Automate VIOS updates in NIM environment with Ansible

AIX VIOS update with Ansible

December 5, 2017

Updating AIX VIOSes in a large scale infrastructure is easier than ever with the use of Ansible. Dedicated Ansible playbooks allow updating NIM client VIOSes using a specific lpp source.

Introduction

This article details how to use Ansible for **VIOS update 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 update 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 update dual VIOS without service interruption, this operation being 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, …).

Our development supports a NIM (Network Installation Management) environment in **PUSH mode**. VIOS updates playbooks are available on **AIXOSS GitHub repository**.
The **AIXOSS GitHub repository** 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 update with Ansible, and typical playbooks. These playbooks can be used as templates for your own purposes.

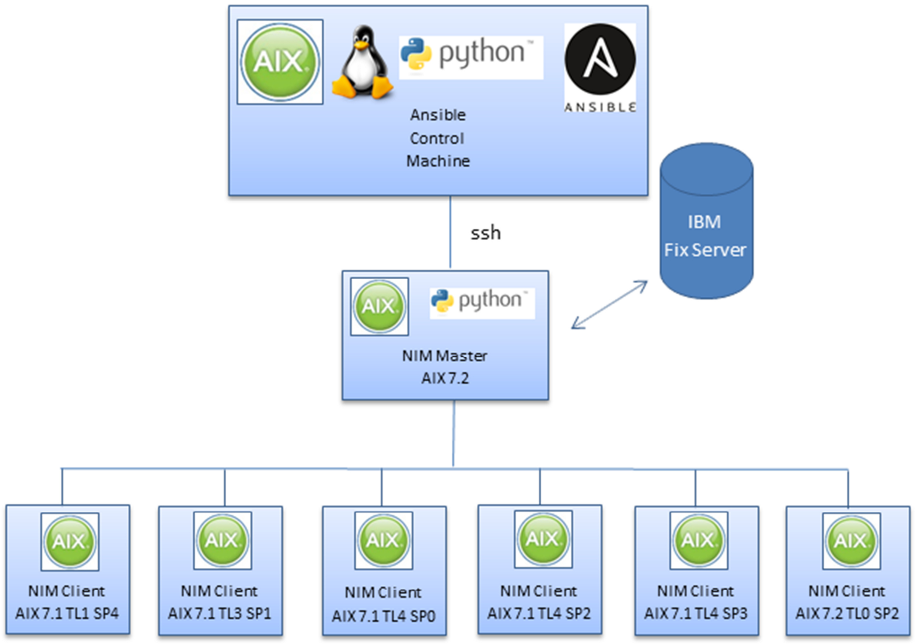
Ansible is agentless. The Ansible control machine connects to Ansible managed node (the NIM server/master in our case) thru ssh. Python is required on both the Ansible control machine and the Ansible managed node.

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 must have access to the internet in order to download fixes and updates through HTTP and FTP protocol. It 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.


AIX
python
ANSIBLE
Ansible Control Machine
ssh
IBM Fix Server
python
NIM Master AIX 7.2
NIM Client AIX 7.1 TL1 SP4
NIM Client AIX 7.1 TL3 SP1
NIM Client AIX 7.1 TL4 SP0
NIM Client AIX 7.1 TL4 SP2
NIM Client AIX 7.1 TL4 SP3
NIM Client AIX 7.2 TL0 SP2

Download AIX Patch Management Playbooks from AIXOSS GitHub

In order to take advantage of the **AIXOSS GitHub repository**:

```
$ /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_vios_altdisk.yml  playbook_aix_suma.yml
playbook_aix_nim_vios_hc.yml       README.md
playbook_aix_nim_vios_update.yml

$ ls library
aix_flrtvc.py  aix_nim_updateios.py      aix_nim_vios_hc.py
aix_nim.py     aix_nim_vios_alt_disk.py  aix_suma.py

$
```

(Material dedicated to the VIOS update 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

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


$ cat playbook_aix_nim_vios_hc.yml
---
- name: "VIOS health check on AIX"
  hosts: all
  gather_facts: no

  tasks:

    - name: "AIX HEALTH CHECK"
      aix_nim_vios_hc:
        description: 'playbook_aix_vios_health_check'
        targets: "(gdrh9v1,gdrh9v2) (gdrh10v1,gdrh10v2)"
        action: health_check

      register: hc_output

    - debug: var=hc_output


$ ansible-playbook playbook_aix_nim_vios_hc.yml

        "output": [
            "VIOS CHECK operation for (gdrh9v1,gdrh9v2), (gdrh10v1,gdrh10v2)",
            "Targets list:[('gdrh9v1', 'gdrh9v2'), ('gdrh10v1', 'gdrh10v2')]",
            "VIOS CHECK Status",
            " gdrh9v1-gdrh9v2 : SUCCESS-HC",
            " gdrh10v1-gdrh10v2 : FAILURE-HC"
        ],
        "status": {
            "gdrh10v1-gdrh10v2": "FAILURE-HC",
            "gdrh9v1-gdrh9v2": "SUCCESS-HC"
        },
        "targets": [
            [
                "gdrh9v1",
```

Task to execute: “aix_nim_vios_hc”.

The action (“health_check”) and “targets” must be specified.

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:
`"(gdrh9v1, gdrh9v2) (gdrh10v1, gdrh10v2)”`
`A single VIOS tuple is specified as: (gdrh11v0)`

Action to execute: “health_check”.

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

```
                "gdrh9v2"
            ],
            [
                "gdrh10v1",
                "gdrh10v2"
            ]
        ]
```

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

$ 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:
    - name: "AIX VIOS ALT DISK COPY"
      aix_nim_vios_alt_disk:
        description: 'Perform an alternate disk copy on VIOS'
        targets: "(gdrh9v1,hdisk1,gdrh9v2,hdisk2)"
        action: alt_disk_copy
        vars: "{{ vars }}"
        #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 (gdrh9v1,hdisk1,gdrh9v2,hdisk2)",
   " Targets list: [('gdrh9v1', 'hdisk1', 'gdrh9v2', 'hdisk2')]",
   " Check the alternate disk hdisk1 on gdrh9v1",
   " Using hdisk1 as alternate disk on gdrh9v1",
   " Alternate disk copy on gdrh9v1",
   " Check the alternate disk hdisk2 on gdrh9v2",
   " Using hdisk2 as alternate disk on gdrh9v2",
   " Alternate disk copy on gdrh9v2",
   "VIOS Alternate disk operation status:",
   " gdrh9v1-gdrh9v2 : SUCCESS-ALTDC"
],
"status": {
   "gdrh9v1-gdrh9v2": "SUCCESS-ALTDC"
},
"targets": [
```

Task to execute: “aix_nim_vios_alt_disk”

The action (“alt_disk_copy” or “alt_disk_clean”) must be specified.

Action to execute: “alt_disk_copy”. The action (“alt_disk_copy” or “alt_disk_clean”) must be specified

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)

A time_limit could be specified.

The time is checked when starting to perform the “alt_disk_copy” for each tuple

```
   [
     "gdrh9v1",

     "hdisk1",
     "gdrh9v2",
     "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`

Perform the VIOS update

Create a playbook to perform the VIOS update and run it

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



$ cat playbook_aix_nim_updateios.yml
- name: "NIM ios update on AIX playbook"
  hosts: all
  gather_facts: no
  vars:
    log_file: "/tmp/ansible_updateios_debug.log"

  tasks:
    - name: "update ios"
      Aix_nim_updateios:
        targets: "(gdrh9v1,gdrh9v2)"
        #filesets: "none"
        installp_bundle: "__smit_bundle_8323538"
        lpp_source: "VIOS_225_30-lpp_source"
        accept_licenses: "yes"
        action: "install"
        preview: "no"
        #time_limit: "mm/dd/yyyy hh:mm"

      register: result

    - debug: var=result


$ ansible-playbook playbook_aix_nim_updateios.yml

    "msg": "NIM updateios operation completed successfully",
    "output": [
        "Updateios operation for (gdrh9v1,gdrh9v2)",
        "Targets list:[('gdrh9v1', 'gdrh9v2')]",
        "Any installp_bundle or filesets have been discarded",
        " Updating VIOS: gdrh9v1",
        " VIOS gdrh9v1 successfully updated",
        " Updating VIOS: gdrh9v2",
```

Task to execute: “aix_nim_updateios”.

List of dual VIOSes to update in a tuple format.
To perform an update on dual VIOSes, specify the dual VIOSes in the same tuple element as:
`"(gdrh9v1, gdrh9v2)`

`A single VIOS tuple is specified as: (gdrh10v1)`

Action to execute: type of the NIM operation to operate on the VIOSes.

The value is one of “install”, “commit”, “reject”, “cleanup”, “remove” values

```
        " VIOS gdrh9v2 successfully updated",
        "NIM updateios operation status:",

        " gdrh9v1-gdrh9v2 : SUCCESS-UPDT"
    ],
    "status": {
        "gdrh9v1-gdrh9v2": "SUCCESS-UPDT"
    },
    "targets": "(gdrh9v1,gdrh9v2)"
```

The optional **time_limit** attribute limits new update operation if the specified date/time is reached. Its format is month/day/year hour:minutes like the following: `mm/dd/yyyy hh:mm`

For more information on the **filesets, installp_bundle, lpp_source, accept_licenses** and **preview** attributes you can refers to the IBM® Knowledge Center documentation about the "Using the NIM updateios operation"
The **filesets** and **installp_bundle** attributes are mutually exclusive.
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` 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").
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".

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"
  hosts: all
  gather_facts: no
  vars:
    log_file: "/tmp/ansible_vios_update_debug.log"

  tasks:

    - name: "AIX VIOS HEALTH CHECK"
      aix_nim_vios_hc:
        description: 'Check the VIOS(es) can be updated'
        targets: "(gdrh10v1) (gdrh10v2) (gdrh9v1,gdrh9v2)"
        action: "health_check"
        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) (gdrh9v1,hdisk1,gdrh9v2,hdisk1)"
        action: "alt_disk_copy"
        vios_status: "{{ hc_result.status }}"
        vars: "{{ vars }}"

      register: altd_result
```

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

First, execute the "aix_nim_vios_hc" task.

Register for the status: **`hc_result`**

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.

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 update ios"
    aix_nim_updateios:

      targets: "(gdrh10v1) (gdrh10v2) (gdrh9v1,gdrh9v2)"

      #filesets: "none"
      installp_bundle: "__smit_bundle_8323538"
      lpp_source: "VIOS_225_30-lpp_source"
      accept_licenses: "yes"
      action: "install"
      preview: "no"
      #time_limit: "mm/dd/yyyy hh:mm"
      vios_status: "{{ altd_result.status }}"
      vars: "{{ vars }}"

    register: updt_result


  ## Uncommented this section to perfom a cleanup, but be careful
  ## because this would remove the alternate rootvg which is your
  ## backup.
  #- name: "AIX VIOS ALT DISK CLEANUP"
  #- aix_nim_vios_alt_disk:
  #    description: 'Remove the altinst_rootvg from the alternate disk'
  #    targets: "(gdrh10v1,hdisk1) (gdrh10v2,hdisk2) (gdrh9v1,hdisk1) (gdrh9v2,hdisk1)"
  #    action: alt_disk_clean
  #    vios_status: "{{ updt_result.status }}"
  #    vars: "{{ vars }}"

  #- register: altdisk_result

  - debug: var=hc_result.output
  - debug: var=altd_result.output
  - debug: var=updt_result.output
  #- debug: var=altdisk_result.output


$ ansible-playbook playbook_aix_nim_vios_update.yml
….

…
```

Potentially (but not recommended) run a “aix_nim_vios_alt_disk“ task with action “alt_disk_clean” to remove the alternate rootvg.

A good practice is to do this action latter when the update is verified.

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

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

In case of failure, it is one of: FAILURE-HC, FAILURE-ALTDCOPY1, FAILURE-ALTDCOPY2, FAILURE-UPDT1, FAILURE-UPDT2, FAILURE-ALTDCLEAN1, FAILURE-ALTDCLEAN2, FAILURE-NO-PREV-STATUS

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 cleanup can be done only if the copy was successful because otherwise there should not be an alternate disk copy to cleanup.

It is not recommended to perform the `alt_disk_clean` action in the same playbook as the updateios. You may want to be sure the updated VIOS is stable before removing your alternate disk copy made for backup. You can use a separate playbook to perform the cleanup after a validation period.