Staged Evolution of Integrating with Redfish

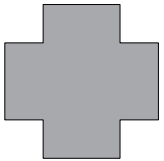
Interacting with hardware resources from a software perspective

Bryan Gartner
Sr. Technology Strategist
SUSE / bryan.gartner@suse.com

Redfish Integration / Usage

Agenda

- Introduction
- Redfish overview from an Open Source software person’s context
- Then, an evolving progression of
  - Accessing the Redfish API and Data Model contents
  - Start manipulating the target hardware to match what the overall use case requires
  - Leveraging all the pieces for an end-to-end deployment / solution



Overview of Redfish

From a software person's context

- Yet another way to access a Baseboard Management Controller (BMC)
  - Bonus points
    - Provides a superset of functionality compared to IPMI
    - Delivers a standardized approach across hardware partner platforms
- Provides / utilizes a REST API approach
  - Bonus points
    - Enables lots of possible ways to integrate
    - Creates a composable, converged, hybrid-IT option to extend the software defined data center concept
    - Feels almost cloud-native like, given that a versioned API approach exists to manage the hardware that software lands upon

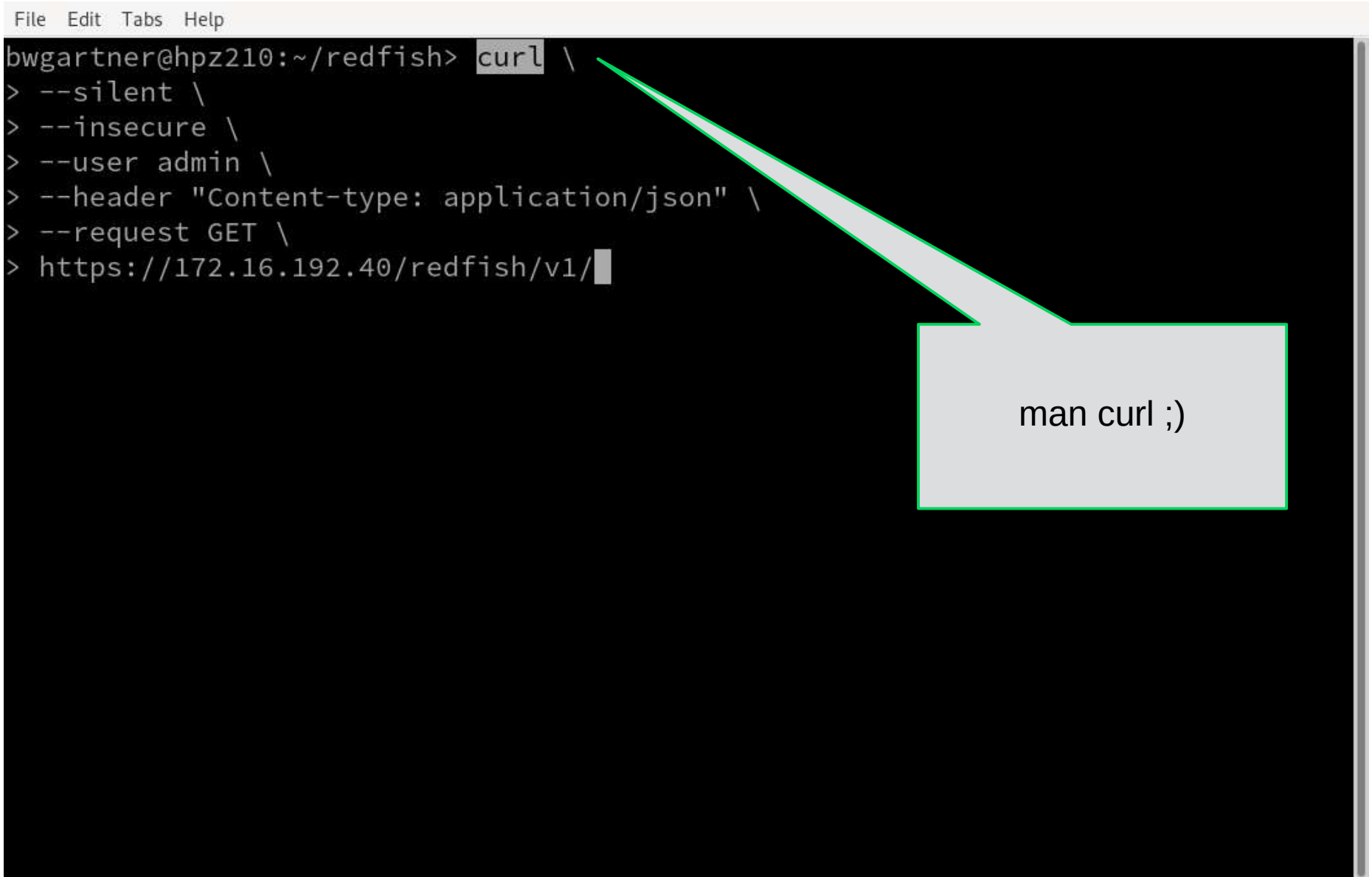
~~Crawl~~ Float

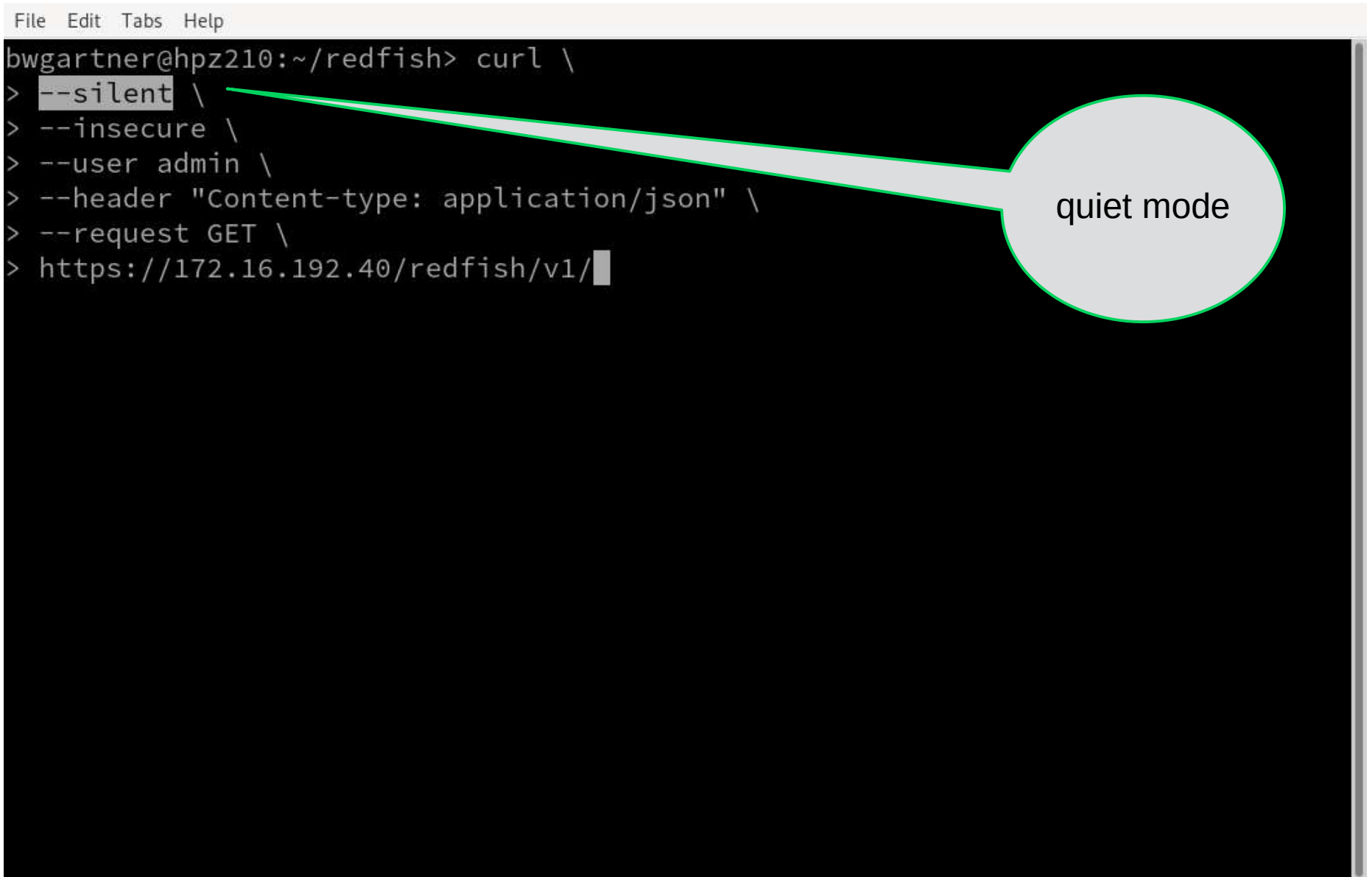
Float : accessing the API/Data Model

Start simple

- Via curl, interactive to scripted CLI walk through
  - literally started with a Google “linux redfish curl examples” search
    - Setup curl options
    - Validated access URL and credentials
    - Formatted output into readable (JSON)
    - Explored a subset of the data model
    - Scripted a poll across several systems

Accessing the Redfish API


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
man curl ;)


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
quiet mode


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
deal with
self-signed
BMC certificate


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
BMC
user credential
(password will be
prompted for)


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
extra header request


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
just request data


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
BMC
IP Address
(and protocol)


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/
Use the Redfish top of API path and current version

File Edit Tabs Help

```
l360g9-a-ilo.suse.de","HostName":"dl360g9-a-ilo","IPManager":{"BiosManaged":fals
e,"FirmwareManaged":false,"ManagerProductName":"HPE OneView","ManagerType":"OneV
iew","ManagerUrl":{"xref":"https://172.16.250.127"},"ManagerVersion":"4.20.01.01
","Name":"Management Console Information","OvManagesiLOIP":false,"SppVersion":nu
ll,"StorageManaged":false,"Type":"HPQ_iLOM                    0","iLOManaged":t
rue,"type":"IpManagerBlob"},"Languages":[                    lationName":"Eng
lish","Version":"2.61"}],"ManagerFirmwa                    rType":"iLO 4"}]
,"Sessions":{"CertCommonName":"dl360g9-                    nabled":false,"L
DAPAuthLicenced":true,"LDAPEnabled":fals                    true,"LoginFailure
Delay":0,"LoginHint":{"Hint":"POST to /Se                    ng the following JSO
N object:","HintPOSTData":{"Password":"passwo                    ame":"username"}},"Securi
tyOverride":false,"ServerName":""},"Type":"HpiLOServiceExt.1.0.0","links":{"Reso
urceDirectory":{"href":"/redfish/v1/Reso                    Directory/"}}}},"RedfishVersion":"1.
0.0","Registries":{"@odata.id":"/redfish/v1/Registries/"},"ServiceVersion":"1.0.
0","SessionService":{"@odata.id":"/redfish/v1/SessionService/"},"Systems":{"@oda
ta.id":"/redfish/v1/Systems/"},"Time                    2019-08-13T20:57:57Z","Type":"ServiceRoot
.1.0.0","UUID":"110fe98a-318c-5283-8572-21f6c0ab0955","links":{"AccountService":
{"href":"/redfish/v1/AccountService/"},"Chassis":{"href":"/redfish/v1/Chassis/"}
,"EventService":{"href":"/redfish/v1/EventService/"},"Managers":{"href":"/redfis
h/v1/Managers/"},"Registries":{"href":"/redfish/v1/Registries/"},"Schemas":{"hre
f":"/redfish/v1/Schemas/"},"SessionService":{"href":"/redfish/v1/SessionService/
"},"Sessions":{"href":"/redfish/v1/SessionService/Sessions/"},"Systems":{"href":
"/redfish/v1/Systems/"},"self":{"href":"/redfish/v1/"}}}
bwgartner@hpz210:~/redfish>
```

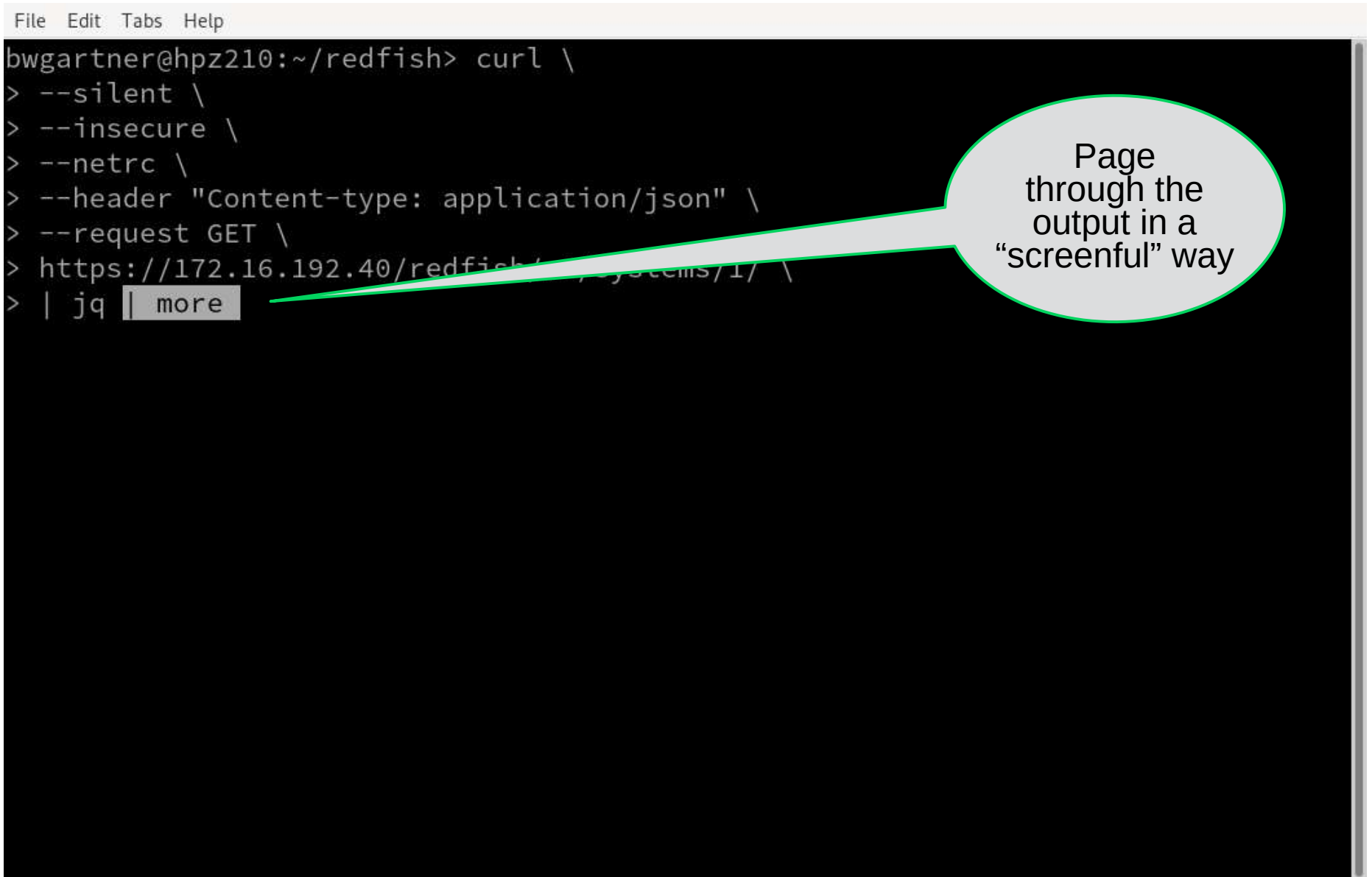
Ok … worked
… but output not entirely human readable


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --user admin \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/ \
> | jq
man jq ;)

Exploring the Data Model


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --netrc \
> --header "Content-type: application/j
> --request GET \
> https://172.16.192.40/redfish/v1/Systems/1/ \
> | jq | more
Read authentication credentials from a file


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --netrc \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/v1/Systems/1/ \
> | jq | more
Grab sub-tree of data model


File Edit Tabs Help
bwgartner@hpz210:~/redfish> curl \
> --silent \
> --insecure \
> --netrc \
> --header "Content-type: application/json" \
> --request GET \
> https://172.16.192.40/redfish/
> | jq | more
Page through the output in a “screenful” way

Simplified Scaling of Information Gathering

File Edit Tabs Help

```
#! /bin/sh

IPSub="172.16"

for i in 192 195
  do
    for j in 36 35 34 33 32
      do
        echo "=== Node BMC - ${IPSub}.${i}.${j} ==="
        curl \
                --silent \
                --insecure \
                --netrc \
                --header "Content-type: application/json" \
                --request GET \
                https://${IPSub}.${i}.${j}/redfish/v1/Systems/1/ \
        | jq '{Model}'
      done
  done
~
~
~
~
                                                    1,1           All
```

Wrap into a shell script

File Edit Tabs Help

```
#! /bin/sh

IPSub="172.16"

for i in 192 195
  do
    for j in 36 35 34 33 32
      do
        echo "=== Node BMC - ${IPSub}.${i}.${j} ==="
        curl \
                --silent \
                --insecure \
                --netrc \
                --header "Content-type: application/json" \
                --request GET \
                https://${IPSub}.${i}.${j}/redfish/v1/Systems/1/ \
        | jq '{Model}'
      done
  done
~
~
~
~
                                                    1,1           All
```

Loop through several BMC IP ranges

File Edit Tabs Help

```
#! /bin/sh

IPSub="172.16"

for i in 192 195
  do
    for j in 36 35 34 33 32
      do
        echo "=== Node BMC - ${IPSub}.${i}.${j} ==="
        curl \
                --silent \
                --insecure \
                --netrc \
                --header "Content-Type: application/json" \
                --request GET \
                https://${IPSub}.${i}.${j}/redfish/v1/Systems/1/ \
        | jq '{Model}'
      done
  done
~
~
~
~
                                                    1,1           All
```

Extract a specific name/value item

[X] Float

Of course, a lot more ways this can be also exercised

- Redfish API
- Exploring Data Model
  - Redfish Developer Hub (see Mockups)
- Programmatic Interfaces
  - Language bindings : C, Javascript, Powershell, Python, Ruby, …
  - DevOps : Ansible, Chef, Nagios, Puppet, ...

Float (additional references)

Homework exercises left for the reader

- Dell-related
  - Knowledge Base - Redfish
- Fujitsu
  - iRMC Redfish API Specifications
  - Redfish White Paper
- HPE-related
  - iLO RESTful API
  - iLO RESTful API Explorer
- Intel
  - Redfish, RESTful and x-UEFI
- Lenovo-related
  - xClarity Controller Redfish REST API
- Supermicro
  - Server Management (Redfish API)
- ...

~~Walk~~ Tread

Tread(ing Water) : understand the target

Helping the hardware-challenged (aka software folks)

- Beyond the on-line Mockups ...
  - Visit GitHub openStack/python-redfish
    - git clone
      - Install a container run-time engine
      - In dmtf/mockup*, build, run, use the container
  - Homework left as an exercise for the reader
    - You can install (from src, PyPi, or packages the redfish-client)

Tread

Other techniques and/or target resources ...

- SUSE Manager / Uyuni
  - Opensource software management solution
  - Leverages Saltstack, and starting development of a Redfish integration - openSUSE/redfish
    - Goal is to be able to query/select/configure + de-configure/de-select/return to a known state the hardware needed to match the desire software workloads as part of the overall deployment lifecyle
      - *salt-call redfish.set_property IndicatorLED “Blinking”* … (or “Off”)
  - Terraform
    - Starting to leverage this technology, which matches quite well with the underlying infrastructure
      - A restapi provider to interact with Redfish
      - A terraform-provider-oneview overlay that works with the HPE Composable Infrastructure APIs

[kinda] Tread

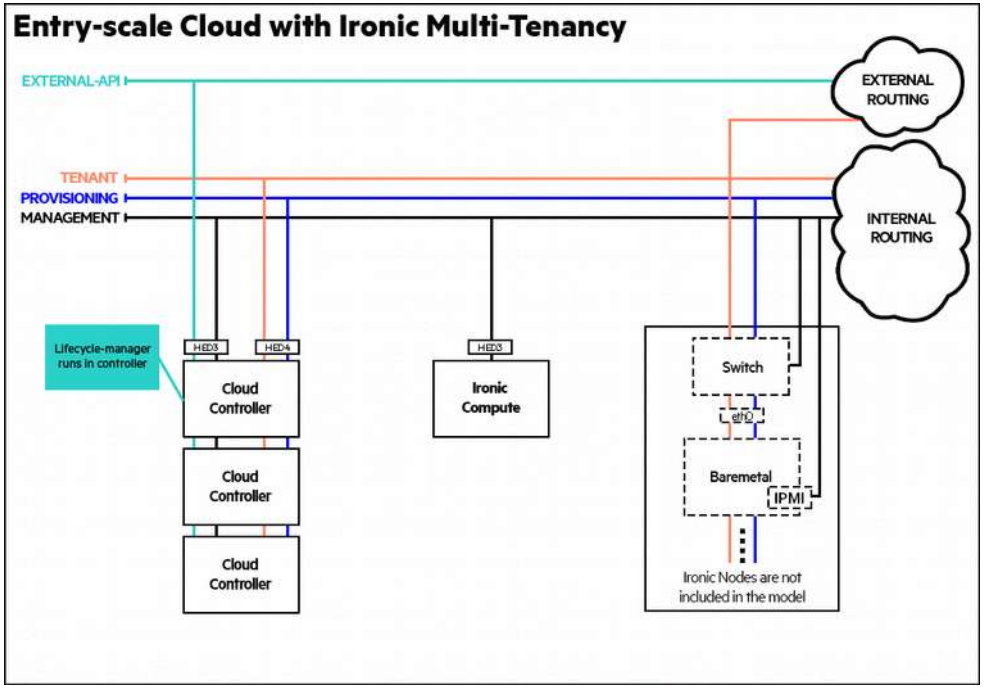
Continually exploring some new and some existing options

- In the end, the true value proposition of open source for users is “freedom of choice”
  - So with the trends of
    - Software-Defined Infrastructure
    - Migration to Infrastructure-as-Code
    - Cloud-Native computing principles (everything is really an API/version)
  - Providing choices in each matrix element and layer approach is highly desirable

~~Run~~ Swim

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from within SUSE OpenStack Cloud



Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor
    - Network port(s)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor … based upon system memory, storage, cpu architecture, boot mode
    - Network port(s) … based upon system MAC Address(es)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor … based upon system memory, storage, cpu architecture, boot mode
    - Network port(s) … based upon system MAC Address(es)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system
  - Power on the node to
    - Enable a PXE boot from a designated NIC (and desired flat/multi-tenant Neutron network) to a specific kernel/initrd image
    - Transfer the complete disk image to the node's local storage
    - Reboot the node and use as desired

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor … based upon system memory, storage, cpu architecture, boot mode
    - Network port(s) … based upon system MAC Address(es)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system
  - Power on the node to
    - Enable a PXE boot from a designated NIC (and desired flat/multi-tenant Neutron network) to a specific kernel/initrd image
    - Transfer the complete disk image to the node's local storage
    - Reboot the node and use as desired

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor … based upon system memory, storage, cpu architecture, boot mode
    - Network port(s) … based upon system MAC Address(es)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system
  - Power on the node to
    - Enable a PXE boot from a designated NIC (and desired flat/multi-tenant Neutron network) to a specific kernel/initrd image
    - Transfer the complete disk image to the node's local storage
    - Reboot the node and use as desired
  - And when done with the workload
    - Power off the system, reset BIOS / secure boot keys / iLO and credentials, erase devices, possibly update firmware, ...

Swim : manage end-to-end deployment

Deploy a baremetal node using Ironic + Redfish from SUSE OpenStack Cloud

- Stepwise Process Approach (see Deployment Guide using Cloud Lifecycle Manager)
  - Setup access to the BMC (utilizing Redfish API via OpenStack Sushy python library)
  - Enroll the node into Ironic, to provide access to / visibility of the system resources
  - Create
    - Nova flavor … based upon system memory, storage, cpu architecture, boot mode
    - Network port(s) … based upon system MAC Address(es)
    - Glance complete disk image for the target solution workload
    - Key pair to later login to the running system
  - Power on the node to
    - Enable a PXE boot from a designated NIC (and desired flat/multi-tenant Neutron network) to a specific kernel/initrd image
    - Transfer the complete disk image to the node's local storage
    - Reboot the node and use as desired
  - And when done with the workload
    - Power off the system, reset BIOS / secure boot keys / iLO and credentials, erase devices, possibly update firmware, ...

[X] Swim

One complete example of a solution end-to-end deployment

- Entirely possible to “make it so”,
  - Matching what one needs to have in place
  - At the desired time
  - Then recycling as needed to the next “need”

Summary

So interesting to explore / discover / leverage

- Redfish integration is an ever expanding utility / frontier
- Allows boundary crossing from developers
  to operations and across the classic IT silos
- ~~Game~~ Meet On!

Questions

Thank You