3-RedfishAnsibleUsingHpePlaybooks

March 15, 2021

1 HPE Redfish Ansible playbooks

Version 0.134

1.1 Introduction

The goal of this notebook is to present another approach for creating Redfish Ansible Playbooks compared to the approach used in the previous notebook.

The HPE Ansible Redfish Gitub repository proposes three sets of Ansible Redfish Playbooks examples using:

- the iLOrest tool
- the Ansible Redfish Galaxy collection
- Ansible modules derived from the examples in the `python-ilorest-library`.

In this Jupyter Notebook, you will study a Redfish Ansible Playbook based upon the set_uid_light.py example of the the HPE python-ilorest-library. This python example has been modified to become an Ansible module similarily (but slightly differently) to the one in the HPE GitHub site.

1.2 Environment preparation

The following cell sets environment variables and checks the connectivity toward the various BMCs used in this notebook.

[1]:
```
########## Environment preparation (Version: 0.132) #############

# Set Student ID number
stdid=825
Id=$(id --user --name)
NbId=3
InvFile=${NbId}/hosts

# location and ports variables
IloSyBasePort=46000
let iLO5SimulatorBasePort=$IloSyBasePort
let iLO5SimulatorPort=${iLO5SimulatorBasePort}+${stdid}

iLO5SimulatorIP=ilo5simulators
```

```
iLOSimulator=${iLO5SimulatorIP}:${iLO5SimulatorPort}
iLO5SimulatorURI=https://${iLOSimulator}

# Fake Credentials as we are testing against a BMC simulator
OvSsoToken="FakeOvSsoToken"

# Miscellaneous
WorkshopDir=$PWD
HpePythonRedfishVenv="${NbId}/HpePythonRedfishVenv"
export PYTHONPATH="${WorkshopDir}/${NbId}/library/"
w=$(basename $PWD)

alias ResetSimulators="../create-globalbmc.shc.x &>/dev/null; sleep 1"

# Verify we can reach the remote Bmcs on the right HTTPS ports.
for bmc in iLO5Simulator ; do
    ip="${bmc}IP" ; port=$(echo ${bmc}Port)
    nc -vz  $(eval echo "\$${ip}") $(eval echo "\$${port}") &>/dev/null &&
        echo -e "\n\tGood News: $bmc is reachable" \
        || echo "WARNING: Problem reaching $bmc"
done


# Create the Ansible inventory file
cat > ${InvFile} << __EOF__
[OneViewManagedBmcs]
${iLO5SimulatorIP} ansible_port=${iLO5SimulatorPort}

[OneViewManagedBmcs:vars]
ansible_python_interpreter=${WorkshopDir}/${HpePythonRedfishVenv}/bin/python3
ansible_search_path=${HpePythonRedfishVenv}

# Below is a fake session token as we are testing against an iLO 5 simulator.
# In real life, you should populate this variable with the token obtained
# lab 1 of this workshop.
token="${OvSsoToken}"
__EOF__
```

```
Good News: iLO5Simulator is reachable
```

1.3 Virtual Python environment creation

In order to completely isolate this notebook environment from other notebooks or student python environments, it is safer to create your dedicated Python virtual environment.

> NOTE: This `Venv` creation can take up to **2 minutes**. Just wait until message `Finished creating Venv` is displayed

[2]:
```
# Create Virtual Python environment (Venv) [Version 0.111]
[ -d ${HpePythonRedfishVenv} ] && rm -r ${HpePythonRedfishVenv} &>/dev/null
python3 -m venv ${HpePythonRedfishVenv}                         &>/dev/null
source ${HpePythonRedfishVenv}/bin/activate                     &>/dev/null
PS1="[PEXP\[\]ECT_PROMPT>"                                      # Avoid Venv␣
 ↪long prompt messing up outputs

# Populate Python Venv with the HPE python-ilorest-library
pip install wheel                                               &>/dev/null
pip install python-ilorest-library                              &>/dev/null

# Install latest Ansible in the Venv
pip install jmespath                                            &>/dev/null
pip install ansible                                             &>/dev/null

echo -e "\n\n\tFinished creating Venv\n\n"
```

```
(HpePythonRedfishVenv)

        Finished creating Venv
```

1.3.1 Restart iLO 5 simulator

If you need or desire to reset your iLO 5 simulator to restart this workshop from scratch or for other reasons, run the following cell at any time.

[ ]:
```
# iLO 5 Simulator restart
ResetSimulators

# Verify we can reach the remote Bmcs on the right HTTPS ports.
for bmc in iLO5Simulator ; do
    ip="${bmc}IP" ; port=$(echo ${bmc}Port)
    nc -vz  $(eval echo "\$${ip}") $(eval echo "\$${port}") &> /dev/null &&
        echo "$bmc is reachable" \
        || echo "WARNING: Problem reaching $bmc"
done
```

1.4 Get and set Redfish properties using HPE's python-ilorest-library

In the previous notebook, to discover the `Chassis` collection and the value of the indicator LEDs, you had to crawl the Redfish tree using Ansible and its built-in `uri` module.

When using a custom Ansible module derived from a `python-ilorest-library` example, the Redfish tree crawling is performed inside the Ansible ((pyhon) module), not in the `.yml` playbook.

HPE `python-ilorest-library` has been loaded in your Python `Venv` in the second cell of this notebook. It will allow the creation of a Redfish object with a (fake) authentication token. The

`python-ilorest-library` contains as well all the needed HTTP requests for getting, setting and performing Redfish actions.

You will find your custom Ansible modules and the get_resource_directory.py file in your `library` sub-folder. The get_resource_directory.py file contains a `get_resource_directory` function which is used to speed up the crawling of HPE iLO Redfish trees.

ILO Redfish implementations offer an HPE specific directory containing information (i.e. location) about all the data types present in the Redfish implementation. You can substantially improve the performance of your scripts using this directory to find resources in the Redfish tree.

1.4.1 Indicator LED

The next cell calls an Ansible Playbook that toggles the chassis UID/LED of a Synergy compute node, as well as its enlosure.

A convenient way to study this playbook is to open it in a different view in this pane. Right click on this Notebook tab and select `New View for Notebook` to open a new view:

Then, click on this file link.

If you need more space, type `Ctrl-B` (or `Command-B` on a Mac) to hide the left pane. You can make it reappear by hitting `Ctrl-B` again.

You can as well study the get_uid_light.py, the set_uid_light.py Ansible modules and the get_resource_directory Python file.

[3]:
```
# Modify IndicatorLED(s) using a custom Ansible Python module against an HPE␣
↪Synergy Gen10 ilo5
ansible-playbook -i ${InvFile} ${NbId}/SetIndicatorLEDUsingiLOrestLibrary.yml
```

```
PLAY [OneViewManagedBmcs] ******************************************************

TASK [1.0- Get and print IndicatorLED values] **********************************
ok: [ilo5simulators]

TASK [debug] *******************************************************************
ok: [ilo5simulators] => {
    "msg": {
        "ChassisLEDValues": {
            "/redfish/v1/Chassis/1/": "Lit",
            "/redfish/v1/Chassis/enclosurechassis/": "Lit"
        },
        "changed": false,
        "failed": false
    }
}

TASK [2.0- Toggle Indicator LED] ***********************************************
changed: [ilo5simulators]
```

```
TASK [3.0-Get and print IndicatorLED values again] ****************************
ok: [ilo5simulators]

TASK [debug] ******************************************************************
ok: [ilo5simulators] => {
    "msg": {
        "ChassisLEDValues": {
            "/redfish/v1/Chassis/1/": "Off",
            "/redfish/v1/Chassis/enclosurechassis/": "Off"
        },
        "changed": false,
        "failed": false
    }
}

PLAY RECAP ********************************************************************
ilo5simulators             : ok=5    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

1.4.2 Test the same playbook against a rack-mount server

The following cell switches your environment toward an **HPE DL360 Gen10** simulator and then runs again the Ansible Playbook. You will notice that the same playbook works for both a Synergy compute node and a rack-mount server although it is not enclosed in any frame or enclosure chassis.

> **NOTE**: In a real and physical environment, session token authentication against HPE iLO 5 rack mount servers is supported when managed by a OneView appliance. If not managed by OneView, you have to modify the Ansible Python modules (`[get,set]_uid_light.py`) as well as the playbook with `username` and `password` parameters.

[4]:
```
# location and ports variables
IloDlBasePort=45000
let iLO5SimulatorBasePort=$IloDlBasePort
let iLO5SimulatorPort=${iLO5SimulatorBasePort}+${stdid}

iLO5SimulatorIP=ilo5simulators
iLOSimulator=${iLO5SimulatorIP}:${iLO5SimulatorPort}
iLO5SimulatorURI=https://${iLOSimulator}

# Adapt the Ansible inventory file
cat > ${InvFile} << __EOF__
[OneViewManagedBmcs]
${iLO5SimulatorIP} ansible_port=${iLO5SimulatorPort}

[OneViewManagedBmcs:vars]
```

```
ansible_python_interpreter=${WorkshopDir}/${HpePythonRedfishVenv}/bin/python3
ansible_search_path=${HpePythonRedfishVenv}
# Below is a fake session token as we are testing against an iLO 5 simulator
token="${OvSsoToken}"
__EOF__


# Modify IndicatorLED(s) using a custom Ansible Python module against an HPE␣
 ↪DL360 Gen10 ilo5
ansible-playbook -i ${InvFile} ${NbId}/SetIndicatorLEDUsingiLOrestLibrary.yml
```

```
PLAY [OneViewManagedBmcs] ******************************************************

TASK [1.0- Get and print IndicatorLED values] **********************************
ok: [ilo5simulators]

TASK [debug] *******************************************************************
ok: [ilo5simulators] => {
    "msg": {
        "ChassisLEDValues": {
            "/redfish/v1/Chassis/1/": "Lit"
        },
        "changed": false,
        "failed": false
    }
}

TASK [2.0- Toggle Indicator LED] ***********************************************
changed: [ilo5simulators]

TASK [3.0-Get and print IndicatorLED values again] *****************************
ok: [ilo5simulators]

TASK [debug] *******************************************************************
ok: [ilo5simulators] => {
    "msg": {
        "ChassisLEDValues": {
            "/redfish/v1/Chassis/1/": "Off"
        },
        "changed": false,
        "failed": false
    }
}

PLAY RECAP *********************************************************************
ilo5simulators             : ok=5    changed=1    unreachable=0    failed=0
```

```
skipped=0    rescued=0    ignored=0
```

1.5 Summary

In this workshop, you used an Ansible Python module containing calls to the HPE `python-ilorest-library` to get and set the same Redfish resources as in the previous notebook. The advantage of this method is to move the complexity of crawling the Redfish tree from the playbook into the Python module. This allows you to use the power and flexibility of the Python language in terms of authentication, data manipulation and error handling. You validated as well the same code against two different types of server, proving its portability.

Read this article if you want to use HPE's `python-ilorest-library` in a Tower or AWX environment.

You are now ready to go through the next Notebook.