, see the section “About Persistent Reservations” in the latest edition of Managing Serviceguard available at: hpe.com/info/linux-serviceguard-docs.

HPE Serviceguard for Linux supports various cluster layouts with FC and iSCSI shared storage, refer Summary of supported cluster deployment models with various storage configurations. Further in this section, we will discuss how to configure FC and iSCSI shared storage in KVM guest.
Configure iSCSI devices as shared storage for KVM guest nodes

Serviceguard for Linux supports iSCSI devices exposed using software initiator only. The iSCSI devices must be directly exposed to the software initiator configured in the guest as shown in the figure 11.

Serviceguard for Linux supports use of iSCSI shared storage with KVM guests carved out from SLES (KVM), RHEL (KVM) and RHEV-H.

![High-level architecture diagram for using iSCSI shared storage in Serviceguard cluster environments](image)

The steps to discover and login to iSCSI targets from the guests can be found in the appropriate operating system manuals as listed below.

For RHEL refer to:
- Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide
- Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide

For SUSE refer to:
- suse.com/documentation/sles-12/pdfdoc/stor_admin/stor_admin.pdf

It is also recommended to have multipath configured for iSCSI devices.

Configure Fibre Channel devices as shared storage for KVM guest nodes

When using FC devices as shared storage, it must be configured to allow SCSI-3 PR operations from the guest. To allow SCSI-3 PR operations, the FC devices must be configured as pass-through devices as shown in figure 12. The OS image’s disk need not be configured as pass-through.
Note
VM created using RHEV-H do not support FC devices as shared storage.
Table 5. Snapshot of supported Serviceguard deployment models with different type of Adapter (HBA) using which FC device can be exposed to KVM guests

<table>
<thead>
<tr>
<th>Supported cluster models</th>
<th>Physical</th>
<th>Adapter (HBA) type</th>
<th>Virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single NPIV enabled vHBA on a physical HBA</td>
<td>Multiple NPIV enabled vHBA on same physical HBA</td>
<td></td>
</tr>
<tr>
<td>Cluster in a box</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster across box (host shared)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster across box (host exclusive)</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Hybrid cluster (host shared)</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Hybrid cluster (host exclusive)</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Cluster with VM as a package</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Extended Distance Cluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Metrocluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Continentalcluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

“virsh attach-device” command uses XML configuration file to create SCSI LUN on guest. Serviceguard requires certain attributes to set as mentioned in Table 6 and Table 7 for a XML configuration:

1. The `<shareable>` element must be set in XML configuration to share a LUN between guests.
2. Required value for each attribute in XML configuration must be set.

For exposing a block device using a physical HBA to KVM guest as a SCSI LUN

Table 6. Attribute name and required values can be used in XML

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>block</td>
<td>To define the type of source device</td>
</tr>
<tr>
<td>device</td>
<td>lun</td>
<td>To define type of target device. The name “Target dev” must be unique and a device of the same name should not be exist in KVM guests</td>
</tr>
<tr>
<td>sgio</td>
<td>unfiltered</td>
<td>To allow guest-issued SGIO ioctl commands to pass-through, for persistence reservation implementation</td>
</tr>
<tr>
<td>bus</td>
<td>scsi</td>
<td></td>
</tr>
</tbody>
</table>

For exposing a storage pool volume using a NPIV enabled HBA to KVM guest as a SCSI LUN

Table 7. Attribute name and required values can be used in XML

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>volume</td>
<td>To define the type of source device</td>
</tr>
<tr>
<td>device</td>
<td>lun</td>
<td>To define type of target device. The name “Target dev” must be unique and a device of the same name should not be exist in KVM guests</td>
</tr>
<tr>
<td>sgio</td>
<td>unfiltered</td>
<td>To allow guest-issued SGIO ioctl commands to pass-through, for persistence reservation implementation</td>
</tr>
<tr>
<td>bus</td>
<td>scsi</td>
<td></td>
</tr>
</tbody>
</table>

The steps to expose the RHEL host’s shared FC disks to guests as pass-through and configuring persistent NPIV/vHBA, can be found in this link: Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide.
Below example to illustrate how to expose an FC device having two paths to the host (in this case light4), as SCSI pass-through device to KVM guest (in this case nickel3), which can be used for Cluster across box (host exclusive) deployment model. This is just a reference for any change in command and XML attribute, please refer latest version of Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide.

1. Check for a pre-existing SCSI controller on the KVM guest (nickel3), run the following command on KVM hypervisor (light4).

```bash
# virsh dumpxml nickel3 | grep controller.*scsi
```

If a device controller is present, the command will list one or more lines similar to the following:

```xml
<controller type='scsi' model='virtio-scsi' index='0'/>
```

2. If previous step did not show a device controller on guest (nickel3), create the description for a new controller in a new file and add it to the virtual machine, using the following steps:

   a. Create the device controller by writing a `<controller>` element in a new file and save this file with an XML extension. virtio-scsi-controller.xml on host (light3), for example.

   ```xml
   <controller type='scsi' model='virtio-scsi'/>
   ```

   b. Attach the device controller you just created in virtio-scsi-controller.xml with your guest virtual machine (nickel3), run the following command on the host (light3):

   ```bash
   # virsh attach-device --persistent nickel3 controller_scsi.xml
   ```

   **Note**

   The “virsh attach-device” command supports multiple options out of which only the “--persistent” option must be used. This is to ensure that added controllers will continue to exist on KVM guests even after a guest or host reboot.

3. Expose all required paths for the FC device individually to the KVM guests. To expose an FC device path from host to KVM guest, find the corresponding stable path under “/dev/disk/by-path” directory.

   For example if “/dev/sdb” is one of the FC device paths on the host, then find the stable path using the following command:

   ```bash
   # ls -l /dev/disk/by-path/ | grep sdb
   lwxcwxicwx.. 1 cot cot 5 Aug 10 21:28 pci-0000:08:00.0-fc-0x500143b011382ad8-lun-1-> ../../sdb
   ```

4. Create an empty XML file `<sdb> xml and populate with information as shown below.
The "source dev" used here is the stable path found in step 3 and "target dev" is the device name as it appears in the guest after attaching the device. In some cases, if the name is already used by the guest, then a different name will be automatically selected. That may appear on KVM guest after the attach.

Make sure specify all required values for corresponding attribute in XML as explained in table 6.

5. Attach a device path to KVM guest, run "virsh attach-device" command on host and make it persistent using the "--persistent" option:

```bash
# virsh attach-device --persistent nickel3 sdb.xml
Device attached successfully
```

Note
"virsh attach-device" command supports multiple options out of which only the "--persistent" option must be used. This is to ensure that all exposed FC device paths will continue to exist on KVM guests even after a guest reboot.

6. After successful execution of above command on host as explained in step 5, the device path must be visible to the KVM guest; verify it by running "fdisk" command on KVM guests (below snapshot a new device name "sda" has appeared):

```bash
# fdisk -l |grep -i disk
Disk /dev/vda: 8589 MB, 8589934592 bytes Disk
  identifier: 0x00058bac
Disk /dev/mapper/vg_nickel2-lv_root: 7205 MB, 7205814272 bytes Disk
  identifier: 0x00000000
Disk /dev/mapper/vg_nickel2-lv_swap: 855 MB, 855638016 bytes Disk
  identifier: 0x00000000
Disk /dev/sda: 2147 MB, 2147483648 bytes Disk
  identifier: 0x00000000
```

Note
Device path name appearing on the guest may not be the same as specified in the XML.
7. To expose another path of FC disk from host to KVM guest, repeat steps 3 to 5.

In this example, the FC disk used has two paths. After successful execution of steps 3 to 5, another device path must be made visible in the KVM guest. This can again be verified by running the “fdisk” command on the KVM guests (the following snapshot shows the new device name “sdb”):

```
# fdisk -l |grep -i disk
Disk /dev/vda: 8589 MB, 8589934592 bytes Disk
  identifier: 0x00058bac
Disk /dev/mapper/vg_nickel2-lv_root: 7205 MB, 7205814272 bytes Disk
  identifier: 0x00000000
Disk /dev/mapper/vg_nickel2-lv_swap: 855 MB, 855638016 bytes Disk
  identifier: 0x00000000
Disk /dev/sda: 2147 MB, 2147483648 bytes Disk
  identifier: 0x00000000
Disk /dev/sdb: 2147 MB, 2147483648 bytes Disk
  identifier: 0x00000000
```

Repeat steps 3 to 5, to expose all required paths to a KVM guest. It is recommended to have multipath configured for FC devices using **Device Mapper Multipath Software**.

8. Repeat all the steps from 1–8, to expose all required paths to all KVM guests from where the shared device will be accessed.

**Restrictions and exclusion**

1. Lock LUN as arbitration is not supported with iSCSI disk; use Quorum server for arbitration.
2. Live migration of KVM guests that are configured as Serviceguard cluster nodes is not supported.
3. FC devices cannot be used as shared storage, when guests created using RHEV-H and SLES are used as Serviceguard cluster nodes.
4. NPIV enabled FC cannot be used as shared storage when guests created using RHEV-H and SLES are used as cluster nodes.
5. Serviceguard supports NPIV enabled FC devices exposed to guest via persistent vHBA only.
6. Serviceguard supports exposing FC devices to multiple KVM guest via multiple NPIV enabled vHBAs created on a physical HBA, here for each guest one NPIV enabled vHBA must be created on a physical HBA.
Summary
This white paper describes best practices for deploying HPE Serviceguard for Linux cluster in a KVM environment. It is not the intent of this document to duplicate the strategies and best practices of other HPE or Red Hat or SLES technical white papers. The strategies and best practices offered here are presented at a very high level to provide general knowledge. Where appropriate, you are referred to specific documentation that provides more detailed information.

References
- Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide
- Red Hat Enterprise Linux 6 Virtualization Getting Started Guide
- Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide
- Red Hat Enterprise Linux 6 Virtualization Administration Guide
- Red Hat Enterprise Virtualization Guide
- SUSE Linux Enterprise Server 11 SP4 Storage Administration Guide
- SUSE Linux Enterprise Server 12 SP2 Storage Administration Guide
- SUSE Linux Enterprise Server 12 SP2 Virtualization Guide

Learn more at
hpe.com/info/linux-serviceguard-docs