5-RedfishAnsibleUsingIlorest

September 7, 2021

1 HPE iLOrest Ansible playbooks

Version 0.17

1.1 Introduction

This notebook invokes a playbook that uses the Ansible Shell built-in module to launch the HPE iLOrest Command Line Interface (CLI) to get and set the chassis Unit Identification light (UID) status of two HPE iLO 5 based servers: an HPE Synergy compute and a rack server. The content of the Ansible Playbook used in this notebook is derived from the examples present in the ansible-ilorest-role HPE GitHub.

1.1.1 Cache considerations

In order to increase performance, the iLOrest tool uses a cache directory to store various data specific to managed servers. This cache directory is created during the authentication process in a default location unless a specific location is specified on the command line.

As Ansible can run a given task in parrallel on several hosts, you must specify a dedicated cache directory for each host. If you don't specify a dedicated cache for each host, the default cache directory will be overwritten by each new connection and, only the last host will be able to complete the task with a valid cache.

To keep the (`host, cache`) tuple coherent, your cache directory name will use a combination of the IP address of the global iLO5 simulator server and the specific TCP port number of your private simulators (`SimulIP:SimulPort`).

A generic Ansible `cache_dir` variable is created in the `hosts` inventory file populated on the fly in the Environment Preparation cell of this notebook. This variable is used in the Ansible Playbook.

1.1.2 Authentication considerations

As iLOrest does not support the OneView Single Sign On (SSO) we are not able to use the token obtained in the first notebook of this workshop.

Moreover, the `ilorest` login process is commented out, in the Ansible Playbook because the iLO 5 simulators do not support any authentication process. Skipping the iLOrest authentication process also means that the cache directory is not populated. To overcome this, your environment holds compressed cache files (one per simulator) that are uncompressed in the right location.

The `ilorest logout` process is also commented out in this Ansible Playbook, in case you want to run multiple times the playbook. The reason is because an `ilorest logout` command triggers a complete deletion of the cache directory, which would prevent you to re-run the playbook again without playing the Environment Preparation cell.

1.2 Environment preparation

The following cell sets environment variables, uncompresses iLOrest cache files, creates an Ansible inventory file and checks the connectivity with the iLO 5 simulators.

[1]:
```
########## Environment preparation (Version: 0.135) ############

# Set Student ID number and more
export stdid=776
Id=$(id --user --name)
NbId=5
InvFile=${NbId}/hosts


# locations, IP and port variables
iLO5DlBasePort=45000
iLO5SyBasePort=46000

iLO5SySimulatorIP=ilo5simulators
iLO5DlSimulatorIP=ilo5simulators

iLO5DlSimulatorPort=$((iLO5DlBasePort+stdid))
iLO5SySimulatorPort=$((iLO5SyBasePort+stdid))

iLO5DlSimulator=${iLO5DlSimulatorIP}:${iLO5DlSimulatorPort}
iLO5SySimulator=${iLO5SySimulatorIP}:${iLO5SySimulatorPort}

iLO5DlSimulatorURI=https://${iLO5DlSimulator}
iLO5SySimulatorURI=https://${iLO5SySimulator}

iLO5SyMockup="iLO5Sy480g10.tgz"
iLO5DlMockup="iLO5Dl360g10.tgz"

# Miscellaneous
alias ResetSimulators="../create-globalbmc.shc.x &>/dev/null; sleep 1"
WorkshopDir=$PWD
w=$(basename $PWD)

# iLOrest cache preparation
CacheLocation="${WorkshopDir}/${NbId}"
echo
for s in iLO5Dl iLO5Sy ; do
   BasePort="${s}BasePort" ; SimulIP="${s}SimulatorIP";␣
 ↪SimulPort="${s}SimulatorPort"
   exp="s?$(eval echo \$$BasePort)?$(eval echo \$$SimulPort)?"
```

```
  CacheDir="${CacheLocation}/$(eval echo \$$SimulIP):$(eval echo \$$SimulPort)"
  Mockup=$(eval echo "\$${s}Mockup")
  [ -d ${CacheDir} ] && rm ${CacheDir}                           &>/dev/null
  mkdir -p ${CacheDir}/cache/                                    &>/dev/null
  tar -C ${CacheDir}/cache -x -f ${NbId}/${Mockup}
  sed -i -e "$exp" ${CacheDir}/cache/*
  echo -e "\tPopulated $s cache directory under ${NbId}/$(basename␣
↪${CacheDir})"
done

# Create the Ansible inventory file with a rack-mount and an HPE Synergy␣
↪compute node
cat > ${InvFile} << __EOF__
[OneViewManagedBmcs]
iLO5DlSimulator ansible_host=${iLO5DlSimulatorIP}␣
↪ansible_port=${iLO5DlSimulatorPort}
iLO5SySimulator ansible_host=${iLO5SySimulatorIP}␣
↪ansible_port=${iLO5SySimulatorPort}

[OneViewManagedBmcs:vars]
ansible_python_interpreter=/usr/bin/python3
cache_dir="${WorkshopDir}/${NbId}/{{ ansible_host }}:{{ ansible_port }}"
__EOF__

echo -e "\n\tCreated Ansible inventory file at ${NbId}/hosts"


# Verify we can reach the remote Bmcs on the right HTTPS ports.
echo
for bmc in iLO5SySimulator iLO5DlSimulator ; do
    ip="${bmc}IP" ; port=$(echo ${bmc}Port)
    nc -vz  $(eval echo "\$${ip}") $(eval echo "\$${port}") &>/dev/null &&
        echo -e "\tGood News: $bmc is reachable" \
        || echo "WARNING: Problem reaching $bmc"
done

echo
# Retrieve iLO firmware versions from ServiceRoot (no credentials needed)
for bmc in iLO5SySimulator iLO5DlSimulator; do
    ip="${bmc}IP" ; port="${bmc}Port"
    echo -n -e "\t${bmc} firmware version: "
     curl --silent --insecure -X GET https://$(eval echo \$${ip}):$(eval echo␣
↪\$${port})/redfish/v1 | \
         jq  '[.Oem.Hpe.Manager[]] | .[] | .ManagerFirmwareVersion'
done
```

```
Populated iLO5Dl cache directory under 5/ilo5simulators:45776
Populated iLO5Sy cache directory under 5/ilo5simulators:46776

Created Ansible inventory file at 5/hosts

Good News: iLO5SySimulator is reachable
Good News: iLO5DlSimulator is reachable

iLO5SySimulator firmware version: "2.47"
iLO5DlSimulator firmware version: "2.44"
```

1.2.1 Restart iLO 5 simulators

If you need or desire to reset your iLO 5 simulators 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 iLO5SySimulator iLO5DlSimulator ; 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
```

1.3 Get and set Redfish properties using HPE iLOrest

In previous notebooks, you had to crawl the Redfish tree to get and set Redfish properties, either in the Ansible Playbook or in an Ansible Python module. In this notebook, the crawl process is performed by iLOrest.

IlOrest is invoked by the Ansible `Shell` built-in module. Each invocation contains the data type associated to the property you want to get or set (`--select Chassis.`). As a programmer, you can use either the DMTF standard schemas documents or the HPE API Reference document to identify the correct data type.

The data type is used by iLOrest to get or set the requested properties without assumming their location.

1.3.1 Indicator LED

The next cell calls an Ansible Playbook that toggles the chassis UIDs of an HPE Synergy compute node and a rack-mount server (chassis and enclosure if any).

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.

[2]:
```
# Modify IndicatorLED(s) using the `shell` Ansible module against
# an HPE Synergy Gen10 iLO5 and a rack-mount iLO 5 based server
ansible-playbook -i ${InvFile} ${NbId}/SetIndicatorLEDUsingIlorest.yml
```

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

TASK [get Indicator LEDs status] ************************************************
changed: [iLO5DlSimulator]
changed: [iLO5SySimulator]

TASK [debug] ********************************************************************
ok: [iLO5DlSimulator] => {
    "msg": [
        "IndicatorLED=Lit"
    ]
}
ok: [iLO5SySimulator] => {
    "msg": [
        "",
        "IndicatorLED=Lit",
        "",
        "IndicatorLED=Lit"
    ]
}

TASK [Toggle Indicator LEDs] ****************************************************
changed: [iLO5DlSimulator]
changed: [iLO5SySimulator]

TASK [debug] ********************************************************************
ok: [iLO5DlSimulator] => {
    "msg": []
}
ok: [iLO5SySimulator] => {
    "msg": []
}

TASK [get Indicator LEDs status] ************************************************
changed: [iLO5DlSimulator]
changed: [iLO5SySimulator]

TASK [debug] ********************************************************************
ok: [iLO5DlSimulator] => {
    "msg": [
        "IndicatorLED=Off"
    ]
}
ok: [iLO5SySimulator] => {
    "msg": [
        "",
        "IndicatorLED=Off",
        "",
        "IndicatorLED=Off"
    ]
}

TASK [Refresh cache] ***********************************************************
changed: [iLO5DlSimulator]
changed: [iLO5SySimulator]

PLAY RECAP *********************************************************************
iLO5DlSimulator            : ok=7    changed=4    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
iLO5SySimulator            : ok=7    changed=4    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

1.4 Summary

In this workshop, you used the built-in Ansible `Shell` module calling the HPE ilOrest tool to get and set the same Redfish resources as in the previous notebooks. The advantage of this method is to move the complexity of crawling the Redfish tree from the playbook or an home made Python module, into iLOrest.

In addition, you used the powerful iLOrest cache feature to perform actions on multiple target compute nodes.

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 need more details about the iLOrest cache directory.

If you are finished, you can to the conclusion.