Controlling VMware Virtual Machines from the Command Line with vmrun

From Virtuatopia

Previous | Table of Contents
VMware Server 2.0 Security - Access, Roles and Permissions

For all the ease of use provided by VMware Server's graphical tools, perhaps the most powerful tool supplied with various VMware products is a little known command line utility called *vmrun*. Installed as standard with VMware Server, *vmrun* allows almost total control over virtual machines from the command line, either locally on the VMware Server host, or remotely over a network or internet connection. Capabilities of *vmrun* include starting, stopping, pausing, resuming and resetting virtual machines, executing commands within guest operating systems and taking snapshots. The primary goal of this chapter is to focus on these and other features of *vmrun*.

Contents

The Basics and Syntax of vmrun

The *vmrun* tool is installed by default with the VMware Server and VMware Workstation products. Assuming that a standard installation is performed, the *vmrun* executable is located in *\Program Files\VMware\VMware Server* on Windows hosts and */usr/bin* on Linux.

The basic command line syntax for using *vmrun* varies between host platforms and VMware product, but may be generally summarized as follows:

vmrun *<host authentication flags> <guest authentication flags> <command> <parameters>*

The *host authentication flags* are required to provide host information and the login and password for the host system to perform management tasks on the virtual machines. These are essentially the same credentials that would be used when accessing the VI Web Access management interface. These flags are required only on VMware Server hosts, and are not needed for VMware Workstation:

Flag	Description
-h	The https URL of the host to which *vmrun* is required to connect. Must also include the */sdk* sub-directory and, optionally the port number (unless the *-p* flag below is used). For example, https://hostname:8333/sdk.
-P	The port number used by the host for virtual machine management. By default this will be 8333. If the port is specified in the URL (see above) this flag is not required.
-T	The type of VMware product which is running on the host. Options are *ws* for VMware Workstation, *server* for VMware Server 2 and *server1* for VMware Server 1.
-u	The user name on the host to be used to log into the VMware management interface. The same as the user name that would be used when logging in using the VI Web Access interface.
-p	The password corresponding to the user name specified with *-u* as outlined above.

The *guest authentication flags* are specified if *vmrun* is required to log into the guest operating system to perform tasks such as work with guest files or execute commands. If such operations are not required, these flags may be omitted from the command line:

Flag	Description
-gu	A suitable user name via which vmrun may log into the specified guest operating system
-gp	The corresponding password for the guest user specified using the *-gu* flag.

The *command* argument instructs *vmrun* on the task to be performed. For example, *start*, *stop* and *reset* are all valid commands. Other valid commands include those to perform tasks within a guest operating system, such as *runProgramInGuest* and *deleteFileInGuest*.

The *parameters* arguments specify additional information required for a specific command. If, for example, the *start* command is specified, the parameters argument is used to reference the .vmx file of the virtual machine which is to be started. In the case of running guest commands, both the .vmx file of the target virtual machine and the path to the program to be executed must be provided as parameters.

When specifying the virtual machine on which a command is to be executed, the location of the virtual machine *.vmx* configuration file must be provided. For VMware Workstation, this involves specifying the full path of the file. For example:

```
"C:\VMware\VMachines\win2008.vmx"
```

In the case of VMware Server 2.0, which introduced the concept of *datastores*, the name of the datastore in which the virtual machines resides must be provided together with the path of the .vmx file within that datastore. For example, to reference a virtual machine stored in the *win2008* sub-directory of the *Vol1* datastore, the vmrun path parameter would be defined as follows (note that the datastore name is enclosed in square brackets and separated from the sub-directory by a space character):

```
"[Vol1] win2008/win2008.vmx"
```

Bringing all of these different command line arguments together, a typical *vmrun* command to start a virtual machine on a host named *hostname* running VMware Server 2 might appear as follows:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword stop
                    "[Vol1] win2008/win2008.vmx"
```

Performing General Administrative Tasks with vmrun

The *vmrun* tool can be used to perform a number of general administrative tasks such as listing running virtual machines, installing VMware Tools in a guest operating system and registering virtual machines.

To list the currently running virtual machines on a VMware Server host:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword list
Total running VMs: 2
[Vol1] win2008-1/win2008-1.vmx
[Vol1] CentOS 5.2/CentOS 5.2.vmx
```

The VMware Tools installation process may be initiated from the command line as follows, although completion of the installation will need to be performed at the console of the guest operating system:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword installTools "[Vol1] CentOS 5.2/CentOS 5.2.vmx"
```

To register a virtual machine with the inventory on a specific VMware host, use the *register* commands as follows:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword register "[Vol1] CentOS 5.2/CentOS 5.2.vmx"
```

Similarly, to unregister a virtual machine from the host inventory:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword unregister "[Vol1] CentOS 5.2/CentOS 5.2.vmx"
```

Controlling Virtual Machine Power States with vmrun

The power states of virtual machines can be controlled either locally or remotely using the *vmrun* tool. The commands associated with these tasks are as follows:

- **start** - Powers on the specified virtual machine. The virtual machine may started either with a GUI console visible, or no console using the *gui* and *nogui* options.
- **stop** - Powers off the specified virtual machine. The power off process can be specified as *hard* or *soft*.
- **reset** - Reboots the specified virtual machine. The reboot process can be specified as *hard* or *soft*.
- **Suspend** - Suspends a virtual machine allowing fast restart via the **start** command. The suspend process can be specified as *hard* or *soft*.
- **pause** - Pauses the specified virtual machine.

- **unpause** - Unpauses a paused virtual machine.

When the *hard* option is specified for the stop and suspend commands, the virtual machine state changes without giving the guest operating system the opportunity to execute shutdown or hibernate procedures (analogous to disconnecting the power to a physical computer system). To ensure an ordered power off or suspension, be sure to specify the *soft* option. For example, to power off a specific virtual machine residing on host named *hostname* using the soft method:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword stop
        "[Vol1] win2008/win2008.vmx" soft
```

Similarly, to subsequently start the same virtual machine:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword reset
         "[Vol1] win2008/win2008.vmx"
```

Virtual Machine Snapshots with vmrun

VMware allows snapshots of virtual machines to be taken, whereby the state of a virtual machine (including the guest operating system) may be saved and then restored at a later date. This feature is particularly useful when performing operating system or application testing. Whilst snapshots may be taken and restored using the VI Web Access interface, it is often quicker and more convenient to use the *vmrun* tool. As with other *vmrun* commands, the appropriate authentication, host and virtual machine arguments must be provided along with a snapshot name.

The following example takes a snapshot (named MySnapshot) of a virtual machine using the *snapshot* command:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword snapshot
        "[Vol1] CentOS 5.2/CentOS 5.2.vmx" MySnapshot
```

The corresponding command to revert the virtual machine to the snapshot state uses the *revertToSnapshot* command and would read as follows:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword revertToSnapshot
        "[Vol1] CentOS 5.2/CentOS 5.2.vmx" MySnapshot
```

Note that if the virtual machine is running at the time the *revertToSnapshot* command is issued, the virtual machine will be placed into the suspended state while the restoration process is performed. To resume the suspended virtual machine using the restored state, it may be resumed using the *vmrun* start command as follows:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword start
        "[Vol1] CentOS 5.2/CentOS 5.2.vmx"
```

Finally, an existing snapshot may be deleted using the *deleteSnapshot* command:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword deleteSnapshot
         "[Vol1] CentOS 5.2/CentOS 5.2.vmx" MySnapshot
```

Running and Managing Guest Programs with vmrun

One of the most useful features of the *vmrun* command is the ability to remotely run and manage programs on the guest operating system of a virtual machine.

A listing of the current process running on a guest operating system can be obtained using the *listProcessesinGuest* command. In order for this command to work, a valid guest user name and password must be provided. The following example show a typical command used to obtain a process listing from a Windows Server 2008 system, together with part of the output listing:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu administrator -gp guestpasswd
     listProcessesInGuest "[Vol1] win2008-1/win2008-1.vmx"

Process list: 39
pid=0, owner=, cmd=[System Process]
pid=4, owner=, cmd=System
pid=392, owner=, cmd=smss.exe
pid=468, owner=, cmd=csrss.exe
pid=512, owner=, cmd=csrss.exe
pid=520, owner=, cmd=wininit.exe
pid=548, owner=, cmd=winlogon.exe
pid=596, owner=, cmd=services.exe
.
.
.
```

To run a program within a guest operating system, use the runProgramInGuest command, together with the path to the program which is to run. For example:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu administrator -gp guestpaswd
  runProgramInGuest "[Vol1] win2008-1/win2008-1.vmx" -activeWindow "c:\windows\system32\cmd.exe"
```

Note that in order to run a graphical application on a Windows Vista or Windows Server 2008 guest, the *-interactive* command must be used. Without this command, the guest Windows system will display an *Interactive services dialog detection* warning dialog stating that the program cannot be displayed on the desktop.

By default, graphical programs started using *vmrun* on Windows guests will start in a minimized state (in other words the program will appear in the Windows task bar, but will not be displayed on the desktop). To force the program to appear on the desktop when it starts, the *-activeWindow* flag must be used. The following example shows the combined use of the *-interactive* and *-activeWindow* flags to run and display the Windows Notepad program on a Windows Server 2008 guest system:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu administrator -gp guestpasswd
     runProgramInGuest "[Vol1] win2008-1/win2008-1.vmx"
     -activeWindow -interactive "c:\windows\system32\notepad.exe"
```

To terminate a program running in a guest operating system, use the *listProcessesInGuest* command described above to ascertain the process id (pid) of the program in question and use that id (2368 in the following example) together with the *killProcessInGuest* command:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu administrator -gp guestpasswd
            killProcessInGuest "[Vol1] win2008-1/win2008-1.vmx" 2368
```

Running Scripts Within a Guest

In addition to running executable programs on a guest operating system, *vmrun* also supports the running of scripts. Scripts are generally text based files containing instructions which are executed by an interpreter. For example, a Perl script would be executed using the *perl* interpreter and Ruby script by the *ruby* interpreter. Whilst some operating systems and scripting languages allow scripts to be executed just as any other program is executed (and without referring to the interpreter in the command line) this is not always the case. In such situations the command line to execute the script must provide the path to the script as a command line argument to the interpreter. For example, suppose that the following Ruby script contained in a file called *hello.rb* located in */home/ruby* needs to be executed:

```
print "hello ruby!\n"
print "goodbye ruby!\n"
```

In order to execute this script, the hello.rb file needs to be passed as a command line argument to the *ruby* interpreter program as follows:

```
/usr/bin/ruby /home/ruby/hello.rb
```

In essence, the task of executing a script in this way using *vmrun* is very similar. In the following example, the *hello.rb* script is executed using the *runScriptInGuest* command:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu administrator -gp guestpasswd
     runScriptInGuest "[Vol1] CentOS 5/Centos 5.vmx" /usr/bin/ruby /home/ruby/hello.rb
```

Using vmrun to Work with Guest Operating Systems Files and Directories

A particularly powerful feature of *vmrun* is the ability to interact with files residing in guest operating systems from either the host, or a remote system. For example, vmrun can be used to identify the existence of a file in a specific virtual machine guest, delete a file, create a directory and copy files between guest and host. In each case, a user name and password valid for logging into the guest operating system must be provided using the *-gu* and *-gp* flags.

The following commands require that VMware Tools be installed in the guest operating system. For more details on VMware Tools refer to the chapter entitled Understanding and Installing VMware Tools.

To check for the existence of a particular file:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
          fileExistsInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyFile

The file exists.
```

To delete a file on a guest:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
           deleteFileInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyFile
```

To create a new directory in a guest filesystem:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
           createDirectoryInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyNewDir
```

To remove a directory from a guest filesystem:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
           deleteDirectoryInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyNewDir
```

To list the files in a guest system directory:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpassword
           listDirectoryInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred
Directory list: 5
.esd_auth
.dmrc
.gconfd
.recently-used.xbel
Desktop
```

To copy a file from the host system to the guest operating system (i.e copy the file /home/fred/MyFile on the guest to /tmp/MyFile on the host):

```
 vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
      copyFileFromGuestToHost "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyFile /tmp/MyFile
```

Conversely, to copy a file from the host to the guest:

```
 vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
      copyFileFromHostToGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /tmp/MyFile /home/fred/MyFile
```

To rename a file on the guest:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd
     renameFileInGuest "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/MyFile /home/fred/MyNewFile
```

Capturing a Virtual Machine Screenshot

A screenshot of the console of a running virtual machine may be captured and written to a file on the host using the vmrun *captureScreen* command. The name of the file to which the image is to be written is supplied as a parameter to the command. Note that the image is created in PNG format, so the file should be given the appropriate *.png* filename extension. For example:

```
vmrun -T server -h https://hostname:8333/sdk -u root -p mypassword -gu fred -gp guestpasswd captureScreen
      "[Vol1] CentOS 5.2/CentOS 5.2.vmx" /home/fred/images/vm_screen.png
```

Retrieved from "https://www.virtuatopia.com/index.php?title=Controlling_VMware_Virtual_Machines_from_the_Command_Line_with_vmrun&oldid=2015"