Redpaper

Paul Sutera

Provisioning Linux on IBM System z with Tivoli Service Automation Manager

This IBM® Redpaper™ document describes a methodology that Linux® on IBM System z® users can employ to perform system provisioning tasks while creating the system management infrastructure required for cloud computing. Cloud computing offers dynamically scalable IT resources, on demand self-service, network access, rapid up and down scalability, resource pooling, flexibility, and pay per use.

The paper outlines the use of a subset of IBM Tivoli® Service Automation Manager functions for rapid installation (provisioning) and management of Linux on System z virtual servers. Tivoli Service Automation Manager software supports several of the Linux on System z distributions at one or more of the recent versions of these products.

Many companies face a rapidly changing IT landscape in which the information technology assets and environments require significant staff and budgets to install, configure, and manage. Tivoli Service Automation Manager can be used to rapidly create, configure, provision, and de-provision servers, thus saving time and reducing costs.

Overview

In the typical IT environment resource requirements and utilization rates are highly variable. Test platforms often make up between 30% and 50% of the servers in the environment. Some servers are required only on a temporary basis to coincide with business cycles and seasonal fluctuations in processing; the resource utilization on these servers might be less than 10%. In such cases, a cost-effective way to manage hardware resources is by using a virtual server running under a hypervisor to meet the demands of peak processing cycles as well as providing test platforms.

Many IBM System z customers use Linux on System z in a z/VM® hypervisor environment to achieve the benefits of server virtualization. While some of these customers have used home-grown tools to install, configure, and manage their Linux servers, there is a need for standardized management tools to automate these system management tasks.

Tivoli Service Automation Manager provides such a set of tools. This solution enables you to rapidly create, configure, provision, and de-provision Linux on System z servers running in the IBM z/VM host environment. It also provides the tools to help you make a staged entrance into cloud computing. *Cloud computing* is an IT modelthat facilitates better use of existing IT resources, with fewer provisioning processes and lower capital and operating expenses.

Introduction to Tivoli Service Automation Manager

Tivoli Service Automation Manager assists in the automated provisioning, management, and deprovisioning of hardware servers, networks, operating systems, middleware, and application-level software. Several virtualization environments (hypervisors) are supported in the process of virtual server provisioning on IBM System x®, System p®, and System z.

Tivoli Service Automation Manager also provides management support for services consisting of a specific set of middleware (WebSphere® ND), in combination with AIX® and Linux (System x and System z). Many automation service definition templates are provided, including specialized job plans or workflows. Tivoli Service Automation Manager also exploits the entire spectrum of Tivoli process automation engine tools.

Tivoli Service Automation Manager helps you define and automate services that are life cycle oriented. For example, an IT server network can be made available for a limited period of time to run a test or handle a temporary increase in workload.

A service definition template specifies the overall framework for a service offering. This service definition template can then be customized and made available as an offering that can be selected when a service is needed.

Tivoli Service Automation Manager provides a Self-Service Virtual Server Management environment, which allows a user to request the provisioning of projects comprising virtual servers based on IBM System x, System p, or System z, as well as the WebSphere Cloudburst Appliance product.

Preparing for Linux provisioning on System z with Tivoli Service Automation Manager

This paper demonstrates Linux on System z provisioning using the Tivoli Service Automation Manager 7.2 and Linux under IBM z/VM. The primary focus is the Tivoli Service Automation Manager setup required to enable provisioning and managing Linux on System z in a z/VM hosting environment. The specifics for obtaining and installing Tivoli Service Automation Manager 7.2 are beyond the scope of this paper. The z/VM-specific setup, including possible RACF® configuration steps, is also beyond the scope of this paper. These subjects are covered in detail in the online documentation for Tivoli Service Automation Manager 7.2:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.tsam.doc_7.2/

In addition, you must install the MAPSRV Linux server and also install Linux images that serve as the “golden masters” to be used during provisioning. This paper does not cover the basic steps needed to install Linux or add software to existing Linux systems; it is assumed that the reader has the requisite skills. Details about these topics, as well as the steps needed to compile and run Tivoli Provisioning Manager workflows, are presented in the Tivoli Provisioning Manager 7.1 online documentation available at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v11r1/index.jsp

Using Tivoli Service Automation Manager to provision Linux on System z

Tivoli Service Automation Manager 7.2 and the fully integrated Tivoli Provisioning Manager 7.1 provisioning product can be used to create and rapidly provision Linux on System z guests in a z/VM hosting environment. The procedures to install and configure Linux, which as a manual process can take an hour or more, can be automated and reduced to a matter of minutes, depending on the devices configured for the guest and the speed of disk-to-disk copy operations on the storage system hardware. Tivoli Service Automation Manager 7.2 can be hosted on System z or non-System z hardware to manage System z hardware, software, and operating systems. The system provisioning functions discussed in this paper are only a small part of the overall IT landscape management capabilities built into the Tivoli Service Automation Manager product.

The following sections describe the steps needed to set up Tivoli Service Automation Manager before it can be used to provision new Linux on System z guests under z/VM 5.4 or later. The setup presented here uses only the IBM Tivoli Provisioning Manager workflows with the Tivoli Service Automation Manager graphical user interface. Tivoli Provisioning Manager is the engine that runs workflows on behalf of the Tivoli Service Automation Manager product. The Tivoli Service Automation Manager user interface drives most of its functions through Tivoli Provisioning Manager workflows. Some of the provisioning steps shown here might not yet be fully supported under Tivoli Service Automation Manager. For example, at this writing minidisks are the only supported disk type for provisioning systems, but both minidisks and dedicated disks are fully supported at the Tivoli Provisioning Manager workflow layer.

Topology of Tivoli Service Automation Manager provisioning using Tivoli Provisioning Manager workflows

The topology of the Tivoli Service Automation Manager is shown in Figure 1. The Tivoli Service Automation Manager server is a separate server here, but it could also be a Linux on System z guest running under z/VM.

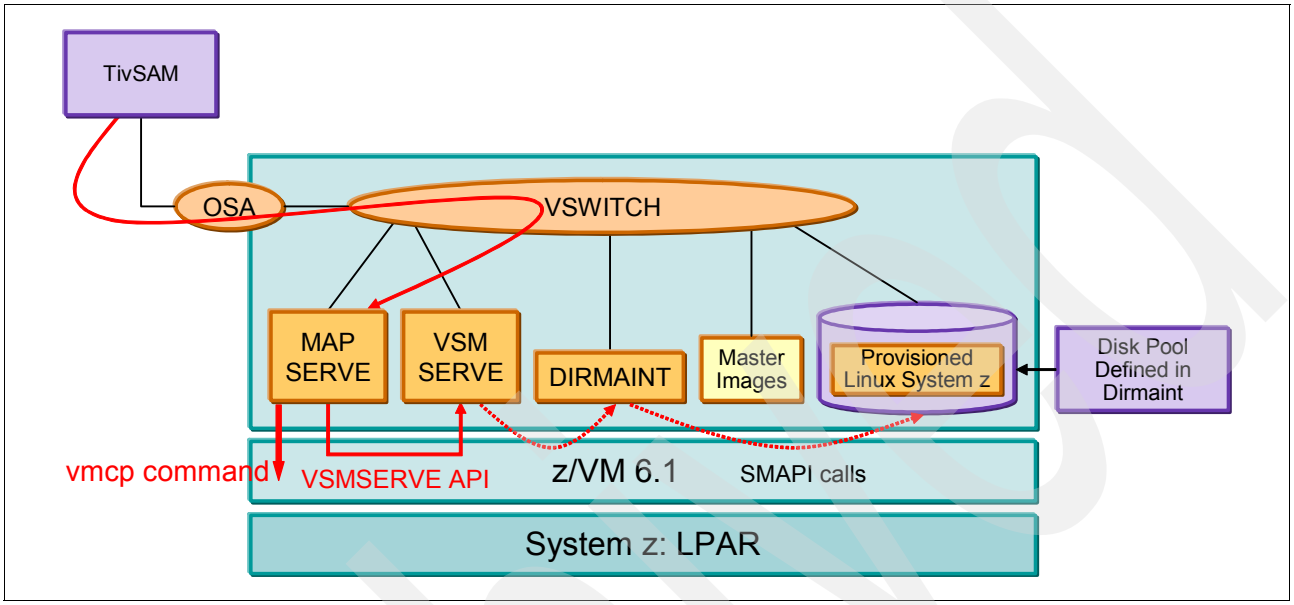

Figure 1 Tivoli Service Automation Manager topology

Tivoli Service Automation Manager communicates using Secure Sockets Layer (SSL) to the z/VM network. The network in our implementation is a VSWITCH. Tivoli Provisioning Manager workflows are run on the Tivoli Service Automation Manager server and the z/VM and Linux-specific provisioning steps are passed to the MAPSRV Linux guest. MAPSRV acts as a central point of control, executing z/VM commands directly or passing VM SMAPI commands to the VSMSERVE z/VM service machine through remote procedure calls (RPC). Some VM SMAPI commands generate DIRMAINT commands that are then passed to the DIRMAINT service machine. MAPSRV uses the “VMCP” interface to run z/VM CP and CMS commands directly, usually under the MAPAUTH user ID. While a DIRMAINT disk pool is shown, dedicated disks can also be used when a Linux golden master image has been created using a dedicated disk. The DATAMOVE service machine also plays a role in some of the disk-based operations. The configuration of these z/VM components is beyond the scope of this document. For z/VM, DIRMAINT, and other system setup specific to Tivoli Service Automation Manager provisioning on System z, consult the Tivoli Service Automation Manager 7.2 documentation library:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.tsam.doc_7.2/

Consult the IBM z/VM online library for z/VM-specific setup information:

http://www.vm.ibm.com/library

Configure the XML for Tivoli Service Automation Manager and Tivoli Provisioning Manager

Tivoli Service Automation Manager distributes the following sample XML files with the product:

- Configure network for Linux guests: 01_Networks_Template.xml
- Configure z/VM-specific information for guests: 02_VirtualServerTemplates_Template.xml
- Configure Tivoli Service Automation Manager to point to the MAP server as its “boot server”: 03_BootServers_Template.xml
- Configure Linux-specific SLES10 operating system information: 04_zLinux_SLES10_image_Template.xml
- Configure available hardware resources, disks, IP address pools, security: 05_Hostplatforms_Template.xml

These files must be tailored to the local Linux and z/VM environment. The resulting XML input files are then incorporated into the Tivoli Service Automation Manager database, the DCM (Data Center Model). This is performed using the xmlImport.sh utility; consult the Tivoli Service Automation Manager documentation for the correct syntax. Next, Tivoli Service Automation Manager/Tivoli Provisioning Manager workflows can be called manually to provision a single Linux guest. A sample workflow is provided in this document to verify the Tivoli Service Automation Manager and z/VM setup. After a simple Linux system is successfully provisioned, the Tivoli Service Automation Manager graphical interface can be used to enlarge the scope and capabilities of the Linux on System z provisioning environment. It is possible to provision simple Linux endpoints without making use of every property or tag shown in the sample XML distributed with the Tivoli Service Automation Manager product. Every XML file should contain the header shown in Example 1, along with a trailing </datacenter> tag.

Example 1 Sample XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE datacenter PUBLIC "-//Think Dynamics//DTD XML Import//EN"
    "xmlimport.dtd">
-->
<datacenter>
XML Contents go here -
</datacenter>
```

Network configuration

The network configuration template file 01_Networks_Template.xml distributed by Tivoli Service Automation Manager was used as the basis for configuring the sample DCM XML for a network configuration as shown in Example 2.

Example 2 XML for a network configuration

```
<subnetwork address-space="DEFAULT" name="zBMC Cloudnet" ipaddress="129.40.178.0"
netmask="255.255.255.0">
    <property component="KANAHA" name="gateway" value="129.40.178.254" />
  </subnetwork>
 <switch name="CLOUDSWC" locale="en_US" failed="false" number-of-ports="16">
<property component="KANAHA" name="device-model" value="zVM Virtual Switch" />
    <property component="KANAHA" name="host-platform" value="MAPSRV" />
  </switch>
```

The first network component defined is the subnetwork. It is named “zBMC Cloudnet,” with a subnetwork value of 129.40.178.0 and a netmask of 255.255.255.0. A property named “gateway” is also defined as the subnetwork’s gateway. A property is represented as a variable in Tivoli Provisioning Manager. A switch entry is then defined which represents the z/VM VSWITCH on the system. The name of the switch on the z/VM System is CLOUDSWC. Executing the z/VM command Q VSWITCH gives the following output:

```
VSWITCH SYSTEM CLOUDSWC Type: VSWITCH Connected: 3    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    ETHERNET                  Accounting: OFF
  VLAN Unaware
  MAC address: 02-00-00-00-00-01
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Isolation Status: OFF
  RDEV: 3840.P00 VDEV: 3840 Controller: DTCVSW1
```

In this example CLOUDSWC is the name defined for the VSWITCH. The device-model is z/VM Virtual Switch. The host-platform property (a variable in the DCM) points to the Linux host name of the Linux guest running on the MAPSRV z/VM user ID.

When Linux images are created, new entries are added to the switch. An entry such as a switch-module can be created. The XML can also be configured to support a hipersocket network.

VLAN entries are also created, but because the test system set up for this paper did not use VLANs, the VLAN entry that is added when a new z/VM user ID or “host container” is created is blank:

```
<property component="KANAHA" name="S10SP201-VLANS" value="" />
```

Host platform definition

The host platform for Linux running under z/VM refers to the special Linux server that runs the MAPSRV component. The MAPSRV runs most of the commands needed for Linux provisioning. The primary installable component needed to make a Linux server a MAP is the IBM-System-z.MAPSRV-1.0-1.s390x rpm.

The next XML example contains some entries that can be used by Tivoli Service Automation Manager but might not be used in every provisioning scenario. The XML for this section was adapted from the Tivoli Service Automation Manager sample XML file named 05_Hostplatforms_template.xml.

Example 3 uses minidisks rather than dedicated disks. This assumes that a minidisk pool of DASD disks has already been created with DIRMAINT. For environments where you want to use dedicated disks instead of minidisks, see “Dedicated disks” on page 12.

Example 3 Sample XML for Tivoli Service Automation Manager

```
<server name="MAPSRV" locale="en_US" is-device-model="zVM_Host_Platform"
ignored-by-resource-broker="false" failed="false" tcp-port="-1">
    <network-interface failed="false" managed="false" management="true"
dynamic-ipaddress="false" ipaddress="129.40.178.250" netmask="255.255.255.0"
address-space="DEFAULT" allocation="none" />
    <nic name="NIC-ConnectedTo-CloudSWC" locale="en_US" failed="false"
managed="true" management="false" netboot-enabled="false" group-name="cloudswc"
is-vlan-aware="false">
```

```
      <property component="KANAHA" name="zVM_NetworkPortType" value="VSwitch"
description="Type of NIC connection - VSwitch" />
      <property component="KANAHA" name="zVM_NetworkVSLanName" value="CLOUDSWC"
description="VSwitch name" />
    </nic>
    <property component="DEPLOYMENT_ENGINE" name="POOL.GUESTNAME.LINUX.1"
value="S1OSP202" />
    <property component="DEPLOYMENT_ENGINE" name="POOL.GUESTNAME.LINUX.2"
value="S1OSP20A" />
    <property component="DEPLOYMENT_ENGINE" name="POOL.GUESTNAME.LINUX.3"
value="S1OSP203" />
    <property component="DEPLOYMENT_ENGINE" name="POOL.IPADDRESS.1"
value="129.40.178.233" />
    <property component="DEPLOYMENT_ENGINE" name="POOL.IPADDRESS.2"
value="129.40.178.234" />
    <property component="DEPLOYMENT_ENGINE" name="POOL.IPADDRESS.3"
value="129.40.178.235" />
          <property component="DEPLOYMENT_ENGINE" name="host-platform-type"
value="zVM" />
    <property component="KANAHA" name="physicalLocation" />
    <property component="DEPLOYMENT_ENGINE" name="vsmserve-address"
value="172.16.0.1" description="Guest Lan IP address for MAP to contact VSMSERVE
on port 845 with MAPAUTH's userid and password" />
    <property component="DEPLOYMENT_ENGINE" name="vsmserve-port" value="845" />
    <sap -- SAP credentials go here – see below  </sap>
    <host-platform />
    <resource – Hardware Resources go here – see below </resource>
  </server>
```

Host platform network information

This section helps you define the network-specific parts of the host platform definition.

The server name refers to the z/VM user ID that hosts the System z Linux-based MAP component. The name “MAPSRV” is used, following the usage found in Tivoli Service Automation Manager System z documentation. This is known to Tivoli Service Automation Manager as the “zVM_Host_Platform.” The network-interface line points to the network information needed for Tivoli Provisioning Manager to contact and direct work to the MAP component. The IP address (129.40.178.250) and netmask for the MAPSRV Linux guest are also provided.

The NIC is then defined and named “NIC-ConnectedTo-CloudSWC” and this is given a group name of “cloudswc,” though any name could have been used for either attribute. Two properties are added that become variables in Tivoli Service Automation Manager: zVM_NetworkPortType and zVM_NetworkVSLanName. The zVM_NetworkPortType is set to “VSwitch” and the zVM_NetworkVSLanName is CLOUDSWC, the z/VM-defined name for the VSWITCH. Note the spelling of “VSwitch” is very specific and the first two letters must be capitalized. The values “GuestLAN” and “Hipersocket” are also supported for these kinds of networks. There are other variables that can be defined if the zVM_NetworkPortType is a hipersocket or a guest LAN network.

After this section there are many user IDs and IP addresses defined. Although these are not used in the simple Tivoli Service Automation Manager provisioning scenario, they can be used by other Tivoli Service Automation Manager operations to provision. They provide a way to create a pool of z/VM User IDs and IP addresses for multiple provisioning operations where

the new user ID and IP address are not directly specified but are instead selected from this pre-established pool.

```
<property component="DEPLOYMENT_ENGINE" name="host-platform-type" value="zVM"/>
<property component="DEPLOYMENT_ENGINE" name="vsmserve-address"
value="172.16.0.1"/>
<property component="DEPLOYMENT_ENGINE" name="vsmserve-port" value="845"/>
```

The host-platform-type of “zVM” is defined along with the IP address for the Guest LAN that is connected to the VSMSERVE z/VM service machine. The port used by VSMSERVE is defined under vsmserve-port and has a z/VM-defined value of 845. VSMSERVE services all of the System Management API calls (SMAPI). Some of these function calls generate DIRMAINT commands that are passed to the DIRMAINT service machine.

Security and access credentials

Another part of the MAP server definition in 05_Hostplatforms_template.xml is the XML that creates Service Access Points (SAPs). SAPs are a Tivoli Service Automation Manager construct that supply the credentials for a user ID to perform certain tasks or establish sessions with other users. See Example 4.

Example 4 XML to create a service access point

```
<sap name="PING" is-device-model="ICMP Ping Service Access Points"
locale="en_US" protocol-type="ipv4" app-protocol="ICMP" context="NOCONTEXT"
port="0" auth-compulsory="true" role="host">
      <default-sap operation-type="ping" />
      <credentials search-key="master" is-default="true">
        <password-credentials username="root"
password="8cKiIv3NuQl8KJPvEZppQQ==" is-encrypted="true" />
      </credentials>
    </sap>
    <sap name="zVM SSH" is-device-model="SSH Service Access Point"
locale="en_US" protocol-type="ipv4" app-protocol="SSH" context="NOCONTEXT"
port="22" auth-compulsory="true" role="host">
      <default-sap operation-type="execute-command" />
      <credentials search-key="master" is-default="true">
        <password-credentials username="root"
password="8cKiIv3NuQl8KJPvEZppQQ==" is-encrypted="true" />
      </credentials>
    </sap>
    <sap name="SMAPI" locale="en_US" protocol-type="ipv4"
app-protocol="LOCAL-EXEC" context="NOCONTEXT" port="845" auth-compulsory="true"
role="host">
      <credentials search-key="master" is-default="true">
        <password-credentials username="MAPAUTH"
password="mGt65zyQr5F8KJPvEZppQQ==" is-encrypted="true" />
      </credentials>
    </sap>
```

The ping SAP is used internally by Tivoli Service Automation Manager to establish initial addressability to another user ID, such as a newly provisioned endpoint. The z/VM SSH Service Access Point is used for several workflows, such as copying a file to an endpoint or executing a shell script on an endpoint. Tivoli Service Automation Manager is able to kick off shell scripts or copy files from one Linux endpoint to the MAP endpoint (also a Linux image) via this access point. As such, a shared credential key called “master” is used and the Linux root user ID “root” and root password for the MAP is provided. The SMAPI SAP is also

defined. Here the credential key “master” is also used. The z/VM user ID supplied here is MAPAUTH, and MAPAUTH’s z/VM password is provided. Tivoli Service Automation Manager uses the MAPAUTH user ID to run SMAPI commands on the MAP server. Passwords are encrypted inside the DCM. The XML shown here was partly obtained through the Tivoli shell script, dcmExport.sh. This shell script dumps the contents of the Data Center Model (DCM) into an XML file that can be used as a rough backup of the DCM contents.

MAP Server hardware resources

The hardware resources section is a part of the 05_Hostplatforms_template.xml file and specifies the totality of available provisioning resources managed by the system. A sample configuration of this section is shown in Example 5.

Example 5 MAP Server hardware resources

```
 <resource name="Platform" resource-type="platform" managed="true"
partitionable="false">
      <property component="KANAHA" name="platform.architecture" value="390" />
    </resource>
    <resource name="CPU" resource-type="cpu" managed="true"
partitionable="true">
      <property component="KANAHA" name="cpu.family" value="s390" />
      <property component="KANAHA" name="cpu.size" value="2" />
      <property component="KANAHA" name="cpu.type" value="64-bit" />
    </resource>
    <resource name="Mem" resource-type="memory" managed="true"
partitionable="true">
      <property component="KANAHA" name="memory.size" value="1024" />
    </resource>
    <resource name="POOL0" resource-type="disk" group-name="mypool"
managed="true" partitionable="true">
      <property component="KANAHA" name="disk.size" value="20" />
    </resource>
```

The hardware resources are associated with the MAPSRV user ID, but MAPSRV does not actually possess these resources in a z/VM sense. Instead, it is the Management Access Point (MAP) for the resources.

The DCM contains information about the platform but it also details the known supply of hardware resources that the MAP server is able to allocate to newly provisioned servers. As servers are provisioned, the amount of available resources, such as disk space in a minidisk pool, are decremented to reflect a new use of that resource. Other resources are virtualized on z/VM so there is no concept of a finite (exhaustible) supply of these resources (CPU, memory). These resources are not diminished by the provisioning of a new server. The memory.size parameter of 1024 means that each newly provisioned server will have a virtual memory size of 1024 Megabytes, or 1 Gigabyte.

The cpu.size is 2, which means that two virtual CPUs are allocated to each new server. The disk resource is a DIRMAINT pool of minidisks, but could also be defined as individual dedicated disks. MAPSRV can manage both pools of minidisks and individual dedicated disks, and both might be present in the MAPSRV’s hardware resources. For minidisks, it is important that the managed and partitionable attribute be set to true so that Tivoli Service Automation Manager can partition the minidisk pool into individual allocations for newly created guests. The name of the resource “POOL0” matches the name chosen by the z/VM system programmer for the minidisk pool. This is defined in the EXTENT CONTROL z/VM control file for DIRMAINT’s minidisk allocations. The disk.size of 20 means there are a total of

20 GB of disk space available in the minidisk pool. This doesn't have to match the true available space exactly. A resource group name of "mypool" is chosen and is used by Tivoli Service Automation Manager to group disk resources. If dedicated disks are used the unit address of the dedicated disk is provided instead of the minidisk pool name. The disk.size should be the size of the DASD device provided. The managed attribute is set to true and the partitionable attribute for minidisks is true; for dedicated disks it should be false. The resource group name for dedicated disks can be the same as the group name used for the minidisk pool.

Virtual Server templates

The Virtual Server template contains the information needed to create the z/VM user ID that is the host container for the provisioned Linux guest. It contains information relating to resource allocations that are generally implemented at the z/VM level. These can be found in the <USER> DIRECT file that DIRMAINT maintains for each z/VM User ID. Tivoli Service Automation Manager distributes sample virtual server templates in 02_VirtualServerTemplates_Template.xml. Example 6 shows XML for a Virtual Server template.

Example 6 XML for a Virtual Server template

```
<virtual-server-template name="Default VST - 1 NIC - 2 CPUs - 1gb Storage">
    <resource-requirement resource-type="platform" how-many="0" size="0.0"
is-shared="true" />
    <resource-requirement resource-type="cpu" how-many="2" size="0.0"
is-shared="true">
      <property component="KANAHA" name="cpu.family" value="s390" />
    </resource-requirement>
    <resource-requirement resource-type="memory" how-many="1024" size="0.0"
is-shared="true" />
    <resource-requirement resource-type="nic" how-many="1" size="0.0"
group-name="cloudswc" is-shared="true">
      <property component="KANAHA" name="zVM_NetworkDeviceRange" value="3" />
     <property component="KANAHA" name="zVM_NetworkPortDeviceNumber" value="4420" />
      <property component="KANAHA" name="zVM_NetworkPortType" value="VSwitch" />
    </resource-requirement>
    <resource-requirement resource-type="disk" how-many="0" size="6.9"
group-name="POOL0" is-shared="false">
      <property component="KANAHA" name="zVM_DiskDeviceNumber" value="0201" />
      <property component="KANAHA" name="zVM_DiskType" value="Minidisk" />
    </resource-requirement>
    <property component="KANAHA" name="host-platform-type" value="zVM" />
    <property component="KANAHA" name="zVM_Prototype" value="LINUX"
description="Optional - initial z/VM prototype file used to create a virtual
server" />
    <property component="KANAHA" name="zVM_Userid" value="S10SP201"
description="First Userid" />
  </virtual-server-template>
```

CPU, memory, networking

The first few XML stanzas concern the resource allocations requested for the new server. These resources must match the available resources defined in the MAP host platform definition.

The CPU and memory requirements outline the requests for CPU and memory for the provisioned server. The property zVM_NetworkPortType is “VSwitch” and the zVM_NetworkPortDeviceNumber is the virtual address defined in z/VM for the OSA card. This corresponds to the address to be defined in the NICDEF statement in the <USER> DIRECT file. In our example the following NICDEF statement is in each user’s directory:

```
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
```

Device 4420 represents the OSA virtual triplet 4420,4421,4421 that will be used by the new guest to connect to the network. The group-name parameter in the NIC definition is “cloudswc.” This name could have been any name but it must agree with the group-name in the NIC definition specified on the host platform XML definition.

Disks and cloning

At this writing, only a single-disk provisioned system is supported. That means that the Master system used as the basis for cloning must also be a single-disk system. Additional disks can be added to provisioned guests later, using manual processes. IBM development is aware of the situation and is actively considering support for multiple disk master systems, which would then be included in a fixpack.

To understand z/VM Linux provisioning it helps to understand the underlying process: A source disk defined to the master is copied to a target disk that will be owned by the newly provisioned server. The z/VM FLASHCOPY is used to copy the disk if it is available and configured. If the FLASHCOPY fails, a DDRCOPY EXEC on DIRMAINT’s CF1 disk is called that runs the DDR copy command. If the source and target disk are in different storage subsystems, FLASHCOPY is not supported and DDR is used instead. FLASHCOPY allows for a much faster disk-to-disk copy than a DDR copy, and that can greatly shorten the provisioning time.

In the Virtual Server template, disk resources are requested under the Hard Disk resources tab. Minidisks are requested by specifying “Minidisk” for the zVM_DiskType. The available space in the DIRMAINT minidisk disk pool must be at least as large as the request. Also, the requested space needs to be able to fit on a single disk. A requested size of 6.9 Gigabytes is an exact fit for a 3390-9, so the minidisk pool needs to have 3390-9 or larger DASD disks available to match this request. Sometimes when specifying a size for a target disk, due to rounding, the value specified for disk.size might not work. It might be necessary to round up the Gigabyte count to the next whole number. The disk in the minidisk pool with the contiguous free space amount that most closely matches the space request will be used to “carve out” a new minidisk for the target system.

The zVM_DiskDeviceNumber is set to the minidisk virtual address that needs to be defined for the newly allocated disk. This will generate a statement in the user’s z/VM user directory, for example:

```
MDISK 0201 3390 1 10016 V64M02 MR PASSWORD PASSWORD PASSWORD
```

In our experiments it was decided to use the same minidisk address (0201) for the new filesystem as was used on the master to hold the root filesystem. This 0201 minidisk is, of course, distinct from the 0201 minidisk owned by the golden master user ID.

Dedicated disks

If dedicated disks are requested (specify “Dedicated” for zVM_DiskType), a search for free disks in the MAPSRV hardware resources is initiated during provisioning. The disks are listed in the Tivoli Service Automation Manager Disks tab under the Hardware tab of the MAPSRV host platform.

There can be multiple dedicated disks in the MAPSRV’s disk resources. Some dedicated disks defined in MAPSRV’s hardware resources must match the disk type of the master’s source disk. For example, if a master is defined with a 3390-Mod 18 disk, then there must be at least one available disk in the MAPSRV hardware resources that is also a 3390-Mod 18 disk, or provisioning will fail. It is possible that a larger disk might work as a target for a smaller disk, but this has not been verified.

For dedicated disks, the unit of the master’s source disk unit is set in zVM_DiskDeviceNumber. This will be virtualized in the user’s directory with the following statement:

```
DEDICATE <Master Source Disk Unit> <Target Disk Unit>
```

The target unit is chosen from the list of available hard disks specified in the Hardware tab of MAPSRV, the host platform. To exactly control the disk chosen for a target system, add one disk at a time to the list of available dedicated disks. If there is only one disk available, it can be guaranteed that this disk is chosen for the target system. For the Software Image, the disk specified for the zVM_CloneDisks and zVM_SystemDisk variables will both point to the z/VM unit address of the root filesystem on the golden master. For further information see “Software images ” on page 14.

After a new image is brought up for the first time, the personalization scripts are run that “personalize” the new guest with its own network and disk information. Personalization scripts are part of the IBM-System-z.MASTER-1.0-1 installed rpm that is copied from the master system during the cloning operation. Currently the scripts handle disk address personalization by requiring that the zVM_DiskDeviceNumber be set to the unit address of the master disk. For dedicated disks the target disk of the new server is then virtualized to the unit address of the master. In this way no changes need to be made to the /etc/sysconfig, /etc/zipl.conf (nor to /etc/udev/rules.d for SLES11, nor to /etc/modprobe.conf for RHEL) filesystem during personalization because the master’s unit is already configured in the filesystem when the filesystem is copied from the master image.

If the root filesystem designation in /etc/zipl.conf contains a device number, however, it might be advisable to change the parameter so that it refers to the device name. For example, if the root partition is on device /dev/dasda1, the following statement would appear in the master's ipl stanza in /etc/zipl.conf:

```
parameters = "root=/dev/dasda1 TERM=dumb".
```

This would replace any device number that might have appeared (for example) as follows:

```
parameters = "root=/dev/disk/by-path/ccw-0.0.7171-part1 TERM=dumb"
```

z/VM user ID, password, PROTODIR

The z/VM user ID and password properties are provided in the Virtual Server template. Tivoli Service Automation Manager provisioning depends on the presence of the zVM_User ID and zVM_Password properties. The Tivoli Service Automation Manager GUI can provide inputs to override this user ID and password and typically will create a temporary copy of a template with a changed z/VM user ID and password property (variable). In our sample workflow, however, the z/VM User ID must be changed directly in the Virtual Server Template. Existing password rules for the security mechanism in place definitely do apply here. The

zVM_Prototype property contains the name of the PROTODIR file that is used to create the z/VM User ID. The z/VM user directory created for the new user will use this PROTODIR file to create the new user. Typically the PROTODIR does an INCLUDE for a default z/VM user ID. In this installation, the PROTODIR file is named LINUX PROTODIR so the value of zVM_Prototype is “LINUX.”

Software stack

The software stack is a Tivoli Service Automation Manager structure that supplies additional information for installable operating system images and software resource templates. Information identifying the operating system and the hardware platform is contained in the software stack. An installable image can then point to a software stack as its “software module.” This is discussed in the section on installable images. A sample software stack for SLES 10 SP2 64-bit, shown in Example 7, was adapted from the Tivoli Service Automation Manager distributed sample, 04_zLinux_SLES10_image_Template.xml.

Example 7 Sample software stack

```
<software-stack name="SLES10 GM OS" locale="en_US" version="N/A"
stack-type="Declared">
    <software-capability type="OS" name="os.family" value="Linux" />
    <software-capability type="OS" name="os.distribution" value="SLES10 s390x" />
    <software-capability type="OS" name="os.name" value="SLES10 for IBM S/390 and
IBM zSeries" />
    <software-capability type="OS" name="os.version" value="10" />
    <software-capability type="OS" name="os.servicepack" value="SP2" />
    <software-requirement name="cpu.family" type="HARDWARE"
enforcement="MANDATORY" hosting="false" accept-non-existing="false">
      <software-requirement-value value="s390" />
    </software-requirement>
    <software-requirement name="cpu.type" type="HARDWARE" enforcement="MANDATORY"
hosting="false" accept-non-existing="false">
      <software-requirement-value value="64-bit" />
    </software-requirement>
 <software-resource-template name="SLES10-S390X-SP2"
software-resource-type="INSTALLATION" multiplicity-type="N"
software-configuration-type="Regular" is-selected="true"
software-resource-device-model="zVM_Software_Installation" is-default="false"
is-deployable="true">
          <template-param name="zVM_Password" value="VMPASSME"
parameter-type="String" multiplicity-type="One" is-hidden="false"
is-changeable="true" is-encrypted="false">
        <template-param-value value="VMPASSME" is-default="true" />
       </template-param>
       <template-param name="zLinux_RootPassword" value="pa55word"
parameter-type="String" multiplicity-type="One" is-hidden="false"
is-changeable="true" is-encrypted="false">
         <template-param-value value="pa55word" is-default="true" />
       </template-param>
           </software-resource-template>
    </software-resource-template>
    <software-stack-iterator priority="1" />
  </software-stack>
```

The software capabilities are outlined as os.family, os.distribution, os.name, os.version, os.servicepack. The software requirements of cpu.family and cpu.type, (s390, 64-bit) are used by Tivoli Service Automation Manager to enforce that only 64-bit enabled System z hardware is eligible for the installation of any images that are a part of this software stack. The software resource template SLES10-S390X-SP2 is another part of the software stack. The template contains parameter values that reference additional information needed for provisioning. This template has two parameters, the z/VM password and the Linux root password. When a new Linux guest is created during provisioning, a z/VM user ID (and virtual machine) is created first, with the password specified for zVM_Password. After this, the Linux portion of the guest is created, and the root user is given a password that is assigned the value specified in zLinux_RootPassword. The user ID used to create the z/VM user ID is currently a variable in the Virtual Server template. Under the full-function Tivoli Service Automation Manager provisioning (not exercised in this paper), a temporary template is created and other user IDs can then be substituted.

Software images

The software image represents to Tivoli Service Automation Manager the installable image that can be provisioned. On z/VM the image lives on a disk or disks that are owned by a master user ID. Sample XML for image definitions is distributed in 04_zLinux_SLES10_Image_Template.xml. Our sample XML for a SLES 10 SP2 software image is shown in Example 8.

Example 8 XML for SLES software image

```
<image name="SLES10 SP2 Minidisk" locale="en_US" version="1.0"
description="Prepared for TiVSAM" boot-server="MAPSRV-bootserver"
image-type="Golden_Master" status="tested" software-module="SLES10 GM OS"
priority="1" is-device-model="image">
    <property component="KANAHA" name="zVM_CloneDisks" value="201" />
    <property component="KANAHA" name="zVM_DiskOwnerId" value="SL10MSTR" />
    <property component="KANAHA" name="zVM_Prototype" value="LINUX" />
    <property component="KANAHA" name="zVM_SystemDisk" value="201"
description="The bootable disk containing the operating system to be then mounted
on /boot" />
  </image>
```

The image XML points to the software stack through the use of the software-module field. The software-module field matches the name of the software stack so it can be defined as an installable image under the software stack. The boot-server is also identified in the boot-server field.

This allows the boot-server to maintain a list of installable images as well that are under the control of the MAP server. The other properties listed are important for pointing to the owning z/VM user ID and disk addresses that contain the Linux operating system installed on the master. This is sometimes referred to as a “golden master.” The zVM_CloneDisks property is designed to contain multiple disks that must be cloned from the Master. The zVM_DiskOwnerId is the z/VM user ID that owns the disks to be cloned. One of the disks to be cloned is the disk that contains the root partition, /. It therefore also contains the /boot partition. The current level of support on Tivoli Service Automation Manager 7.2.1 only supports a single clone disk, so that single clone disk must also appear as the z/VM_SystemDisk. There is also a zVM_Prototype property that in our implementation will be the same z/VM PROTODIR file that was also referenced in the Virtual Server Template. The value LINUX is used, which refers to a file named LINUX PROTODIR as explained in the Virtual Server template section. For dedicated disks, the disk address for both of these is the

single dedicated disk containing the root filesystem of the golden master, the operating system image.

MAP Bootserver

The Bootserver is a Tivoli Service Automation Manager construct that is used to point to installable images as well as the MAP or management access point construct. The following Bootserver XML sample was developed from the Tivoli Service Automation Manager sample, 03_BootServers_Template.xml:

```
<boot-server name="MAPSRV-bootserver" locale="en_US"
is-device-model="zVM_BootServer" type="zVM" failed="false">
    <property component="KANAHA" name="Hostplatform" value="MAPSRV" />
  </boot-server>
```

The device-model defines the Tivoli Service Automation Manager workflows that are attached to a given Tivoli Service Automation Manager construct, and can be used by Tivoli Service Automation Manager to invoke certain workflows that are pertinent to a structure. For example, the device-model zVM_BootServer causes workflows UHub_CreateVirtualServer and Uhub_DestroyVirtualServer to be associated with the Bootserver. That is appropriate for a z/VM Bootserver because these workflows are unique to provisioning System z Linux on z/VM.

Verification of provisioning setup environment

With the sample XML now tailored with the specific IT information for z/VM Linux systems to be managed, the setup must be verified to see whether a newly provisioned guest can be created. The following steps outline a procedure for verifying the setup and provisioning of the first Linux system.

Sample workflow for a simple all-in-one Linux provisioning

The XML files can be imported into the DCM using Tivoli Service Automation Manager's xmlimport.sh shell script. The files can be combined into one file, but it might be preferable to leave them in the five XML files outlined previously and import them, one file at a time, into Tivoli Service Automation Manager.

Once the XML is imported, a sample workflow can be created that calls the three Tivoli Service Automation Manager workflows, with arguments that should result in a successfully provisioned Linux guest. Usually there is a significant amount of debugging at the z/VM setup level, so it might be wise to call the three workflows inside the "all-in-one" workflow individually before attempting to run all three at once. The Tivoli Service Automation Manager provisioning process might well change and might call different workflows besides the three main workflows: UHub_CreateVirtualServer, UHub_AddMgmtInterface, and UHub_OSInstall. The sample workflow is listed in Example 9. Edit and compile a new workflow so that the sample will be available for execution on Tivoli Service Automation Manager.

Example 9 Workflow to call three Tivoli Service Automation Manager workflows

```
@doc Create a virtual server on a z/VM Hostplatform, provision using the Golden Master @doc
image and then activate the z/VM Userid - calling 3 successive workflows
javaimport com.ibm.ism.pmzhb.os.common.MessageLoader
```

```
workflow UHub_AllInOneProvision(in HostPlatformID, in ServerTemplateID, in Name, in
BootServerID,in ImageID, in SoftwareResourceTemplateID, in interfaceName, in ipAddress, in
hostname, in domainSuffix, in gateway, in primaryDNS) LocaleInsensitive

@doc Create a virtual server on a z/VM Hostplatform, provision using the Golden Master @doc
image and then activate the z/VM Userid - calling 3 successive workflows

javaimport com.ibm.ism.pmzhb.os.common.MessageLoader

workflow UHub_AllInOneProvision(in HostPlatformID, in ServerTemplateID, in Name, in
BootServerID,in ImageID, in SoftwareResourceTemplateID, in interfaceName, in ipAddress, in
hostname, in domainSuffix, in gateway, in primaryDNS) LocaleInsensitive

# ServerID to be filled in by UHub_CreateVirtualServer

var ServerID = ""

#Change to match your installation's subnet Id.
var subnetId = 7819

var nicId = 0

var secondaryDNS=""

var adminVlanId=""

var groupName = "cloudswc"

log info "Starting UHub_CreateVirtualServer workflow"

UHub_CreateVirtualServer(HostPlatformID, ServerTemplateID, Name, ServerID)

nicId=DCMQuery(/server[@id=$ServerID]/nic[@groupName=$groupName]/@id)

log info "Starting UHub_AddMgmtInterface"

UHub_AddMgmtInterface( ServerID, nicId, interfaceName, ipAddress, hostname, domainSuffix,
adminVlanId, subnetId, gateway, primaryDNS, secondaryDNS )

log info "Starting UHub_OSInstall"

UHub_OSInstall(BootServerID, ImageID, ServerID,  SoftwareResourceTemplateID)

log info "UHub_AllInOneProvision completed"
```

Some parameters are hardcoded as <dcmId>. You should become familiar with the Tivoli Service Automation Manager provisioning graphical interface so that the DCM IDs can be located and hardcoded into this workflow or passed as additional parameters to a slightly rewritten workflow. A sample invocation of the workflow is shown in Figure 2 on page 17.

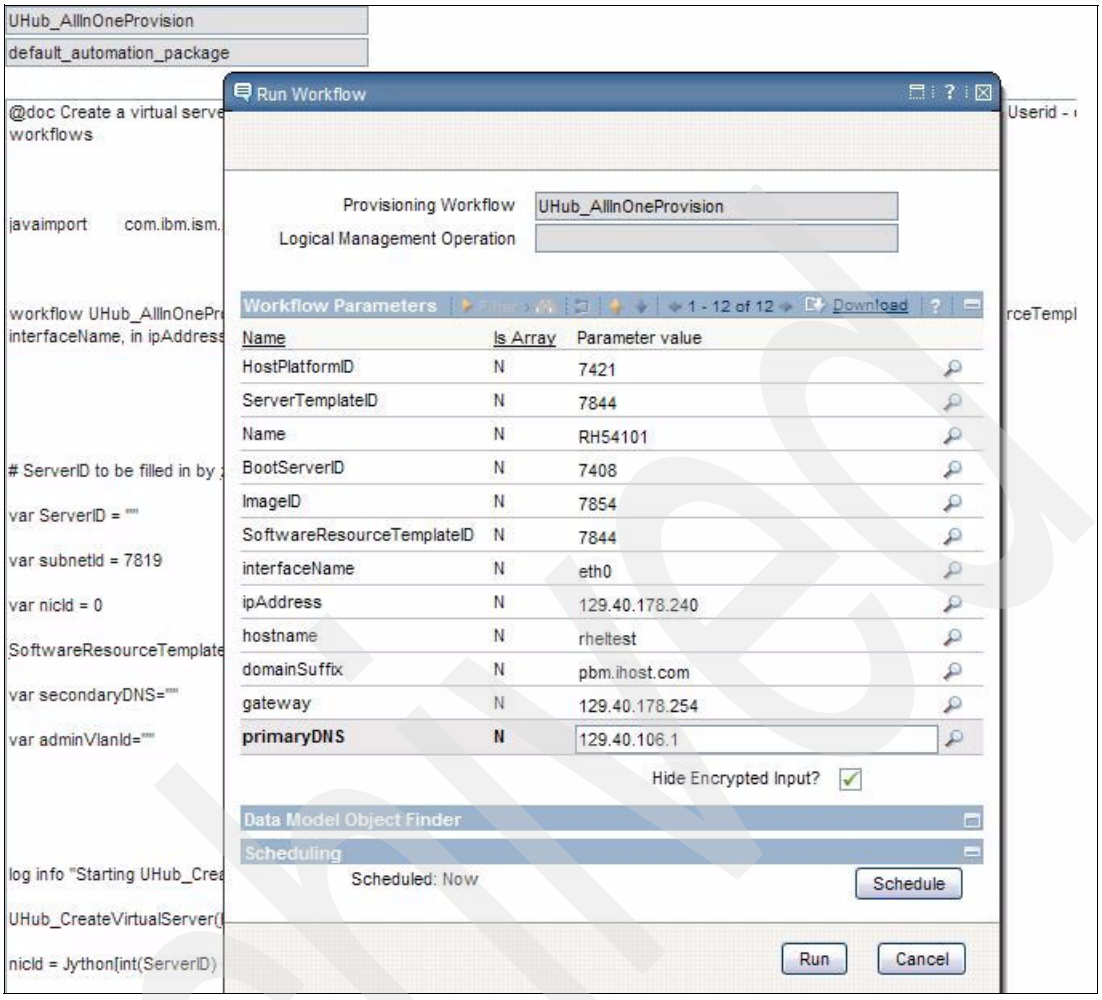

Figure 2 Workflow invocation

Note that while debugging the Tivoli Service Automation Manager and z/VM setup, when the zVM_CreateVirtualServer workflow runs successfully, a z/VM user ID is created. Subsequent steps might still fail, and after correcting the problem you must delete the user ID using DIRMAINT FOR <z/VM user ID> PURGE prior to resuming testing. If the user ID is logged on (typically seen in disconnected or DISC mode), you must LOGOFF the partially provisioned user ID prior to calling DIRMAINT to delete the user ID. The entry for the user ID is likely to have been created in the Tivoli Service Automation Manager database and must be deleted. The DCMID for the guest can be found using the find utility in the Tivoli Service Automation Manager graphical interface and supplying the z/VM user ID name. Use the graphical interface to delete the DCM entry for the user ID.

A look inside of the provisioning process

The provisioning process is a multiphase process starting on Tivoli Service Automation Manager and progressing through Tivoli Provisioning Manager workflows. Many of the System z workflows, which begin with the prefix “Uhub,” call shell scripts that reside on the Management Access Point server. These scripts in turn temporarily mount target disks by attaching them to the MAPSRV user ID. Here more configuration work is done and the system is prepared for the personalization process. In this process the provisioned guest gains a separate identity from the Master system from which it was cloned.

Personalization

Some shell scripts are called on the MAP server near the end of the provisioning process. These can be found on the MAP server in the file: /opt/ibm/ztsam/workflow. A log file named “logfile” is produced in this directory during the execution of these shell scripts. The copying of source disks from the golden master to target disks belonging to the new server creates the Linux image on the new z/VM user ID. The image still needs to be customized with disk and network information. This process is called personalization.

The personalization scripts are installed on the master image when the Master RPM is installed. The RPM is called IBM-System-z.MASTER-1.0-1.s390x.rpm; the personalization scripts are found in /opt/IBM/AE. When the master is cloned, the personalization scripts are also copied onto the new Linux guest. When the Linux guest is booted for the first time, the scripts run and perform personalization. Networking and DASD and other information is changed from the golden master’s values to the values appropriate to the new guest. Personalization on different Linux distributions and versions work differently because the location of network and disk information is different.

The results of the personalization are placed in /tmp/logfile on the provisioned server before being sent back to the MAP server and appended to its logfile in /opt/ibm/ztsam/workflow. If the personalization failed, it is possible to run the personalization in manual mode by issuing this command with the listed parameters.

```
sh /opt/IBM/AE/AE.sh -i /opt/IBM/AE/AP/final.ap
```

The file final.ap is usually removed after a successful personalization but it might still be present after a failure. It is also possible to run the personalization in command mode:

```
sh /opt/IBM/AE/AS/personalize.sh -hostname "sl11test" -domain "pbm.ihost.com"
-password "mypass" -openports "22" -credentials "" -securitylevel "Low" -nics
"1" -portno0 "4420" -porttype0 "VSwitch" -ipaddr0 "129.40.178.240" -netmask0
"255.255.255.0" -gateway0 "129.40.178.254"
```

Deprovisioning

To deprovision a system, find the server using the Tivoli Service Automation Manager “find” dialog in the Data model object finder. Delete the computer from the Tivoli Data center model. Then, after ensuring that the z/VM User ID that contains the Linux guest is logged off, delete the user ID using DIRMAINT, for example:

```
DIRM FOR <ZVM_USER> PURGE CLEAN
```

The CLEAN option is optional and the disk can be reformatted by other means. Be sure the user ID is logged off prior to deletion. Otherwise a user ID purge will be required to get rid of the still-running address space after the user ID is deleted.

Using one z/VM system to provision for another

It is likely to be a common practice that a test z/VM system is set up for provisioning Linux guests for another z/VM system. The z/VM user IDs created on the provisioning system become throw-away user IDs because they are created automatically during provisioning to install the Linux root filesystem on disks that will ultimately be dedicated to another system.

Use dedicated disks for “cross-system” provisioning, rather than Minidisks. Certain conventions must be followed to ensure that, when the disks are made available to the production z/VM system, a Linux guest will be both bootable and network addressable.

On the provisioning system, when a z/VM user ID is created with dedicated disks, a DEDICATE statement is automatically generated in the user directory. The DEDICATE statement virtualizes the disk unit defined as the master's system disk to the unit that will contain the root partition of the new guest. To prepare for provisioning, first build a master on the provisioning z/VM system on a particular disk that is part of the storage unit destined to also contain the root filesystems for newly provisioned guests. The Linux guests built will then use the master's disk address and virtualize it to their own unit that will contain the root filesystem. For example, if the master's disk is unit 9060, then each guest will get created with a DEDICATE statement automatically generated, such as:

```
DEDICATE 9060 <Devno>
```

In this example, Devno is the device address of the guest's root partition. Each guest system will then boot with the same unit, 9060. The user directory for the user could contain an IPL statement as follows, or it could go into a PROFILE EXEC:

```
IPL 9060 CL
```

Once the Linux systems are built on the provisioning system, z/VM user IDs must be manually created on the production system. The definitions for these user IDs on the production system should closely match the same z/VM userid on the provisioning system. The virtual addresses for networking, and the root filesystem disk in particular, need to match the original z/VM userid. The Linux operating system has already been built with these addresses deeply imbedded in the /etc filesystem, and will not function unless their z/VM user entries match those on the provisioning system.

If DIRMAINT is in use on the target system, then the user directories on the provisioning system can be used to help manually create user IDs on the production system. Most of the z/VM security permissions performed automatically on the provisioning system will have to be repeated manually on the production system.

Before bringing up the production z/VM user, you must LOGOFF the z/VM userid on the provisioning system or the disk might not be available on the production system. A better idea is to LOGOFF the user ID, then ensure that the provisioning system can no longer bring the disk online and that it is just online to the target z/VM system.

Volume serial and label considerations

A shell script on the MAPSRV, basicInstall.sh, contains the FLASHCOPY command to copy the master disk to a target disk destined to be the root filesystem on the new guest. The default behavior of FLASHCOPY is to copy the VOLSER Label from the source disk to the target disk. In a provisioning environment with dedicated disks, this can create many "DUPLICATE VOLID" messages because multiple disks have the same volume serial number as the master disk. Adding the "SAVELABEL" parameter to the flashcopy command in basicInstall.sh reduces the need to manually re-clip the target volumes. With the DDR copy command, manual re-clipping of volumes is still required. DDR copy is used when FLASHCOPY is not available or when the source and target disks are on different storage subunits.

Summary

Through the use of the Tivoli Service Automation Manager workflows you can rapidly provision and deprovision Linux virtual servers on IBM z/VM from pre-built Linux master images. Multiple Linux distributions are supported, offering a diverse selection of Linux providers and versions. Depending on the disk-to-disk copying speed, a manually intensive

Linux installation that can take over an hour can be reduced to a matter of minutes through the cloning of pre-installed Linux masters and the personalization of these new servers with their system-unique configuration values.

The creation of the Tivoli Service Automation Manager Data Center Model (DCM) will also support a staged entry into the broader Cloud computing capabilities of the Tivoli Service Automation Manager product. These capabilities include more complex provisioning environments, with multiple hardware devices, software interfaces, resource pooling, and network interfaces. Also, by integrating IT processes with the use of Tivoli Service Automation Manager, the full spectrum of Tivoli service template definition and process automation engine tools becomes available to manage your diverse IT resources and assets.

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paul Sutera is a Software Engineer at the IBM System z Poughkeepsie Benchmark Center. He has been focused on the area of System z rapid provisioning in setting up environments and running Linux on System z performance benchmarks and proof of concept exercises.

Thanks to the following people for their contributions to this project:

Mike Ebbers
IBM International Technical Support Organization, Poughkeepsie Center

Eugene Ong
Stephen McGarril
IBM System z Poughkeepsie Benchmark Center

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author – all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts

- Follow us on twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4663-00 was created or updated on May 3, 2010.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks