Welcome to the Synergy Roundtable

created by Dirk Derichsweiler and Stephan Koch
do not hesitate to contact us: derdirk@hpe.com stephan.koch@hpe.com

jupyter Notebook can be found:
https://github.com/dderichswei/synergy/blob/master/Synergy%20Roundtable.ipynb (https://github.com/dderichswei/synergy/blob/master/Synergy%20Roundtable.ipynb)

requirements

Python + hpOneView Python Library (see: https://github.com/HewlettPackard/python-hpOneView/wiki/HPE-OneView-Python-Windows-Setup-Guide (https://github.com/HewlettPackard/python-hpOneView/wiki/HPE-OneView-Python-Windows-Setup-Guide))

pip3 install redfish pip install python-ilorest-library

additional information

On your HPE OneView appliance, or online
https://10.0.20.50/api-docs/current/ (https://10.0.20.50/api-docs/current/)
http://www.hpe.com/info/oneview/docs (http://www.hpe.com/info/oneview/docs)
https://developer.hpe.com/ (https://developer.hpe.com/)

python specific

https://hewlettpackard.github.io/python-hpOneView/index.html (https://hewlettpackard.github.io/python-hpOneView/index.html)
https://github.com/HewlettPackard/python-hpOneView (https://github.com/HewlettPackard/python-hpOneView)

Login

import the python OneView library

In [1]:

```
from hpOneView.oneview_client import OneViewClient
from pprint import pprint
```

configure your environment:

In [2]:

```
config = {
    "api_version": "1200",
    "ip": "10.0.20.50",
    "credentials": {
        "userName": "XXXX",
        "authLoginDomain": "local",
        "password": "XXXXXX"
    }
}
```

login:

create new object oneview_client with the config (see above)

In [3]:

```
oneview_client = OneViewClient(config)
```

ADVANCED: which functions are available?

In [ ]:

```
dir(oneview_client)
#dir(oneview_client.storage_volume_templates) # drill down
dir(oneview_client.connection)
```

show existing networks

In [4]:

```
roundtable_networks=oneview_client.ethernet_networks.get_all()
# print(roundtable_networks)

print("The following networks exists:")
for net in roundtable_networks:
    # print(net['name'] + " VLAN ID:" + str(net['vlanId']))
    print(net['name'] + "    URI: "  + net['uri'] )
```

```
The following networks exists:
Roundtable - Test Ethernet Network    URI: /rest/ethernet-networks/00b85db
0-d1a3-4cfc-8f8e-55dcee071fcc
Storage iSCSI 0042    URI: /rest/ethernet-networks/07223e04-9793-414f-bfdc
-066405c81f58
Production 0103    URI: /rest/ethernet-networks/0c6bfc24-9cea-4d25-b6a7-97
90fbfecd93
acs Test VLAN (Multicast, MLAG)    URI: /rest/ethernet-networks/10cacc73-3
0c9-4f3a-8cff-8077fdd97fa3
Production 0108    URI: /rest/ethernet-networks/12492fe9-215a-4e5f-92c8-0b
121acafb2b
acs Test VLAN2    URI: /rest/ethernet-networks/16450273-cac1-41a3-a8f3-b9f
2a42fd824
CTC NSX 03 0804    URI: /rest/ethernet-networks/1d57ab1a-606a-409b-bdf2-e4
2e7e37a2fd
Stratoscale_1 0056    URI: /rest/ethernet-networks/2330f794-f167-47eb-a178
-b73dc9293ab7
Stratoscale Guest 0054    URI: /rest/ethernet-networks/2ccff264-df23-4499-
93db-f93162080d53
CTC NSX 01 0802    URI: /rest/ethernet-networks/2f036536-23b0-4d7c-b94e-5b
230c5179aa
Storage Management 0040    URI: /rest/ethernet-networks/30c981bb-894f-460f
-9060-a994e97a3173
Image Streamer 0019    URI: /rest/ethernet-networks/4022370f-1fd3-4b70-b72
4-87a76f4e760f
CTC VCF Management 0070    URI: /rest/ethernet-networks/41790d66-3817-4a45
-b3c1-580d6cc5c4d4
DevOps FrontEnd 0047    URI: /rest/ethernet-networks/59229bbb-01b5-4611-80
38-061a281c5428
Production 0100    URI: /rest/ethernet-networks/5a4be04a-b38d-4c22-9bb8-a3
98ee8d9db4
CTC VCF 0066    URI: /rest/ethernet-networks/6f4e71d7-fdae-4b80-b5d1-408a3
cf49478
Production 0105    URI: /rest/ethernet-networks/74b54ec9-e36d-4c8c-bb0a-db
a77b98fbc5
Production 0107    URI: /rest/ethernet-networks/775fcd17-a7ea-4434-af40-2a
3d0a7f4460
Stratoscale_2 0057    URI: /rest/ethernet-networks/7765ff9a-e5a1-4682-a268
-c98b73c19127
CTC NSX 02 0803    URI: /rest/ethernet-networks/77e5df8e-7d6b-42f7-ae17-4f
9532038b9a
CTC NSX 04 0805    URI: /rest/ethernet-networks/7ab16bc2-1d35-4503-86e6-75
660fd599be
Production 0102    URI: /rest/ethernet-networks/863d2e02-9206-4ee8-8570-31
d8283e6946
SDS    URI: /rest/ethernet-networks/8e87d82f-a315-46ec-aa5d-3c30c5a7ed0f
CTC-Demo 030    URI: /rest/ethernet-networks/93a8bd1c-d714-4a72-af34-8aae8
51904f8
CTC VCF 0069    URI: /rest/ethernet-networks/97a3ef7f-a5dd-42a7-83c0-99c6c
6c89d5e
Hypervisor Tunnel    URI: /rest/ethernet-networks/9e921fda-4f38-42db-ba01-
1726f7df9b01
Stratoscale Edge 0055    URI: /rest/ethernet-networks/a1a4e931-1207-4bd2-8
511-3b4bad67e37e
Production 0109    URI: /rest/ethernet-networks/a226be73-529e-428e-abae-a7
b4f975cf3f
Stratoscale Data 0052    URI: /rest/ethernet-networks/a69559bd-bd4c-4d2c-b
440-8a14ae46ae23
CTC VCF 0064    URI: /rest/ethernet-networks/a7f0285c-e7dc-40c7-8758-7c474
ebb06ba
Production 0106    URI: /rest/ethernet-networks/ac9040cf-2187-4584-8fad-34
```

```
1e189209aa
Production 0101    URI: /rest/ethernet-networks/ad16c3dd-c0eb-4167-b1c1-5c
63580ae1a9
CTC VCF 0068    URI: /rest/ethernet-networks/af062166-62ac-4739-8114-92563
5922904
Stratoscal Control 0053    URI: /rest/ethernet-networks/afe011a5-43f0-4988
-ba96-8ff5863a3d1c
CTC VCF 0065    URI: /rest/ethernet-networks/b7a05515-7e4f-486c-9389-5b0dc
e7d481d
vMotion 0800    URI: /rest/ethernet-networks/b7ea506f-e31c-4309-9e20-7a00e
076489a
0033_Cloud    URI: /rest/ethernet-networks/c28d4471-836e-4dc1-b3c5-7bf90a2
9ee24
DevOps BackEnd 0048    URI: /rest/ethernet-networks/c888078f-8688-44c4-ab2
e-98955bbb0655
Stratoscale_4 0059    URI: /rest/ethernet-networks/cb6aebec-70d4-4047-a7cc
-2a186bcfd362
Stratoscale Access 0051    URI: /rest/ethernet-networks/cba84168-4b7f-491d
-975e-cffe4ba5abd1
Stratoscale_3 0058    URI: /rest/ethernet-networks/cd730342-ee8b-4623-ab25
-e6fed1198e1f
CTC VCF 0067    URI: /rest/ethernet-networks/ce6f61f8-5281-4d40-a8e6-33b9f
4ebc017
Storage Replication 0041    URI: /rest/ethernet-networks/d0b050ee-7802-43a
4-977d-46d4c1d13644
Production 0104    URI: /rest/ethernet-networks/d39fa870-9342-4206-bba6-36
d7663d959f
CTC VCF 0063    URI: /rest/ethernet-networks/d3c3728e-c83f-41f6-b207-5d998
9f47b56
Management 0020    URI: /rest/ethernet-networks/d6b3e24a-c948-4d76-93d8-c7
855bce4c81
UNTAGGEDNW    URI: /rest/ethernet-networks/da967af3-294e-4ad5-9a3a-b45a471
55806
CTC VCF vSAN 0801    URI: /rest/ethernet-networks/e71c4ee9-73d6-46c5-948b-
3a6371062b60
CTC VCF 0061    URI: /rest/ethernet-networks/eabdc806-ff6f-4776-b11e-4e435
44baa7f
VDI    URI: /rest/ethernet-networks/ed76b66c-0dad-4d59-bfa7-1b496385732b
CTC VCF 0062    URI: /rest/ethernet-networks/f24f6ac5-2024-497e-9a1d-19133
1a050f4
```

In [ ]:

```
```

create network

In [6]:

```
options = {
    "name": "Roundtable - Test Ethernet Network",
    "vlanId": 200,
    "ethernetNetworkType": "Tagged",
    "purpose": "General",
    "smartLink": False,
    "privateNetwork": False,
    "connectionTemplateUri": None
}

ethernet_network = oneview_client.ethernet_networks.create(options)
if ethernet_network:
    print("Created ethernet-network: '{name}'.\n   uri = '{uri}'" .format(**ethernet_ne
twork.data))



#print("Created ethernet-network " + ethernet_network['name'] + "   URI: " + ethernet_n
etwork['uri'] + " successfully.")
```

```
Created ethernet-network: 'Roundtable - Test Ethernet Network'.
   uri = '/rest/ethernet-networks/b1076d6d-c06d-46c9-b71f-183dc08d6ed9'
```

delete network

In [ ]:

```
ethernet_network = oneview_client.ethernet_networks.get_by_name("Roundtable - Test Ethe
rnet Network")
if ethernet_network:
    print("Delete ethernet-network by name: '{name}'.\n   uri = '{uri}'" .format(**ethe
rnet_network.data))
    ethernet_network.delete()


#oneview_client.ethernet_networks.delete(ethernet_network)
```

create bulk network

How to set up multiple networks at once.

In [ ]:

```
options_bulk = {
    "vlanIdRange": "1-5,7,100-109,200",
    "purpose": "General",
    "namePrefix": "Roundtable-Ethernet",
    "smartLink": False,
    "privateNetwork": False,
    "bandwidth": {
        "maximumBandwidth": 10000,
        "typicalBandwidth": 2000
    }
}


ethernet_nets_bulk = oneview_client.ethernet_networks.create_bulk(options_bulk)
for net in ethernet_nets_bulk:
  print("created network: " + net['name'] + " URI: " + net['uri'])
```

delete bulk network

as it´s not needed for the demo

In [ ]:

```
for net in ethernet_nets_bulk:
    oneview_client.ethernet_networks.get_by_uri(net['uri']).delete()
    print("delete network: " + net['name'])


#for net in ethernet_nets_bulk:
#    oneview_client.ethernet_networks.delete(net)
#    print("delete network: " + net['name'])
```

show configured/existing storage (systems and pools)

In [7]:

```
storage_system = oneview_client.storage_systems.get_all()
storage_pools = oneview_client.storage_pools.get_all()

# storage_pool_name = 'FC_r1'
# storage_pools = oneview_client.storage_pools.get_by('name', storage_pool_name)[0]

print("Storage Systems:")
for stor in storage_system:
  print(stor['displayName'] + "    URI: " + stor['uri'])

print("\nStorage Pools:")
for storpool in storage_pools:
  print(storpool['name'] + "    URI: " + stor['uri'])
```

```
Storage Systems:
3PAR8200    URI: /rest/storage-systems/CZ38277F39
nimbleaf-grp    URI: /rest/storage-systems/d05e1db0-b118-41f9-a645-ab0a00d
26e01
primera630    URI: /rest/storage-systems/1c3cfa75-7dd6-4b93-84fe-ab5d008e1
472
Primera650    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1
194

Storage Pools:
FC_r1    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
FC_r5    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
FC_r6    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
fs_cpg    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
default    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
SSD_r6    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
SSD_r6    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
SSD_r6_6    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f119
4
FC_test    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f1194
FC_test2    URI: /rest/storage-systems/baa16c9f-2d85-4241-85a1-ab5d008f119
4
```

In [ ]:

```
print (storpool)
```

show volume templates

In [8]:

```
storage_volume = oneview_client.storage_volume_templates.get_all(filter="\"isRoot='Fals
e'\"")
for storvol in storage_volume:
    print(storvol['name'])
```

```
VDI-ESXi-Boot
TEST-Terraform
Roundtable Volume Template2
```

create volume template

In [ ]:

```
storage_pool_name = 'FC_r5'

# Get the storage pool by name to use in options
storage_pool = oneview_client.storage_pools.get_by('name', storage_pool_name)[0]

# Gets the first Root Storage Volume Template available to use in options
root_template = oneview_client.storage_volume_templates.get_all(filter="\"isRoot='True'
\"")[0]
print(root_template['uri'])

options = {
    "name": "Roundtable Volume Template",
    "description": "",
    "rootTemplateUri": root_template['uri'],
    "properties": {
        "name": {
            "meta": {
                "locked": False
            },
            "type": "string",
            "title": "Volume name",
            "required": True,
            "maxLength": 100,
            "minLength": 1,
            "description": "A volume name between 1 and 100 characters"
        },
        "size": {
            "meta": {
                "locked": False,
                "semanticType": "capacity"
            },
            "type": "integer",
            "title": "Capacity",
            "default": 1073741824,
            "maximum": 17592186044416,
            "minimum": 268435456,
            "required": True,
            "description": "The capacity of the volume in bytes"
        },
        "description": {
            "meta": {
                "locked": False
            },
            "type": "string",
            "title": "Description",
            "default": "",
            "maxLength": 2000,
            "minLength": 0,
            "description": "A description for the volume"
        },
        "isShareable": {
            "meta": {
                "locked": False
            },
            "type": "boolean",
            "title": "Is Shareable",
            "default": False,
            "description": "The shareability of the volume"
        },
```

```
        "storagePool": {
            "meta": {
                "locked": False,
                "createOnly": True,
                "semanticType": "device-storage-pool"
            },
            "type": "string",
            "title": "Storage Pool",
            "format": "x-uri-reference",
            "required": True,
            "description": "A common provisioning group URI reference",
            "default": storage_pool['uri']
        },
        "snapshotPool": {
            "meta": {
                "locked": True,
                "semanticType": "device-snapshot-storage-pool"
            },
            "type": "string",
            "title": "Snapshot Pool",
            "format": "x-uri-reference",
            "default": storage_pool['uri'],
            "description": "A URI reference to the common provisioning group used to cr
eate snapshots"
        },
        "provisioningType": {
            "enum": [
                "Thin",
                "Full",
                "Thin Deduplication"
            ],
            "meta": {
                "locked": True,
                "createOnly": True
            },
            "type": "string",
            "title": "Provisioning Type",
            "default": "Thin",
            "description": "The provisioning type for the volume"
        }
    }
}

volume_template = oneview_client.storage_volume_templates.create(options)
print("Storage volume created...")
```

delete storage template

""" don´t use it, for the demo.. """

In [ ]:

```
storage_volume = oneview_client.storage_volume_templates.get_all(filter="\"name='Roundt
able Volume Template'\"")
for storvol in storage_volume:
    oneview_client.storage_volume_templates.delete(storvol)
```

show Enclosure / Server Hardware / Bay

In [ ]:

```
print ("show enclosure group (enclosure_group_uri)")
enclosure = oneview_client.enclosure_groups.get_all()
for enc in enclosure:
    print(enc['name'] + " " + enc['uri'])

print("\nshow server hardware (server_hardware_type_uri)")
server_hardware_types = oneview_client.server_hardware_types.get_all(sort='name:descend
ing')
# print(server_hardware_types)
for serverhw in server_hardware_types:
    # print(' %s ' % serverhw['model'])
    print(serverhw['model'] + " URI: " + serverhw['uri'] )

print("\nEnclosure/Bay (server_hardware_uri)")
server_hardware = oneview_client.server_hardware.get_all()
for server in server_hardware:
    print(server['name'] + " " + server['model'] + " " + server['uri'])
```

show Serverprofile

In [9]:

```
print("show server profile templates:")
all_srv_templates = oneview_client.server_profile_templates.get_all()
for srv_tmp in all_srv_templates:
    print(srv_tmp['name'] + "   URI:   " + srv_tmp['uri'])

print("\nshow server profiles:")
all_profiles = oneview_client.server_profiles.get_all()
for profile in all_profiles:
    print(profile['name']+ "   URI:  " + srv_tmp['uri'])

# my_profile = oneview_client.server_profiles.get_by_name("Roundtable - API Demo Templa
te (DirkD)")
```

```
show server profile templates:
CTC VCF Management   URI:   /rest/server-profile-templates/003b73aa-e5d4-4
a97-ad38-4613600f9e85
CTC ESXi 6.7 U3 SY480G10 incl. SAN SPT BfS   URI:   /rest/server-profile-t
emplates/33bb8750-b682-4786-89b7-223c8029aad2
CTC ESXi 6.7 U3 SY480G10 I3S SAN GrowCluster SPT   URI:   /rest/server-pro
file-templates/4aaea88a-198e-487e-806c-037d4ab2e5f1
CTC Windows Server 2019 BfS   URI:   /rest/server-profile-templates/5e5c4f
97-9db7-4439-9e59-7fcf6b9f71a7
CTC_RHEL7_Docker_SPT   URI:   /rest/server-profile-templates/84876834-19ca
-4f2b-8be6-274f15011346
ANSIBLE_OS_Deploy_via_iLO   URI:   /rest/server-profile-templates/950ac7c2
-30f9-47f4-927d-480d255f0361
CTC ESXi 6.7 U3 SY480G10 incl. SAN SPT   URI:   /rest/server-profile-templ
ates/b9bfd793-f3fd-4242-9ee3-82802f91d76f
acs_Test   URI:   /rest/server-profile-templates/bef761b7-ad01-42b7-a256-7
5849ad3ae0d
CTC_PP_ESX   URI:   /rest/server-profile-templates/c4f511e7-3700-417f-989f
-b67c13ef5b7c
TF_Grid   URI:   /rest/server-profile-templates/c9062c0c-3321-458c-9d78-e4
ed6e3ef69f
ESX i TPL SAN   URI:   /rest/server-profile-templates/efbe51ef-2e7d-4f44-a
7cc-7843328582fd
CTC VCF Workload Domain   URI:   /rest/server-profile-templates/f09cdfaa-9
85f-4775-ac2b-c5e2572c6c47
Demo-ServerProfileTemplate   URI:   /rest/server-profile-templates/f927b64
d-b43a-4a6e-a216-412874bc0900

show server profiles:
H5HE05B11-VDI   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a2
16-412874bc0900
acs_WinServerTest   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6
e-a216-412874bc0900
SUO04CTCVCFESX01   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e
-a216-412874bc0900
SP-created by Ansible2   URI:  /rest/server-profile-templates/f927b64d-b43
a-4a6e-a216-412874bc0900
H5HE05B6-VDI   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a21
6-412874bc0900
SUO04CTCVCFESX03   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e
-a216-412874bc0900
SUO04CTCW20215 BfS   URI:  /rest/server-profile-templates/f927b64d-b43a-4a
6e-a216-412874bc0900
SUO04CTCE20203   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a
216-412874bc0900
SUO04CTCE20201   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a
216-412874bc0900
Synergy Grow Cluster 02   URI:  /rest/server-profile-templates/f927b64d-b4
3a-4a6e-a216-412874bc0900
CTC-PP-ESX1   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a216
-412874bc0900
SUO04CTCW20216 BfS   URI:  /rest/server-profile-templates/f927b64d-b43a-4a
6e-a216-412874bc0900
SUO04CTCVCFESX02   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e
-a216-412874bc0900
SUO04CTCE20204   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a
216-412874bc0900
CTC-PP-ESX2   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a216
-412874bc0900
SUO04CTCVCFESX04   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e
-a216-412874bc0900
```

```
CTC_API_RHEL7_EFI_GoldenImage   URI:  /rest/server-profile-templates/f927b
64d-b43a-4a6e-a216-412874bc0900
SUO04CTCE20202   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a
216-412874bc0900
API-ESX   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a216-412
874bc0900
TF_node1   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a216-41
2874bc0900
CTC-Storage-WIN   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-
a216-412874bc0900
H5HE01B5-SRT   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-a21
6-412874bc0900
SUO04CTCW20217 BfS   URI:  /rest/server-profile-templates/f927b64d-b43a-4a
6e-a216-412874bc0900
SUO04CTCESX9x30   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-
a216-412874bc0900
SUO04CTCESX9x33   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-
a216-412874bc0900
SUO04CTCESX9x31   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-
a216-412874bc0900
CTC ESXi 6.7 U3 Reference SRV GoldImage   URI:  /rest/server-profile-templ
ates/f927b64d-b43a-4a6e-a216-412874bc0900
SUO04CTCESX9x32   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e-
a216-412874bc0900
SUO04CTCESX12001   URI:  /rest/server-profile-templates/f927b64d-b43a-4a6e
-a216-412874bc0900
```

create server profile based on an Server Profile Template (takes 2-3 minutes)

In [10]:

```
#server = server_hardwares.get_by_name(server_name)
#server_hardwares = oneview_client.server_hardware

powerOn = {
    "powerState": "On",
    "powerControl": "MomentaryPress"
}

powerOff = {
    "powerState": "Off",
    "powerControl": "PressAndHold"
}

template_name = "ANSIBLE_OS_Deploy_via_iLO"
profile_name = "Roundtable - API Demo Server"


template = oneview_client.server_profile_templates.get_by_name(template_name)

server = oneview_client.server_hardware.get_by_name( "CTC H5 HE11, bay 2")
server_hardware_uri=server.data['uri']


basic_profile_options = template.get_new_profile()
pprint(basic_profile_options)

basic_profile_options['name'] = profile_name
basic_profile_options['serverHardwareUri'] = server_hardware_uri


#server_template_uri = server_template_data.data['uri']

#print(server_template_uri)


server_power = server.update_power_state(powerOff) # turn off server


#try:
print ("create server profile")


#basic_profile_options = dict(
#        name=profile_name,
#        serverProfileTemplateUri=server_template_data.data["uri"],
#        serverHardwareTypeUri=hardware_type.data["uri"],
#        enclosureGroupUri=enclosure_group.data["uri"]
#    )


profile = oneview_client.server_profiles.create(basic_profile_options)


#except:
#    print(server_name + " Server already exists")
```

```
#server_power = server.update_power_state(powerOn) # turn off server
#print ("server powered on .........")
```

```
{'affinity': 'Bay',
 'associatedServer': None,
 'bios': {'consistencyState': 'Unknown',
          'manageBios': False,
          'overriddenSettings': [],
          'reapplyState': 'NotApplying'},
 'boot': {'manageBoot': True, 'order': ['SD']},
 'bootMode': {'manageMode': True,
              'mode': 'UEFIOptimized',
              'pxeBootPolicy': 'Auto',
              'secureBoot': 'Unmanaged'},
 'category': 'server-profiles',
 'connectionSettings': {'connections': [{'allocatedMbps': 0,
                                         'allocatedVFs': None,
                                         'boot': {'priority': 'NotBootabl
e'},
                                         'functionType': 'Ethernet',
                                         'id': 1,
                                         'interconnectPort': 0,
                                         'interconnectUri': None,
                                         'ipv4': None,
                                         'isolatedTrunk': False,
                                         'lagName': None,
                                         'mac': None,
                                         'macType': None,
                                         'managed': True,
                                         'maximumMbps': 0,
                                         'name': 'Net33',
                                         'networkName': None,
                                         'networkUri': '/rest/ethernet-net
works/c28d4471-836e-4dc1-b3c5-7bf90a29ee24',
                                         'portId': 'Mezz 3:1-a',
                                         'privateVlanPortType': 'None',
                                         'requestedMbps': '10000',
                                         'requestedVFs': '0',
                                         'state': 'Unknown',
                                         'status': 'Unknown',
                                         'wwnn': None,
                                         'wwpn': None,
                                         'wwpnType': None},
                                        {'allocatedMbps': 0,
                                         'allocatedVFs': None,
                                         'boot': {'bootVolumeSource': 'Man
agedVolume',
                                                  'priority': 'Primary'},
                                         'functionType': 'FibreChannel',
                                         'id': 2,
                                         'interconnectPort': 0,
                                         'interconnectUri': None,
                                         'ipv4': None,
                                         'isolatedTrunk': False,
                                         'lagName': None,
                                         'mac': None,
                                         'macType': None,
                                         'managed': True,
                                         'maximumMbps': 0,
                                         'name': '',
                                         'networkName': None,
                                         'networkUri': '/rest/fc-networks/
10f72439-00f4-45b2-81b5-ac917f80f7fa',
                                         'portId': 'Mezz 3:1-b',
```

```
                                          'privateVlanPortType': 'None',
                                          'requestedMbps': '8000',
                                          'requestedVFs': None,
                                          'state': 'Unknown',
                                          'status': 'Unknown',
                                          'wwnn': None,
                                          'wwpn': None,
                                          'wwpnType': None},
                                         {'allocatedMbps': 0,
                                          'allocatedVFs': None,
                                          'boot': {'bootVolumeSource': 'Man
agedVolume',
                                                   'priority': 'Secondar
y'},
                                          'functionType': 'FibreChannel',
                                          'id': 3,
                                          'interconnectPort': 0,
                                          'interconnectUri': None,
                                          'ipv4': None,
                                          'isolatedTrunk': False,
                                          'lagName': None,
                                          'mac': None,
                                          'macType': None,
                                          'managed': True,
                                          'maximumMbps': 0,
                                          'name': '',
                                          'networkName': None,
                                          'networkUri': '/rest/fc-networks/
6da9c8be-081f-440b-b60a-c00bad899543',
                                          'portId': 'Mezz 3:2-b',
                                          'privateVlanPortType': 'None',
                                          'requestedMbps': '8000',
                                          'requestedVFs': None,
                                          'state': 'Unknown',
                                          'status': 'Unknown',
                                          'wwnn': None,
                                          'wwpn': None,
                                          'wwpnType': None}],
                          'reapplyState': 'NotApplying'},
 'created': None,
 'description': '',
 'eTag': None,
 'enclosureBay': None,
 'enclosureGroupUri': '/rest/enclosure-groups/796b7388-1d6a-4d14-9147-f7a2
3491d0c6',
 'enclosureUri': None,
 'firmware': {'consistencyState': 'Unknown',
              'firmwareActivationType': None,
              'firmwareBaselineUri': None,
              'firmwareInstallType': None,
              'firmwareScheduleDateTime': None,
              'forceInstallFirmware': False,
              'manageFirmware': False,
              'reapplyState': 'NotApplying'},
 'hideUnusedFlexNics': True,
 'inProgress': False,
 'iscsiInitiatorName': None,
 'iscsiInitiatorNameType': 'AutoGenerated',
 'localStorage': {'controllers': [],
                  'reapplyState': 'NotApplying',
                  'sasLogicalJBODs': []},
```

```
 'macType': 'Virtual',
 'managementProcessor': {'manageMp': True,
                         'mpSettings': [{'args': {'localAccounts': [{'disp
layName': 'p',
                                                                    'host
BIOSConfigPriv': True,
                                                                    'host
NICConfigPriv': True,
                                                                    'host
StorageConfigPriv': True,
                                                                    'iLOC
onfigPriv': True,
                                                                    'logi
nPriv': True,
                                                                    'pass
word': None,
                                                                    'remo
teConsolePriv': True,
                                                                    'user
ConfigPriv': True,
                                                                    'user
Name': 'skoch',
                                                                    'virt
ualMediaPriv': True,
                                                                    'virt
ualPowerAndResetPriv': True}]},
                                         'settingType': 'LocalAccounts'}],
                         'reapplyState': 'NotApplying'},
 'modified': None,
 'name': None,
 'osDeploymentSettings': None,
 'refreshState': 'NotRefreshing',
 'sanStorage': {'hostOSType': 'RHE Linux (5.x, 6.x, 7.x)',
                'manageSanStorage': True,
                'reapplyState': 'NotApplying',
                'sanSystemCredentials': [],
                'volumeAttachments': [{'associatedTemplateAttachmentId':
'9135e03d-e8a3-4104-aad2-3598ad03fae9',
                                       'bootVolumePriority': 'Primary',
                                       'id': 2,
                                       'lun': None,
                                       'lunType': 'Auto',
                                       'state': None,
                                       'status': None,
                                       'storagePaths': [{'connectionId':
2,
                                                         'isEnabled': Tru
e,
                                                         'targetSelector':
'Auto',
                                                         'targets': []},
                                                        {'connectionId':
3,
                                                         'isEnabled': Tru
e,
                                                         'targetSelector':
'Auto',
                                                         'targets': []}],
                                       'volume': {'initialScopeUris': Non
e,
                                                  'isPermanent': False,
```

```
                                                    'properties': {'descript
ion': '',
                                                                   'isDataRe
ductionEnabled': False,
                                                                   'isSharea
ble': False,
                                                                   'name':
'Ansbible_Deploy_Boot-Vol',
                                                                   'size': 4
2949672960,
                                                                   'snapshot
Pool': '/rest/storage-pools/08E4A609-988E-4C88-BF26-AB5D008EDEC0',
                                                                   'storageP
ool': '/rest/storage-pools/08E4A609-988E-4C88-BF26-AB5D008EDEC0',
                                                                   'template
Version': '3.0'},
                                                    'templateUri': '/rest/st
orage-volume-templates/1d3e0085-ee68-4ce3-9d9c-ab5d008edec6'},
                                          'volumeStorageSystemUri': '/rest/st
orage-systems/1c3cfa75-7dd6-4b93-84fe-ab5d008e1472',
                                          'volumeUri': None}]},
 'serialNumber': None,
 'serialNumberType': 'Virtual',
 'serverHardwareReapplyState': 'NotApplying',
 'serverHardwareTypeUri': '/rest/server-hardware-types/D5DAB541-6AC3-413D-
A69B-C50C654D81E0',
 'serverHardwareUri': None,
 'serverProfileTemplateUri': '/rest/server-profile-templates/950ac7c2-30f9
-47f4-927d-480d255f0361',
 'state': 'Creating',
 'status': 'OK',
 'taskUri': None,
 'templateCompliance': 'Unknown',
 'type': 'ServerProfileV11',
 'uri': None,
 'uuid': None,
 'wwnType': 'Virtual'}
create server profile
```

In [11]:

```
# get the ilo IP and an Token to login to ilo

import re
#server = oneview_client.server_hardware.get_by_name( "CTC H5 HE11, bay 2")
#pprint(server.data['mpHostInfo']['mpIpAddresses'][1]['address'])

remote_console_url = server.get_remote_console_url()
pprint(remote_console_url['remoteConsoleUrl'])

ssoRootUriHostAddressMatchObj = re.search( r'addr=(\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})', r
emote_console_url['remoteConsoleUrl'], re.M|re.I)
ssoTokenMatchObj = re.search( r'sessionkey=(\S*)$', remote_console_url['remoteConsoleUr
l'], re.M|re.I)  # This will get the session token that you will then use to pass to th
e iLO RedFish interface


server_address = ssoRootUriHostAddressMatchObj.group(1)
pprint(server_address)
Token = ssoTokenMatchObj.group(1)

pprint(Token)
redFishSsoSessionObject = { "RootUri": server_address, "Token": Token }
```

```
'hplocons://addr=10.0.20.68&sessionkey=c81c9796e875c65b633e533d202d900d'
'10.0.20.68'
'c81c9796e875c65b633e533d202d900d'
```

Mount an Installation ISO at the virtual media of an ilo and Boot the server

In [12]:

```
# with uid + pwd
import sys
import json
from redfish import RedfishClient
from redfish.rest.v1 import ServerDownOrUnreachableError

SYSTEM_URL = ("https://" + server_address)
LOGIN_ACCOUNT = "XXXXXX"
LOGIN_PASSWORD = "XXXXXX"
MEDIA_URL = "http://osdepl.demo.local/centos/centos7custom.iso"

def get_resource_directory(redfishobj):

    try:
        resource_uri = redfishobj.root.obj.Oem.Hpe.Links.ResourceDirectory['@odata.id']
    except KeyError:
        sys.stderr.write("Resource directory is only available on HPE servers.\n")
        return None

    response = redfishobj.get(resource_uri)
    resources = []

    if response.status == 200:
        sys.stdout.write("\tFound resource directory at /redfish/v1/resourcedirectory"
+ "\n\n")
        resources = response.dict["Instances"]
    else:
        sys.stderr.write("\tResource directory missing at /redfish/v1/resourcedirector
y" + "\n")

    return resources


def mount_virtual_media_iso(_redfishobj, iso_url, media_type, boot_on_next_server_reset
):

    virtual_media_uri = None
    virtual_media_response = []

    resource_instances = get_resource_directory(_redfishobj)
    if DISABLE_RESOURCE_DIR or not resource_instances:
        #if we do not have a resource directory or want to force it's non use to find t
he
        #relevant URI
        managers_uri = _redfishobj.root.obj['Managers']['@odata.id']
        managers_response = _redfishobj.get(managers_uri)
        managers_members_uri = next(iter(managers_response.obj['Members']))['@odata.id'
]
        managers_members_response = _redfishobj.get(managers_members_uri)
        virtual_media_uri = managers_members_response.obj['VirtualMedia']['@odata.id']
    else:
        for instance in resource_instances:
            #Use Resource directory to find the relevant URI
            if '#VirtualMediaCollection.' in instance['@odata.type']:
                virtual_media_uri = instance['@odata.id']

    if virtual_media_uri:
        virtual_media_response = _redfishobj.get(virtual_media_uri)
        for virtual_media_slot in virtual_media_response.obj['Members']:
```

```
            data = _redfishobj.get(virtual_media_slot['@odata.id'])
            if media_type in data.dict['MediaTypes']:
                virtual_media_mount_uri = data.obj['Actions']['#VirtualMedia.InsertMedi
a']['target']
                post_body = {"Image": iso_url}

                if iso_url:
                    resp = _redfishobj.post(virtual_media_mount_uri, post_body)
                    if boot_on_next_server_reset is not None:
                        patch_body = {}
                        patch_body["Oem"] = {"Hpe": {"BootOnNextServerReset": \
                                                     boot_on_next_server_reset}}
                        boot_resp = _redfishobj.patch(data.obj['@odata.id'], patch_body
)
                        if not boot_resp.status == 200:
                            sys.stderr.write("Failure setting BootOnNextServerReset")
                    if resp.status == 400:
                        try:
                            print(json.dumps(resp.obj['error']['@Message.ExtendedInfo'
], indent=4, \
                                                                                    sor
t_keys=True))
                        except Exception as excp:
                            sys.stderr.write("A response error occurred, unable to acce
ss iLO"
                                             "Extended Message Info...")
                    elif resp.status != 200:
                        sys.stderr.write("An http response of \'%s\' was returned.\n" %
resp.status)
                    else:
                        print("Success!\n")
                        print(json.dumps(resp.dict, indent=4, sort_keys=True))
                break

if __name__ == "__main__":



    #specify the type of content the media represents
    MEDIA_TYPE = "CD" #current possible options: Floppy, USBStick, CD, DVD
    #specify if the server should attempt to boot this media on system restart
    BOOT_ON_NEXT_SERVER_RESET = True

    # flag to force disable resource directory. Resource directory and associated opera
tions are
    # intended for HPE servers.
    DISABLE_RESOURCE_DIR = False

    try:
        # Create a Redfish client object
        REDFISHOBJ = RedfishClient(base_url=SYSTEM_URL, username=LOGIN_ACCOUNT, \
                                                                        password=LO
GIN_PASSWORD)
        # Login with the Redfish client
        REDFISHOBJ.login()
    except ServerDownOrUnreachableError as excp:
        sys.stderr.write("ERROR: server not reachable or does not support RedFish.\n")
        sys.exit()

    mount_virtual_media_iso(REDFISHOBJ, MEDIA_URL, MEDIA_TYPE, BOOT_ON_NEXT_SERVER_RESE
```

```
T)
    REDFISHOBJ.logout()
```

```
        Found resource directory at /redfish/v1/resourcedirectory

Success!

{
    "error": {
        "@Message.ExtendedInfo": [
            {
                "MessageId": "Base.1.4.Success"
            }
        ],
        "code": "iLO.0.10.ExtendedInfo",
        "message": "See @Message.ExtendedInfo for more information."
    }
}
```

In [14]:

```
OSIP="10.0.33.131"
HOSTNAME="centos01"

#create kickstart File on Webserver Directory
f= open("/persistent/osdepl/centos/centos7ks.cfg","w+")



f.write('lang en_US.UTF-8\n')
f.write('keyboard us\n')
f.write('timezone --utc America/New_York\n')
f.write('text\n')
f.write('install\n')
f.write('skipx\n')
f.write('network  --bootproto=static --ip=%s --netmask=255.255.255.0 ' % OSIP)
f.write(' --gateway=10.0.33.254 --nameserver=10.0.20.5 --hostname=%s\n' %  HOSTNAME)
f.write('authconfig --enable shadow --enablemd5\n')
f.write('firstboot --enable\n')
f.write('cdrom\n')
f.write('rootpw HP1nvent\n')
f.write('ignoredisk --only-use=/dev/disk/by-id/dm-name-mpatha\n')
f.write('zerombr\n')
f.write('clearpart --all --initlabel\n')
f.write('autopart --type=lvm\n')
f.write('reboot\n')
f.write('\n')
f.write('user --name=vagrant --plaintext --password vagrant --groups=vagrant,wheel\n')
f.write('\n')
f.write('#repo --name=docker --baseurl=https://download.docker.com/linux/centos/docker-
ce.repo\n')
f.write('\n')
f.write('# Disable firewall and selinux\n')
f.write('firewall --disabled\n')
f.write('selinux --disabled\n')
f.write('\n')
f.write('%pre\n')
f.write('%end\n')
f.write('\n')
f.write('%packages\n')
f.write('@Base\n')
f.write('@Core\n')
f.write('%end\n')
f.write('\n')
f.write('\n')
f.write('%post\n')
f.write('echo "vagrant        ALL=(ALL)       NOPASSWD: ALL" >> /etc/sudoers.d/vagrant
\n')
f.write('sed -i "s/^.*requiretty/#Defaults requiretty/" /etc/sudoers\n')
f.write('\n')
f.write('/bin/mkdir /home/vagrant/.ssh\n')
f.write('/bin/chmod 700 /home/vagrant/.ssh\n')
f.write('/bin/chown vagrant:vagrant /home/vagrant/.ssh\n')
f.write('/bin/echo -e "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDKphudM9WIBRid2DKz/UlQ+t99
bKMBfmynwy0Fj3ugolElu0lsCr0wRMeopHr5NyUz0EI4diO1CKSwu53axvQr8Lquu8W4/fi39r027efu0xMsCf2
eJFY+b7a8wyC8Y+UhXRfFxXWixuLxC06vlrew26Z7UXzk+WRCb/ixiN8wfRryUIROZ4RrV4cUt/gcobMSyvNVKJ
ksHfy/1MAGbwzene6dlHXeSrw7ipc721AqYgvdiGAc5UryDSJZpFTdAMY1aLQUOP7FlUNH30tHOyZLrp9HBhtQ3
gZO7rsHJwgtIIw5DnRF8BRmDq5AKvyDZRrEDEHirMTAt+BetokBA6DF skoch@ansible" > /home/vagran
t/.ssh/authorized_keys')
f.write('/bin/chown -R vagrant:vagrant /home/vagrant/.ssh\n')
```

```
f.write('  \n')
f.write('/usr/bin/yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo\n')
f.write('/usr/bin/yum install docker-ce -y\n')
f.write('/usr/bin/systemctl enable docker\n')
f.write('\n')
f.write('/usr/sbin/usermod  -a -G docker vagrant\n')
f.write('/usr/bin/yum -y install epel-release\n')
f.write('/usr/bin/yum -y install python-pip\n')
f.write('/usr/bin/pip install docker-py\n')
f.write('%end\n')
f.write('\n')

f.close()


#powerOn = {
#    "powerState": "On",
#    "powerControl": "MomentaryPress"
#}



#server = oneview_client.server_hardware.get_by_name( "CTC H5 HE11, bay 1")

## Power on Server and start Installation by booting from virtual media
server_power = server.update_power_state(powerOn) # turn off server
```

application deployment

Webserver NGINX running on docker.

What we need:

IP address of deployed system.

In [23]:

```
import os
import paramiko
import time

ssh = paramiko.SSHClient()

#server_name = "Roundtable - API Demo Server (DirkD)"
username = 'root'
password = 'HP1nvent'


ip_address=OSIP
print("We are pinging the server: "+ ip_address +" to wait till it´s online..."     )

# wait until Server is up .....

waiting=True
counter=0


import socket
def isOpen(ip,port):
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   try:
      s.connect((ip, int(port)))
      s.shutdown(2)
      return True
   except:
      return False


while not isOpen(OSIP,"22"):
        time.sleep(10)
        print("waiting to finish boot ")

time.sleep(20)

print("Login with user: " + username + " Server:" + ip_address)
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())         # add unknown Host-Ke
ys
ssh.connect(ip_address, username=username, password=password)     # login
# ssh.exec_command('systemctl restart docker')                      # workaround for ou
r docker environment

ssh.exec_command('docker run -d --name nginx -p 80:80 nginx')
time.sleep(10)
stdin, stdout, stderr = ssh.exec_command("docker exec -it nginx sed -i '\''s/nginx/Disc
over More/g'\'' /usr/share/nginx/html/index.html", get_pty=True)
time.sleep(5)
stdin, stdout, stderr = ssh.exec_command("docker exec -it nginx sed -i '\''s/nginx/Disc
over More/g'\'' /usr/share/nginx/html/index.html", get_pty=True)
print("http://" + ip_address)
# print(stdout.read())
# print(stderr.read())
```

```
We are pinging the server: 10.0.33.131 to wait till it´s online...
Login with user: root Server:10.0.33.131
http://10.0.33.131
```

Preperation for the CentOS kickstart Installation

Get an CentOS ISO image and mount it on an Linux Server

cp -pR it to an folder

Edit the grub.cfg under root of the CDROM and within EFI/BOOT

```
menuentry 'Install CentOS 7' --class fedora --class gnu-linux --class gnu --clas
s os {
    linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=CentOS\x207\x20x86_64
 quiet inst.ks=http://osdepl.demo.local/centos/centos7ks.cfg
    initrdefi /images/pxeboot/initrd.img
```

write an new customized iso with:

```
mkisofs -o /tmp/centos7custom.iso -b isolinux/isolinux.bin -J -R -l -c isolinux/
boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -eltorito-alt-boot -e
 images/efiboot.img -no-emul-boot -graft-points -V "CentOS7 Server.x86_64" .
```

Place customized iso and kickstart file on an reachable web server

dhcsp server must be available

create an Server Profile Template

```
with an ilo user in in
with an network connection where the dhcp request could be handled
and in my case with an SAN disk and boot from SAN configuration
```

Addtional stuff

example: how to get the values from json

In [ ]:

```
    server_hardware = oneview_client.server_hardware.get_all()
   # print(server_hardware)

    for server in server_hardware:
        for ports in server['portMap']['deviceSlots']:
            for mac in ports['physicalPorts']:
                for wwnn in mac['virtualPorts']:
                    print(server['name']+ " model:" + server['model'] + " " + str(serve
r['memoryMb']) + " MB mac: " + str(mac['mac']) + " wwnn: " + str(wwnn['wwnn']))
```

Excel Export

xlswriter: https://xlsxwriter.readthedocs.io/ (https://xlsxwriter.readthedocs.io/)

example: we extrace some data to excel XLS

In [ ]:

```
import xlsxwriter
workbook = xlsxwriter.Workbook('roundtable.xlsx')
worksheet = workbook.add_worksheet()

# Add a bold format to use to highlight cells.
bold = workbook.add_format({'bold': True})

# Text with formatting.
worksheet.write(0,0, 'Synergy Roundtable', bold)

# Start from the first cell below the headers.
row = 4
worksheet.write(row, 0, "Servername", bold)
worksheet.write(row, 1, "Model", bold)
worksheet.write(row, 2, "Memory", bold)
worksheet.write(row, 3, "MAC address", bold)
worksheet.write(row, 4, "WWN address", bold)
worksheet.write(row, 5, "Status", bold)
row += 1

server_hardware = oneview_client.server_hardware.get_all()
#print(server_hardware)

for server in server_hardware:
 col = 0
 for ports in server['portMap']['deviceSlots']:
     for mac in ports['physicalPorts']:
         for wwnn in mac['virtualPorts']:
             # print(server['name']+ " model:" + server['model'] + " " + str(server['me
moryMb']) + " MB mac: " + str(mac['mac']) + " wwnn: " + str(wwnn['wwnn']))
             worksheet.write(row,col, server['name'])
             worksheet.write(row,col+1, server['model'])
             worksheet.write(row,col+2, server['memoryMb'])
             worksheet.write(row,col+3, mac['mac'])
             worksheet.write(row,col+4, wwnn['wwnn'])
             worksheet.write(row,col+5, server['status'])
             row += 1
workbook.close()
print ('Excel File roundtable.xlsx created')
```

Delete Serverprofile

!!! do not use, if not necessary !!!

In [ ]:

```
server_power = oneview_client.server_hardware.update_power_state(powerOff, server_hardw
are_uri) # turn on server
oneview_client.server_profiles.delete(profile)
```

internal

In [ ]:

```
#### mount iso on ilo with sso token, not supported by python redfish sdk
####with token #### NOT WORKING
import sys
import json
from redfish import RedfishClient
from redfish.rest.v1 import ServerDownOrUnreachableError

#from get_resource_directory import get_resource_directory


def get_resource_directory(redfishobj):

    try:
        resource_uri = redfishobj.root.obj.Oem.Hpe.Links.ResourceDirectory['@odata.id']
    except KeyError:
        sys.stderr.write("Resource directory is only available on HPE servers.\n")
        return None

    response = redfishobj.get(resource_uri)
    resources = []

    if response.status == 200:
        sys.stdout.write("\tFound resource directory at /redfish/v1/resourcedirectory"
+ "\n\n")
        resources = response.dict["Instances"]
    else:
        sys.stderr.write("\tResource directory missing at /redfish/v1/resourcedirector
y" + "\n")

    return resources


def mount_virtual_media_iso(_redfishobj, iso_url, media_type, boot_on_next_server_reset
):

    virtual_media_uri = None
    virtual_media_response = []

    resource_instances = get_resource_directory(_redfishobj)
    if DISABLE_RESOURCE_DIR or not resource_instances:
        #if we do not have a resource directory or want to force it's non use to find t
he
        #relevant URI
        managers_uri = _redfishobj.root.obj['Managers']['@odata.id']
        managers_response = _redfishobj.get(managers_uri)
        managers_members_uri = next(iter(managers_response.obj['Members']))['@odata.id'
]
        managers_members_response = _redfishobj.get(managers_members_uri)
        virtual_media_uri = managers_members_response.obj['VirtualMedia']['@odata.id']
    else:
        for instance in resource_instances:
            #Use Resource directory to find the relevant URI
            if '#VirtualMediaCollection.' in instance['@odata.type']:
                virtual_media_uri = instance['@odata.id']

    if virtual_media_uri:
        virtual_media_response = _redfishobj.get(virtual_media_uri)
        for virtual_media_slot in virtual_media_response.obj['Members']:
            data = _redfishobj.get(virtual_media_slot['@odata.id'])
```

```
            if media_type in data.dict['MediaTypes']:
                virtual_media_mount_uri = data.obj['Actions']['#VirtualMedia.InsertMedi
a']['target']
                post_body = {"Image": iso_url}

                if iso_url:
                    resp = _redfishobj.post(virtual_media_mount_uri, post_body)
                    if boot_on_next_server_reset is not None:
                        patch_body = {}
                        patch_body["Oem"] = {"Hpe": {"BootOnNextServerReset": \
                                                     boot_on_next_server_reset}}
                        boot_resp = _redfishobj.patch(data.obj['@odata.id'], patch_body
)
                        if not boot_resp.status == 200:
                            sys.stderr.write("Failure setting BootOnNextServerReset")
                    if resp.status == 400:
                        try:
                            print(json.dumps(resp.obj['error']['@Message.ExtendedInfo'
], indent=4, \
                                                                                    sor
t_keys=True))
                        except Exception as excp:
                            sys.stderr.write("A response error occurred, unable to acce
ss iLO"
                                             "Extended Message Info...")
                    elif resp.status != 200:
                        sys.stderr.write("An http response of \'%s\' was returned.\n" %
resp.status)
                    else:
                        print("Success!\n")
                        print(json.dumps(resp.dict, indent=4, sort_keys=True))
                break

if __name__ == "__main__":


    SYSTEM_URL = "https://10.0.20.65"
    LOGIN_ACCOUNT = "XXXXX"
    LOGIN_PASSWORD = "XXXXXX"

    MEDIA_URL = "http://osdepl.demo.local/centos/centos7custom.iso"
    #specify the type of content the media represents
    MEDIA_TYPE = "CD" #current possible options: Floppy, USBStick, CD, DVD
    #specify if the server should attempt to boot this media on system restart
    BOOT_ON_NEXT_SERVER_RESET = True

    # flag to force disable resource directory. Resource directory and associated opera
tions are
    # intended for HPE servers.
    DISABLE_RESOURCE_DIR = False

    try:
        # Create a Redfish client object
        #REDFISHOBJ = RedfishClient(base_url=SYSTEM_URL, username=LOGIN_ACCOUNT, \
        #                                                                        password=L
OGIN_PASSWORD)
        REDFISHOBJ = RedfishClient(base_url=SYSTEM_URL, session=Token)

        # Login with the Redfish client
        REDFISHOBJ.login(auth="session")
    except ServerDownOrUnreachableError as excp:
```

```
        sys.stderr.write("ERROR: server not reachable or does not support RedFish.\n")
        sys.exit()

    mount_virtual_media_iso(REDFISHOBJ, MEDIA_URL, MEDIA_TYPE, BOOT_ON_NEXT_SERVER_RESE
T)
    REDFISHOBJ.logout()
```