# OpenNebula - Feature #216
# Additional hook types

04/20/2010 02:43 PM - Maciej Kruk

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 04/20/2010 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | Core & System | **Estimated time:** | 0.50 hour |
| **Target version:** | | | |
| **Resolution:** | wontfix | **Pull request:** | |

**Description**

Hello!

When using OpenNebula 1.4 as a base for our cloud, we encountered a problem with key injection. We needed to launch an injection script after the image was copied to the host, but before it started booting. There was no hook type that allowed to do that, therefore, we added a new one - TRANSFERRED.

Another functionality we required was a hook launched after machine was shutdown, but before it was deleted (for saving purpose). We didn't know which VM state to use, so we added it on suspend and are now suspending the machine if we want to save it.

It was pretty easy after we found the place where it resides. The problem is we had to recompile the code and we would like to avoid doing that with each release. And key injection is such a popular feature, I think we're not the only ones that are going to use it.

Could you add the following hook types (and preferably another one, launched after the VM is shutdown, before it's deleted) to the next release?

Thanks,
makhan

file: vm/VirtualMachinePool.cc
line: 89

```
else if ( on == "TRANSFERRED" )
{
    VirtualMachineStateHook * hook;

    hook = new VirtualMachineStateHook(name, cmd, arg, remote,
        VirtualMachine::BOOT, VirtualMachine::ACTIVE);
    add_hook(hook);

    state_hook = true;
}
else if ( on == "SUSPENDED" )
{
    VirtualMachineStateHook * hook;

    hook = new VirtualMachineStateHook(name, cmd, arg, remote,
            VirtualMachine::LCM_INIT, VirtualMachine::SUSPENDED);
    add_hook(hook);

    state_hook = true;
```

```
}
```

## History

#### #1 - 04/22/2010 02:01 PM - Ruben S. Montero

Hi,

I would say that we can add this new hook (although I'd call it on "BOOT"). Your hook is executed when the VM enters the BOOT state, so the hypervisor would probably trying to boot the VM. I am just wondering how the key injection is done?. Alsp have you checked the CONTEXT attribute, we are actually using it for key injection, as well as put custom programs scripts....

thanks for the feedback!

#### #2 - 04/22/2010 11:40 PM - Maciej Kruk

Hi,

Thank you for your response. We are using external script to inject keys, executed on the host. It tries to mount the image as a loop device, and copy the key into it. We are not using the CONTEXT script in order to keep our interference with images as low as possible.
As a consequence of mounting image file, the script must be executed before machine boots and mounts the drive we are injecting keys into. As far as we can tell, doing this at the beggining of BOOT state is early enough.
We didn't write the script ourselves, we actually found it in Eucalyptus and just tuned a bit (added logging, suid and made it compatible with OpenNebula). You can find the original version here:
http://www.sfr-fresh.com/linux/misc/eucalyptus-1.5.1-src-online.tar.gz:a/eucalyptus-1.5.1/tools/add_key.sh

Thanks for responding,
Maciej

#### #3 - 04/23/2010 09:27 AM - Ruben S. Montero

Hi,

Would not it be safer to add that in tm_clone.sh script?. tm_clone.sh is the program called by OpenNebula to clone the image to the cluster node, so you could just call the script after the copy is done. When the tm_clone.sh succeed the VM is moved to BOOT state. In this way you'd be sure that the VM is not boot and so that you are not trying to mount the image at the same time as the VM.

In the same way tm_delete.sh is called to delete the VM images, in case you want to do something before actually deleting the VM images...

The tm_* scripts ara in some way hooks for image manipulation so you can tune those to do image related tasks. Note also that the adaptations you do to those scripts are compatible with future OpenNebula releases...

Does it make sense?

Cheers

#### #4 - 04/23/2010 11:54 AM - Maciej Kruk

Hi,

Yeah, it makes sense, but there is one problem. We need to pass some arguments to the script and as far as I can tell, tm_clone (in opposition to hooks) doesn't have access to VM variables (passed in the configuration file).

Cheers

#### #5 - 04/23/2010 11:55 AM - Maciej Kruk

Hi,

Yeah, it makes sense, but there is one problem. We need to pass some arguments to the script and as far as I can tell, tm_clone (in opposition to hooks) doesn't have access to VM variables (passed in the configuration file).

Cheers

#### #6 - 04/29/2010 03:32 PM - Ruben S. Montero

Hi

You can use onevm show <ID> from the tm scripts to grab what you need. The ID is passed to the scripts.

Cheers

#### #7 - 05/04/2010 10:43 AM - Gyula Csom

Hi
We met the same question, eg. how to retrieve vm definition from the transfer scripts?

I've made some investigation and found the following:

1. oned's tm subsystem passes the VM_ID and the prolog file path to the tm_mad:

```
32:    os << "TRANSFER " << oid << " " << xfr_file << endl;
```
source:/src/tm/TransferManagerDriver.cc

```
48:    def action_transfer(number, script_file)
```
source:/src/tm_mad/one_tm.rb

2. However the tm_mad doesn't pass the VM_ID to the TMScript when executing the prolog file:

```
48:    def action_transfer(number, script_file)
...
56:            script=TMScript.new(script_text, log_method(number))
57:            res=script.execute(@plugin)
```
source:/src/tm_mad/one_tm.rb

3. Thus the TMScript executes just what it finds within the prolog file:

```
140:    def execute(plugin)
...
143:        result = @lines.each {|line|
144:            res = plugin.execute(@logger, *line)
source:/src/tm_mad/TMScript.rb
```

4. Unfortunately it seems that the tm subsystem (source:/src/tm/TransferManager.cc) doesn't put the vm id into the prolog file when cloning images. It just passes the source, the destination and optionally the size. For instance here's a sample prolog file:

```
CLONE node0:/mnt/vm-images/arch.cow node1:/mnt/vm-images/one/45/images/disk.0
CONTEXT /opt/one/var/45/context.sh /mnt/vm-images/context/arch-context-init.sh node1:/mnt/vm-images/one/45/images/disk.1
```

After all it seems that the VM_ID is not passed to the clone script. But again I could be wrong.

Cheers,
Gyula

#### #8 - 05/04/2010 12:23 PM - Gyula Csom

The tm_mad might be patched in order to pass vm_id to transfer scripts. Something like this:

TM_MAD patch for passing WM_ID to the transfer scripts in the following format:  VM_ID = $VM_ID

Patch for source:/src/tm_mad/one_tm.rb: passing vm id to TMScript:

```
48:   def action_transfer(number, script_file)
...
56:           script=TMScript.new(script_text, log_method(number))
57:           res=script.execute(@plugin, number)
```

Patch for source:/src/tm_mad/TMScript.rb: appending VM_ID to the command:

```
140: def execute(plugin, number)
143:        result = @lines.each {|line|
144:            line << "VM_ID=#{number}"
145:            res = plugin.execute(@logger, *line)
```

Then a transfer script might set the VM_ID like this:

```
#!/bin/bash

for arg; do
  case $arg in
    VM_ID=*)
      eval $arg
```

```
        ;;
    esac
done
```

*Note that the VM_ID cannot be passed simply as the last argument, since the there may be optional transfer arguments (like SIZE).*

#### #9 - 05/04/2010 02:58 PM - Gyula Csom

Corrections:

1./ I reviewed my last comment regarding passing the VM_ID as the last argument. I guess it is not really true.

Generally it seems that it's the transfer script what really matters, eg. whether it accepts optional arguments or not, and if yes, how it handles them.

Especially I think that if a transfer script handled an optional parameter (expecting it in a fixed position) then the above patch might still break it. Meanwhile if none of the scripts handled optional arguments then simply passing the VM_ID as the last argument should work, eg.:

```
140: def execute(plugin, number)
143:       result = @lines.each {|line|
144:           line << number.to_s
145:           res = plugin.execute(@logger, *line)
```

I've ran through the builtin clone and ln scripts. It seems that none of them handles optional arguments. For instance the ssh, nfs and vmware clone scripts don't handle the size parameter at all, meanwhile the lvm clone script handles it as a required parameter, expecting it in the 3rd position. So I guess, at least the patch might work against cloning.

2./ Finally I guess that the clone destination path contains the VM_ID. For instance 45 means the VM_ID in the above example:

```
node1:/mnt/vm-images/one/45/images/disk.0'
```

.

Thus the VM_ID might be extracted from the original path arguments. This may not be elegant, but works without deep modification. Is this correct?:)

#### #10 - 05/04/2010 03:58 PM - Ruben S. Montero

Hi

My first thought was to obtain the VMID from the DST argument passed to the scripts. This can be safely done as the vm_directory is **always** build using the VMID to build the path. Maybe we can add a funtion (get_vmid) in tm_common.sh...

#### #11 - 05/05/2010 11:36 AM - Gyula Csom

A get_vmid function would be useful. Though it's not a request from me, yet, since our project is mostly in the analysis phase...

Our use case is the following. We are using diskless hosts with external storage accessible thru iSCSI. Storage alocation is dynamic, eg. creating iSCSI targets/LUN-s happen when vms are created. Thus it's clear that we have some storage preparation tasks before the vm is booted, however it's not clear whether it requires the detailed knowledge of the vm definition. My colleague already started to implement the iSCSI driver, and it is expected to enter into the alpha state in this month, so I'll be smarter at the end of this month:)

#### #12 - 05/06/2010 11:17 PM - Ruben S. Montero

Hi

It sounds very interesting :) please also consider sharing your developments through the ecosystem. Also you may want to take a look to issue #32 , specially the iscsi driver from Sander . It is not what your are doing but maybe you can get some ideas...

Keep us updated with your progress!

#### #13 - 06/23/2010 10:34 PM - Gyula Csom

Hi, a quick progress report:)

- iSCSI LN is implemented, eg. we can link a previously created image to a newly created vm
- iSCSI MV is under development (checkpoint file handling is missing, otherwise it works)
- Also we've implemented image resize, but it is not yet integrated into the TM MAD (features: the script can increase the last partition, and the filesystem on it, it supports ext2/3/4, reiserfs and ntfs). It might be of interest within the 1.4.2 development line as well, eg.: http://dev.opennebula.org/issues/179

The functions above are currently under test.

Note also that we've found an impediment in our way. When using iSCSI within live migration the new host must login to the iSCSI target before the migration starts and the previous host must logoff when it ends. However there is no TM callback during live migration, and the hook system cannot pass host parameters either. So it seems that we need some update within the core. See the mail sent to the list by my colleague responsible for iSCSI-related development: http://lists.opennebula.org/pipermail/users-opennebula.org/2010-June/002229.html

Cheers,
Gyula

#### #14 - 12/22/2010 05:14 PM - Ruben S. Montero

*- Status changed from New to Closed*
*- Resolution set to wontfix*

Seems that the new hooks are not really needed. Closing...