Make It Real
DIGITAL TRANSFORMATION
Using Ansible and Redfish to automate systems management

Jose Delarosa
May 9, 2018
Before we start

• Thank you for coming to this session
• Please ask questions: It’s OK to interrupt
• If time runs out, happy to talk to you afterwards
Who am I?

- Linux Engineer
- Part time technology evangelist
- Part-time systems engineer
- Part-time developer
- @jdelaros1
Why are we all here?

• Wrote some code using some really cool tools that will make your life easier.
• If you manage servers (i.e. sysadmin, SRE) in a lab or data center, this is for you.
• If you are an open source developer in IT, this is for you.
• If you like experimenting with new tools, this is definitely for you!
Motivation

• I had a need to scale OOB management
• I had a need to automate OOB management
• Needed to be open source
• Could have used shell scripting for some of it, but wanted something different
Agenda

1. Out-of-Band Management
2. Redfish (scalability)
3. Ansible (automation)
4. Scalability + Automation =
Out-of-Band Management Overview
What is Out-of-Band (OOB) management?

• Server management independent of the server’s operating system and main power
• Provided by an embedded chip, has its own Ethernet port, usually connected to a separate management network
• Goes by many names: iDRAC, iLO, IMM, BMC
• Management includes:
  – Device inventory
  – Hardware failure detection
  – System event logs
  – BIOS configuration
Login
Dashboard

System Health
- Batteries
- CPUs
- Cooling
- Intrusion
- Memory
- Power Supplies
- Removable Flash Media
- Voltages
- Miscellaneous

System Information
- Power State: ON
- Model: PowerEdge R740
- Host Name
- Operating System
- Operating System Version
- Service Tag: BPQDHL2
- BIOS Version: 1.0.7
- iDRAC Firmware Version: 3.00.00.00
- iDRAC MAC Address: f8:ca:b6:ff:b5:ea

Virtual Console

Launch Virtual Console
Hard Drives
## Thermal

### Temperatures

#### Temperature Probes

<table>
<thead>
<tr>
<th>Status</th>
<th>Probe Name</th>
<th>Reading</th>
<th>Min</th>
<th>Max</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>CPU1 Temp</td>
<td>29 °C (84.2 °F)</td>
<td>N/A</td>
<td>N/A</td>
<td>3 °C (37.4 °F)</td>
<td>89 °C (192.2 °F)</td>
</tr>
<tr>
<td>✔️</td>
<td>CPU2 Temp</td>
<td>28 °C (82.4 °F)</td>
<td>N/A</td>
<td>N/A</td>
<td>3 °C (37.4 °F)</td>
<td>89 °C (192.2 °F)</td>
</tr>
<tr>
<td>✔️</td>
<td>System Board Exhaust Temp</td>
<td>29 °C (84.2 °F)</td>
<td>8 °C (46.4 °F)</td>
<td>75 °C (167 °F)</td>
<td>3 °C (37.4 °F)</td>
<td>80 °C (176 °F)</td>
</tr>
<tr>
<td>✔️</td>
<td>System Board Inlet Temp</td>
<td>26 °C (78.8 °F)</td>
<td>3 °C (37.4 °F)</td>
<td>43 °C (109.4 °F)</td>
<td>-7 °C (19.4 °F)</td>
<td>47 °C (116.6 °F)</td>
</tr>
</tbody>
</table>
## System Event Logs

### System Event Log

**Instructions:** The System Event Log contains information about the managed system. To sort the log by column, click a column header.

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>The process of installing an operating system or hypervisor is successfully completed.</td>
<td>Tue 26 Sep 2017 14:31:50</td>
</tr>
<tr>
<td>✔️</td>
<td>The process of installing an operating system or hypervisor is started and is in progress.</td>
<td>Tue 26 Sep 2017 14:21:16</td>
</tr>
<tr>
<td>✔️</td>
<td>The input power for power supply 2 has been restored.</td>
<td>Tue 26 Sep 2017 13:33:06</td>
</tr>
<tr>
<td>✔️</td>
<td>The power supplies are redundant.</td>
<td>Tue 26 Sep 2017 13:33:05</td>
</tr>
<tr>
<td>⚠️</td>
<td>Power supply redundancy is lost.</td>
<td>Tue 26 Sep 2017 13:33:00</td>
</tr>
<tr>
<td>⚠️</td>
<td>The power input for power supply 2 is lost.</td>
<td>Tue 26 Sep 2017 13:32:53</td>
</tr>
<tr>
<td>✔️</td>
<td>Log cleared.</td>
<td>Sat 16 Sep 2017 10:37:59</td>
</tr>
</tbody>
</table>
Simple OOB Management

1. Not scalable

2. Not automated
Redfish Overview
Redfish Overview

• Open source, open industry standard specification published by the DMTF for hardware management.

• Provides a RESTful API used to obtain information and exert control over servers via an OOB controller.

• Built on a modern tool-chain which includes HTTPS, JSON and the OData standard.

• A Redfish request is sent as an URI, so a client could be any application on a server, workstation or mobile device.
What can we do with Redfish?

- Retrieve server health status
- Retrieve hardware and firmware inventory
- Power up, power down, warm boot, cold boot
- Change BIOS settings
- Change boot settings
- Configure OOB controller (i.e. users, network settings)
- Configure RAID
- Firmware updates

https://www.dmtf.org/standards/redfish
Example: System Health

```bash
{
  "Health": "OK",
  "HealthRollUp": "OK"
}
```
Example: Hard Drives

```bash
$ curl https://<OOB>/redfish/v1/Systems/System.Embedded.1/Storage/Controllers/RAID.Slot.4-1
   --user root:password | jq .Devices
```

```json
[
  {
    "CapacityBytes": 599550590976,
    "Manufacturer": "SEAGATE",
    "Model": "ST600MM0238",
    "Name": "Physical Disk 0:1:0",
    "Status": {
      "Health": "OK",
      "HealthRollup": "OK"
    }
  },
  {
    "CapacityBytes": 599550590976,
    "Manufacturer": "SEAGATE",
    "Model": "ST600MM0238",
    "Name": "Physical Disk 0:1:1",
    "Status": {
      "Health": "OK",
      "HealthRollup": "OK"
    }
  }
]
```
Example: Thermal


```
{
  "name": "CPU1 Temp",
  "readingCelsius": 29,
  "health": "OK"
}
{
  "name": "CPU2 Temp",
  "readingCelsius": 28,
  "health": "OK"
}
{
  "name": "System Board Exhaust Temp",
  "readingCelsius": 29,
  "health": "OK"
}
..
Example: System Event Logs


--- snip ---

```json
{
  "date": "2017-09-26T13:33:00-05:00",
  "message": "Power supply redundancy is lost.",
  "severity": "Critical"
}
{
  "date": "2017-09-26T13:32:53-05:00",
  "message": "The power input for power supply 2 is lost.",
  "severity": "Critical"
}
{
  "date": "2017-09-16T10:37:59-05:00",
  "message": "Log cleared.",
  "severity": "Ok"
}
```
Redfish API tree structure

- `/redfish/v1/Systems`
  - Logical system view
  - `/redfish/v1/Systems/<id>`
    - General system status, Disks, NICs, PSUs Fans, Processors

- `/redfish/v1/Chassis`
  - Chassis global inventory
  - `/redfish/v1/Chassis/<id>`
    - Fans, storage, NICs, Pwr, Thermal

- `/redfish/v1/Managers`
  - iDRAC operations
  - `/redfish/v1/Managers/<id>`
    - IDRAC or BMC settings
      - Virtual media, Firmware, Logfiles

- `/redfish/v1/EventService`
  - Alerting
  - `/redfish/v1/EventService`
    - Subscription to be notified in case of specific events

- `/redfish/v1/Sessions`
  - iDRAC session functionality
  - `/redfish/v1/Sessions`
    - View / Create / Delete iDRAC sessions

- `/redfish/v1/TaskService`
  - Running tasks
  - `/redfish/v1/TaskService`
    - Task operations
## System APIs

<table>
<thead>
<tr>
<th>Redfish API URIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>/redfish/v1</td>
</tr>
<tr>
<td>/redfish/v1/Systems</td>
</tr>
<tr>
<td>/redfish/v1/Systems/&lt;ServiceTag+nodeid&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Bios</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/BootSources</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Processors</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Processors/&lt;Processor-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces/&lt;EthernetInterface-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/EthernetInterfaces/&lt;EthernetInterface-FQDD&gt;/Vlans</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Storage/Controllers</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Power/PowerSupplies</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/SecureBoot</td>
</tr>
<tr>
<td>/redfish/v1/Systems/System.Embedded.1/Sensors/Fans</td>
</tr>
<tr>
<td>Redfish API URIs</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>/redfish/v1/Chassis</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Thermal</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Fans</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Fans/&lt;Fan-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Temperatures</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Temperatures/&lt;Sensor-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Power</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Power/PowerControl</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Voltages</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Sensors/Voltages/&lt;Voltage-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Power/PowerSupplies</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Power/PowerSupplies/&lt;PSU-FQDD&gt;</td>
</tr>
<tr>
<td>/redfish/v1/Chassis/System.Embedded.1/Power/Redundancy/&lt;PSRedundancy-FQDD&gt;</td>
</tr>
</tbody>
</table>
More than just GET!

Example: Reboot Server

$ curl https://<OOB>/redfish/v1/Systems/System.Embedded.1/Actions/ComputerSystem.Reset \
   --request POST \
   --header "Content-Type: application/json" \
   --data '{"ResetType":"GracefulRestart"}' \
   --user root:password

Example: Change boot mode to UEFI

$ curl https://<OOB>/redfish/v1/Systems/System.Embedded.1/Bios/Settings \
   --request PATCH \
   --header "Content-Type: application/json" \
   --data '{"Attributes":{"BootMode":"Uefi"}}' \
   --user root:password
Redfish Roadmap

• Version 1.x focused on servers. Will expand to cover storage and network infrastructure.

• Will expand APIs over time to cover new technologies such as NVDIMMs and Multifunction Network Adapters.

• SNIA is developing ‘Swordfish’ to address advanced storage devices.

• DMTF expanding open source efforts (http://github.com/dmtf)
  – Client libraries (Python, Java, PowerShell)
  – Redfish Mockup Creator / Server
  – RedfishTool (CLI utility similar to ipmitool)
Redfish provides scalability to OOB management

https://<OOB>/redfish/v1/Systems/Systems.Embedded.1

1

Management
Network

2

{ "Health": "OK",
   "HealthRollup": "OK"
}
Ansible Overview
Ansible 101

- Automation software → makes repetitive tasks easy
- Agentless → minimum footprint
- No database backend → easy to install
- Remote tasks are run in parallel → fast & efficient
- Easier to learn and use than shell scripts
Ansible use cases

OpenStack
- Compute nodes
- Storage nodes
- Controller nodes

IT Security Hardening
- Firewall rules
- Remove unused login IDs
- Install updates

Container Management
- Stop/remove containers
- Refresh container images
- Deploy with new images

✓ 1-to-n management
✓ Executes tasks in parallel
Ansible concepts

- **Task:** A task is the smallest unit of work. Examples: “install a package”, “remove a user”, “create firewall rule” or “copy a file to this directory”.

- **Play:** A play is made up of tasks. Example: the play “*Prepare a database*” is composed of tasks:
  - Task 1: “Install the database package”
  - Task 2: “Set password”
  - Task 3: “Create database”

- **Playbook:** A playbook is composed of plays. Example: the playbook “*Setup my web application*” has plays 1) “Set up database server” and 2) “Set up web server”.

---

<table>
<thead>
<tr>
<th>Playbook: Setup my web application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Play 1: Setup database</strong></td>
</tr>
<tr>
<td>Task 1: Install mysql package</td>
</tr>
<tr>
<td>Task 2: Create database ‘customer_db’</td>
</tr>
<tr>
<td><strong>Play 2: Setup web server</strong></td>
</tr>
<tr>
<td>Task 1: Install httpd package</td>
</tr>
<tr>
<td>Task 2: Configure site for TLS</td>
</tr>
</tbody>
</table>
Simple implementation example

Say you provision 100 servers every day and you run these commands in each server:

```
$ groupadd admin
$ useradd -c "Sys Admin" -g admin -m sysman
$ mkdir /opt/tools
$ chmod 755 /opt/tools
$ chown sysman /opt/tools
$ yum -y install httpd
$ yum -y update
$ systemctl enable httpd
$ systemctl start httpd
$ rm /etc/motd
```

The same commands can be placed in an Ansible playbook and executed in 100 servers:

```
daily_tasks.yml
- name: daily tasks
  hosts: my_100_daily_servers
  tasks:
  - group: name=admin state=present
  - user: name=sysman comment="Sys Admin" group=admin
  - file: path=/opt/tools state=directory owner=sysman mode=0755
  - yum: name=httpd state=latest
  - yum: name=* state=latest
  - service: name=httpd state=started enabled=yes
  - file: path=/etc/motd state=absent
```

```
$ ansible-playbook daily_tasks.yml
```
Ansible module

- Whereas a playbook is where you specify the tasks to run; a module is the code to implement those tasks.
- Modules can be written in any language, but most popular is Python.
- If you are a system administrator, you will work mostly with playbooks.
- If you are a developer, you will work mostly with modules.

```
- name: daily tasks
  hosts: my_100_daily_servers
  tasks:
  - group: name=admin state=present
  - user: name=sysman comment="Sys Admin" group=admin
  - file: path=/opt/tools state=directory owner=sysman mode=0755
  - yum: name=httpd state=latest
  - service: name=httpd state=started enabled=yes
  - file: path=/etc/motd state=absent
```
Scalable and automated OOB management

https://<OOB>/redfish/v1/Systems/Systems.Embedded.1

1

ANSIBLE

Management Network

2

{  
  "Health": "OK",
  "HealthRollup": "OK"
}

3
Coming together:
Ansible module for Redfish
Ansible module for Redfish

• Use it to manage your entire IT infrastructure (compute, network & storage) from one controller.
• Automated inventory, monitoring & provisioning at scale.
• It’s open source, so you can write your own extensions and contribute back to the community.
• Working to submit upstream.
• DMTF will extend it and support it.
Key Lifecycle Management Tasks

- Device Inventory
- iDRAC Configuration
- BIOS Configuration
- Firmware Update
- Event Logs
- Health Reporting
- Power Management
https://github.com/dell/redfish-ansible-module

Pull requests accepted
Example: get system inventory

1. Playbook

```yaml
---
- hosts: myhosts
  name: System Inventory
gather_facts: False

tasks:
  - name: Define output file
    include_tasks: create_output_file.yml type=Syst
  - name: Getting system inventory
    local_action: >
      redfish [category=Inventory_command=GetSystem]
    baseurl={{baseurl}} user={{user}} password={{password}}
    register: result
  - name: Copy results to output file
    local_action: copy content={{ result | to_nice
    dest={{template}}}.json
```

2. Execute Playbook

```
$ ansible-playbook system-inventory.yml
PLAY [PowerEdge IDRAC Get System Inventory]
 ************************************
TASK [Define timestamp]                        
ok: [r740-1]
ok: [r630]
ok: [r648-1]

TASK [Define file to place results]          
ok: [r630]
ok: [r648-1]
ok: [r740-1]

TASK [Create dropoff directory for host]     
changed: [r740-1 -> localhost]
changed: [r630 -> localhost]
changed: [r648-1 -> localhost]

TASK [Getting system inventory]              
ok: [r740-1 -> localhost]
ok: [r648-1 -> localhost]
ok: [r630 -> localhost]

TASK [Copying results to file]              
changed: [r630 -> localhost]
changed: [r740-1 -> localhost]
changed: [r648-1 -> localhost]

PLAY RECAP
 ************************************
r630: ok=5  changed=2  unreachable=0
r648-1: ok=5  changed=2  unreachable=0
r740-1: ok=5  changed=2  unreachable=0
```

3. Result

```
```
```json
{
  "changed": false,
  "result": {
    "AssetTag": "",
    "BiosVersion": "1.0.7",
    "BootSourceOverrideMode": "UEFI",
    "CpuCount": 2,
    "CpuHealth": "OK",
    "CpuModel": "Intel(R) Xeon(R) Silver 4108 CPU",
    "HostName": "",
    "Manufacturer": "Dell Inc.",
    "MemoryHealth": "OK",
    "MemoryTotal": 128.0,
    "Model": "PowerEdge R640",
    "PartNumber": "088R9MA02",
    "PowerState": "On",
    "SerialNumber": "CNIW08B347",
    "ServiceTag": "3M92",
    "Status": "OK",
    "SystemType": "Physical"
  }
}
```
### Example: Inventory spreadsheet

<table>
<thead>
<tr>
<th>Server</th>
<th>iDRAC IP</th>
<th>Model</th>
<th>IP address</th>
<th>BIOS</th>
<th>CPU</th>
<th>Type</th>
<th>RAM</th>
<th>Service Tag</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>webserver-1</td>
<td>192.168.2.10</td>
<td>PowerEdge R630</td>
<td>10.0.1.30</td>
<td>2.3.4</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
<td>128</td>
<td>5W14Q52</td>
<td>OK</td>
</tr>
<tr>
<td>webserver-2</td>
<td>192.168.2.11</td>
<td>PowerEdge R630</td>
<td>10.0.1.31</td>
<td>2.3.4</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
<td>128</td>
<td>5XXYQ32</td>
<td>OK</td>
</tr>
<tr>
<td>webserver-3</td>
<td>192.168.2.12</td>
<td>PowerEdge R630</td>
<td>10.0.1.33</td>
<td>2.3.2</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz</td>
<td>128</td>
<td>5XT3QYY</td>
<td>OK</td>
</tr>
<tr>
<td>appserver-1</td>
<td>192.168.2.13</td>
<td>PowerEdge R830</td>
<td>10.0.1.34</td>
<td>2.3.2</td>
<td>4</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.60GHz</td>
<td>512</td>
<td>5XR7QXY</td>
<td>OK</td>
</tr>
<tr>
<td>dbserver-1</td>
<td>192.168.3.10</td>
<td>PowerEdge R740</td>
<td>10.0.2.30</td>
<td>1.2.11</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz</td>
<td>256</td>
<td>5XR7Q88</td>
<td>OK</td>
</tr>
<tr>
<td>dbserver-2</td>
<td>192.168.3.11</td>
<td>PowerEdge R740</td>
<td>10.0.2.31</td>
<td>1.1.7</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz</td>
<td>256</td>
<td>5WEYQ37</td>
<td>OK</td>
</tr>
<tr>
<td>dbserver-3</td>
<td>192.168.3.12</td>
<td>PowerEdge R740</td>
<td>10.0.2.32</td>
<td>1.2.11</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.33GHz</td>
<td>256</td>
<td>5WR4Q12</td>
<td>Fail</td>
</tr>
<tr>
<td>dbserver-4</td>
<td>192.168.3.13</td>
<td>PowerEdge T640</td>
<td>10.0.2.33</td>
<td>1.1.3</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.33GHz</td>
<td>512</td>
<td>5TEEQ21</td>
<td>OK</td>
</tr>
<tr>
<td>dbserver-5</td>
<td>192.168.3.14</td>
<td>PowerEdge T640</td>
<td>10.0.2.34</td>
<td>1.2.11</td>
<td>2</td>
<td>Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.33GHz</td>
<td>512</td>
<td>5TT1Q26</td>
<td>OK</td>
</tr>
</tbody>
</table>
Example: Set controller’s NTP server

1. Playbook

```yaml
---
- hosts: myhosts
  name: Set Manager NTP settings
  gather_facts: False

  vars:
    - ntpserver1: ntp.domain.com

  tasks:
    - name: Enable NTP
      local_action: >
        redfish category=Manager command=SetAttributes
        user={{user}} password={{password}} baseuri={{baseuri}}
        mgr_attr_name=NTPConfigGroup.1.NTPEnable mgr_attr_value=Enabled
        ignore_errors: yes

    - name: Set NTP server 1
      local_action: >
        redfish category=Manager command=SetAttributes
        user={{user}} password={{password}} baseuri={{baseuri}}
        mgr_attr_name=NTPConfigGroup.1.NTP1 mgr_attr_value={{ntpserver1}}
        ignore_errors: yes

# Add more NTP servers as needed
# To get exact attributes names, run the getattributes task first
```

2. Execute Playbook

```bash
$ ansible-playbook set_manager_ntp.yml
```

```
PUBLISH [Set Manager NTP settings] *****************************************************

TASK [Enable NTP] **********************************************************************
ok: [red1 -> localhost]
ok: [red4 -> localhost]
ok: [red2 -> localhost]
ok: [red3 -> localhost]
fatal: [t620 -> localhost]: FAILED! => {"changed": false, "msg": "Resource not supported"}...ignoring

TASK [Set NTP server 1] ***************************************************************
ok: [red1 -> localhost]
ok: [red4 -> localhost]
ok: [red2 -> localhost]
ok: [red3 -> localhost]
fatal: [t620 -> localhost]: FAILED! => {"changed": false, "msg": "Resource not supported"}...ignoring

PLAY RECAP ***********************************************************************
red1 : ok=2  changed=0  unreachable=0  failed=0
red2 : ok=2  changed=0  unreachable=0  failed=0
red3 : ok=2  changed=0  unreachable=0  failed=0
red4 : ok=2  changed=0  unreachable=0  failed=0
t620 : ok=2  changed=0  unreachable=0  failed=0

Playbook run took 0 days, 0 hours, 0 minutes, 37 seconds
```
https://github.com/dell/redfish-ansible-module

So important I put it twice! 😊
Resources

• Getting started with Ansible: http://bit.ly/2oCj5xy
• PowerEdge Redfish API Overview: http://dell.to/2odsH1p
• iDRAC Redfish API Reference Guide: http://dell.to/2oyjMTy
• jq JSON parser: https://stedolan.github.io/jq/
Conclusion

• Automation + scalability are useful when managing hardware.

• Module tested mostly on Dell EMC platforms, but *should* work on any controller that implements the Redfish API standard.

• Module was designed to be as simple as possible.

• Thank you for listening, hope this was useful.
Thank you

Q & A