Skynet your Infrastructure with QUADS

EuroPython 2017

Will Foster • @sadsfae • hobo.house
Systems Design & Engineering, Red Hat

What do we do? A race car analogy.

- High performance computer servers are race cars.
- High performance networks are race tracks.
- Race car races are performance/scale product testing.
- Race car drivers are performance/scale engineers.
- We are the Pit crew/track engineers.

Track Conditions

There are many races happening all the time with different sets of race cars on different race tracks, scheduled as efficiently as possible for as far out in the future as possible.

QUADS helps us automate and document the scheduling, management and mayhem of the races, race cars and race tracks.

What do we do? (readers digest version)

- <u>2-person DevOps / Sysadmin Team</u>
  - Manage 300+ high-performance servers, switches, infra
    - Accommodates many parallel perf/scale product testing workloads and isolated scale & performance tests
    - Sets of machines, network/VLAN change hands constantly, spin-up / spin-down

- **We have automated our jobs with Python**
- **We spend our time improving the automation instead**

QUADS - What it is (and isn't).

- QUADS is not an installer
- QUADS is not a provisioning system
- QUADS bridges several interchangeable tools together
- QUADS uses Foreman (or something else)
- QUADS can prep systems/networks for OpenStack deployment.
- QUADS helps us automate boring, manual things
- QUADS documents things for us that we might mess up.

Don't build the systems, build the system that builds the systems.

QUADS - What is it?

Set of tools to help automate the scheduling, management and end-to-end provisioning of servers and networks.

- Programmatic, YAML-driven scheduling
- Automated systems provisioning
- Automated network/VLAN provisioning
- Automated documentation
- Automated usage and status generation

QUADS - What Does it Do *(details)*

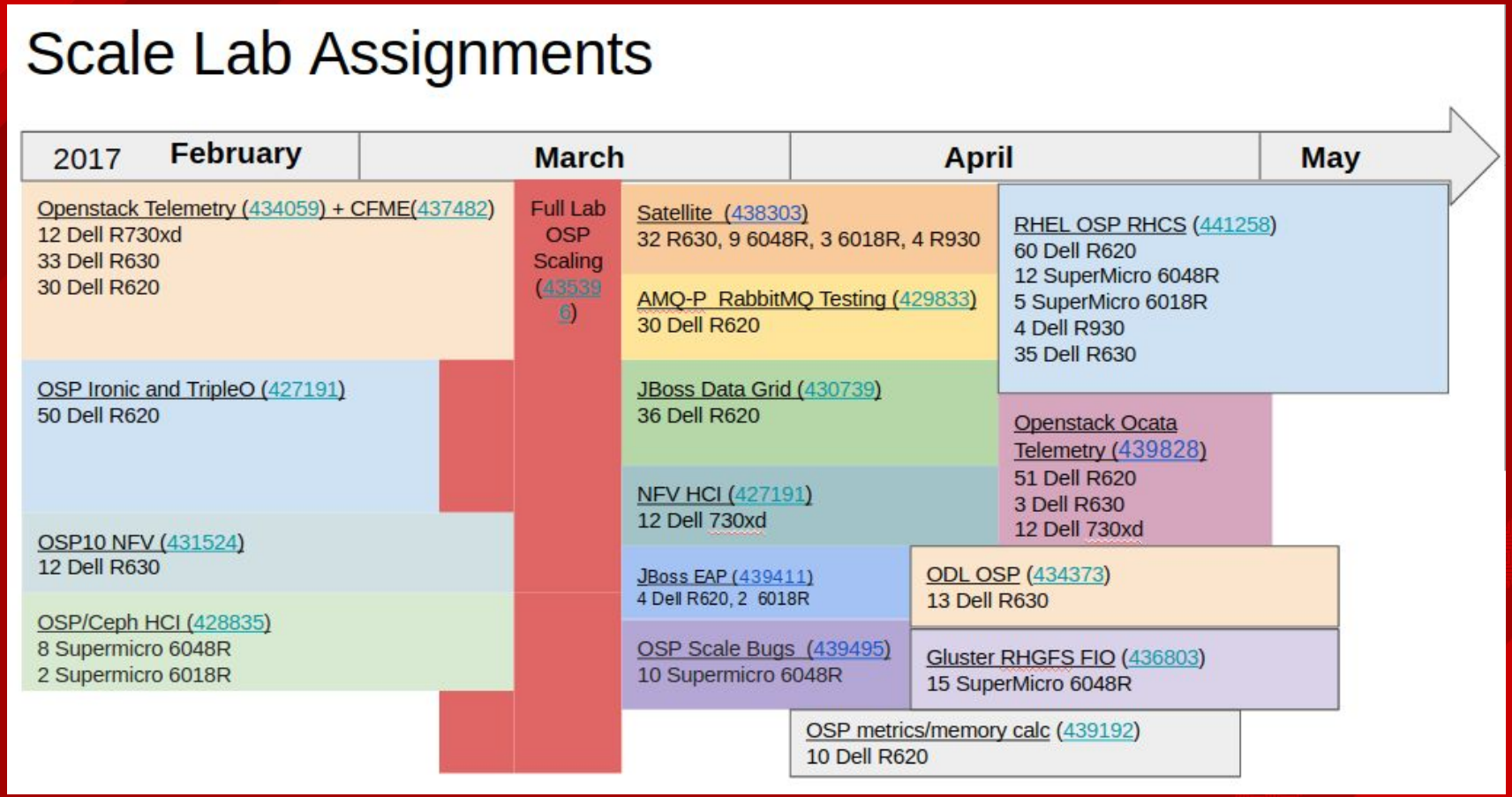
- Create and manage a date/time based YAML schedule for machine allocation and provisioning for unlimited schedules in the future.
- Automate flexible system assignments based on schedules.
- Drive system provisioning and network switch changes based on workload assignments and requirements automatically.
- Generates appropriate instackenv.json for OpenStack environments
- Automated documentation generation published to an internal Wordpress instance via Python API
  - Name, location, macaddr, IP, IPMI, assignment
  - Current workloads and assignments, status, runtime, duration
  - Available and faulty systems

QUADS - How is it used?

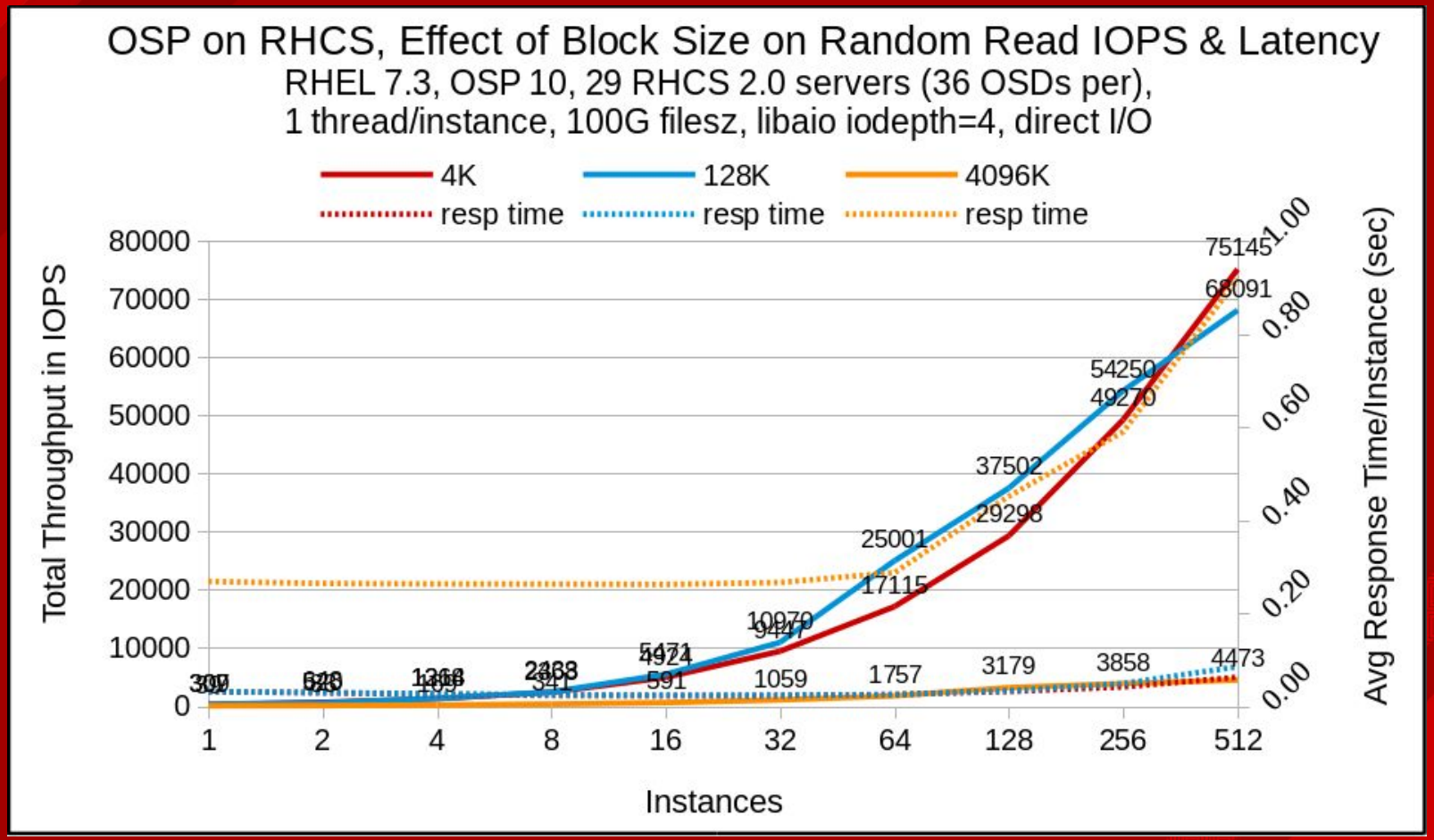
Red Hat Scale Lab

- 300+ node high-performance R&D lab
- Testing/vetting Red Hat and partner products at scale
- Demanding spin-up/down requirements
- Rapid provisioning for short-term usage (maximum 4weeks)
- 16-20+ isolated, parallel systems assignments ongoing 24/7

QUADS - Scheduling in Action



QUADS - Scale Lab Workload Result Example



What problems are we solving?

QUADS - What does it solve?

No more server hugging

- Greater desire for hardware resources than hardware resources
- Clearly defined scheduling for server assignment and reclamation
- Published future schedule and visualizations for planning and audit

QUADS - What does it solve?

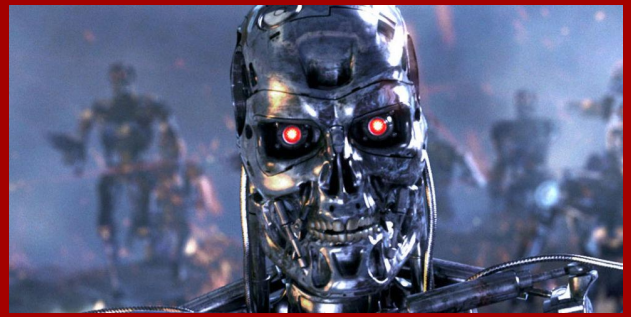
Less human error, more automation.

Give control over to the machines, what's the worst that can happen?

- Automated documentation
- Programmatic scheduling and provisioning
- Automatic network switch changes

QUADS - What does it solve?

Maximize idle machine cycles.



Automated spin-up of machines to run weekend workloads/tests.

- YAML-based scheduling maximizes computing cycles
- Machines power down when not in active use.

QUADS - What does it solve?

More airbnb, less hobo house.

Clearly defined operating guidelines, maximum residency limit.

- Maximum reservation of 4 weeks.
- Uniform hardware specs/sizes
- Must have proven workload at smaller scale elsewhere.

QUADS - What does it solve?

Significant time savings: time is money friend.

What if 100 servers change hands and we did everything manually?

- ~ 90 hours of work
  - Current 2 person team: ~ 45 hours
  - Triple staff: 6 person team: ~ 15 hours (2 working days)
  - Full Ops team: 12 person team: ~ 7.5 hours (1 working day)

QUADS - What does it solve?

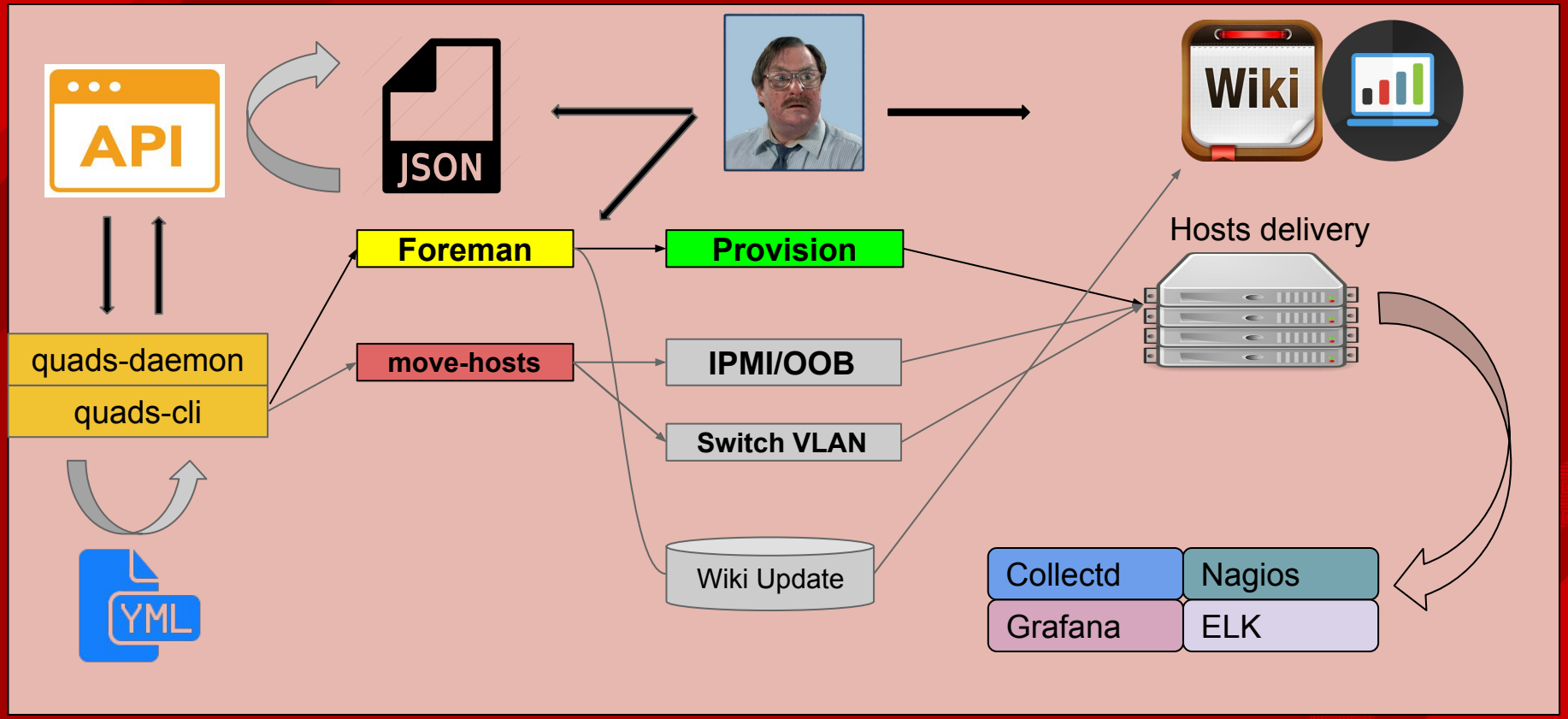
Significant time savings: time is money friend. (continued)

What if 100 servers change hands and we did everything manually?

- Switchport reconfiguration (2 min/port; 4 ports per server)
- Reprovision servers (5 min/server)
- Setup foreman accounts per environment (20 min / environment)
- Reconfigure each server boot order (15 min / server)
- Change iDrac password (1 min / server)
- Update wiki docs / inventory (5 min / server)
- Validate systems / networks (5 min / server)
- Send notifications (15 min / environment)
- Update notifications (15 min / environment)

Let’s look at some pictures. (QUADS arch)

QUADS Scheduler Workflow



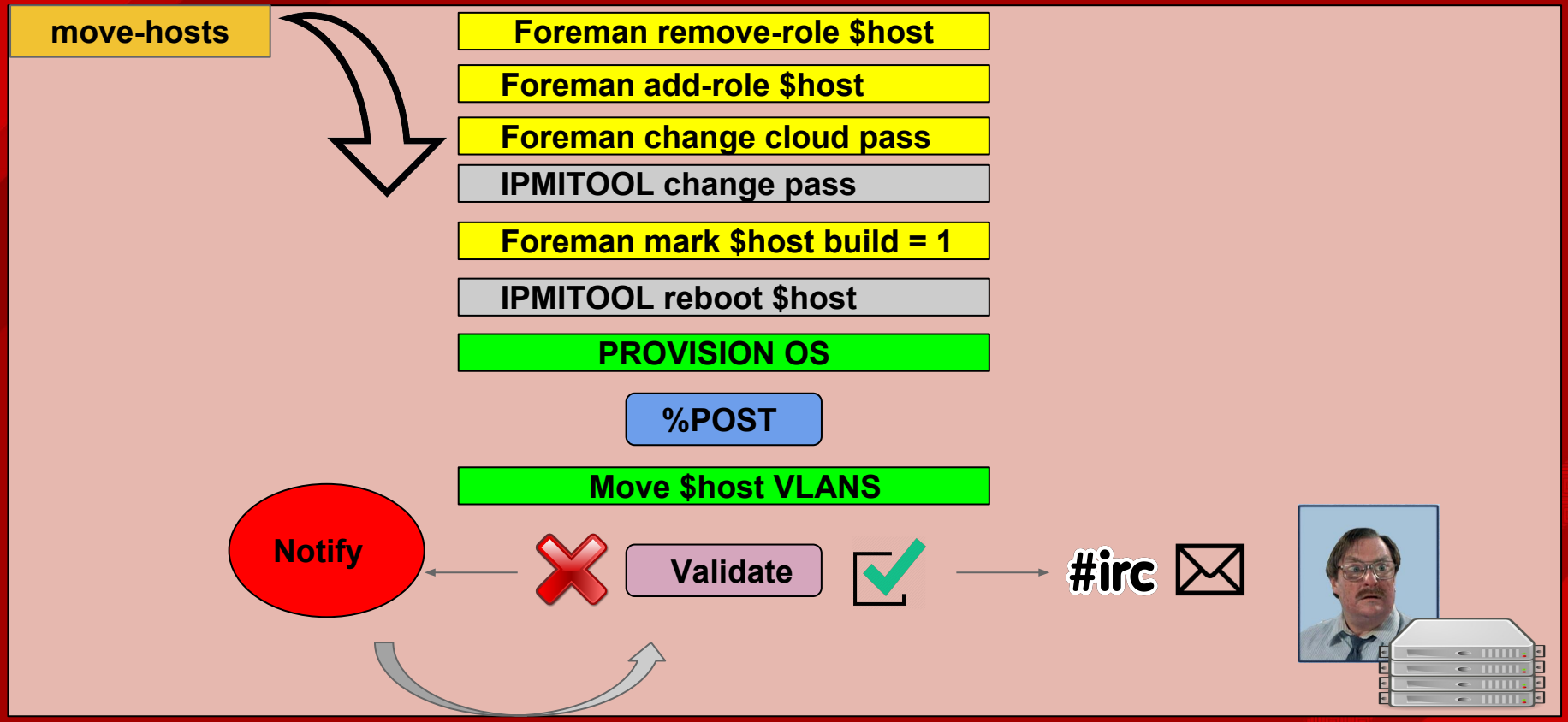
quads --move-hosts

- Execute host migration or provisioning based on schedule
- Only executes an action if one is needed
- Fires off all backend provisioning
  - Use Foreman or plug into an existing provisioning backend

```
quads --move-hosts
```

```
quads.Quads - INFO: Moving c08-h21-r630.example.com from cloud01 to
cloud02
```

QUADS Provisioning (move-hosts)



quads --ls-schedule

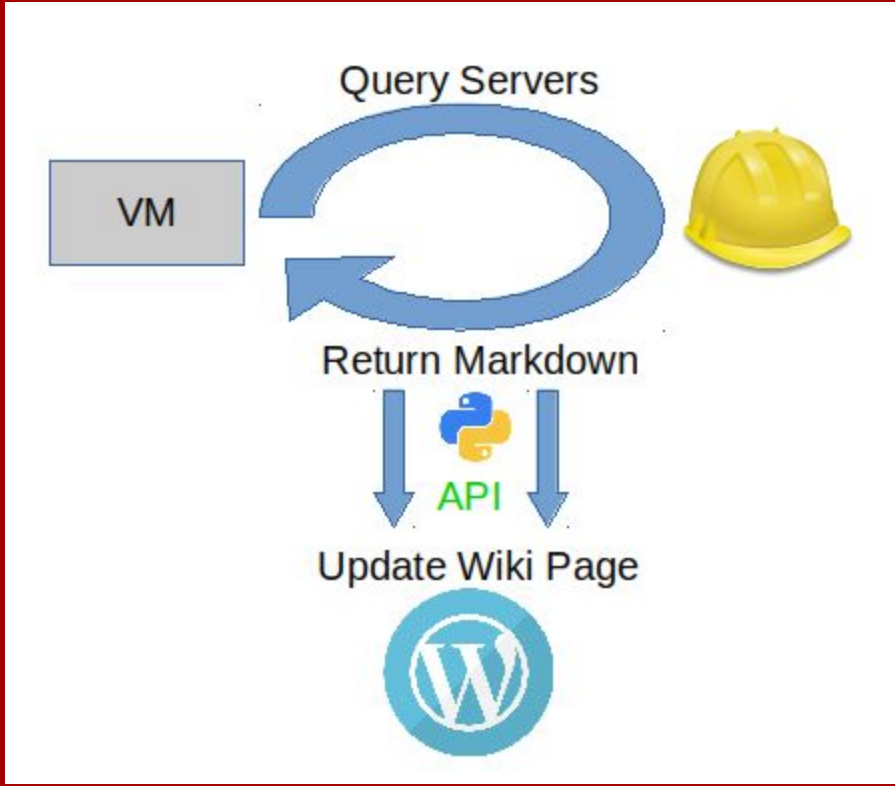
- List scheduling information for a given host

```
quads --ls-schedule --host c08-h21-r630.example.com
```

```
Default cloud: cloud01
Current cloud: cloud02
Current schedule: 5
Defined schedules:
  0| start=2016-07-19 18:00,end=2016-07-20 18:00,cloud=cloud02
  1| start=2016-08-15 08:00,end=2016-08-16 08:00,cloud=cloud02
  2| start=2016-10-12 17:30,end=2016-10-26 18:00,cloud=cloud02
  3| start=2016-10-26 18:00,end=2017-01-09 05:00,cloud=cloud10
  4| start=2017-02-06 05:00,end=2017-02-27 05:00,cloud=cloud05
  5| start=2017-06-28 12:00,end=2017-07-06 05:00,cloud=cloud02
```

Automated Documentation with QUADS

QUADS Dynamic Wiki Auto-generation



QUADS Dynamic Wiki Auto-generation

RDU Scale Lab Dynamic Inventory

Scale Calendar | Support | Docs | Usage | FAQ | Request Tracker | Assignments | QUADS | Monitoring | System Logs

Rack B09

U	ServerHostname	Serial	MAC	IP	IPMIADDR	IPMIURL	IPMIMAC	Workload	Owner	Graph
01	b09-h01-r620	HLPQM02	ec:f4:bb:c1:db:38	10.12.66.225	10.12.66.193	console	f8:bc:12:44:75:dc	cloud06	rbryant	
02	b09-h02-r620	HLPPM02	ec:f4:bb:c1:c7:a8	10.12.66.227	10.12.66.195	console	f8:bc:12:44:75:da	cloud06	rbryant	
03	b09-h03-r620	HLQYL02	ec:f4:bb:c1:ce:88	10.12.66.229	10.12.66.197	console	f8:bc:12:44:70:2a	cloud06	rbryant	
05	b09-h05-r620	6F987Y1	b8:ca:3a:62:81:38	10.12.66.231	10.12.66.199	console	e0:db:55:15:ca:52	cloud06	rbryant	
06	b09-h06-r620	CQWW6X1	b8:ca:3a:5f:ae:d8	10.12.66.233	10.12.66.201	console	e0:db:55:15:ca:e2	cloud06	rbryant	
07	b09-h07-r620	H5SW6X1	b8:ca:3a:5f:ad:c0	10.12.66.235	10.12.66.203	console	e0:db:55:15:ca:72	cloud06	rbryant	
09	b09-h09-r620	G2TW6X1	b8:ca:3a:61:45:38	10.12.66.237	10.12.66.205	console	e0:db:55:15:c8:c2	cloud06	rbryant	
10	b09-h10-r620	1GWW6X1	b8:ca:3a:61:42:20	10.12.66.239	10.12.66.207	console	e0:db:55:14:e2:7a	None		

QUADS Dynamic Wiki - Assignment Summary

assignments

SUMMARY

NAME	SUMMARY	OWNER	REQUEST	INSTACKENV
cloud01	6 (Pool of available servers)	nobody		cloud01
cloud02	9 (OSP Newton Testing)	jtaleric	428270	cloud02
cloud03	5 (Keystone Newton Response/API)	akrzos	427008	cloud03
cloud04	52 (Ceph deployment)	bengland	423424	cloud04
cloud05	7 (Neutron Control Plane)	smalleni	426282	cloud05
cloud06	13 (OVN and OpenStack ML2/OVS)	rbryant	426080	cloud06
cloud07	8 (GlusterFS Scale Testing)	ssaha	426254	cloud07
cloud09	18 (Gnocchi/Telemetry)	akrzos	427010	cloud09
cloud10	58 (Openshift + OSPD testing)	jeder	423401	cloud10

QUADS Dynamic Wiki - Assignment Details

cloud02 : 9 (OSP Newton Testing) — jtaleric

SystemHostname	OutOfBand	DateStartAssignment	DateEndAssignment	TotalDuration	TimeRemaining
b10-h06-r620	console	2016-11-22 16:30	2016-12-16 08:00	23 day(s), 15 hour(s)	8 day(s), 07 hour(s)
b10-h10-r620	console	2016-11-22 16:30	2016-12-16 08:00	23 day(s), 15 hour(s)	8 day(s), 07 hour(s)
b10-h13-r620	console	2016-11-22 16:30	2016-12-16 08:00	23 day(s), 15 hour(s)	8 day(s), 07 hour(s)

QUADS Dynamic Wiki Auto-generation

Unassigned systems

SystemHostname	OutOfBand

Faulty systems

SystemHostname	OutOfBand
b09-h10-r620	console
b09-h13-r620	console
b09-h21-r620	console

QUADS Dynamic Wiki - Calendar View

October 2016

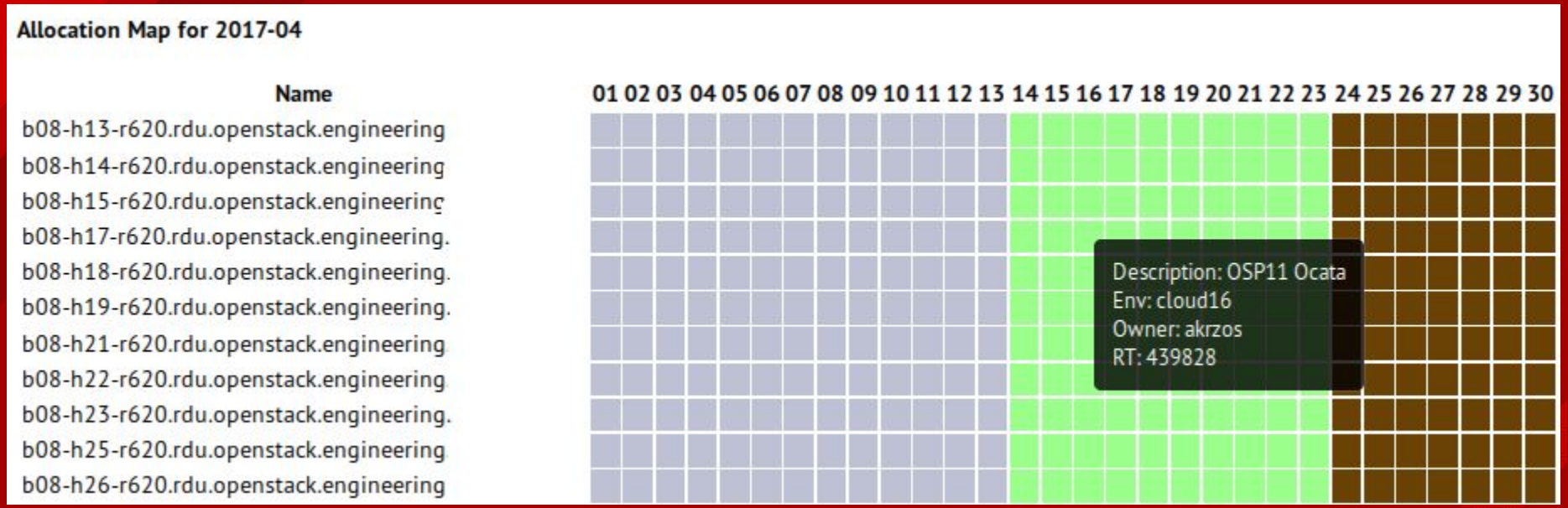
Scale Lab Allocations

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
25 2016-09-25 Scale Lab	26 2016-09-26 Scale Lab	27 2016-09-27 Scale Lab	28 2016-09-28 Scale Lab	29 2016-09-29 Scale Lab	30 2016-09-30 Scale Lab	1 2016-10-01 Scale Lab
2 2016-10-02 Scale Lab	3 2016-10-03 Scale Lab	4 2016-10-04 Scale Lab	5 2016-10-05 Scale Lab	6 2016-10-06 Scale Lab	7 2016-10-07 Scale Lab	8 2016-10-08 Scale Lab
9 2016-10-09 Scale Lab	10 2016-10-10 Scale Lab	11 2016-10-11 Scale Lab	12 2016-10-12 Scale Lab	13 2016-10-13 Scale Lab	14 2016-10-14 Scale Lab	15 2016-10-15 Scale Lab
16 2016-10-16 Scale Lab	17 2016-10-17 Scale Lab	18 2016-10-18 Scale Lab	19 2016-10-19 Scale Lab	20 2016-10-20 Scale Lab	21 2016-10-21 Scale Lab	22 2016-10-22 Scale Lab
23 2016-10-23 Scale Lab	2016-10-2 Scale Lab			27 2016-10-27 Scale Lab	28 2016-10-28 Scale Lab	29 2016-10-29 Scale Lab
30 2016-10-30 Scale Lab	2016-10-31 Scale Lab	2016-11-01 Scale Lab	2016-11-02 Scale Lab	3 2016-11-03 Scale Lab	4 2016-11-04 Scale Lab	5 2016-11-05 Scale Lab

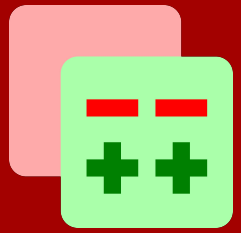
2016-10-17
Location: Scale Lab

cloud01 : 77 (Pool of available servers)
cloud02 : 12 (Small OSPD deployment -- jtaleric)
cloud03 : 0 (OSPD deployment -- dwilson)
cloud04 : 60 (Ceph deployment -- bengland)
cloud05 : 0 (Small OSPD deployment -- jkilpatr)
cloud06 : 0 (Small OSPD deployment -- smalleni)
cloud07 : 10 (Small OSPD deployment -- akrzos)
cloud08 : 0 (Private VLAN testing -- kambiz)
cloud09 : 5 (Keystone OSPD Deployment -- akrzos)
cloud10 : 12 (Openshift + OSPD testing -- jeder)

QUADS Dynamic Wiki - Map Visualization



QUADS Code Review and CI



- Gerrit Code Review
  - IRC: Freenode #quads : announces new reviews

- Jenkins: (automated testing of new gerrit reviews)
  - Votes on all commits in gerrit
    - All CI Tests must pass to get +1
      - Functional tests (quads commands)
      - Shellcheck
      - flake8

Current Status : What's Working?

- Automated system/switch provisioning
- IRC, email notifications and RT ticket integration
- IPMI/out-of-band provisioning
  - HW RAID config
  - User accounts
  - Network Interface ordering, PXE enable/disable
- Wiki Page update/generation
- Calendar and visualization update/generation
- instackenv.json generation per OpenStack cloud
- Post-deployment system automation
- Full CI for every patch via Gerrit / Jenkins
- Foreman UI views for users (self-service)
- JSON API
- Automated network validation

Future Updates: What are we working on?

- Automated IaaS deployment (OpenStack) (RFE #14)
- Flask web interface to scheduler (RFE #86 - 2.0 milestone)
- Per-switchport bandwidth graphing (RFE #86 - 1.1 milestone)
- ~~JSON HTTP API~~
- ~~Automated network validation (RFE #5)~~
- Self-service machine image backups (RFE #84 - 1.1 milestone)
- Automated deployment of OpenStack (RFE #13)
- Ansible-driven facts resource backend (RFE #95)

QUADS Methodology

- Share a pool of hardware across different development groups
  - Cost savings, re-usability

- The QUADS parts may be more useful than the sum of its parts
  - Production bare-metal hardware may not move often but automated documentation is always useful
  - Keep it modular

- Stick to K.I.S.S. principles because they work

- Spend your time building/improving a system that builds the systems, don’t just build the systems (and networks, over and over).

QUADS Usage and Interest

- NASDAQ has shown interest in adoption for their dev/test bare-metal environments
- Ongoing integration with the MOC / HIL (Boston Univ, Harvard, MIT)

Question • Comments • Feedback

https://github.com/redhat-performance/quads
GitHub // Gerrit Code Review // waffle.io

Will Foster • @sadsfae • https://hobo.house