Univention Corporate Server

Extended virtualization documentation

Table of Contents

1. Operating storage pools using iSCSI 4
   1.1. Requirements 4
   1.2. Integration of iSCSI storage pools 4
   1.3. Use of iSCSI storage pools 5
2. Xen deprecated in UCS-4 6
   2.1. Switch to QEMU/KVM 6
   2.2. Switch to Debian-Xen 11
3. Network setup for virtual machines 13
   3.1. Migrating virtual machines between hosts 14
   3.2. UVMM profiles 15
Bibliography 16

Chapter 1. Operating storage pools using iSCSI

iSCSI is a standard for connecting SCSI storage devices via a TCP-based network connection. iSCSI makes it possible to use a wide range of professional storage solutions for the storage of virtual machines. This documentation describes the connection of an iSCSI storage device (also known as a *target*) as a storage pool for the UCS Virtual Machine Manager.

1.1. Requirements

Feedback

- The ***open-iscsi*** package must be installed on the virtualization servers.
- The iSCSI storage device must be configured in such a way that the virtualization servers can access it: The servers use the name stored in the `/etc/iscsi/initiatorname.iscsi` file as the initiator name. The name may still need to be generated automatically, which can be done by running `/etc/init.d/open-iscsi restart` once.
- The libvirt interface used by UVMM does not currently support any authentication. If CHAP is used, the `/etc/iscsi/iscsid.conf` file must be adjusted by hand.

1.2. Integration of iSCSI storage pools

Feedback

UVMM does not yet support the creation of storage pools via the UMC. The creation of storage pools is described below.

To make an additional storage pool available on a virtualization server, it is necessary to create an XML file with a description. The creation of storage pools is documented in the virtualization chapter of the UCS manual. Below, a storage pool is defined which accesses an iSCSI storage device.

- The *type* attribute must be set to the `iscsi` value in the *pool* element.
- A *source* element is used to specify the source servers (iSCSI target).
- The *target* element is used to define the path to the device files. The `/dev/disk/by-path` should be used there for stable names.

In the following example, a storage pool is defined, which mounts the `iqn.2010-08.local.ucs:sdb` iSCSI target from the `xen6.ucs.local` server:

```
<pool type='iscsi'>
 <name>iscsi</name>
 <source>
  <host name='xen6.ucs.local' port='3260'/>
  <device path='iqn.2010-08.local.ucs:sdb'/>
 </source>
 <target>
  <path>/dev/disk/by-path</path>
 </target>
</pool>
```

This description must firstly be saved in a file (for example, in the `iscsi-pool.xml` file). For it to be possible to use this storage pool, the following commands must be run:

```
virsh pool-define iscsi-pool.xml
virsh pool-start iscsi
virsh pool-autostart iscsi
```

The UVMM service will detect the new storage pool automatically after a certain period of time. Alternatively, the service can also be restarted once so that this new storage pool information is detected immediately:

```
/etc/init.d/univention-virtual-machine-manager-daemon restart
```

1.3. Use of iSCSI storage pools

Feedback

Hard disks in UVMM can then be used from the storage pool. When doing so, ensure that no new images are saved in an iSCSI storage pool and only existing ones are selected. The individual LUNs are offered as possible images.

If new iSCSI LUNs are set up, the iSCSI storage pool must still be re-imported manually at present. This can be done using the following command:

```
virsh pool-refresh "$pool_name"
```

For the example above:

```
virsh pool-refresh iscsi
```

Chapter 2. Xen deprecated in UCS-4

For UCS-4.0 Univention decided to drop support for the Xen hypervisor in favor of QEMU/KVM. UCS-4 no longer contains the package ***xen-4.1*** as previously provided by Univention, but only the package ***xen*** as maintained by Debian.

Managing Xen virtual machines in no longer possible using UCS Virtual Machine Manager from UCS-4. Customers using Xen can continue using existing UCS-3 virtualization servers with UVMM from UCS-3 by not updating them, but should consider one of the following options long-term:

- switch to the hypervisor QEMU/KVM, which requires updating all virtual machines.
- switch to the Debian based Xen.

Univention recommends to switch to KVM.

2.1. Switch to QEMU/KVM

Feedback

When switching from Xen to KVM the drivers for para-virtual block and network devices must be changed. This switch is easier for Linux based virtual machines, as the Linux kernel already contains drivers for both hypervisors and automatically uses the right one. For Microsoft Windows bases systems the GPLPV drivers must be un-installed completely, before the replacement drivers VirtIO for KVM can be installed.

Univention recommends to setup a new virtualization server using KVM and to copy over the exiting image files to the new server. This has the advantage that the new setup can be tested thoroughly and a fallback to the old hypervisor using Xen is possible.

Procedure 2.1. Convert virtual machines from Xen to KVM

1. Backup all disk images of the virtual machine. The exact file path can be queried from UCS Virtual Machine Manager or by running the command `virsh domblklist "$vm"`, where `$vm` is the name or UUID of one virtual machine.

2. Create a backup of the XML definition of the virtual machine using the command `virsh dumpxml --inactive "$vm" >"$vm.xml"`.

3. Prepare the virtual machine for migration by performing the following actions inside the running virtual machine:

   - For a Linux based virtual machine perform the following steps:

     a. Convert any reference to Xen specific block devices like `/dev/xvda1` in `/etc/fstab`. Use a device name independent format like `LABEL=` or `UUID=` as this is the most reliable format. The use of fixed names like `/dev/vda1` should be avoided as the order of the devices might change between reboots.

        The device name for CD-ROMs with QEMU/KVM is `/dev/sr0` or `/dev/vdb`, depending on the exact setup of the virtual machine.

     b. Make sure GRUB is installed in the directory `/boot/` and in the master boot record (MBR) of the first hard disk. You can use commands like `fdisk /dev/xvda` for MBR or `gdisk /dev/xdva` for GPT based disks to check the partition layout. Use `grub-install /dev/xdva` to install the boot loader into the MBR.

c. Run `update-grub` to create an up-to-date file `/boot/grub/grub.cfg`. Check that file for any Xen specific device names. Modify the file using your preferred editor and change them to the QEMU/KVM equivalent names as described in the first step.

d. Check the file `/etc/inittab` for any getty running on `hvc0` or any other Xen specific console. Either remove those lines or convert them to a traditional serial console `ttyS0`, which is supported by QEMU/KVM.

e. Check all Univention Configuration Registry variables still referencing Xen specific devices and convert them to the appropriate equivalent:

```
ucr search --value '^(/dev/)?xvd[a-z]+[0-9]*$'
```

f. Check the Univention Configuration Registry variable `kernel/modules` for any Xen specific modules and remove them. The variable is a list of semicolon separated values.

- For Windows the *GPLPV drivers for Xen* must be removed completely, otherwise Windows will crash. The procedure is described in the Univention Wiki: Installing signed GPLPV drivers[1].

  After a reboot the new *VirtIO drivers for QEMU/KVM* can be installed. They are provided by the package ***univention-kvm-virtio*** and are available as an ISO image in `/var/lib/libvirt/images/`. The exact procedure differs between Windows versions and is beyond the scope of this documentation. See the virtualization chapter in the [ucs-manual] for further details.

4. Shut down the virtual machine cleanly. Suspending the machine is not enough!

5. The image format for storing the virtual machine data can be changed optionally: Xen only supports “raw” images, while QEMU/KVM supports more advanced formats like “qcow2”, which supports snapshots. If you plan to use those features, the conversion can be done using a command like `qemu-img convert -f raw -O qcow2 *input*.raw *output*.qcow2`.

   Remember to change the format from “raw” to “qcow2” in the domain definition later on.

After this the virtual machine can be migrated to KVM. There are two alternatives:

- The recommended upgrade path is to setup a *new server* using KVM. This has the advantage that the old server is still available if anything goes wrong or the migration is aborted.

  If you are using a shared storage, make sure that you have backups of all files and that you are using the correct images. If you don't have a shared storage, copy the image file to the new server into `/var/lib/libvirt/images/` or any other storage pool you might have configured.

- Otherwise the virtualization server must be upgraded *in-place*.

Warning

This is a one-way upgrade path. If anything goes wrong, the server has to be restored from backup and must be re-joined!

Procedure 2.2. Migrate server from Xen to KVM in-place

1. Shut down all virtual machines running on this server.

2. Uninstall the App “Xen virtualization server”.

[1] http://wiki.univention.de/index.php?title=Installing-signed-GPLPV-drivers

3. Run the *un-join scripts*: Go to the Univention Management Console and open the *Join* module. Click the button to run all pending join scripts.

4. Perform the update to UCS-4 using `univention-upgrade` or the update module of the Univention Management Console.

5. Install the App “KVM virtualization server”.

6. Reboot the server: This removes the Xen hypervisor below the currently running kernel and also loads the updated kernel from UCS-4.

Now the virtual machines must be re-defined. Depending on the number of virtual machines there are two alternatives:

- If only a few virtual machines need migration, it is easier to create new virtual machines which re-use the old images:

  Procedure 2.3. Re-define virtual machines only re-using images

  1. Go to the Univention Management Console of the QEMU/KVM server and open the *UCS Virtual Machine Manager* module.

  2. Create a new virtual machine using the profile matching the guest operating system.

  3. When asked for the disk images, select “Choose existing image” as “Drive type” and select the existing image from the corresponding storage pool.

     Repeat this step for all drives.

     Pay special attention to the “Image format” if you performed the conversion from “raw” to “qcow2” mentioned above.

  4. Finish creating the machine but do not start it yet.

  5. Re-open the virtual machine configuration and go to the “Devices” tab. Edit all network interfaces and change the “MAC address” back to the original one. You can get it for example from the XML definition file. Otherwise the operating system might detect the change as a new network interface and will require further manual updates like deleting `/etc/udev/rules/70-persistent-net.rules` for Linux or re-configuring the network in Windows.

     Also verify that the “Source” setting matches the name of your bridge interface, as this might have changed from `eth0` to `br0`. See Chapter 3 for more details.

  6. Save the modified virtual machine configuration.

- If many virtual machines need updating, the XML definition from Xen can be semi-automatically transformed into a XML definition for QEMU/KVM.

  Procedure 2.4. Re-define virtual machines converting the Xen XML definition

  1. Use the command `virsh dumpxml --inactive "$vm" >"$vm.xml"` to generate the XML file `$vm.xml` describing a single virtual machine on the Xen host.

  2. Perform the following changes to the XML document, which are described using the XPath[2] notation:

[2] http://www.w3.org/TR/xpath/

a. Change the attribute `/domain/@type` from `xen` to `kvm`.

b. QEMU/KVM only supports full-virtualization (HVM).

   i. Set the element `/domain/os/type` to `hvm`.

   ii. Remove the elements `/domain/bootloader` and `/domain/bootloader_args`.

   iii. Make sure there is an element `/domain/features` enabling Advanced Configuration and Power Interface (ACPI) and Advanced Programmable Interrupt Controller (APIC). They are required for shutdown via ACPI power button event and for interrupt handling on multi-processor systems. If not insert the following XML fragment just after `/domain/os`:

```
<features>
 <acpi/>
 <apic/>
</features>
```

c. Remove the element `/domain/os/loader` for HVM domains.

d. The element `/domain/devices/emulator` should point to `/usr/bin/kvm`.

e. Update each block device `/domain/devices/disk`:

   i. In the sub-element `driver` change the following attributes:

      A. Change `@type='tap2'` to `qemu`.

      B. Change `@name='aio'` to `raw`. If you converted the image file as mentioned above, use `qcow2` instead.

      C. If live-migration is used, make sure to change `@cache='none'` or add it if it is missing.

   ii. In the sub-element `target` change the following attributes:

      - Replace any `@bus='xen'` with `virtio` and convert the attribute `@dev` from `xvd`*X* to `vd`*X*.

      - Use `@bus='ide'` and `@dev='hd`*X*`'` for CD-ROM drives or if an emulated IDE device is preferred to the more efficient VirtIO.

f. Update each network interface `/domain/devices/interfaces`:

   i. Change `model/@type` to `virtio` or any other type supported by QEMU/KVM like `e1000` or `rtl8139`.

   ii. Remove any sub-element `script`.

   iii. Check that the attribute `source/@bridge` references a valid bridge interface on the target host system. See Chapter 3 for more details.

g. In any element `/domain/devices/console/target` for consoles changed the attribute from `@type='xen'` to `@type='serial'`.

h. In any element `/domain/devices/input` for absolute pointing devices change the attribute from `@bus='xen'` to `@bus='usb'`.

3. On the target host define the virtual machine using `virsh define "$vm.xml"` where *`$vm.xml`* is the name of the changed XML file.

The following XSLT style sheet file `xen2qemu.xslt` can be used to transform a Xen domain definition into a QEMU/KVM definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
 <xsl:output method="xml" encoding="UTF-8" indent="yes" omit-xml-
declaration="yes"/>
 <xsl:template match="node()" priority="-1">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
   <xsl:apply-templates select="node()|text()|comment()|processing-
instruction()"/>
  </xsl:copy>
 </xsl:template>

 <!-- Convert Xen into QEMU/kvm domain -->
 <xsl:template match="/domain">
  <domain type="kvm">
   <xsl:apply-templates select="*"/>
   <!-- Make sure ACPI is enabled -->
   <xsl:if test="not(features)">
    <features>
     <acpi/>
     <apic/>
    </features>
   </xsl:if>
  </domain>
 </xsl:template>

 <!-- Make sure domain is HVM -->
 <xsl:template match="/domain/os/type">
  <type>hvm</type>
 </xsl:template>

 <!-- Remove boot loader from Xen-PV domains -->
 <xsl:template match="/domain/bootloader"/>
 <xsl:template match="/domain/bootloader_args"/>

 <!-- Remove BIOS loader from Xen-HV domain -->
 <xsl:template match="/domain/os/loader"/>

 <!-- Change emulator to QEMU -->
 <xsl:template match="/domain/devices/emulator">
  <emulator>/usr/bin/kvm</emulator>
 </xsl:template>

 <!-- Convert Xen block-tap2 devices -->
 <xsl:template match="/domain/devices/disk/driver[@name='tap2']">
  <driver name="qemu" type="raw" cache="none">
   <xsl:copy-of select="@*[name()!='type' and name()!='name']"/>
  </driver>
 </xsl:template>
 <xsl:template match="/domain/devices/disk/target[@bus='xen']">
  <target bus="virtio" dev="vd{substring-after(@dev,'xvd')}">
   <xsl:copy-of select="@*[name()!='bus' and name()!='dev']"/>
  </target>
 </xsl:template>

 <!-- Convert Xen netfront devices -->
 <xsl:template match="/domain/devices/interface/
model[@type='netfront']">
  <model type="virtio"/>
 </xsl:template>
 <xsl:template match="/domain/devices/interface/script"/>

 <!-- Convert Xen console -->
 <xsl:template match="/domain/devices/console/target[@type='xen']">
  <target type="serial">
   <xsl:copy-of select="@*[name()!='type']"/>
  </target>
 </xsl:template>

 <!-- Convert Xen specific input devices -->
 <xsl:template match="/domain/devices/input[@bus='xen']">
  <input bus="usb">
   <xsl:copy-of select="@*[name()!='bus']"/>
   <xsl:apply-templates select="*"/>
  </input>
 </xsl:template>
</xsl:stylesheet>
<!-- vim:set ts=2 sw=2 noet foldmethod=marker: -->
```

It requires an XSLT command line processor, for example the one provided by the package ***xsltproc***. Running a command like `xsltproc xen2qemu.xslt "xen/$vm.xml" >"qemu/$vm.xml"` produces an XML file, which can be copied to the QEMU/KVM host to re-define the virtual machine there as described above.

2.2. Switch to Debian-Xen

Feedback

The major differences between the UCS version and Debian version are:

No UCS Virtual Machine Manager integration	Virtual hosts using Xen can not be managed with UVMM in UCS-4 as support for Xen has been removed completely.
Different Xen patch set	The UCS version contained several patches on top of the Debian version, which fixed several issues (Bug >18357[2], Bug 23394[2], Bug 23812[2], Bug 36098[2]). Their status in Debian is unknown.

[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=>18357
[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=23394
[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=23812
[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=36098

Different libvirt patch set	The UCS version of ***libvirt*** contained several patches specifically for Xen, which are no longer applied (Bug 20024[2], Bug 29532[2]).
No blktap Linux kernel module	The UCS Linux kernel contained the module ***blktap*** built-in, which was used to provide block devices for virtual machines. This module is no longer included in the default kernel, but can be self-compiled using DKMS and the package ***blktap-dkms***.
Different network setup	UCS used a custom network script to setup bridging. It is not used by Debian. As such bridging must be setup manually. See Chapter 3 and the chapter on configuring bridges in the [ucs-manual] for more issues.
Different package layout	UCS-3 had its own source code package for Xen, which was used to build several binary packages. This might break an automatic update, as files might be moved between packages. The package manager dpkg will then refuse to install the Debian packages. As such it is advisable to remove the old UCS packages completely before installing the new Debian based packages. The virtual machine definitions and image files are not effected by the removal.

The following commands can be used as a start to migrate to the Debian version of Xen.

Warning

This is a one-way upgrade path. Backup the host and test the procedure in isolation before doing this on a production system!

```
# Remove xen-4.1 before update
apt-get remove xen-4.1 libxenstore3.0 libxen4.1 libxen-dev xen-4.1-dbg
# Upgrade to UCS-4
univention-upgrade
# Re-install xen
ucr set repository/online/unmaintained=yes
univention-install xen
```

After this more manual configuration might be needed, which is beyond the scope of this documentation.

[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=20024
[2] https://forge.univention.org/bugzilla/show_bug.cgi?id=29532

Chapter 3. Network setup for virtual machines

With UCS-3 a script renamed the physical interface `ethX` into `pethX` and created a bridge interface with the old name as a replacement. This is no longer recommended as this script only supports a very basic setup and is confusing to most users. As such UCS was changed to support bridging, bonding and VLANs, which is described fully in the [ucs-manual].

Unfortunately this leads to a user visible change of the network interface, as `ethX` now no longer is the name of the bridge interface but that of the physical interface again. As this interface name is referenced in the domain configurations of all existing virtual machines, they will no longer start. This is further complicated by the fact, that each snapshot and saved virtual machine also contains the old information, which should be updated.

Procedure 3.1. Convert virtual machines to new network bridge setup

1. Repeat the following steps on each host and for each virtual machine to be updated. Use the command `virsh list --all` to get a list of all defined domains.
2. For each virtual machine named *`$vm`* update the persistent domain configuration using the following steps:
    a. Run the command `virsh edit "$vm"` to get the configuration opened in an editor.
    b. In the XML file lookup all network interfaces below `/domain/devices/interfaces/source` and change the attribute `bridge` from `eth0` to `br0` (or whatever).
    c. Quit the editor to save the file.
3. If the virtual machine has snapshots, they need to be updated as well. Use the command `virsh snapshot-list "$vm"` to get a list of all snapshots.

    For each snapshot named *`$snap`* update the domain configuration of that snapshot using the following steps:

    a. Run the command `virsh snapshot-edit "$vm" "$snap"` to get the configuration opened in an editor.
    b. In the XML file lookup all network interfaces below `/domainsnapshot/domain/devices/interfaces/source` and change the attribute `bridge` from `eth0` to `br0` (or whatever).
    c. Quit the editor to save the file.
4. If the domain is suspend to disk, the suspend image also contains a copy of the domain configuration. Use the command `virsh dominfo "$vm"` to check `Managed save`. If it is `yes`, the image must be either updates or discarded.
    - The virtual machine can be resumed and then be shut down cleanly. This removes the suspend image cleanly, so it no longer must be modified.
    - Run the command `virsh managedsave-remove "$vm"` to discard the saved data. This will remove the runtime state of the virtual machine, but the persistent date stored in the image files is not discarded. Normally the operating system will then perform a file system check as it looks like the virtual machine has crashed while running.

- To update the image, run the command `virsh save-image-edit "/var/lib/libvirt/qemu/save/$vm.save"`. Perform the same changes as described in step 2.

3.1. Migrating virtual machines between hosts

Feedback

(Live-)migration also transfers the domain configuration. This breaks when the network interface names differ between the source and destination host. Because of that it is strongly advised to shut down any virtual machine cleanly and to transfer them as described above.

There is a second option available which allows updating virtual machines while they are migrated from a UCS-3 to a UCS-4 host. libvirt provides a hook mechanism, which can be used to rewrite the virtual machine configuration on-the-fly for incoming migrations. This is fully described in the libvirt hook documentation[1].

The following example can be used to convert the network interface name `eth0` to `br0`. The following script must be copied to `/etc/libvirt/hooks/qemu` and be marked as executable using `chmod +x`:

```
#!/bin/sh
object="${1}"
operation="${2}"
sub_operation="${3}"
extra_argument="${4}"

SRC_BRIDGE="eth0"
DST_BRIDGE="br0"

convert_network () {
 xsltproc \
  --stringparam src_bridge "${SRC_BRIDGE}" \
  --stringparam dst_bridge "${DST_BRIDGE}" \
  "${0}.xsl" -
}

case "${operation}/${sub_operation}" in
prepare/begin) ;;
start/begin) ;;
started/begin) ;;
stopped/end) ;;
release/end) ;;
migrate/begin) convert_network ;;
restore/begin) convert_network ;;
reconnect/begin) ;;
attach/begin) ;;
*) echo "${0} ${*}" >&2 ;;
esac
```

The script uses the `xsltproc` XSLT command line processor, which is provided by the package ***xsltproc***. The corresponding style sheet file must be copies to `/etc/libvirt/hooks/qemu.xsl`, which looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
```

[1] http://libvirt.org/hooks.html

```
 <xsl:output method="xml" encoding="UTF-8" indent="yes" omit-xml-
declaration="yes"/>
 <xsl:param name="src_bridge" select="'eth0'"/>
 <xsl:param name="dst_bridge" select="'br0'"/>

 <xsl:template match="node()" priority="-1" name="copy">
  <xsl:copy>
   <xsl:copy-of select="@*"/>
   <xsl:apply-templates select="node()|text()|comment()|processing-
instruction()"/>
  </xsl:copy>
 </xsl:template>

 <!-- Convert network interface name -->
 <xsl:template match="/domain/devices/interface[@type='bridge']/source">
  <xsl:choose>
   <xsl:when test="@bridge=$src_bridge">
    <source bridge="{$dst_bridge}"/>
   </xsl:when>
   <xsl:otherwise>
    <xsl:call-template name="copy"/>
   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
</xsl:stylesheet>
<!-- vim:set ts=2 sw=2 noet foldmethod=marker: -->
```

Warning

Migration can only be done in the direction UCS-3 to UCS-4, not the reverse. As UCS-4 is using a newer version of libvirt, additional settings are added to the virtual machine configuration, which older versions can't process. Make sure to backup the pre-migration configuration using `virsh --inactive dumpxml "$vm" >"$vm.xml"`.

3.2. UVMM profiles

Feedback

In addition to that the UVMM profiles should be updated as well, as they name the network interface which is used to connect newly created virtual machines. This is described in the [ucs-manual] in *UVMM chapter* in the section *Changing default network*.

Bibliography

[ucs-manual] Univention GmbH. 2015. *Univention Corporate Server - Manual for users and administrators*. https://docs.software-univention.de/manual-4.1.html.