

# Skynet your Infrastructure with QUADS

DevOps Pro RU 2017



Will Foster • [@sadsfae](#) • [hobo.house](#) • [github.com/sadsfae](#)  
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.

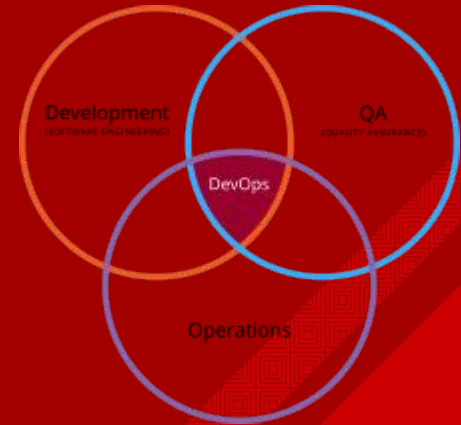
QUADS helps us automate and document the scheduling, management and mayhem of the races, race cars and race tracks.

# What do we do? *(readers digest version)*

- 2-person DevOps / Sysadmin Team
  - 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 and tools**

# How did we DevOps?

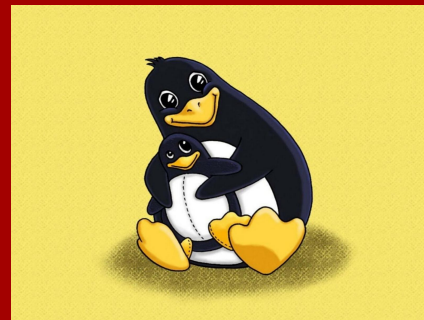
- Embed small Sysadmin/DevOps team (*2 people*) with a much larger Performance/Systems Engineering group (*160 people*).
- Attend relevant development syncs/scrums
- Become intimately familiar with development needs and workflow
- Make transparency paramount (break down silos where possible)
- Develop tooling and automation to get things done better
- Contribute code to core development (if relevant)
- *Team rotation*: development staff come into the DevOps team for 2-month periods.



# What Now? Sharing is Caring.

- All code and tools in public repositories (we use Git)
- Flexible, transparent Kanban task and priority system (Trello)
- Bi-weekly collaboration with DevOps people in similar roles inside of the company.
- Regular communication, demos, knowledge transfer of tooling.
- Be flexible to adjust priorities based on need
- Celebrate successes, learn from failures.
- Document everything!

You are the pit crew, track engineer and enabler.



But basically ..



# QUADS - What it is (*and is not*).

- 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 / visualizations
- 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?

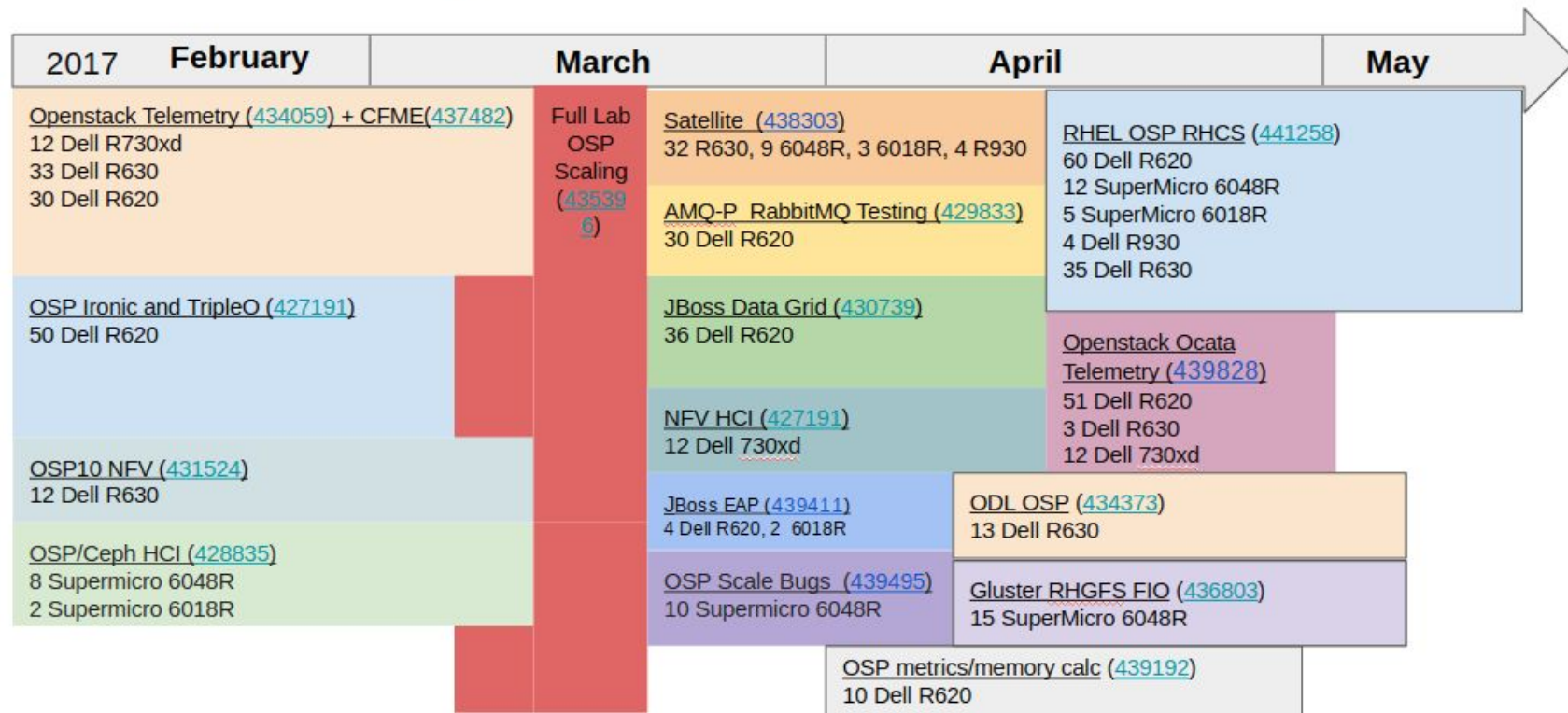
Red Hat Scale Lab



- 300+ bare-metal 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 4 weeks)
- 25+ isolated, parallel systems assignments ongoing 24/7

# QUADS - Scheduling in Action

## Scale Lab Assignments



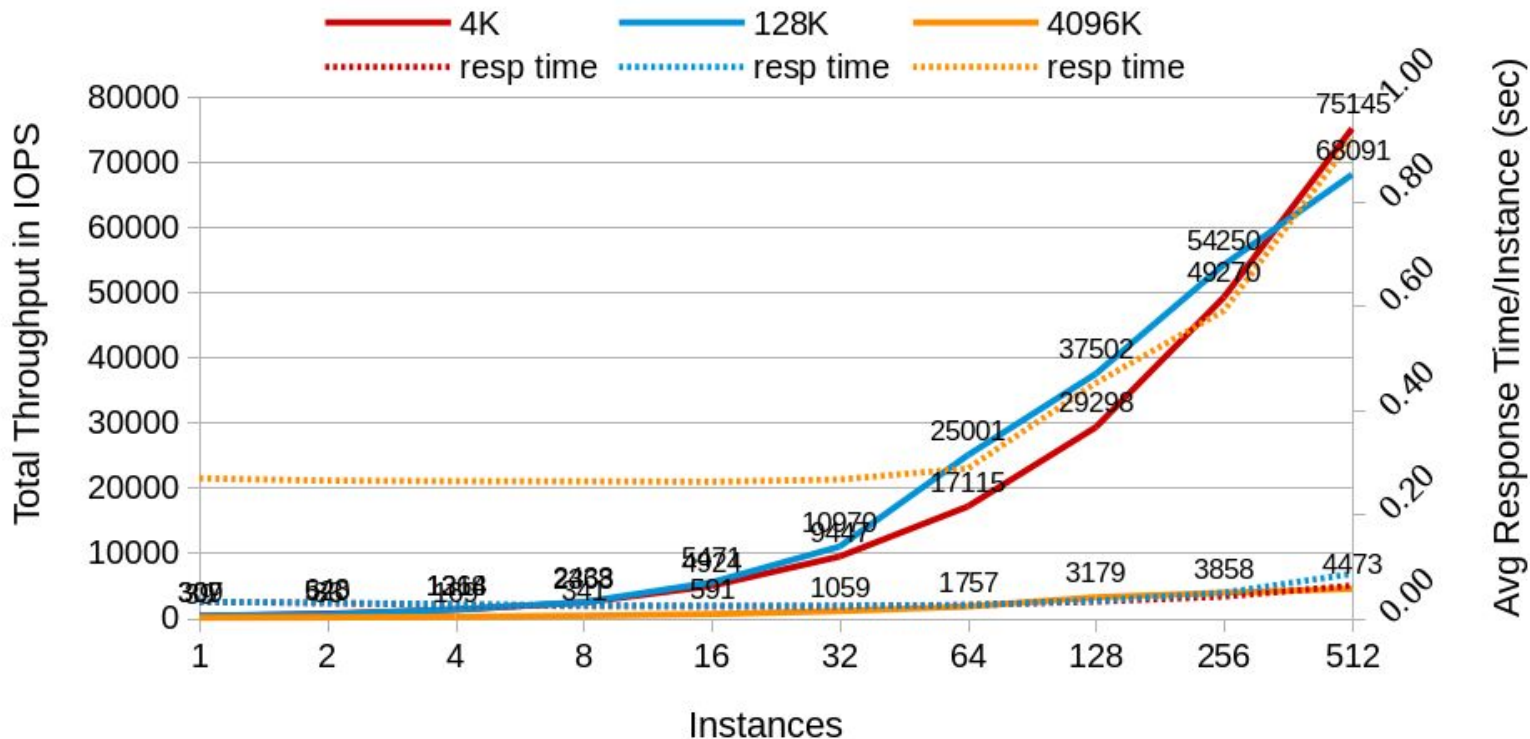
# QUADS - Scheduling in Action

## Scale Lab Assignments

2017	June	July		August	Sept
<u>OpenShift + OVN integration</u> (442508) 26 Dell R620 3 Dell R630	<u>Ansible, OpenShift, OpenStack and Ceph Expansion</u> (444204) 78 Dell R620	<u>OpenStack, OVN</u> (448689) 10 Dell R620	<u>Automated QUADS OpenStack Deploy testing</u> 10 Dell R620		<u>OpenStack</u> (448736) Full Lab 176 machines
<u>OSP Scale BZ</u> (446955) 10 R620		<u>New machine integration into lab</u> R620s offline	<u>Red Hat OpenStack Platform (OSP11 Telemetry)</u> (448731) 50 Dell R620 3 Supermicro 6048R		
<u>Ops Tooling</u> (447235) 18 R620					
<u>Ansible Tower</u> (447978) 24 R620			<u>Red Hat OpenStack Platform 12 Telemetry</u> (449244) 3 Dell R630 12 Dell 730xd 60 Dell R620		
<u>Ansible, OpenShift, OpenStack and Ceph</u> (444204) 4 Dell R930 45 Dell R630 12 Dell 730xd 2 Dell R620	<u>RHV, OSP, OCP, Gluster, Ceph, CloudForms</u> (449126) 4 Dell R930 45 Dell R630 12 Dell 730xd 2 Dell R620 15 Supermicro 6048R				
<u>Ceph+OSP Extension</u> (440297) 3 Dell R620 30 SuperMicro 6048R	<u>Straton Hyperconverged</u> (446799) 12 6048R	<u>Ceph RGW caching + EC</u> (436494) 13 6048R + 1 6018R		<u>RHCS/Ceph and RHEL/dm-cache</u> (438993) 10 6048R + 6018R + 8 R630	

# QUADS - Scale Lab Workload Result Example

OSP on RHCS, Effect of Block Size on Random Read IOPS & Latency  
RHEL 7.3, OSP 10, 29 RHCS 2.0 servers (36 OSDs per),  
1 thread/instance, 100G filesz, libaio iodepth=4, direct I/O



What problems are we solving?



# QUADS - What does it solve?

No more server hugging

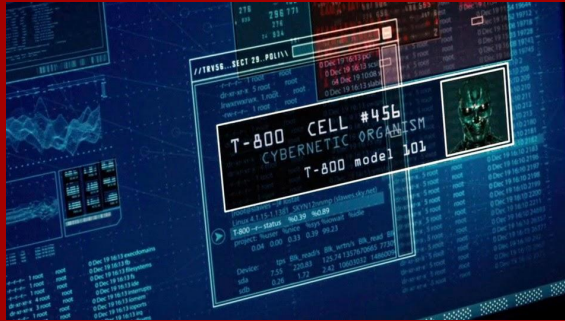


- Greater desire for hardware resources than hardware resources
- Clearly defined scheduling for server assignment and reclamation
- Published future schedule and visualizations for planning and audit



# 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 can 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)*

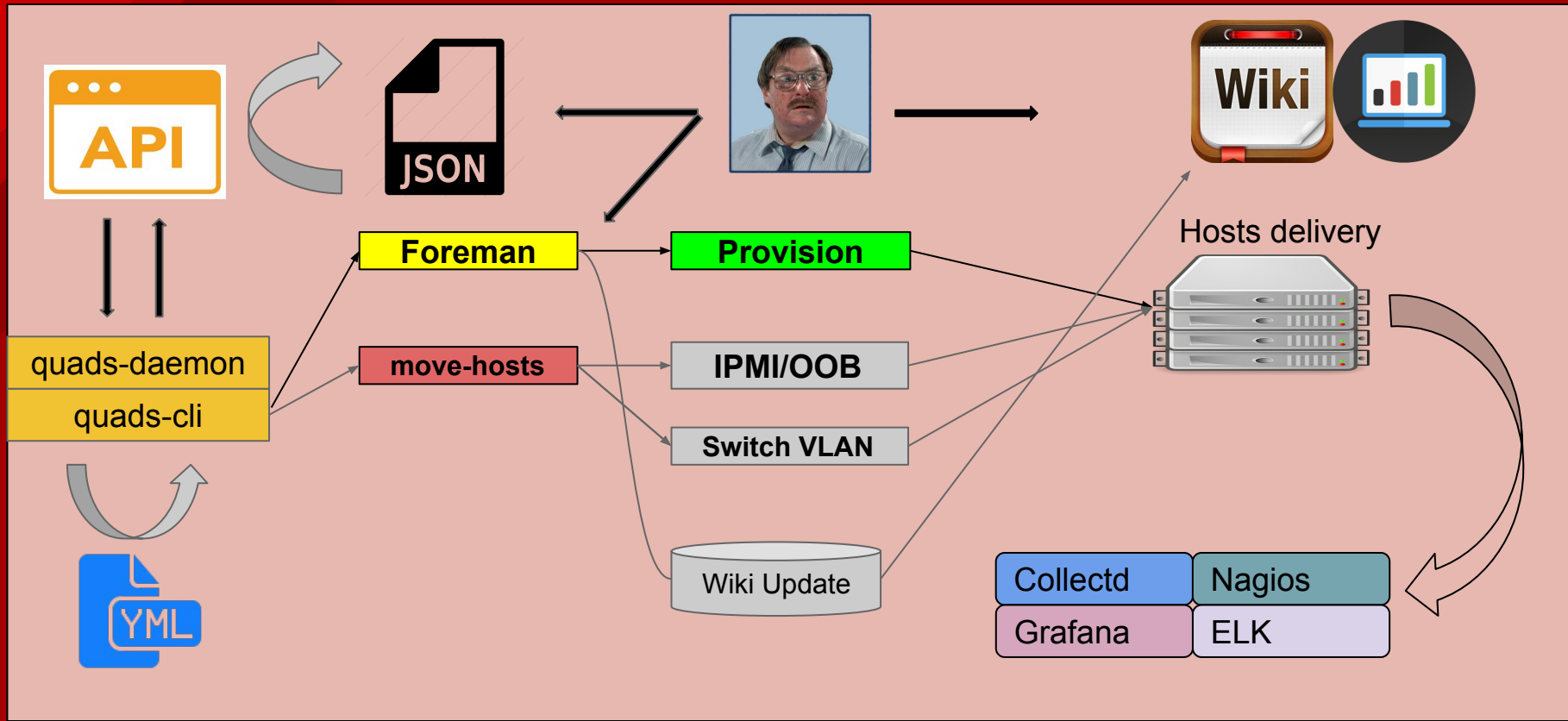
What if 100 servers change hands and we did everything manually?

- 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-cli --move-hosts
```

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

# QUADS Provisioning (move-hosts)

move-hosts

Foreman remove-role \$host

Foreman add-role \$host

Foreman change cloud pass

IPMITOOL change pass

Foreman mark \$host build = 1

IPMITOOL reboot \$host

PROVISION OS

%POST

Move \$host VLANS

Notify



Validate



#irc





# quads --ls-schedule

- List scheduling information for a given host

```
quads-cli --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=2017-11-15 22:00,end=2017-12-06 22:00,cloud=cloud02**

# Network Design / VLAN Automation

- We support automated setup of two types of VLAN designs
  - qinq: 0 mode: (default)
    - Each interface across all hosts in environment in it's own VLAN
      - NIC1 across all hosts in environment = VLAN1
      - NIC2 across all hosts in environment = VLAN2 etc.
  - qinq: 1 mode:
    - All interfaces across all hosts in environment in same VLAN

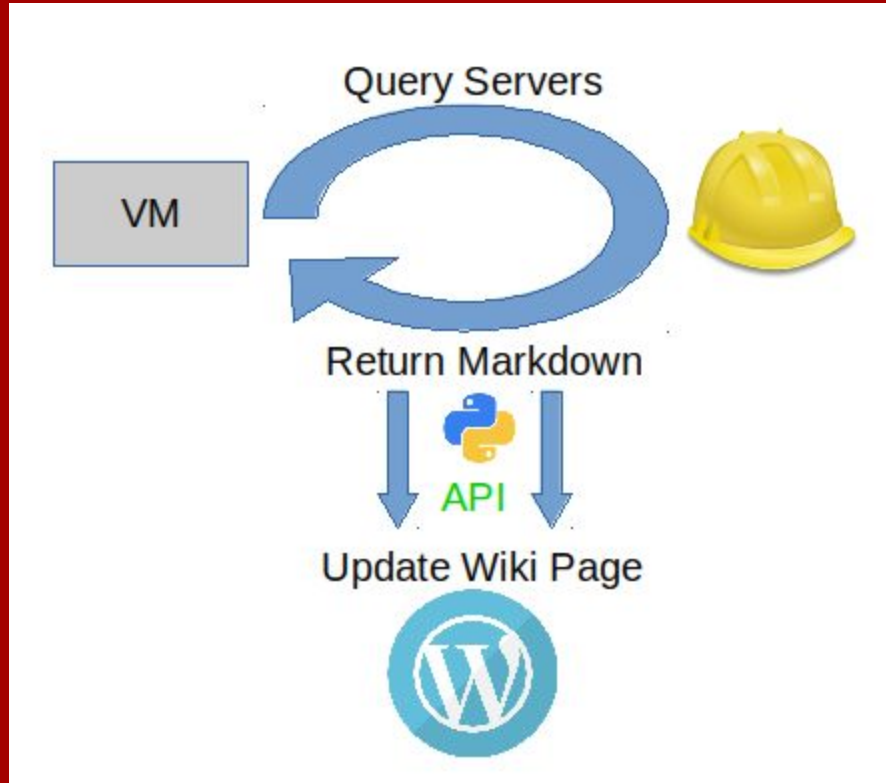
# Network Design / Host Interfaces

System Type	Network 1	Network 2	Network 3	Network 4	Public Network
Dell R620	p2p3	p2p4	em1	em2	em3
Dell R630	em1	em2	em3	em4	p2p1
Dell R720xd	p4p1	p4p2	em1	em2	em3
Dell R730xd	em1	em2	p4p1	p4p2	em3
Dell R930	em1	em2	p1p1	p1p2	em3
SuperMicro 6018r	enp4s0f0	eno2	enp4s0f1	enp4s0f2	eno1
SuperMicro 1028r	ens2f0	ens2f1	ens2f2	ens2f3	eno1
SuperMicro 1029p	enp94s0f0	enp94s0f1	enp94s0f2	enp94s0f3	eno1
SuperMicro 1029u	enp175s0f0	enp175s0f1	enp216s0f0	enp216s0f1	eno1
SuperMicro 6048r	ens3f0	ens3f1	enp131s0f0	enp131s0f1	enp5s0f0

# 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

## SUMMARY

NAME	SUMMARY	OWNER	REQUEST	INSTACKENV	HWFACTS	DELLCFG
cloud01	45 (Available servers)	nobody				
cloud02	36 (Alderaan Allocation)	smalleni	456512	download	inventory	view
cloud03	50 (OpenStack/Director)	jkilpatr	452743	download	inventory	view
cloud07	35 (ODL SDN Controller)	sgaddam	455851	download	inventory	view
cloud17	5 (CEPH IOPS)	acalhoun	461418	download	inventory	view
cloud18	64 (A-MQ Messaging)	mwagner	457259	download	inventory	view
cloud19	37 (Satellite 6.3)	sbadhwar	459829	download	inventory	view
cloud20	6 (TripleO, Open Stack, RHEL)	trwilson	460910	download	inventory	view
cloud22	10 (ODL SDN Controller)	sgaddam	455851	download	inventory	view



# 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
<p>● 2016-09-25 Scale Lab</p>	<p>25 ● 2016-09-26 Scale Lab</p>	<p>26 ● 2016-09-27 Scale Lab</p>	<p>27 ● 2016-09-28 Scale Lab</p>	<p>28 ● 2016-09-29 Scale Lab</p>	<p>29 ● 2016-09-30 Scale Lab</p>	<p>30 ● 2016-10-01 Scale Lab</p>
<p>● 2016-10-02 Scale Lab</p>	<p>2 ● 2016-10-03 Scale Lab</p>	<p>3 ● 2016-10-04 Scale Lab</p>	<p>4 ● 2016-10-05 Scale Lab</p>	<p>5 ● 2016-10-06 Scale Lab</p>	<p>6 ● 2016-10-07 Scale Lab</p>	<p>7 ● 2016-10-08 Scale Lab</p>
<p>● 2016-10-09 Scale Lab</p>	<p>9 ● 2016-10-10 Scale Lab</p>	<p>10 ● 2016-10-11 Scale Lab</p>	<p>11 ● 2016-10-12 Scale Lab</p>	<p>12 ● 2016-10-13 Scale Lab</p>	<p>13 ● 2016-10-14 Scale Lab</p>	<p>14 ● 2016-10-15 Scale Lab</p>
<p>● 2016-10-16 Scale Lab</p>	<p>16 ● 2016-10-17 Scale Lab</p>	<p>17 ● 2016-10-18 Scale Lab</p>	<p>18 ● 2016-10-19 Scale Lab</p>	<p>19 ● 2016-10-20 Scale Lab</p>	<p>20 ● 2016-10-21 Scale Lab</p>	<p>21 ● 2016-10-22 Scale Lab</p>
<p>● 2016-10-23 Scale Lab</p>	<p>23 ● 2016-10-24 Scale Lab</p>	<p>24 ● 2016-10-25 Scale Lab</p>	<p>25 ● 2016-10-26 Scale Lab</p>	<p>26 ● 2016-10-27 Scale Lab</p>	<p>27 ● 2016-10-28 Scale Lab</p>	<p>28 ● 2016-10-29 Scale Lab</p>
<p>● 2016-10-30 Scale Lab</p>	<p>30 ● 2016-10-31 Scale Lab</p>	<p>● 2016-11-01 Scale Lab</p>	<p>● 2016-11-02 Scale Lab</p>	<p>● 2016-11-03 Scale Lab</p>	<p>● 2016-11-04 Scale Lab</p>	<p>● 2016-11-05 Scale Lab</p>

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 -- smallen)  
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

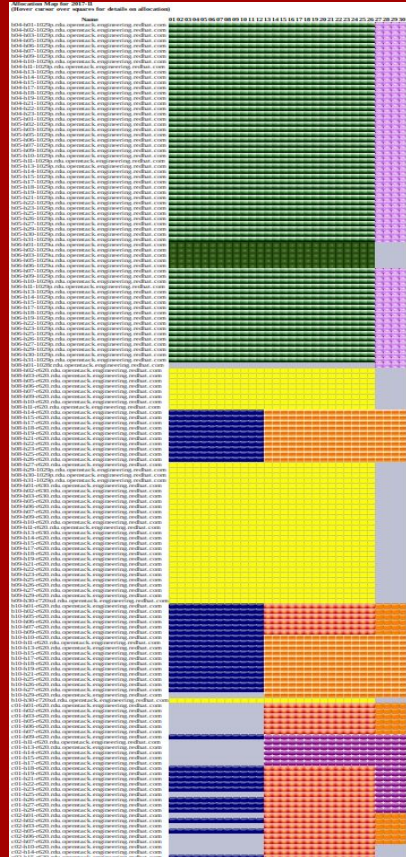
## Allocation Map for 2017-11

(Hover cursor over squares for details on allocation)

Name		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
b04-h01-1029p.rdu.openstack.engineering	.com																														
b04-h02-1029p.rdu.openstack.engineering	.com																														
b04-h03-1029p.rdu.openstack.engineering	.com																														
b04-h05-1029p.rdu.openstack.engineering	.com																														
b04-h06-1029p.rdu.openstack.engineering	.com																														
b04-h07-1029p.rdu.openstack.engineering	.com																														
b04-h09-1029p.rdu.openstack.engineering	.com																														
b04-h10-1029p.rdu.openstack.engineering	.com																														
b04-h11-1029p.rdu.openstack.engineering	.com																														
b04-h13-1029p.rdu.openstack.engineering	.com																														
b04-h14-1029p.rdu.openstack.engineering	.com																														
b04-h15-1029p.rdu.openstack.engineering	.com																														
b04-h17-1029p.rdu.openstack.engineering	.com																														
b04-h18-1029p.rdu.openstack.engineering	.com																														
b04-h19-1029p.rdu.openstack.engineering	.com																														

Description: A-MQ Messaging  
Env: cloud18  
Owner: mwagner  
RT: 457259  
Day: 25

# QUADS Dynamic Wiki - Map Visualization



# Ansible Facts Per Host (Ansible CMDB)

## Host overview

Generated on Mon Oct 30 15:20:12 2017 by root @ walkabout.rdu.openstack.engineering. .com [Clear settings](#)

### Shown columns

Name	Groups	DTAP	Comment	Ext ID	FQDN	Main IP	All IPv4	All IPv6	OS	Kernel	Arch	Virt	CPU type	vCPUs	RAM [GiB]	Mem Usage	Swap Usage	Disk usage
PhysDisk size	Nr of Ifaces	Timestamp																

### Host overview

Search:

Name		FQDN		Main IP	OS	Kernel	Virt	CPU type	vCPUs	RAM [GiB]	Nr of Ifaces
b04-h01-1029p.rdu.openstack.engineering	.com	b04-h01-1029p.rdu.openstack.engineering.	.com	10.12.77.84	RedHat 7.4	3.10.0-693.el7.x86_64	kvm / host	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	64	251.4	15
b04-h02-1029p.rdu.openstack.engineering	.com	b04-h02-1029p.rdu.openstack.engineering.	.com	10.12.77.86	RedHat 7.4	3.10.0-693.el7.x86_64	kvm / host	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	64	251.4	15
b04-h03-1029p.rdu.openstack.engineering	.com	b04-h03-1029p.rdu.openstack.engineering.	.com	10.12.77.88	RedHat 7.4	3.10.0-693.el7.x86_64	kvm / host	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	64	251.4	15
b04-h05-1029p.rdu.openstack.engineering	.com	b04-h05-1029p.rdu.openstack.engineering.	.com	10.12.77.90	RedHat 7.4	3.10.0-693.el7.x86_64	kvm / host	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	64	251.4	15
b04-h06-1029p.rdu.openstack.engineering	.com	b04-h06-1029p.rdu.openstack.engineering.	.com	10.12.77.92	RedHat 7.4	3.10.0-693.el7.x86_64	kvm / host	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	64	251.4	15



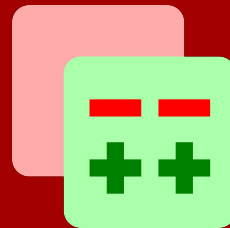
# BIOS, CPU Perf, Boot Order (Dell Only)

Name	CPU Setting	Boot Order
c08-h24-r630.rdu.openstack.engineering. .com	PerfOptimized	NET 5 (Foreman),Hard Drive,NET 2 (Director)
c08-h25-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)
c08-h26-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)
c08-h27-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)
c08-h28-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)
c08-h29-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)
c09-h11-r630.rdu.openstack.engineering. .com	PerfOptimized	Hard Drive,NET 2 (Director),NET 5 (Foreman)

*Oh no, one of the CPU settings isn't optimized!*

Name	CPU Setting	Boot Order
c01-h09-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedOs	NET 5 (Foreman),Hard Drive,NET 2 (Director),NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3
c01-h11-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedOs	NET 2 (Director),Hard Drive,NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3,NET 5 (Foreman)
c01-h13-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedDapc	NET 2 (Director),Hard Drive,NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3,NET 5 (Foreman)
c01-h14-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedOs	NET 2 (Director),Hard Drive,NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3,NET 5 (Foreman)
c01-h15-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedOs	NET 2 (Director),Hard Drive,NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3,NET 5 (Foreman)
c01-h17-r620.rdu.openstack.engineering. .com	PerfPerWattOptimizedOs	NET 2 (Director),Hard Drive,NIC.Slot.2-1,NIC.Slot.2-2,NIC.Slot.2-3,NET 5 (Foreman)

# QUADS Code Review and CI



- Gerrit Code Review
  - IRC: Freenode #quads : announces new reviews
- Jenkins: (automated testing of new gerrit reviews/patchsets)
  - Votes on all commits in gerrit
    - All CI Tests must pass to get +1
      - Functional tests (quads commands)
      - Shellcheck
      - Flake8 for Python

# QUADS CI / CD Pipeline



- Successful merge to master git branch auto-creates RPM
  - COPR service uses webhooks to auto-build versions

Build 661847

ResubmitDelete

General Information

Status:

🟢 succeeded - Successfully built.

Submitted:

2017-11-14 17:44 UTC (14 minutes ago)

Started:

2017-11-14 17:47 UTC

Finished:

2017-11-14 17:51 UTC

Build time:

3 minutes

Networking enabled:

True

Built by:

[quadsdev](#)

Source

Package:

[quads](#)

Version:

0.99.2-20171114

Source Type:

Build from a SCM repository

SCM type:

git

Clone URL:

<https://github.com/redhat-performance/quads>

Committish:

master

Subdirectory:

rpm

Path to .spec file:

quads.spec

Build SRPM with:

rpkg



# Engineering Challenges / QUADS 2.0

- Using flat file architecture provides a simplistic design but comes with downsides. This is on purpose (for now).
  - *Positive:* Easy to debug, test, replicate schedule.yaml
  - *Positive:* Simplistic codebase, system calls are fast (*but don't scale - more on that later*).
  - *Negative:* Systems/Network provisioning done sequentially
  - *Negative:* HTTP API does not provide distributed scale
    - 2-3 hours to deploy 500+ machines/networks
      - We don't care, automated deployments go Sunday
      - Not everyone needs to be Facebook or Google

## Code Example #1: Read without a Database

- Python method ensures routine syncing to disk, checks before every query

```
def config_newer_than_data(self):  
    if os.path.isfile(self.config):  
        if os.path.getmtime(self.config) > self.loadtime:  
            return True  
    return False
```

- Force data integrity by writing in-memory data to disk then query with system calls

```
#return hosts  
def get_hosts(self):  
    if self.config_newer_than_data():  
        self.read_data()  
    return self.quads.hosts.get()
```

## Code Example #2: Write Data without a Database

- Python thread locking is required to ensure write integrity.

```
# define a schedule for a given host
def add_host_schedule(self, schedstart, schedend, schedcloud, host):
    # add a scheduled override for a given host
    self.thread_lock.acquire()
```

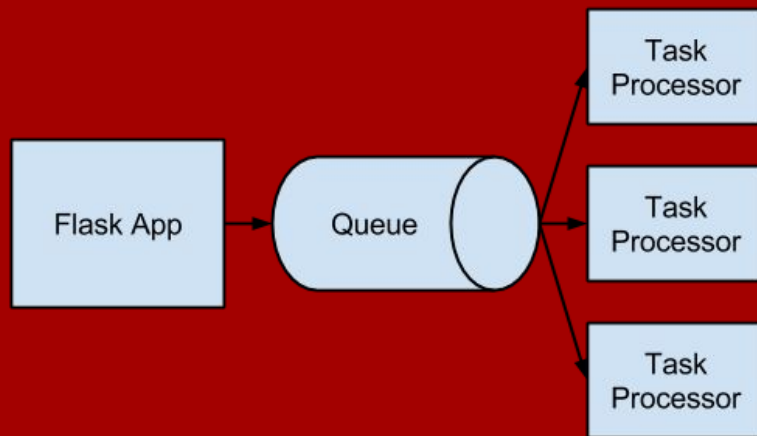
- Threads are released when data is written, command successful (*quads-cli output*)
- System calls are very fast in singular operation, they just don't scale in parallel

```
if self.write_data():
    self.thread_lock.release()
    return ["OK"]
else:
    self.thread_lock.release()
    return ["ERROR"]
```

# QUADS Milestone 2.0

Future move to an asynchronous architecture in milestone 2.0

- Flask UI / self-service front-end in milestone 1.1
- Flask / sqlite / celery backend in milestone 2.0



# 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).

Come to the Red Hat booth after the talk!

<https://github.com/redhat-performance/quads>

Will Foster • @sadsfae • hobo.house • [github.com/sadsfae](https://github.com/sadsfae)



If we have time, let's look at some commands

# quads-cli --define-host

- Define a host to manage in the scheduler, and specify its cloud.
  - Example: place two hosts in the same cloud

```
quads-cli --define-host c08-h21-r630.example.com --default-cloud cloud01 --host-type general  
quads-cli --define-host c08-h22-r630.example.com --default-cloud cloud01 --host-type general
```



# quads-cli --ls-host

- Lists all the current hosts managed by QUADS

```
quads-cli --ls-hosts
```

```
c08-h21-r630.example.com  
c08-h22-r630.example.com
```

# quads-cli --add-schedule

- Create a schedule for the host to perform workloads/tasks.
- Set current future workloads/tests of machines based on date
- Unlimited, consecutive future schedules are supported

```
quads-cli --add-schedule --host c08-h21-r630.example.com --schedule-start "2016-07-11 08:00" --schedule-end "2016-07-12 08:00" --schedule-cloud cloud02
```

# quads-cli --ls-schedule

- List scheduling information for a given host

```
quads-cli --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=2017-01-28 12:00,end=2017-01-29 05:00,cloud=cloud02
```

# quads-cli --summary

- Provide a summary of current system allocations

```
quads-cli --summary
```

```
cloud01 : 9 (Unallocated hardware)  
cloud02 : 198 (Openshift on OpenStack)  
cloud03 : 22 (OSP Newton)
```

# quads-cli --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-cli --move-hosts
```

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

# Auditing Tools

- We include additional auditing tools like find-available
  - Searches for availability based on days needed, machines needed and optionally limit by type

```
bin/find-available.py -c 3 -d 10 -l "620|630"
```

```
First available date = 2016-12-05 08:00  
Requested end date = 2016-12-15 08:00  
hostnames =  
c03-h11-r620.rdu.openstack.example.com  
c03-h13-r620.rdu.openstack.example.com  
c03-h14-r620.rdu.openstack.example.com
```

# Common Configuration File

- We try to make everything a variable in quads.yml

```
cat conf/quads.yml
```

```
install_dir: /opt/quads  
data_dir: /opt/quads/data  
domain: example.com  
  
email_notify: true  
irc_notify: true  
ircbot_ipaddr: 192.168.0.100  
ircbot_port: 5050  
ircbot_channel: #yourchannel
```