IBM

developerWorks®

Automate infrastructure updates in NIM environment with Chef

AIX VIOS update with Chef

December 05, 2017

Updating AIX VIOSes in a large scale infrastructure is easier than ever with the use of Chef.
The aix_nimviosupdate chef resource allows updating NIM client VIOSes using a specific lpp source.

Introduction

This article details how to use Chef for **VIOS update automation** on IBM® AIX® systems. In addition to this document, you may refer to "Chef Automate infrastructure updates in NIM environment" which describes the hardware configuration, the installation process and use cases to use Chef to automate AIX patch management.

This article explains the different steps to securely update VIOS. These steps can be run separately but they also can be combined together. An example at the end of this document shows how to combine these steps to update dual VIOS without service interruption. This operation is called "VIOS rolling update".

The steps to perform a VIOS rolling update are:

a) Verify the state of the VIOSes to update by performing an health check of the VIOSes.
b) Create an alternate disk copy for the rootvg of the VIOS for backup purpose in case of failure.
c) Perform the update using updateios.
d) Cleanup (remove the alternate disk, …).

The Chef resource we use for these steps is named: **aix_nimviosupdate**.
Our development supports a NIM (Network Installation Management) environment in **PUSH mode**. VIOS updates cookbooks and recipes are available on **chef-cookbook repository**.
The health check requires getting the vioshc.py Python script available on **AIXOSS vios-health-checker GitHub repository**. This script must be installed on the Chef Client machine (the NIM master) under /usr/sbin.
The **chef-cookbook repository** contains Open Source Software ported to AIX. It also contains scripts to use with Open Source software to perform specific AIX tasks. You will find a library including the Chef

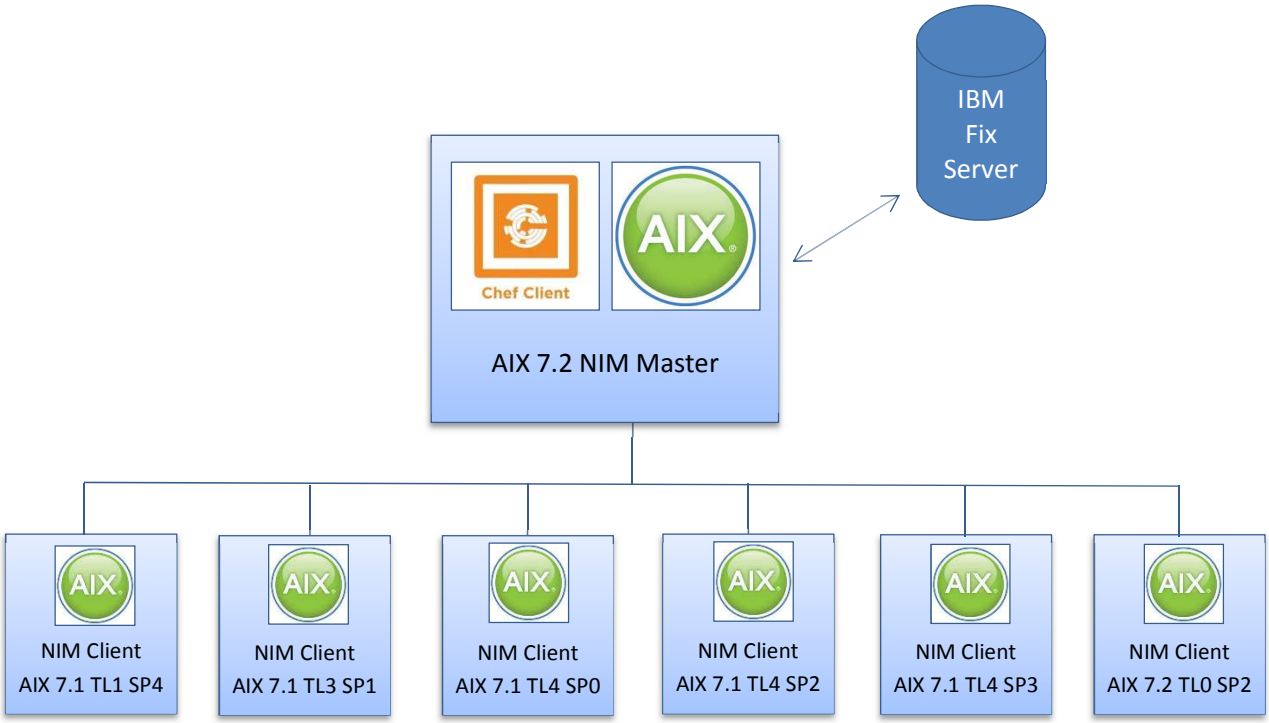
scripts necessary for patch management with Chef, and typical recipes. These recipes can be used as templates for your own purposes.

For ease of use, Chef will be used in **local (or standalone) mode** on the NIM master. But, it may also be configured with a Linux Chef server.

Configuration

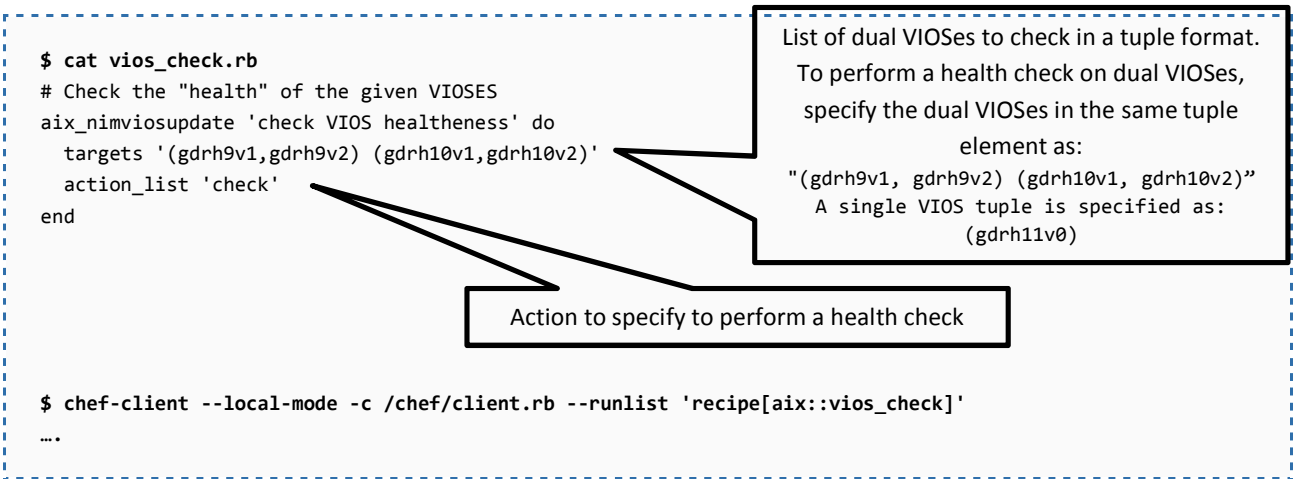
The NIM master has access to the internet in order to download fixes and updates through HTTP and FTP protocol. It has several clients which are at different AIX releases and levels. It needs to be at a level at least as high as the highest level client. As a prerequisite to this demo, it needs to have *wget* and *git* tools installed in order to easily install Chef and AIX VIOS update cookbook and recipes example. But the user can also download and transfer the repository on the NIM client.

The diagram below describes the hardware configuration for the test use case.



Verify the state of the VIOS

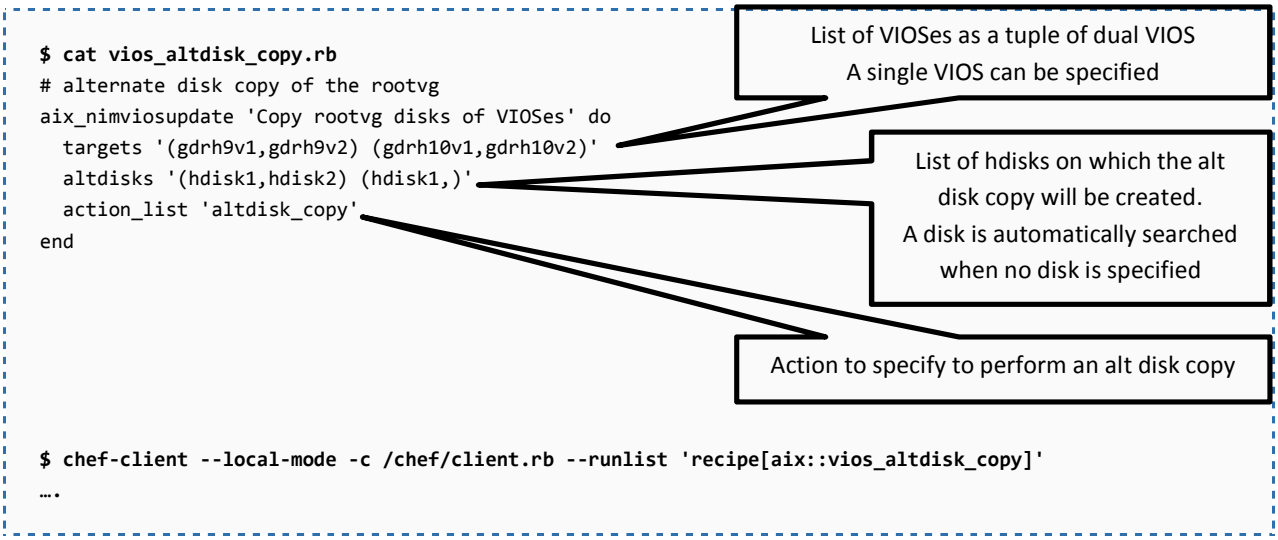
The following recipe shows how to perform a health check of VIOS in order to update them.



The “check” action is used to control that the VIOSes in a pair manage the same objects.

Perform an alternate disk copy

Saving your VIOS rootvg allows you to restore it, in case of problems during the update.



When a disk is not specified for a VIOS in the “altdisks” property list, a free disk (if one exists) is automatically selected. By default the selected disk is the “nearest” in size with the VIOS rootvg. This policy could be modified with the “disk_size_policy” property. When the “altdisks” property is set to “auto”, the automatic disk selection is used for each VIOS.

Note : If the volume group rootvg is mirrored, it will be unmirrored, before creating the alternate disk copy. It will then be mirrored to restore its state.
As a consequence, make sure that all the logical volume included in rootvg are mirrored.
In case of error during the alt_disk_copy operation, the mirroring of rootvg may need to be redone manually by the user.

Perform the VIOS update

The following recipe shows how to perform a NIM updateios operation on each VIOS.

```
$ cat vios_update.rb
# update the VIOSES
aix_nimviosupdate 'Update the VIOSES' do
  targets '(gdrh9v1,gdrh9v2) (gdrh10v1,gdrh10v2)'
  lpp_source 'VIOS_225_30-lpp_source'
  updateios_flags 'install'
  accept_licenses 'yes'
  preview 'no'
  action_list 'update'
end


$ chef-client --local-mode -c /chef/client.rb --runlist 'recipe[aix::vios_update]'
….
```

List of VIOSes as a tuple of dual VIOS
A single VIOS can be specified

Update to apply to the VIOSes

Flags and update options for the updateios command

Action to specify to perform the VIOS update

Note: In case of a tuple in “targets” specifying a couple of VIOS, if one of the VIOS is a node of an active Shared Storage Pool (SSP), the other VIOS must also be part of the same SSP. In addition, both must be in the same SSP state (i.e. “OK” or “DOWN”).
In case of a tuple in “targets” specifying a single VIOS, if this VIOS is part of a SSP, the SSP state for this VIOS must be “DOWN”.

Note: In order to apply the new versions of the lpps, the lpps on the server must be committed. As a result, previous versions of updated lpps in apply state will be committed during the update.

Example to perform all the steps in one recipe

```
$ cat vios_update_all_in_one.rb
# recipe example update
vios = '(gdrh9v1,gdrh9v2) (gdrh10v1,gdrh10v2)'
disks = '(hdisk1,hdisk2) (hdisk1,)'
lpp = 'VIOS_225_30-lpp_source'
update_cmd = 'install'



# check vioses, save rotvg disks and update the vioses
aix_nimviosupdate 'Update the VIOSES' do
  targets vios.to_s
  altdisks disks.to_s
  lpp_source lpp.to_s
  updateios_flags update_cmd.to_s
  accept_licenses 'yes'
  preview 'no'
  action_list 'check, altdisk_copy, update'
end


$ chef-client --local-mode -c /chef/client.rb --runlist 'recipe[aix::vios_update_all_in_one]'
….
```

Definition of variables to use in the different part of the recipe

Use of the defined variables , using the “to_s” method to convert their value to string

List of all the actions to perform in this recipe

Whatever the actions order specified in the “action_list” property, they are always run in the following order: check, altdisk_copy, update and clean.

When multiple VIOSes tuples are specified with the “targets” property, all the actions in the list are executed for each VIOS in the tuple before going to the next tuple.

Example of interactive recipe

```
$ cat vios_update_interactive.rb
# recipe example update

Chef::Recipe.send(:include, AIX::PatchMgmt)

puts '############################################################'
puts 'Available VIOS and their corresponding oslevel are:'
puts vios(node)
puts 'Choose one or two VIOS (comma or space separated) to update?'
puts '(The pair of VIOSes must manage the same LPARs)'
vios = STDIN.readline.chomp
vios_list = vios.to_s.strip.split()
tuple = '(' + vios_list[0].to_s
if vios_list.length > 1
  tuple = tuple + ',' + vios_list[1].to_s
end
tuple = tuple + ')'

hdisk_list = []
vios_list.each do |vios|
  puts '############################################################'
  puts 'Available free disks for #{vios}'
  puts free_vios_disks(vios)
  puts 'Choose a disk to build the alternate disk copy of #{vios}'
  puts 'If no disk is chosen, one will be automatically selected.'
  hdisk = STDIN.readline.chomp
  if hdisk.empty?
    hdisk_list.push(" ")
  else
    hdisk_list.push(hdisk)
  end
end
disks = '(' + hdisk_list[0].to_s
if hdisk_list.length > 1
  disks = disks + ',' + hdisk_list[1].to_s
end
disks = disks + ')'

puts '############################################################'
puts 'Available LPP are:'
puts list_lpp_sources(node)
puts 'Choose one to install?'
lpp = STDIN.readline.chomp
```

```
# check vioses, save rotvg disks and update the vioses
aix_nimviosupdate 'Update the VIOSES' do
  targets tuple.to_s
  altdisks disks.to_s
  lpp_source lpp.to_s
  updateios_flags 'install'
  accept_licenses 'yes'
  preview 'no'
  action_list 'check, altdisk_copy, update'
end
$ chef-client --local-mode -c /chef/client.rb --runlist 'recipe[aix:: vios_update_interactive]'
….
```

This recipe offers the possibility to build the “targets” and “disks” lists interactively and select an existing lpp source to install.