**Automate VIOS upgrade in NIM environment with Ansible**

**AIX VIOS upgrade with Ansible**

December 17, 2018

Upgrading AIX VIOSes in a large-scale infrastructure is easier than ever with the use of Ansible.

Dedicated Ansible playbooks allow upgrading NIM client VIOSes using a specific mksysb image.

# Introduction

This article details how to use [Ansible](https://www.ansible.com/it-automation) for **VIOS upgrade automation** on IBM**®** AIX**®** systems. In addition to this document, you may refer to “Automate infrastructure updates in NIM environment” which describes the hardware configuration, the installation process and use cases to use Ansible to automate AIX patch management.

This article explains the different steps to securely upgrade VIOS. These steps can be run separately. They also can be combined together. An example at the end of this document shows how to combine these steps to upgrade dual VIOS without service interruption, this operation being called “VIOS rolling upgrade”.

The steps to perform a VIOS rolling upgrade are:

1. Verify the state of the VIOSes to upgrade by performing a health check of the VIOSes.
2. Create an alternate disk copy for the rootvg of the VIOS for backup purpose in case of failure.  
   This step is optional, it can be done by the third step, the upgrade step itself using the viosupgrade tool.
3. Perform the upgrade using viosupgrade tool. Tow modes are available:
   1. bosinst: consists to install a new mksysb image on the current rootvg disk.  
      For safety reasons, this mode requires to previously clone the rootvg
   2. altdisk: consists to install the new mksysb image on a new alternate disk.

Our development supports a NIM (Network Installation Management) environment in **PUSH mode**. VIOS upgrades playbooks are available on [**AIXOSS GitHub repository**](https://github.com/aixoss).

The [**AIXOSS GitHub repository**](https://github.com/aixoss) contains Open Source Software ported to AIX. It also contains scripts to use with Open Source software to perform specific AIX tasks. Under “ansible-playbooks”, you will find a library including the Ansible scripts necessary for VIOS upgrade with Ansible, and typical playbooks. These playbooks can be used as templates for your own purposes.

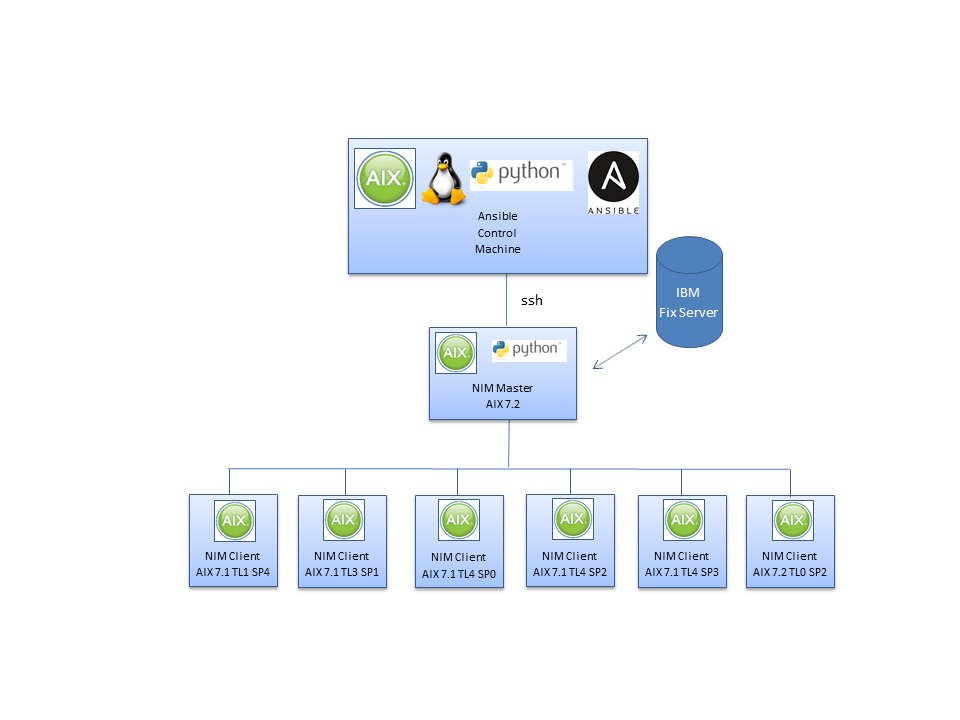
Ansible is agentless. The Ansible control machine connects to Ansible managed node (the NIM server/master in our case) thru ssh. Python is required on both the Ansible control machine and the Ansible managed node.

# Configuration

Ansible can run on AIX or Linux machine. To install Ansible on the targeted machine, *Python, wget* and *git* tools must be installed as a prerequisite.

The NIM master can have several clients with different AIX releases and levels. It needs to be at a level at least as high as the highest-level client.

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



# Download AIX Patch Management Playbooks from AIXOSS GitHub

In order to take advantage of the [**AIXOSS GitHub repository**](https://github.com/aixoss):

**$ /home/users/ansible**

**$ git clone git://github.com/aixoss/ansible-playbooks**

**….**

**$ cd ansible-playbooks**

**$ ls**

**library playbook\_aix\_suma\_nim.yml**

**playbook\_aix\_flrtvc.yml playbook\_aix\_suma\_targets\_all.yml**

**playbook\_aix\_nim\_check.yml playbook\_aix\_suma\_targets\_list.yml**

**playbook\_aix\_nim\_reboot.yml playbook\_aix\_suma\_targets\_range.yml**

**playbook\_aix\_nim\_updateios.yml playbook\_aix\_suma\_targets\_star.yml**

**playbook\_aix\_nim\_viosupgrade.yml playbook\_aix\_suma.yml**

**playbook\_aix\_nim\_vios\_altdisk.yml README.md**

**playbook\_aix\_nim\_vios\_hc.yml**

**playbook\_aix\_nim\_vios\_update.yml**

**$ ls library**

**aix\_flrtvc.py aix\_nim\_updateios.py aix\_nim\_viosupgrade.py aix\_nim\_vios\_hc.py**

**aix\_nim.py aix\_nim\_vios\_alt\_disk.py aix\_suma.py**

**$**

(Material dedicated to the VIOS upgrade automation is highlighted)

# Verify the state of the VIOSes

Create a playbook to perform a health check of the VIOSes to update and run it

Task to execute: “aix\_nim\_vios\_hc”.

The action (“health\_check”) and “targets” must be specified.

**$ cd /home/user/ansible/ansible-playbooks**

**$ cat playbook\_aix\_nim\_vios\_hc.yml**

---

- name: "VIOS health check on AIX"

hosts: all

gather\_facts: no

List of dual VIOSes to check in a tuple format.

To perform a health check on dual VIOSes, specify the dual VIOSes in the same tuple element as:

"(vios1, vios2) (viosA, viosB)”

A single VIOS tuple is specified as: (viosX)

tasks:

- name: "AIX HEALTH CHECK"

aix\_nim\_vios\_hc:

description: 'playbook\_aix\_vios\_health\_check'

targets: "(vios1,vios2) (viosA,viosB)"

action: health\_check

register: hc\_output

- debug: var=hc\_output

Action to execute: “health\_check”.

$ **ansible-playbook** playbook\_aix\_nim\_vios\_hc**.yml**

**"output": [**

**"VIOS CHECK operation for (vios1,vios2), (viosA,viosB)",**

**"Targets list:[('vios1', 'vios2'), ('viosA', 'viosB')]",**

**"VIOS CHECK Status",**

**" vios1-vios2 : SUCCESS-HC",**

**" viosA-viosB : FAILURE-HC"**

**],**

Status of the action.

In case of a multi-tasks playbook, the following steps will only be performed for the “SUCCESS-HC” VIOS

**"status": {**

**"viosA-viosB": "FAILURE-HC",**

**"vios1-vios2": "SUCCESS-HC"**

**},**

**"targets": [**

**[**

**"vios1",**

**"vios2"**

**],**

**[**

**"viosA",**

**"viosB"**

**]**

**]**

You can register a hash table with the name of your choice (here **hc\_output**) to gather debug and data output as well as the health check operation results to control other tasks operations.

The different fields of the hash are:

changed – true is the node state has changed,

msg – brief message explaining the state or the error if any,

targets – the target list effectively used,

nim\_node – NIM info gathered in the module that can be re-used by other modules,

status – result for each target that other modules can check before performing further operation,

debug\_output – output containing debug information,

output – output of the different execution phases of the module.

The “aix\_nim\_vios\_hc” task requires vioshc.py as a prerequisite. vioshc.py is available at

https://github.com/aixoss/vios-health-checker.

# Create an alternate disk copy

Create a playbook to perform the alternate disk copy of the VIOSes to update and run it

**$ cd /home/user/ansible/ansible-playbooks**

List in a tuple format with the 1st element the VIOS and the second element the disk used for the alternate disk copy

To perform the “alt\_disk\_copy” on a dual VIOSes the tuple has the form: (vios1,disk1,vios2,disk2)

For a single VIOS the tuple form is: (vios1,disk1)

Task to execute: “aix\_nim\_vios\_alt\_disk”

The action (“alt\_disk\_copy” or “alt\_disk\_clean”) must be specified.

**$ cat playbook\_aix\_nim\_vios\_altdisk.yml**

---

- name: "VIOS alternate disk copy on AIX"

hosts: all

gather\_facts: no

vars:

log\_file: "/tmp/ansible\_vios\_alt\_disk\_debug.log"

tasks:

Action to execute: “alt\_disk\_copy”.

The action (“alt\_disk\_copy” or “alt\_disk\_clean”) must be specified

- name: "AIX VIOS ALT DISK COPY"

aix\_nim\_vios\_alt\_disk:

description: 'Perform an alternate disk copy on VIOS'

targets: "(vios1,hdisk1,vios2,hdisk2)"

action: alt\_disk\_copy

vars: "{{ vars }}"

A time\_limit could be specified.

The time is checked when starting to perform the “alt\_disk\_copy” for each tuple

#time\_limit: "mm/dd/yyyy hh:mm"

register: altdisk\_copy\_result

- debug: var=altdisk\_copy\_result

$ ansible-playbook playbook\_aix\_nim\_vios\_altdisk.yml

"output": [

"VIOS Alternate disk operation for (vios1,hdisk1,vios2,hdisk2)",

" Targets list: [('vios1', 'hdisk1', 'vios2', 'hdisk2')]",

" Check the alternate disk hdisk1 on vios1",

" Using hdisk1 as alternate disk on vios1",

" Alternate disk copy on vios1",

" Check the alternate disk hdisk2 on vios2",

" Using hdisk2 as alternate disk on vios2",

" Alternate disk copy on vios2",

"VIOS Alternate disk operation status:",

" vios1-vios2: SUCCESS-ALTDC"

],

"status": {

"vios1-vios2": "SUCCESS-ALTDC"

},

"targets": [

[

"vios1",

"hdisk1",

"vios2",

"hdisk2"

]

]

The **vars** attribute passes additional settings to the module. In particular, you can specify a file name where the log will be written. This attribute is optional and is global through the playbook. For a module to use this variable, you may want to specify the following line in the task:

vars: "{{ vars }}"

The optional **time\_limit** attribute limits new alternate disk copy operation if the specified date/time is reached. Its format is month/day/year hour:minutes like the following: mm/dd/yyyy hh:mm

**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 upgrade

Create a playbook to perform the VIOS upgrade and run it

**$ cat /home/user/ansible/ansible-playbooks/playbook\_aix\_nim\_viosupgrade.yml**

**- name: "NIM ios upgrade on AIX playbook"**

**hosts: all**

**gather\_facts: no**

**vars:**

**log\_file: "/tmp/ansible\_viodupgrade\_debug.log"**

Task to execute: “aix\_nim\_viosupgrade”.

**tasks:**

**- name: "upgrade ios"**

**Mandatory**: **list** of strings: VIOSes targets to upgrade in a tuple format.

Specify a dual VIOSes in the same tuple element as: "vios1, vios2" with "," or ":" or space separator.

A single VIOS tuple is specified as: "vios3"

**aix\_nim\_viosupgrade:**

**targets: ["vios1, vios2", "viosA vioB"]**

**Mandatory**: **dictionary**: specify per vios the operation to perform: "**bosint**" or "**altdisk**". **Default** value is specified by **"all\_vios"** key

**actions: {'vios1': 'altdisk', 'all\_vios': 'bosinst'}**

**ios\_mksysb : {'vios1': 'ios\_1844B\_72M', 'all\_vios': 'ios\_1848A\_72M'}**

**Dictionary**: specify per vios the mksysb image to install.

**Default** value is specified by **"all\_vios"** key

**Dictionary** to specify per vios the disks to install the new image or to clone the rootvg

**alt\_disk : {'vios1': 'hdisk1', 'vios2': 'hdisk2 hdisk4'}**

**user\_res: {'vios1': 'script\_name file\_res\_name', 'all\_vios': 'resolv\_conf\_name'}**

**Dictionary**: Specify for each vios and or far all vioses a list of nim resources to use during installation. Supported nim resource type are: resolv\_conf, script, fb\_script, file\_res, image\_data, log.

The file\_res type is not supported for altdisk installation mode

**Force: {'vios1': True, 'all\_vios': False}**

**Dictionary**: Specify for each vios whether the vios will be installed or not in case of the ioslevel == ios\_mksysb level.

The keywords of dictionary are vios\_name or ‘all\_vios’.

The values are Boolean: True or False

**register: upgrade\_result**

**- debug: var= upgrade\_result**

$ **ansible-playbook** **playbook\_aix\_nim\_viosupgrade.yml**

**"upgrade\_result": {**

**"changed": false,**

**"msg": "NIM updateios VIOSUpgrade operation completed successfully",**

**"upgrade\_result.output": [**

**"Log file: /tmp/ansible\_viosupgrade\_debug.log",**

**"VIOSUpgrade operation for ['wrong\_vios', vios1 vios2]",**

**"vios: wron\_vios is not a nim client.",**

**"Then the "wrong\_vios" target will not be selected for upgrade operation",**

**"Remaining Targets list:['vios1 vios2']",**

**"VIOSUpgrade operation on target: "vios1 vios2" end with status: UPGRADE-SUCCESS.",**

**"VIOSUpgrade altdisk operation status on "vios1": UPGRADE-SUCCESS.",**

**"VIOSUpgrade bosinst operation status on "vios2": UPGRADE-SUCCESS.",**

**"VIOSUpgrade operation on target: "wrong\_vios" end with status: UPGRADE-REJECTED.",**

**"VIOSUpgrade operation succeeded"**

**]**

**"upgrade\_result.status": {**

**"p7jufv1 p7jufv2": "UPGRADE-SUCCES",**

**"wrong\_vios": "UPGRADE-REJECTED"**

**}**

**"targets": "(vios1 vios2)"**

**}**

.

You can register a hash table to gather debug and data output as well as the health check operation results to control other tasks operations.

The different fields of the hash are the same as the [aix\_nim\_vios\_hc](#hash_table_result) module.

**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”).

# Example of a full scenario

Generally, an update should be started only when the health-check verifications and an alternate disk copy of the VIOSes are successfully made. The following playbook shows a way to run these tasks one after the other depending on the results of the previous task for each target.

**$ cd /home/user/ansible/ansible-playbooks**

**$ cat playbook\_aix\_nim\_vios\_update.yml**

**- name: "VIOS update on AIX"**

Use the playbook variable “vars” to specify the log file name to use in all tasks

**hosts: all**

**gather\_facts: no**

**vars:**

**log\_file: "/tmp/ansible\_vios\_upgrade\_debug.log"**

**tasks:**

First, execute the “aix\_nim\_vios\_hc” task.

Register for the status: **hc\_result**

**- name: "AIX VIOS HEALTH CHECK"**

**aix\_nim\_vios\_hc:**

**description: 'Check the VIOS(es) can be updated'**

**targets: "(viosA) (gdrh10v2) (vios1,vios2)"**

**action: "health\_check"**

Then, execute the “aix\_nim\_vios\_alt\_disk“ task with action “alt\_disk\_copy”.

Use the “vios\_status” parameter to perform this action only on VIOS tuples that succeed the health check phase.

**vars: "{{ vars }}"**

**register: hc\_result**

**- name: "AIX VIOS ALT DISK COPY"**

**aix\_nim\_vios\_alt\_disk:**

**description: 'Perform the rootvg copy to an alternate disk'**

**targets: "(gdrh10v1,hdisk1) (gdrh10v2,hdisk2) (vios1,hdisk1,vios2,hdisk1)"**

**action: "alt\_disk\_copy"**

**vios\_status: "{{ hc\_result.status }}"**

**vars: "{{ vars }}"**

**register: altd\_result**

Pursue with the “aix\_nim\_updateios“ task.

Use the “vios\_status” parameter to perform this action only on VIOS tuples that succeed the alternate disk copy phase.

**- name: "AIX NIM vios upgrade"**

**aix\_nim\_viosupgrade:**

**targets: ["viosA viosB", "vios1, vios2"]**

**action: {'all\_vios': 'bosinst'}**

**ios\_mksysb: {'all\_vios': 'ios\_1848A\_72M'}**

**user\_res: {'all\_vios': 'master\_net\_conf'}**

**vios\_status: "{{ altd\_result.status }}"**

**vars: "{{ vars }}"**

**register: upgrade\_result**

Finally list the VIOS status at each step of this playbook.

When a task on a tuple succeeds, the status is one of: SUCCESS-HC, SUCCESS-ALTDC or UPGRADE-SUCCESS

In case of failure, it is one of: FAILURE-HC, FAILURE-ALTDCOPY1, FAILURE-ALTDCOPY2, UPGRADE-FAILURE, FAILURE-NO-PREV-STATUS or UPGRADE-REJECTED

**- debug: var=hc\_result.status**

**- debug: var=altd\_result.status**

**- debug: var=upgrade\_result.status**

$ **ansible-playbook** **playbook\_aix\_nim\_vios\_update.yml**

**….**

**…**

The **vios\_status** attribute gets the result of a previous operation. It could be the health check result for example the alternate disk copy operation result. The specified action is performed only if the previous operation result is “success”. Here the upgrade can be done only if the clone rootvg was successful because otherwise there should not be an alternate disk copy to restore.

It is not possible to perform the **alt\_disk\_clean** action in the same playbook as the vios upgrade. You can use a separate playbook to perform the cleanup after a validation period.