Automated Out-of-Band management with Ansible and Redfish

Jose Delarosa – Senior Linux Engineer at Dell EMC
Jake Jackson – Product Field Engineer at Ansible by Red Hat
May 2nd, 2017

Who are we

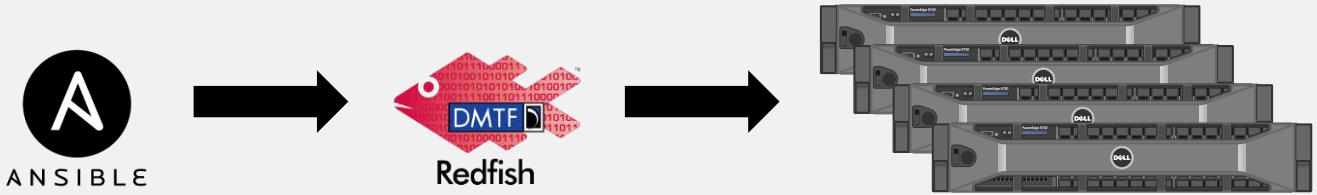
- Jose De la Rosa (@jdelaros1)
  - Linux Engineer at Dell EMC.
  - Systems engineer, emerging technologies evangelist.
- Jake Jackson (@thedoubl3j)
  - Product Field Engineer at Ansible by Red Hat.
  - Works with the Getting Started team helping customers with standing up and getting started with Ansible.

Before we start

1. Thank you for coming.
2. Please ask questions at any time.
3. If time runs out, we will be happy to talk to you after the session.

Content

1. Out-of-band management with PowerEdge iDRAC
2. Scalable out-of-band management with Redfish
3. Automated out-of-band management with Ansible

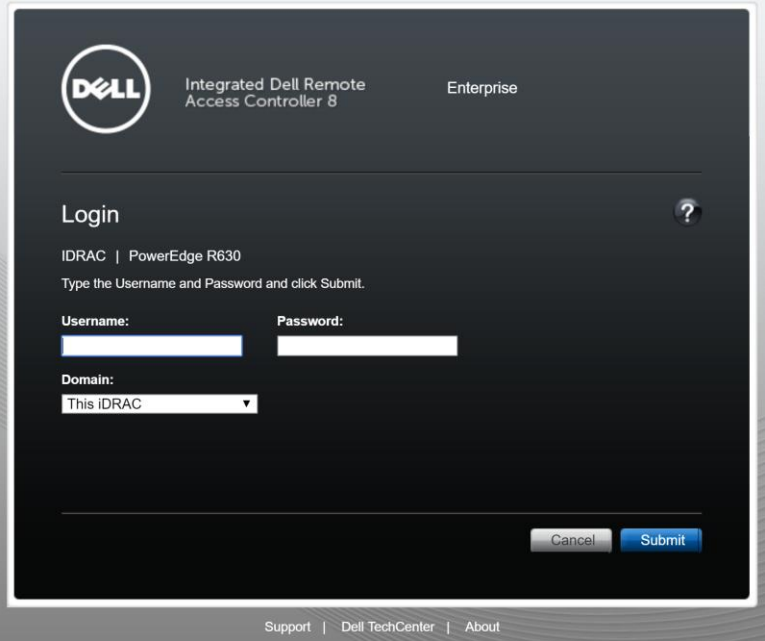


iDRAC Overview

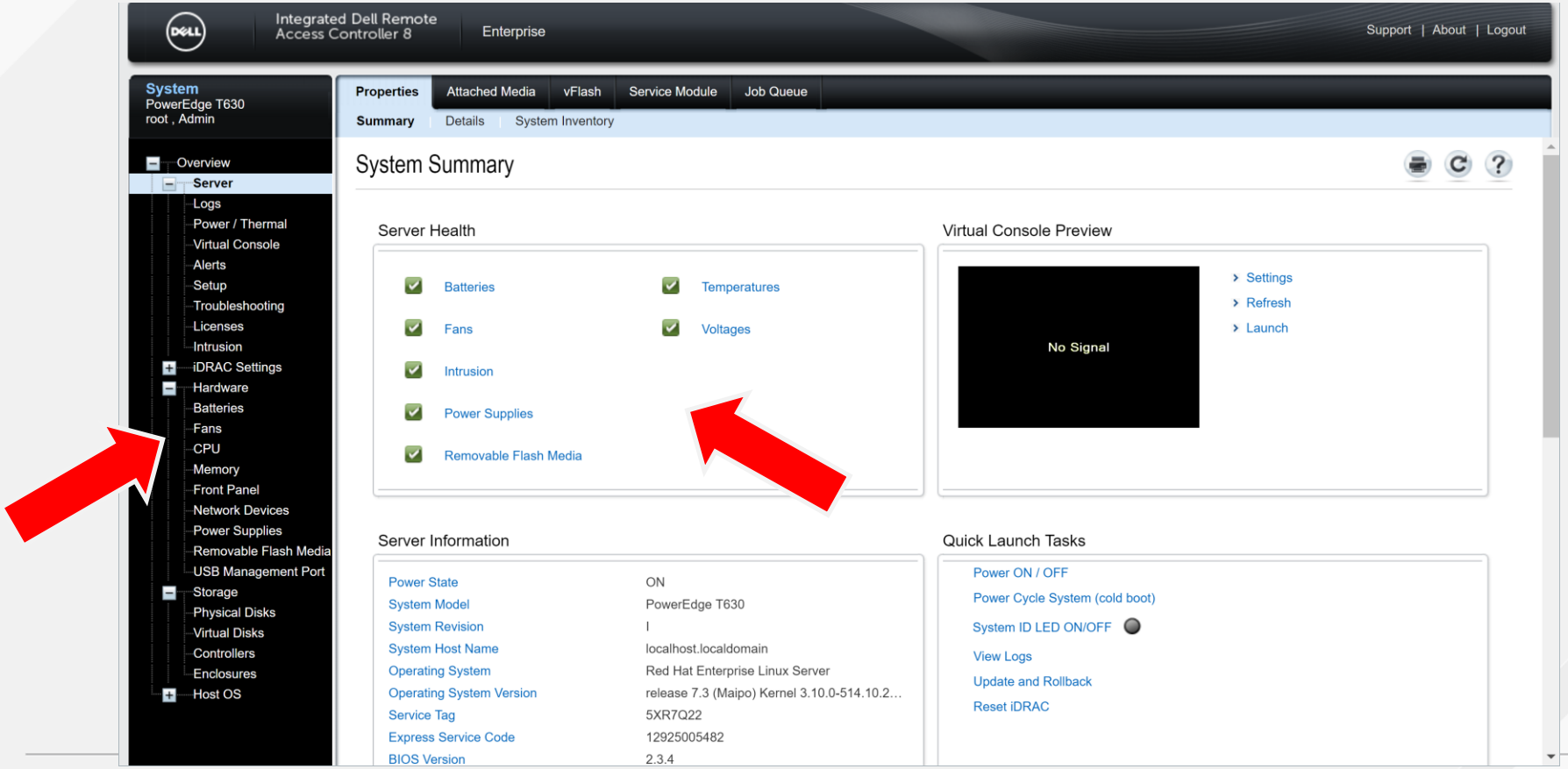
Integrated Dell Remote Access Controller (iDRAC)

- Embedded chip on a PowerEdge server used to run tasks that are independent of the host OS and the rest of the server.
  - Detects hardware failure
  - Manage power: turn off, on, hard reset
  - System event and lifecycle logs
- Has its own ethernet port, usually connected to separate management network.
- Referred to as “out-of-band” management, as opposed to “in-band” management which is provided by the OS.

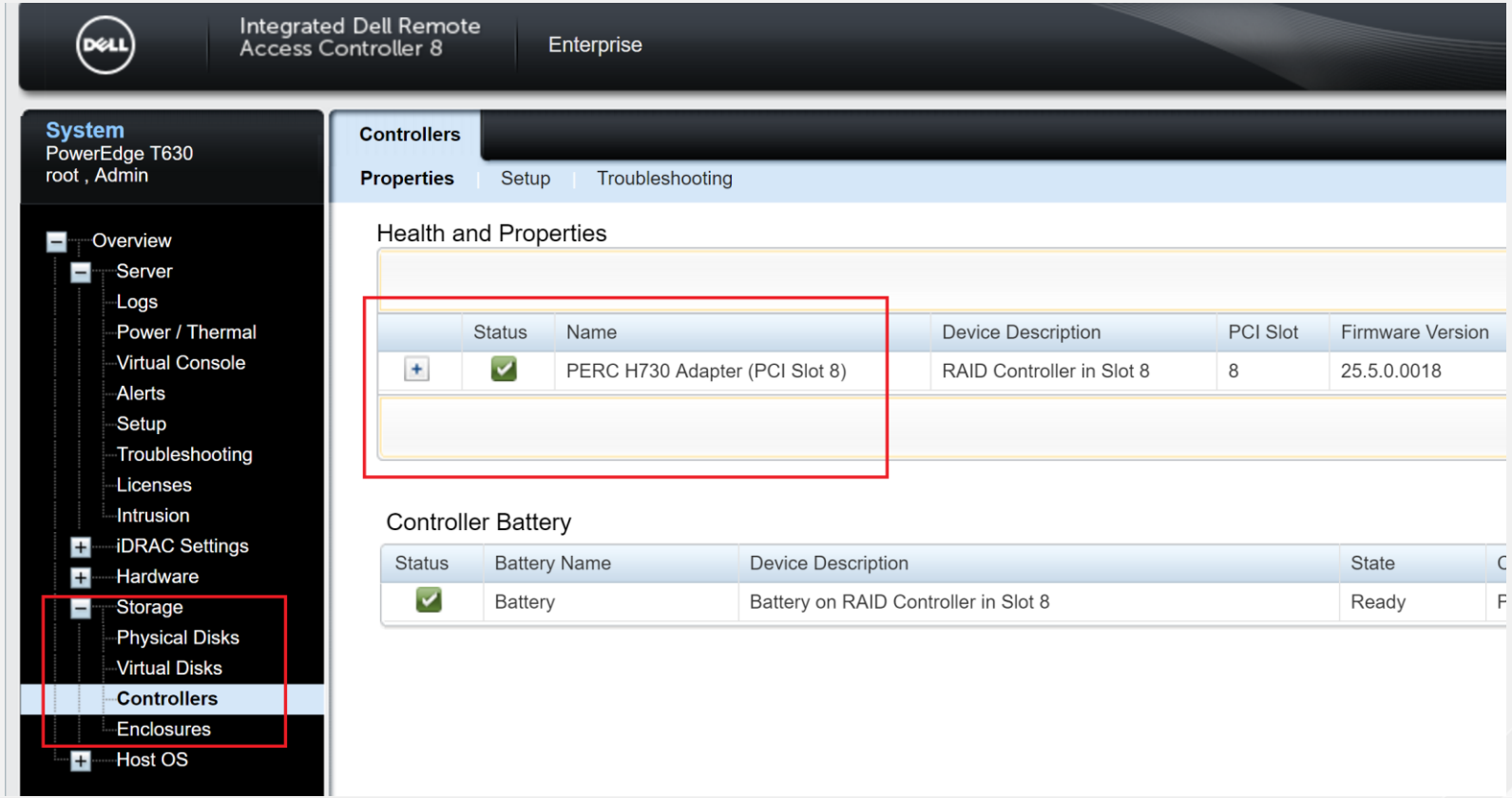
Web UI Login



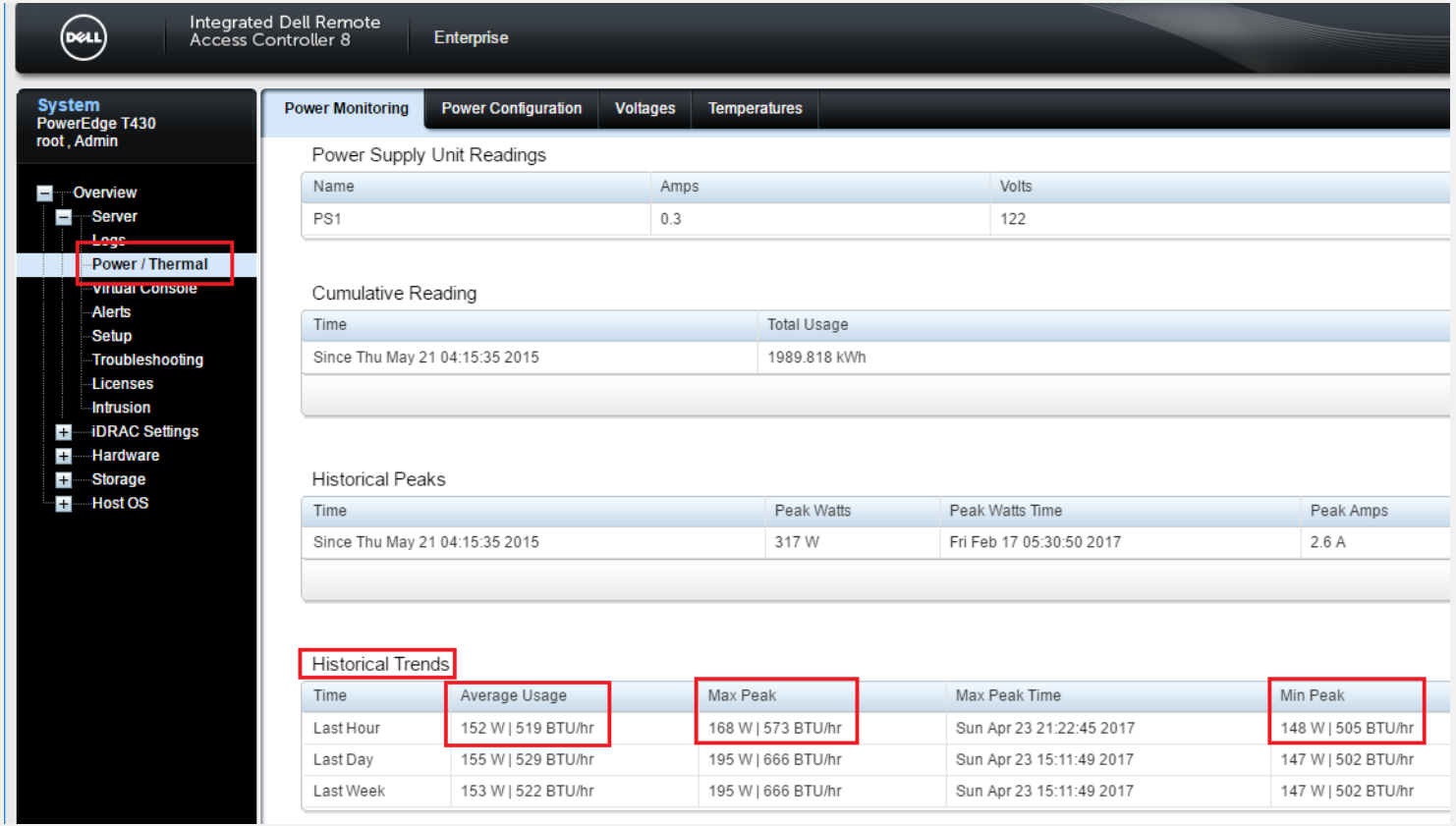
Main page


Integrated Dell Remote Access Controller 8
Enterprise
Support | About | Logout
System
PowerEdge T630
root , Admin
Properties
Attached Media
vFlash
Service Module
Job Queue
Summary
Details
System Inventory
Overview
Server
Logs
Power / Thermal
Virtual Console
Alerts
Setup
Troubleshooting
Licenses
Intrusion
iDRAC Settings
Hardware
Batteries
Fans
CPU
Memory
Front Panel
Network Devices
Power Supplies
Removable Flash Media
USB Management Port
Storage
Physical Disks
Virtual Disks
Controllers
Enclosures
Host OS
System Summary
Server Health
Batteries
Temperatures
Fans
Voltages
Intrusion
Power Supplies
Removable Flash Media
Virtual Console Preview
No Signal
Settings
Refresh
Launch
Server Information
Power State ON
System Model PowerEdge T630
System Revision I
System Host Name localhost.localdomain
Operating System Red Hat Enterprise Linux Server
Operating System Version release 7.3 (Maipo) Kernel 3.10.0-514.10.2...
Service Tag 5XR7Q22
Express Service Code 12925005482
BIOS Version 2.3.4
Quick Launch Tasks
Power ON / OFF
Power Cycle System (cold boot)
System ID LED ON/OFF
View Logs
Update and Rollback
Reset iDRAC

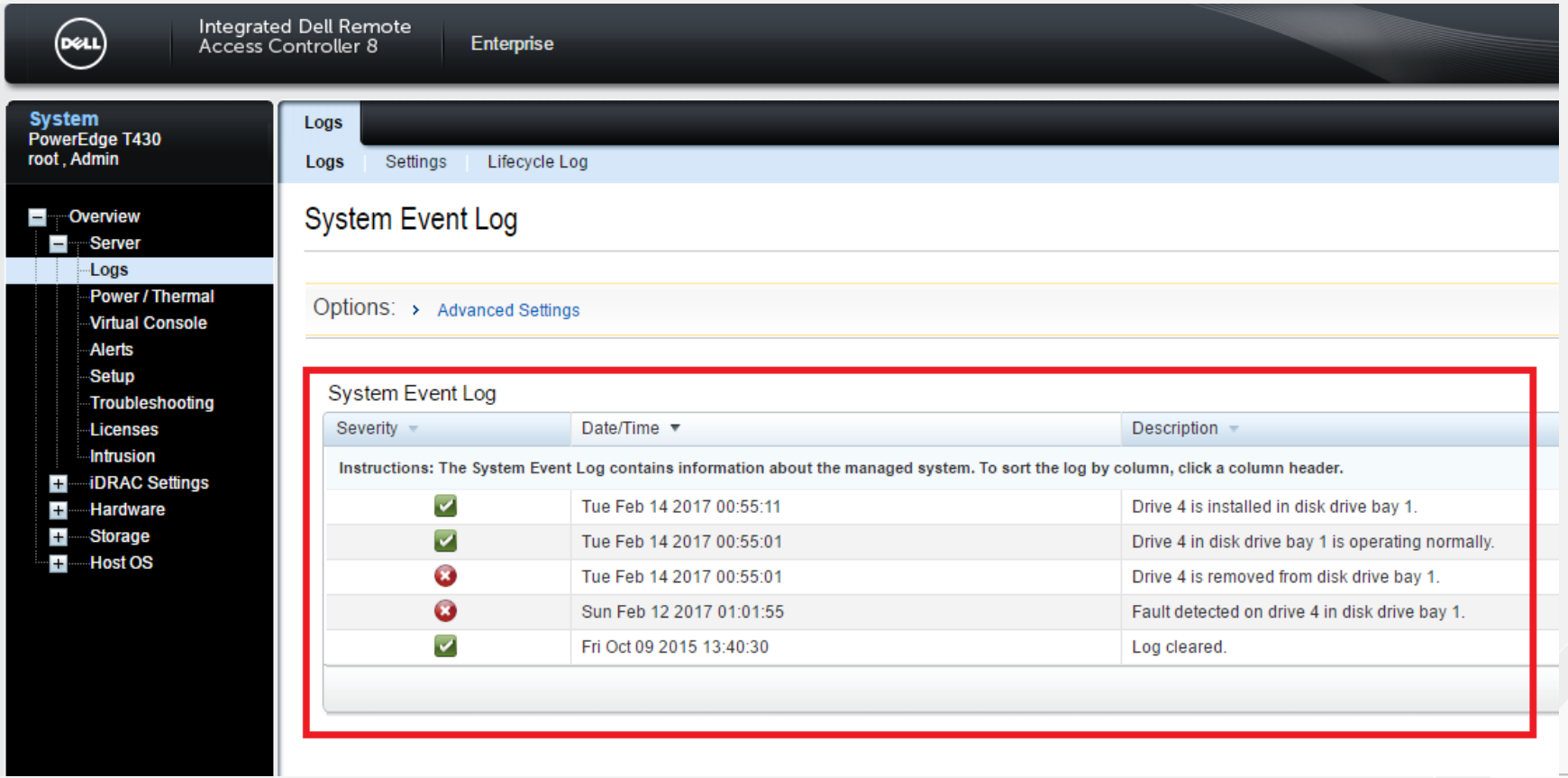
Storage controller status



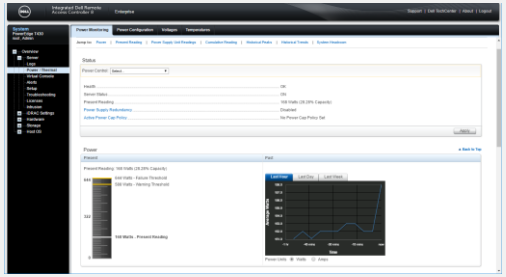
Power & thermal readings

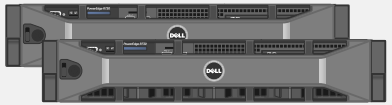


System event logs



Simple Out-of-band management





1- 5 servers

Redfish Overview

What is Redfish?

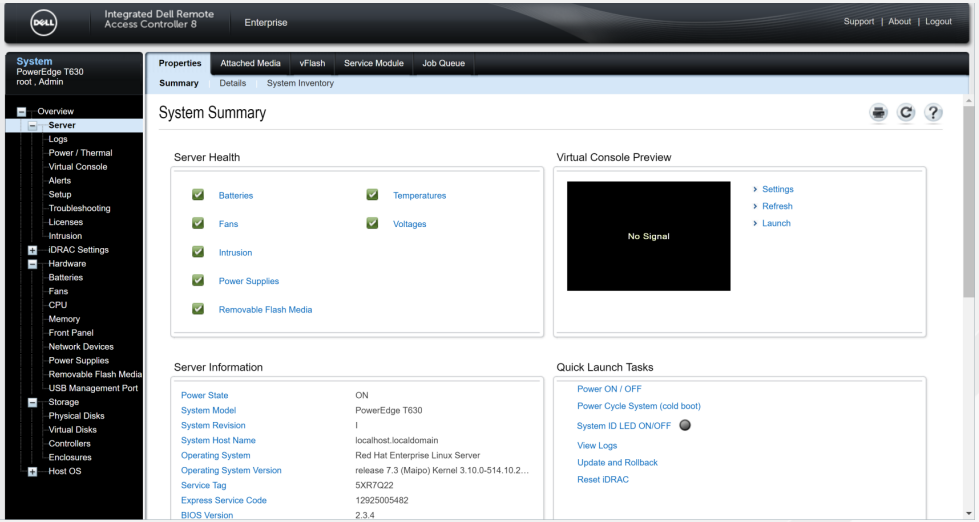
- RESTful API specification defined by the DMTF for hardware management.
- Aims to replace IPMI and vendor-specific interfaces like WSMAN.
- Schema-based but human-readable.
- Clients send URI requests over https to iDRAC, so clients can be any OS or application on a server, workstation or mobile device.

What can Redfish do?

- Monitor server health status
- Alert on server health status changes
- View server hardware inventory and firmware versions
- Reset, reboot, and power control servers
- Access system logs

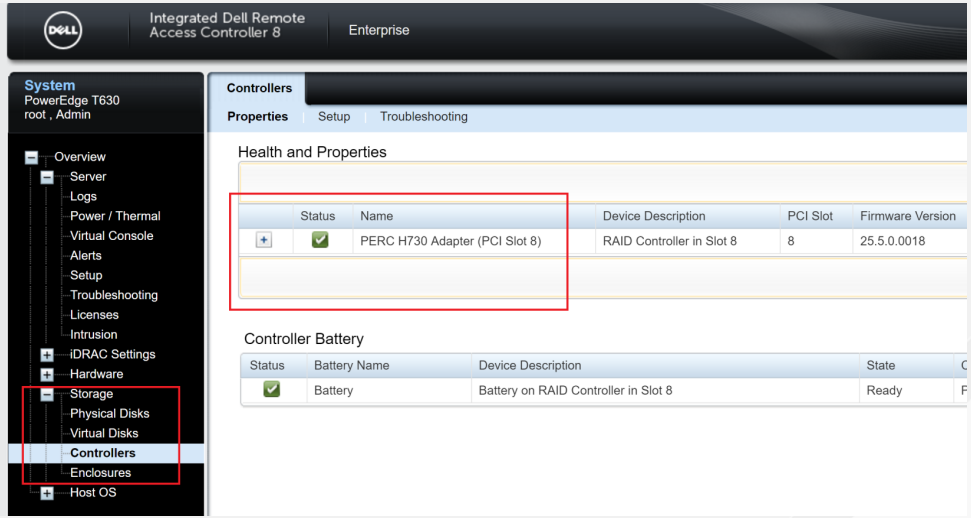
Example: Get system health

```
$ curl -s https://<idrac_ip>/redfish/v1/Systems/System.Embedded.1 -k -u
root:password | python -m json.tool | jq .Status
{
  "Health": "OK",
  "HealthRollUp": "OK",
  "State": "Enabled"
}
```



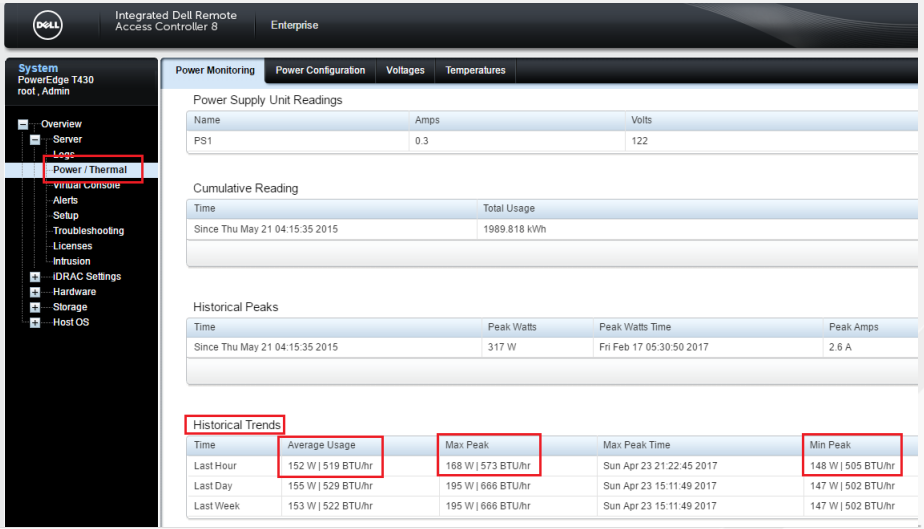
Example: Get storage controller health

```
$ curl -s
https://<idrac_ip>/redfish/v1/Systems/System.Embedded.1/Storage/Controllers/RAID.
Slot.8-1 -k -u root:password | python -m json.tool | jq .Name
"PERC H730 Adapter"
$ curl -s
https://<idrac_ip>/redfish/v1/Systems/
Slot.8-1 -k -u root:password | python
{
  "Health": "OK",
  "HealthRollUp": "OK"
}
```



Example: Get power consumption during last hour

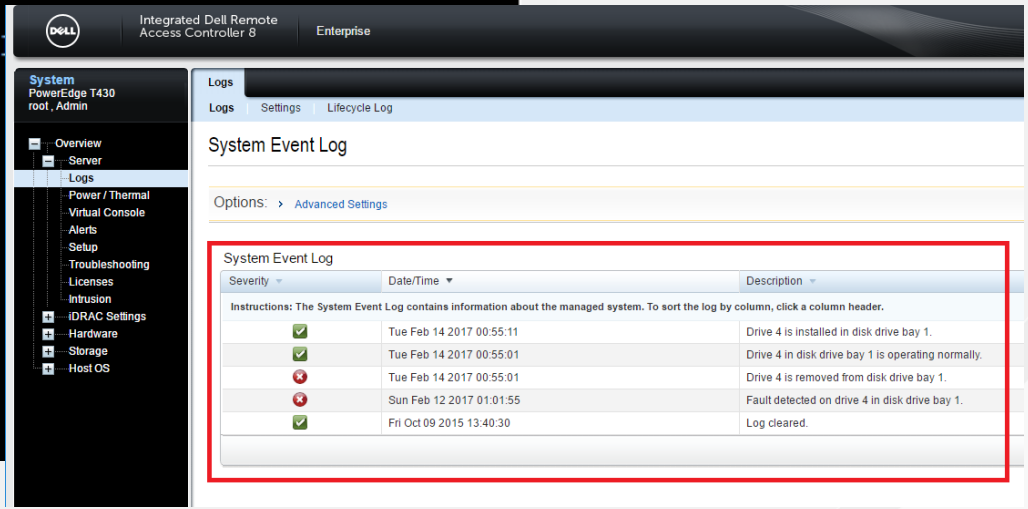
```
$ curl -s
https://<idrac_ip>/redfish/v1/Chassis/System.Embedded.1/Power/PowerControl -k -u
root:password | python -m json.tool | jq .PowerMetrics
{
  "AverageConsumedWatts": 152,
  "IntervalInMin": 60,
  "MaxConsumedWatts": 168,
  "MinConsumedWatts": 148
}
```



Example: Get system event logs

root@ans:/opt/dlr/files/redfish

```
[root@ans redfish]# ./view-system-events.py 192.168.0.53
ip=192.168.0.53, id=root, pw=calvin
uri=https://192.168.0.53/redfish/v1/Managers/iDRAC.Embedded.1/Logs/Sel
 Log Entry 5: 2017-02-13T22:55:11-06:00
 Drive 4 is installed in disk drive bay 1.

 Log Entry 4: 2017-02-13T22:55:01-06:00
 Drive 4 in disk drive bay 1 is operating norma

 Log Entry 3: 2017-02-13T22:55:01-06:00
 Drive 4 is removed from disk drive bay 1.

 Log Entry 2: 2017-02-11T23:01:55-06:00
 Fault detected on drive 4 in disk drive bay 1.

 Log Entry 1: 2015-10-09T12:40:30-05:00
 Log cleared.

[root@ans redfish]#
```



What else can Redfish do?

Retrieve Telemetry
- Basic server identification and asset information
- Health state
- Temperature sensors and fans
- Power consumption and thresholds

Discovery
- Service endpoint (network-based discovery)
- Syste

Basic I/O I
- Host
- Simp

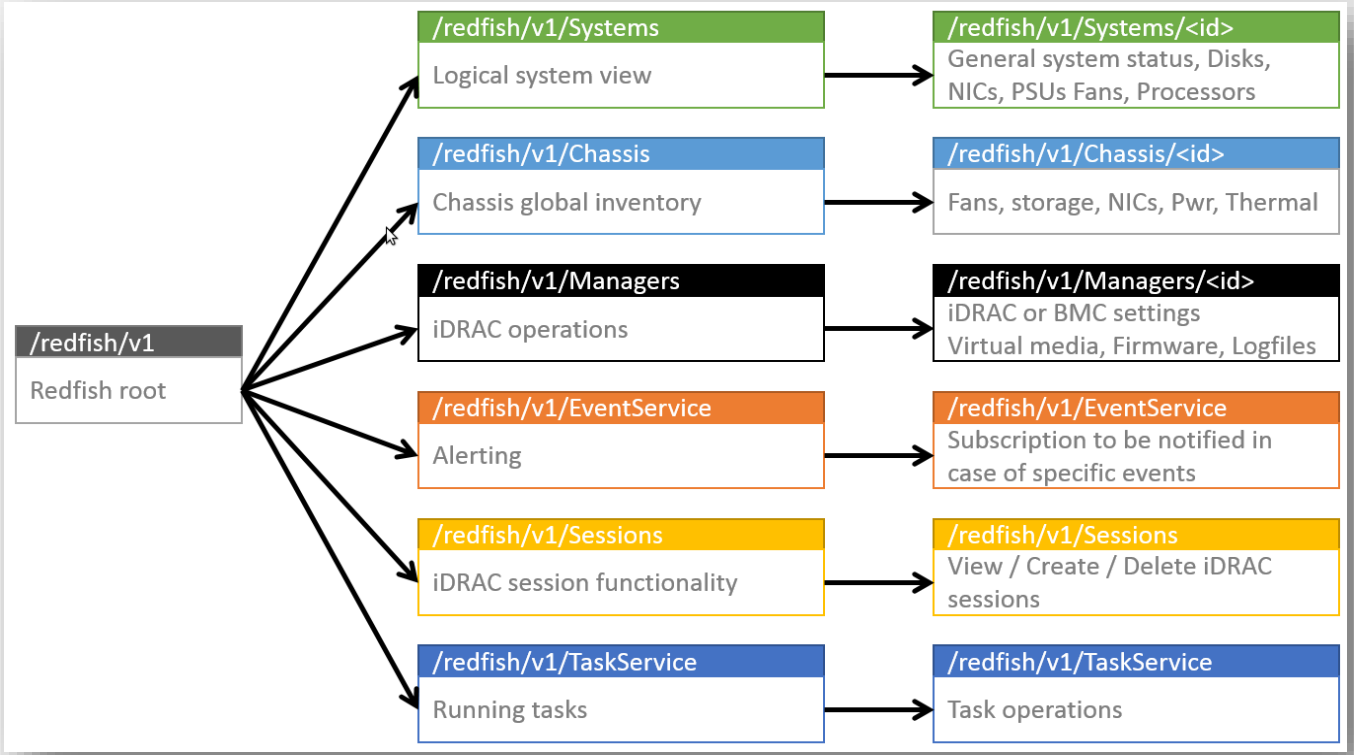
Security
- Session-based leveraging HTTPS

Common Management Actions
- Reboot / power cycle
- Change boot order
- Configure BMC network settings
- Manage user accounts

Access and Notification
- Serial console access via SSH
- Alert / event notification
- Event log access

More Coming Soon!
https://www.dmtf.org/standards/redfish

Redfish API tree structure

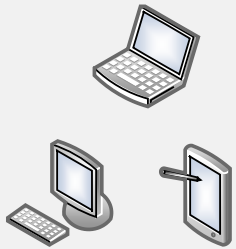


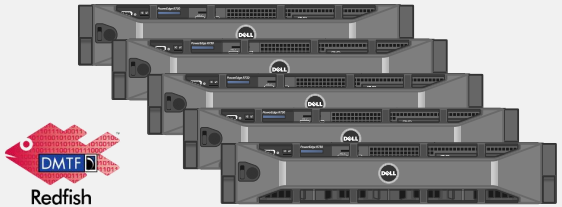
Redfish Roadmap

- Version 1 focused on servers. Expand over time to cover rest of IT infrastructure.
- Will add devices over time to cover new technologies (i.e. NVDIMMs, Multifunction Adapters)
- SNIA is developing *Swordfish*, which builds upon Redfish's local storage management to address advanced storage devices.
- Open source efforts:
  - Client libraries (Python, Java, PowerShell)
  - Command line utility *redfishtool* (similar to *ipmitool*)

Scalable Out-of-band management

https:/ / <idrac-ip>/ redfish/ v1/ Managers/ iDRAC.Embedded.1/ Logs/ Sel





More than 5 servers

Server data

Ansible Overview

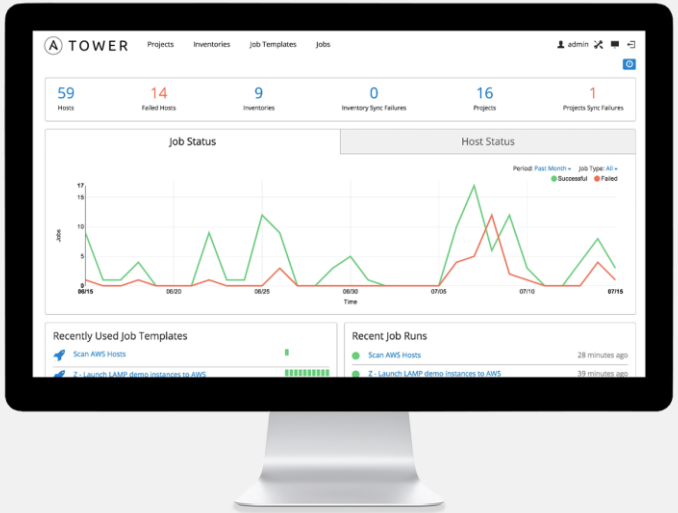
What is Ansible?

ANSIBLE

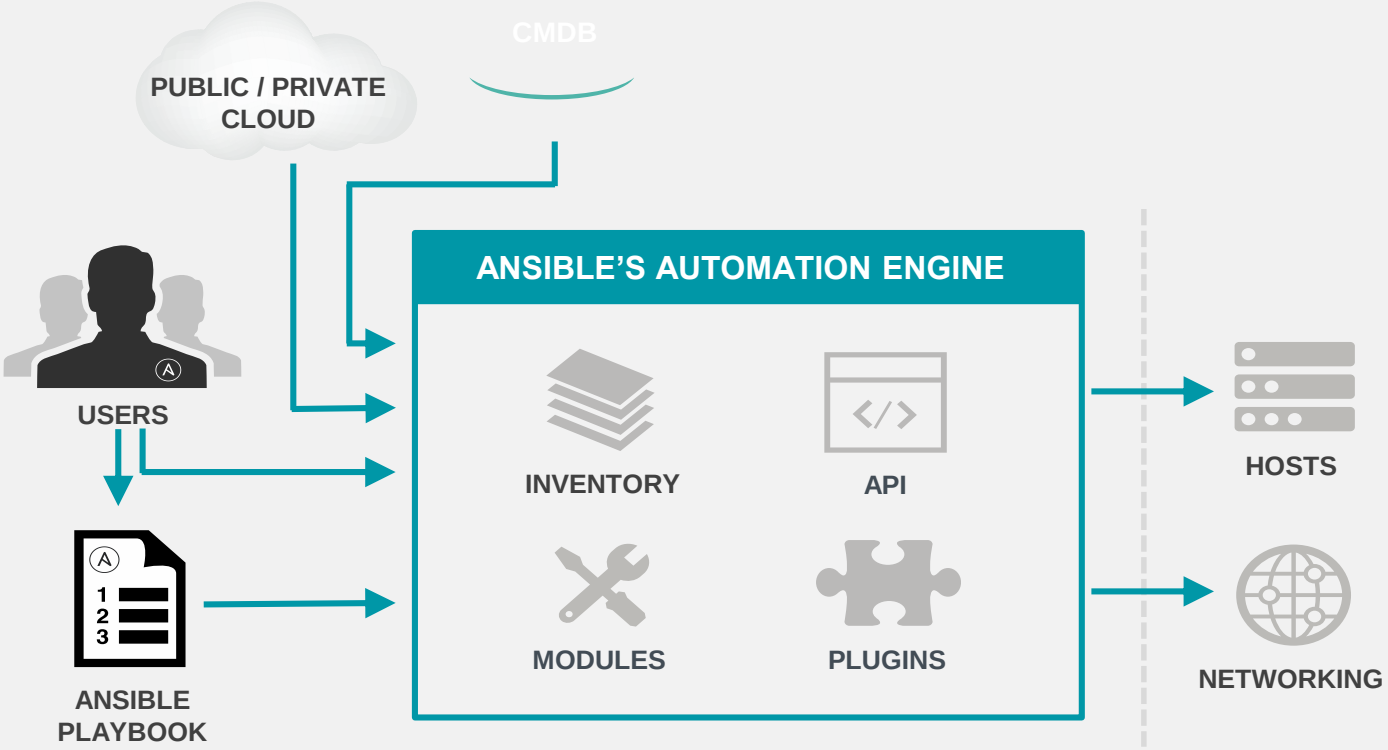
It's a **simple automation language** that can perfectly describe an IT application infrastructure in Ansible Playbooks.

It's an **automation engine** that runs Ansible Playbooks.

Ansible Tower is an **enterprise framework** for controlling, securing and managing your Ansible automation with a **UI and restful API.**



How Ansible works



More about Ansible

- Agentless → minimum footprint
- Make repetitive tasks easy
- Defines a desired state, so OK to run playbook more than once
- Easier to use than writing shell scripts

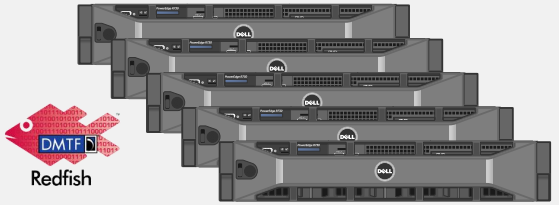
Ansible use cases

- Infrastructure Automation
  - Networking
  - Containers
  - Code Deployment
  - Server / Bare Metal
- Want more information? Visit www.ansible.com and docs.ansible.com

Ansible + Redfish + iDRAC together!

Scalable & Automated Out-of-band management

https:/ / <idrac-ip>/ redfish/ v1/ Managers/ iDRAC.Embedded.1/ Logs/ Sel



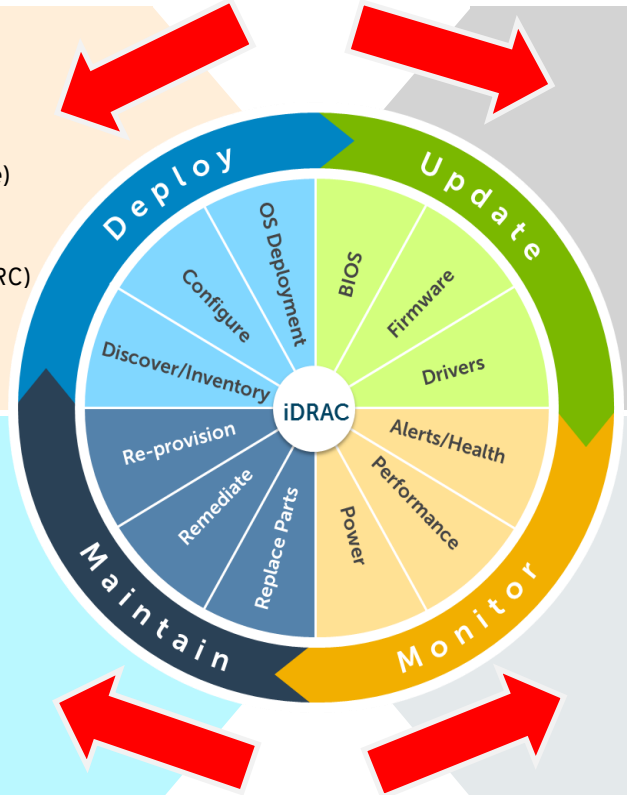
More than 5 servers

Database

Server data

Key Lifecycle Management tasks

- Server Power On/Off; Reboot; Hard Reset
- Install BIOS, Configure BIOS, Reset to Default
- Configure iDRAC (CRUD operations):
  - User & Password Management
  - Certificate Management (import, export, delete)
  - Network Configuration
  - NTP and Time Zone settings
  - Storage (RAID, Virtual Disk, Physical Disks, PERC)
- System Inventory – H/W, Firmware, Sensor
- OS Deployment – remote file share, vMedia
- Import / Export SCP – remote file share, vMedia



- Upgrade using DSU (Dell Server Update) or DUEC (Dell Update Engine for Consoles)
  - Get list of available and applicable updates
  - Firmware Upgrade
  - BIOS Upgrade
  - OS Drivers Upgrade

- Job Management
  - Check JOB status
  - Create JOB
  - Delete JOB
  - Create JOB Queue
  - Delete JOB Queue
- Get Logs
  - Export LC logs
  - Export System Event Logs

- Backup and Restore
  - Server Profiles

Ansible module for iDRAC

- Manage your entire Dell EMC IT infrastructure (servers, routers, switches, storage) from your Ansible Controller.
- Automated monitoring, provisioning, firmware updates at scale.
- Open source, so you can write your own extensions as needed and contribute back to the community.
- Working with Red Hat to include as core Ansible module.

Implementation (playbook)

idrac.yml

```
---
- hosts: myhosts
  name: PowerEdge iDRAC
  gather_facts: False

  # Here we define global variables, but if some servers have different
  # credentials, then place these variables in /etc/ansible/hosts to override
  # for each host
  vars:
     idracuser: root
     idracpswd: calvin

  # Choices available:
  #    Health            Get server health
  #    Model             Get server model
  #    BiosVersion       Get BIOS version
  #    AssetTag          Get asset tag
  #    Memory            Get system memory (GB)
  #    CPU               Get CPU model
  #    PowerRead         Get power consumed (watts)
  #    Selog             Get SELogs

  tasks:

  - include: get_data.yml action={{item}}
    vars:
      json_file: /root/{{host}}-{{action}}
    with_items:
      - Health
      - Model
      - BiosVersion
      - AssetTag
      - Memory
      - CPU
      - PowerRead
      - Selog
```

getdata.yml

```
---

  - name: "Get data: {{ action }}"
    local_action: >
       idrac choice={{ action }} idracuser={{ idracuser }}
            idracpswd={{ idracpswd }} idracip={{ idracip }}
    register: result

  - name: Print inventory file
    local_action: copy content={{ result | to_nice_json }}
                       dest={{ json_file }}

  # Using simple JSON parser jq (https://stedolan.github.io/jq/) to further
  # simplify output. Uncomment once you have it installed.
  - local_action: shell jq .result {{ json_file }} > {{ json_file }}.final
  - local_action: file path={{ json_file }} state=absent
```

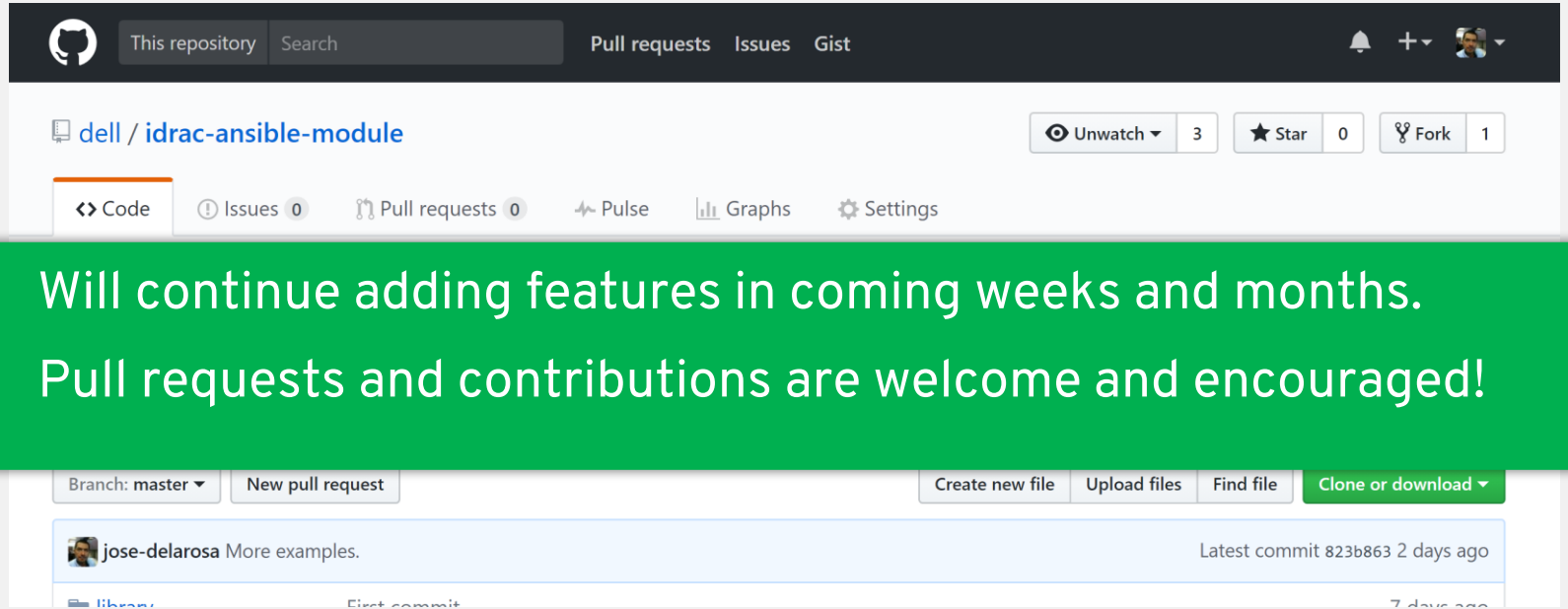
Server inventory

Use case: collect inventory data, maintain in spreadsheet or database

Server	iDRAC IP	Model	IP address	BIOS	CPU	Type	RAM	Service Tag	Status
webserver-1	192.168.2.10	PowerEdge R630	10.0.1.30	2.3.4	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	128	5WT4Q47	OK
webserver-2	192.168.2.11	PowerEdge R630	10.0.1.31	2.3.4	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	128	5XR7Q32	OK
webserver-3	192.168.2.12	PowerEdge R630	10.0.1.33	2.3.2	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	128	5XR7QYY	OK
appserver-1	192.168.2.13	PowerEdge R830	10.0.1.34	2.3.2	4	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.60GHz	512	5XR7QYY	OK
dbserver-1	192.168.3.10	PowerEdge R730	10.0.2.30	2.1.2	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz	256	5XR7Q67	OK
dbserver-2	192.168.3.11	PowerEdge R730	10.0.2.31	2.3.4	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz	256	5WT4Q37	OK
dbserver-3	192.168.3.12	PowerEdge R730	10.0.2.32	2.3.4	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz	256	5WR4Q12	OK
dbserver-4	192.168.3.13	PowerEdge R730	10.0.2.33	2.3.4	2	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz	256	5TT1Q44	OK

Source code repository

- https://github.com/dell/idrac-ansible-module



Resources

- iDRAC with Lifecycle Controller: http://dell.to/2qdBd0y
- Redfish API specification: https://www.dmtf.org/standards/redfish
- Dell EMC PowerEdge Redfish API Overview: http://dell.to/2odsH1p
- iDRAC Redfish API Reference Guide: http://dell.to/2oyjMTy
- Getting started with Ansible: http://docs.ansible.com/ansible/intro_getting_started.html

Q & A

Backup

iDRAC operation APIs

Dell Redfish API URLs	Comments
/redfish/v1/Managers	
/redfish/v1/Managers/iDRAC.Embedded.1	
/redfish/v1/Managers/iDRAC.Embedded.1/Actions/Manager.Reset	Used to perform iDRAC reset
/redfish/v1/Managers/iDRAC.Embedded.1/NetworkProtocol	Reports information about iDRAC's network services. Includes Web server, SNMP, vMedia, Telnet, SSH, IPMI & KVM.
/redfish/v1/ Managers/iDRAC.Embedded.1/SerialInterfaces	iDRAC BMC serial interface
/redfish/v1/ Managers/iDRAC.Embedded.1/SerialInterfaces/<Serial-key>	
/redfish/v1/Managers/iDRAC.Embedded.1/LogServices	
/redfish/v1/Managers/iDRAC.Embedded.1/LogServices/Sel	Access to server System Event Log
/redfish/v1/Managers/iDRAC.Embedded.1/LogServices/Lclog	Access to Lifecycle Controller Log
/redfish/v1/Managers/iDRAC.Embedded.1/LogServices/Sel/Actions/LogService.ClearLog	Used to clear LC Log
/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia	Status of iDRAC virtual media
/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/<media-type>	
/redfish/v1/Managers/iDRAC.Embedded.1/EthernetInterfaces	iDRAC network interface
/redfish/v1/Managers/iDRAC.Embedded.1/EthernetInterfaces/<FQDD>	
/redfish/v1/Managers/iDRAC.Embedded.1/AccountService	
/redfish/v1/Managers/iDRAC.Embedded.1/Accounts	iDRAC user accounts
/redfish/v1/Managers/iDRAC.Embedded.1/Accounts/<Account-Id>	

Chassis inventory APIs

Dell Redfish API URLs	Comments
/redfish/v1/Chassis	
/redfish/v1/Chassis/System.Embedded.1	Top-level URI for server chassis
/redfish/v1/Chassis/System.Embedded.1/Thermal	
/redfish/v1/Chassis/System.Embedded.1/Sensors/Fans	Reports fan status for server and FX2 chassis
/redfish/v1/Chassis/System.Embedded.1/Sensors/Fans/<Fan-FQDD>	
/redfish/v1/Chassis/System.Embedded.1/Sensors/Temperatures	Reports thermal data for server and FX2 chassis
/redfish/v1/Chassis/System.Embedded.1/Sensors/Temperatures/<Sensor-FQDD>	<Sensor-FQDD> addresses each temperature probe
/redfish/v1/Chassis/System.Embedded.1/Power	Power consumption and supply status
/redfish/v1/Chassis/System.Embedded.1/Power/PowerControl	
/redfish/v1/Chassis/System.Embedded.1/Sensors/Voltages	
/redfish/v1/Chassis/System.Embedded.1/Sensors/Voltages/<Voltage-FQDD>	<Voltage-FQDD> addresses each voltage output
/redfish/v1/Chassis/System.Embedded.1/Power/PowerSupplies	
/redfish/v1/Chassis/System.Embedded.1/Power/PowerSupplies/<PSU-FQDD>	<PSU-FQDD> addresses each power supply
/redfish/v1/Chassis/System.Embedded.1/Power/Redundancy/<PSRedundancy-FQDD>	<PSRedundancy-FQDD> addresses power supply redundancy

System status APIs

Dell Redfish API URLs	Comments
/redfish/v1	Top-level API access
/redfish/v1/Systems	Server inventory and status information access
/redfish/v1/Systems/<ServiceTag+nodeid>	
/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset	Server reset operation
/redfish/v1/Systems/System.Embedded.1/Processors	Details on CPUs
/redfish/v1/Systems/System.Embedded.1/Processors/<Processor-FQDD>	
/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces	Reports NIC IP address, DHCP and DNS information.
/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces/<EthernetInterface-FQDD>	Example <EthernetInterface-FQDD> = NIC.Embedded.1-1-1
/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces/<EthernetInterface-FQDD>/Vlans	
/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces/<EthernetInterface-FQDD>/Vlans/<Vlan-FQDD>	
/redfish/v1/Systems/System.Embedded.1/Storage/Controllers	
/redfish/v1/Systems/System.Embedded.1/Storage/Controllers/<Controller-FQDD>	Typical <Controller-FQDD>=RAID.Slot.N-1; describes details of controller, backplane, enclosure, attached drives

Registries, Sessions, Tasks and Event APIs

Dell Redfish API URLs	Comments
/redfish/v1/Registries/Messages/En	PowerEdge message registry
/redfish/v1/odata	Enables OData clients to navigate iDRAC Redfish resources
/redfish/v1/$metadata	Provides a metadata document describing the resources and collections that are available at the iDRAC Redfish service root URI
/redfish/v1/$metadata#<Collection or a single resource>	
/redfish/v1/JSONSchemas	Schema descriptions for all supplied data
/redfish/v1/JSONSchemas/<file>	
/redfish/v1/SessionService	Redfish session management
/redfish/v1/Sessions	
/redfish/v1/Sessions/<SessionId>	
/redfish/v1/TaskService	Redfish internal task management
/redfish/v1/EventService	Redfish event management
/redfish/v1/EventService/Actions/EventService.SubmitTestEvent	
/redfish/v1/EventSubscriptions	
/redfish/v1/EventSubscriptions/<Subscription ID>	

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

LEARN. NETWORK.
EXPERIENCE
OPEN SOURCE.