



Ceph devops operations & deployment

other ceph talks

Block Storage for the Cloud, a Pilot with Enterprise Use-cases at BMW

thu 10am, fri 8:30am

Janos Mattyasovszky, BMW

Andreas Poeschl Project Lead, BMW

Ceph: Much More Than Just a Distributed File System

wed 1pm, thu 10am

Lars Marowsky-Brée, Architect Storage/HA, SUSE

Udo Seidel, O/S Services Manager, Amadeus Data Processing GmbH



infrastructure as code



infrastructure **is** code



scale



scale



SCS

[illegible]









```
for h in host{1,2,3,4}
do
  ssh $h apt-get install ...
done
```



```
dsh -g foo apt-get install ...
```

dsh

```
[0 tv@dreamer ~]$ dsh -g plana uptime
```

```
ubuntu@plana01.front.sepia.ceph.com: 14:19:49 up 167 days, 21:32, 1 user, load average: 0.00, 0.01, 0.05
ubuntu@plana02.front.sepia.ceph.com: 14:19:49 up 1 day, 3:29, 0 users, load average: 0.95, 1.03, 1.01
ubuntu@plana03.front.sepia.ceph.com: 14:19:49 up 1 day, 3:29, 0 users, load average: 1.52, 1.28, 1.17
ubuntu@plana04.front.sepia.ceph.com: 14:19:52 up 18 min, 1 user, load average: 3.63, 3.51, 2.51
ubuntu@plana05.front.sepia.ceph.com: 17:19:52 up 17 days, 22:59, 0 users, load average: 0.05, 0.03, 0.05
ubuntu@plana06.front.sepia.ceph.com: 14:19:53 up 18 days, 0 min, 1 user, load average: 0.31, 0.23, 0.17
ubuntu@plana07.front.sepia.ceph.com: 14:19:53 up 18 min, 1 user, load average: 5.29, 4.98, 3.11
ubuntu@plana08.front.sepia.ceph.com: 14:19:53 up 44 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana09.front.sepia.ceph.com: 14:19:53 up 18 min, 1 user, load average: 6.56, 6.33, 3.93
ubuntu@plana10.front.sepia.ceph.com: 14:19:54 up 48 days, 23:30, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana11.front.sepia.ceph.com: 14:19:54 up 44 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana12.front.sepia.ceph.com: 14:19:54 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana13.front.sepia.ceph.com: 14:19:55 up 43 min, 0 users, load average: 0.01, 0.03, 0.05
ubuntu@plana14.front.sepia.ceph.com: 14:19:55 up 43 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana15.front.sepia.ceph.com: 17:19:55 up 17 days, 22:52, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana16.front.sepia.ceph.com: 14:19:55 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana17.front.sepia.ceph.com: 14:19:55 up 43 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana18.front.sepia.ceph.com: 14:19:57 up 102 days, 1:03, 0 users, load average: 1.07, 0.62, 0.29
ubuntu@plana19.front.sepia.ceph.com: 14:19:57 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana20.front.sepia.ceph.com: 14:19:57 up 43 min, 0 users, load average: 0.00, 0.02, 0.04
ubuntu@plana21.front.sepia.ceph.com: 14:19:58 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana22.front.sepia.ceph.com: 14:19:58 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana23.front.sepia.ceph.com: 14:19:58 up 43 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana24.front.sepia.ceph.com: 14:19:58 up 43 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana25.front.sepia.ceph.com: 14:19:58 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana26.front.sepia.ceph.com: 14:19:59 up 18 days, 1 min, 3 users, load average: 0.11, 0.05, 0.05
ubuntu@plana27.front.sepia.ceph.com: 14:19:59 up 18 days, 0 min, 0 users, load average: 0.04, 0.04, 0.05
ubuntu@plana28.front.sepia.ceph.com: 14:19:59 up 18 days, 0 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana29.front.sepia.ceph.com: 14:19:59 up 18 days, 0 min, 0 users, load average: 0.03, 0.11, 0.13
ubuntu@plana30.front.sepia.ceph.com: 14:20:00 up 43 min, 0 users, load average: 0.04, 0.04, 0.05
ubuntu@plana31.front.sepia.ceph.com: 14:20:00 up 43 min, 0 users, load average: 0.03, 0.04, 0.05
ubuntu@plana32.front.sepia.ceph.com: 14:20:00 up 18 days, 0 min, 0 users, load average: 0.00, 0.02, 0.05
ubuntu@plana33.front.sepia.ceph.com: 14:20:01 up 18 days, 0 min, 0 users, load average: 0.00, 0.01, 0.05
ubuntu@plana34.front.sepia.ceph.com: 14:20:01 up 19 min, 1 user, load average: 1.75, 1.85, 1.21
ubuntu@plana35.front.sepia.ceph.com: 14:20:01 up 43 min, 0 users, load average: 0.00, 0.01, 0.04
ubuntu@plana37.front.sepia.ceph.com: 14:20:01 up 43 min, 0 users, load average: 0.00, 0.01, 0.05
```


cssh

[illegible]



inktank

mkcephfs

mkcephfs

Deploying with mkcephfs

ceph.com/docs/master/config-cluster/mkcephfs/

Ceph documentation » Configuration »

previous | next | modules | index




TABLE OF CONTENTS

- Getting Started
- Installation
- Configuration
 - Hard Disk and File System Recommendations
 - Configuration
 - Deploy with mkcephfs
 - Enable Login to Cluster Hosts as **root**
 - Copy Configuration File to All Hosts
 - Create the Default Directories
 - Run **mkcephfs**
 - Deploy with Chef
 - Storage Pools
 - Authentication
- Operating a Cluster
- Ceph FS
- Block Devices
- RADOS Gateway
- Operations
- Recommendations
- Control commands
- API Documentation
- Ceph Source Code
- Internals
- Manual pages
- Architecture of Ceph

DEPLOYING WITH MKCEPHFS

ENABLE LOGIN TO CLUSTER HOSTS AS ROOT

To deploy with **mkcephfs**, you will need to be able to login as **root** on each host without a password. For each host, perform the following:

```
sudo passwd root
```

Enter a password for the root user.

On the admin host, generate an **ssh** key without specifying a passphrase and use the default locations.

```
ssh-keygen
Generating public/private key pair.
Enter file in which to save the key (/ceph-admin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /ceph-admin/.ssh/id_rsa.
Your public key has been saved in /ceph-admin/.ssh/id_rsa.pub.
```

You may use RSA or DSA keys. Once you generate your keys, copy them to each OSD host. For example:

```
ssh-copy-id root@myserver01
ssh-copy-id root@myserver02
```

Modify your `~/ssh/config` file to login as **root**, as follows:

```
Host myserver01
    Hostname myserver01.fully-qualified-domain.com
    User root
Host myserver02
```

mkcephfs walkthrough

- say “[osd.42] host = mysrv1”
- setup ssh
- put ceph.conf on every node
- create dirs, mount data disks
- `sudo mkcephfs -a`
 `-c /etc/ceph/ceph.conf`
 `-k ceph.keyring`
- `service ceph -a start`

mkcephfs limitations

- can't grow
- can't shrink
- can't upgrade
- can't handle errors
- lots of shell → hard to maintain

sysvinit limitations

all daemons managed by the
same script

can't respawn

pid files

legacy

support for running in source checkout

support for scp'ing the binary
ssh from an init script

“service ceph ssh”?

support for mkfs & mount of btrfs
command for deleting logs

legacy

```
wc -l src/mkcephfs.in  
src/ceph_common.sh
```

```
520 src/mkcephfs.in  
158 src/ceph_common.sh  
678 total
```

legacy

```
cat src/init-ceph.in  
src/ceph_common.sh |wc -l  
516
```

```
wc -l /etc/init.d/*|grep -v  
total|sort -n|tail -3
```

```
314 /etc/init.d/openvpn  
338 /etc/init.d/rc  
533 /etc/init.d/openipmi
```




inktank



manual bootstrap

manual bootstrap

roughly what mkcephfs does
underneath (modernized for presentation)

showing all options on command line
so you can see information flow;
normally would use ceph.conf

create a keyring and monitor key

```
$ ceph-authtool \  
  --create-keyring --gen-key \  
  --name=mon. mon.keyring
```

creating mon.keyring

```
$ cat mon.keyring
```

```
[mon.]
```

```
key = AQBaBThQcLhpLBAANhkkqZ70d23I94xIt80F9A==
```

create admin key

```
$ ceph-authtool --gen-key  
  --name=client.admin --set-uid=0 \  
  --cap mon "allow *" \  
  --cap osd "allow *" \  
  --cap mds "allow" \  
  mon.keyring
```

```
$ cat mon.keyring
```

```
[mon.]  
key = AQBaBThQcLhpLBAANhkkqZ70d23I94xIt80F9A==  
[client.admin]  
key = AQDqBThQcC94FRAA0aUłHAD2IuM/0QdVQIs5ZQ==  
auid = 0  
caps mds = "allow"  
caps mon = "allow *"  
caps osd = "allow *"
```

create a monitor

```
# unique id to identify the whole cluster  
$ uuidgen
```

```
e946cafc-2b52-4858-8cf0-db4619481f72
```

```
# note keyring, fsid, and the address where  
# the monitor will listen, for use in the  
# generated monitor map.
```

```
$ ceph-mon -i a --mkfs --keyring mon.keyring \  
  -m 10.1.2.3:6789 --mon-data /dev/mon.a/ \  
  --fsid e946cafc-2b52-4858-8cf0-db4619481f72
```

```
# start it
```

```
$ ceph-mon -i a --mon-data /dev/mon.a/ \  
  --public-addr 10.1.2.3:6789 \  
  --auth-supported cephx
```

test the monitor

```
$ ceph -m 10.1.2.3:6789 \
  --keyring mon.keyring \
  --auth-supported cephx \
  -S
```

...

allocate a storage daemon

```
# allocate osd id; uuids can be used for  
# idempotency and avoiding waste of  
# numbers on errors  
$ uuidgen
```

```
1199be1c-2ecd-4e25-80c0-107209b05673
```

```
$ ceph -m 10.1.2.3:6789 \  
  --keyring mon.keyring \  
  --auth-supported cephx \  
  osd create --concise \  
  1199be1c-2ecd-4e25-80c0-107209b05673
```

create a storage daemon

```
# also create a key
$ ceph-osd \
  -m 10.1.2.3:6789 \
  --osd-data /media/osd42/ \
  --osd-journal /media/osd42/journal \
  --osd-journal-size 2000 \
  --osd-uuid 1199be1c-2ecd-4e25-80c0-107209b05673 \
  --fsid e946cafc-2b52-4858-8cf0-db4619481f72 \
  -i 42 --mkfs --mkkey
```


authorize the osd key

```
# add the key to the monitor's key
# store, with the right permissions
$ ceph -m 10.1.2.3:6789 \
  --keyring mon.keyring \
  --auth-supported cephx \
  auth add osd.42 \
  osd 'allow *' \
  mon 'allow rwx' \
  -i /media/osd42/keyring
```

set the crush location

```
$ ceph -m 10.1.2.3:6789 \  
  --auth-supported cephx \  
  --keyring /media/osd42/keyring \  
  --name osd.42 \  
osd crush create-or-move \  
42 \                               ← osd id  
1 \                               ← initial weight  
host=mysrv05 \  
rack=rack07 \  
dc=us-west
```

start the storage daemon

```
$ ceph-osd \  
-m 10.1.2.3:6789 \  
--auth-supported cephx \  
--osd-data /media/osd42/ \  
--osd-journal /media/osd42/journal \  
-i 42
```



inktank



coordination

monitors use paxos

majority vote about
everything that needs
consensus

quorum needs

2 of 3

3 of 5

4 of 7

initial quorum

```
mon_initial_members =  
    host1, host2, host3
```

majority of these needed
to form the quorum
for the very first time



osd hotplugging

systematic, devops, angle

focus on admin needs
(would love to hear more from you!)

better integration, automation,
industrial strength shrinkwrap

strong conventions for paths etc

ceph core remains as flexible as ever,
deployment packaging has assumptions;
you can always create your own packaging

disk failure

eject old one

pick new from the replacement pile

plug it in

when failure percentage is above a threshold, bring a cart full of spare disks and visually scan the rack for blinking lights

fan/psu/mobo/etc fails

just redistribute disks to any server
that has slots free

fix/replace the server at leisure

the server is just a chassis for data

re-replication will also handle it, but see
old tale about station wagon full of tapes

implications

osd id not tied to a server anymore
might as well move away from
humans managing osd ids

cluster uuid, osd uuids keep us safe;
still need id for dense array in CRUSH

ceph-disk-prepare

```
$ ceph-disk-prepare /dev/sdg
```

erases all data on that disk!

creates GPT partition with c3ff05d uuid

creates a file system

stores cluster uuid, osd uuid on disk

ceph-disk-activate

```
$ ceph-disk-activate /path/to/osd/data
```

```
$ ceph-disk-activate --mount /dev/sdg1
```

(mount in temp directory)

allocate osd id, if needed

create ceph-osd & authorize its key, if n.

move mount point to /var/lib/ceph/osd

tell upstart to start the service

udev

kernel sees a block device, triggers udev

upstart monitors udev events

upstart runs ceph-disk-activate when
partition with uuid c3ff05d is added

upstart

current implementation is tied to upstart

design is not; core ceph is not

we'll support more once the need is there

good stuff in upstart

no pid files, no double-fork needed

respawn on crash

log stderr until custom log system starts

instance jobs lets us run several daemons
dynamically, no compromises

using upstart

```
$ initctl list | grep ceph
```

```
$ initctl status ceph-osd id=42
```

```
$ stop ceph-osd id=42
```

```
$ initctl status \  
ceph-osd cluster=foo id=42
```

bootstrap keys

how to create a new osd?

need to talk to monitor

need to make monitor trust us

create `client.bootstrap-osd` key that is allowed to allocate osd id, add key with the capabilities an osd needs

teach a computer to fish

capa-what?

capabilities; what that key can do

```
[client.admin]
key = AQDqBThQcC94FRAA0aUlHAD2IuM/0QdVQIs5ZQ==
auid = 0
caps mds = "allow"
caps mon = "allow *"
caps osd = "allow *"
```

```
[client.bootstrap-osd]
```

```
...
```

```
caps mon = "allow command osd create ...; \
    allow command osd crush set ...; \
    allow command auth add *↵
    osd allow\\ *↵
    mon allow\\ rwx; \
    allow command mon getmap"
```

say what?

intricate nesting and delicate whitespace (sorry)

```
caps mon = "  
allow command osd create ...      ← any number of args  
allow command osd crush set ...  
allow command auth add *↓  
    osd allow\ *↓                  ← one level less quoting  
    mon allow\ rwx  
allow command mon getmap  
"
```

example:

```
auth add ANYONEWORD      ← like, "osd.42"  
    osd "allow *"  
    mon "allow rwx"
```



do they speak english in what?

i do hope we visit this some time

but for now, it works

simple case is simple,
complex case is tedious



inktank



devops frameworks

chef

ruby framework, you describe steps to take
with pseudo-declarative helpers to create files etc

json data about node fetched from server
at beginning of run

searches via solr/lucene, fetch other node's json data:
`role:ceph-mon AND chef_environment:#{node.chef_environment}`

json data saved to server at end of successful run

run every 30 min, or when told to

awkward to bring up distributed systems from
scratch; a needs to save attributes, and only then b
can search for them, and even then the search
updates asynchronously

chef



TABLE OF CONTENTS

- Getting Started
- Installation
- Configuration
 - Hard Disk and File System Recommendations
 - Configuration
 - Deploy with mkcephfs
 - Deploy with Chef
 - Clone the Required Cookbooks
 - Add the Required Cookbook Paths
 - Install the Cookbooks
 - Configure your Ceph Environment
 - Configure the Roles
 - Configure Nodes
 - Prepare OSD Disks
 - Run **chef-client** on each Node
 - Proceed to Operating the Cluster
 - Storage Pools
 - Authentication
- Operating a Cluster
- Ceph FS
- Block Devices

DEPLOYING WITH CHEF

We use Chef cookbooks to deploy Ceph. See [Managing Cookbooks with Knife](#) for details on using `knife`. For Chef installation instructions, see [Installing Chef](#).

CLONE THE REQUIRED COOKBOOKS

To get the cookbooks for Ceph, clone them from git:

```
cd ~/chef-cookbooks
git clone https://github.com/opscode-cookbooks/apache2.git
git clone https://github.com/ceph/ceph-cookbooks.git ceph
```

ADD THE REQUIRED COOKBOOK PATHS

If you added a default cookbook path when you installed Chef, `knife` may be able to upload the cookbook you've cloned to your cookbook path directory without further configuration. If you used a different path, or if the cookbook repository you cloned has a different tree structure, add the required cookbook path to your `knife.rb` file. The `cookbook_path` setting takes a string or an array of strings. For example, you can replace a string path with an array of string paths:

```
cookbook_path '/home/{user-name}/chef-cookbooks/'
```

Becomes:

```
cookbook_path [
  '/home/{user-name}/chef-cookbooks/',
  '/home/{user-name}/chef-cookbooks/{another-directory}/',
  '/some/other/path/to/cookbooks/'
]
```

INSTALL THE COOKBOOKS

cooking squid

to add an osd node:

- assign node to right environment
- set crush location of server
- add role ceph-osd

to remove a node:

- chef really doesn't handle this well
- shift the data out
- shut down the daemon
- zap the disk
- remove the osd from monitors

crowbar

bare-metal deploy system on top of chef
quite different from chef in functionality
still managing to reuse code from our
standalone chef cookbook, 4 if `is_crowbar?`

puppet

if you squint hard enough, looks
somewhat like chef

currently community contributions only



inktank

ceph-deploy



there is no spoon

we don't want to force users to use
chef (or any of the alternatives)

just ssh and python libraries on the
admin workstation

no dorodango

deploy logic is put into ceph, not
the recipes or such wrappers
makes cookbooks simpler, too



demo

(he says, hope in his eyes)

```
ceph-deploy new n01 n02
```

```
ceph-deploy install n01 n02
```

```
ceph-deploy mon
```

```
ssh n01 sudo ceph -s
```

```
ceph-deploy osd n02
```

```
ceph-deploy disk n02:vdb n02:vdc
```



inktank



monitoring

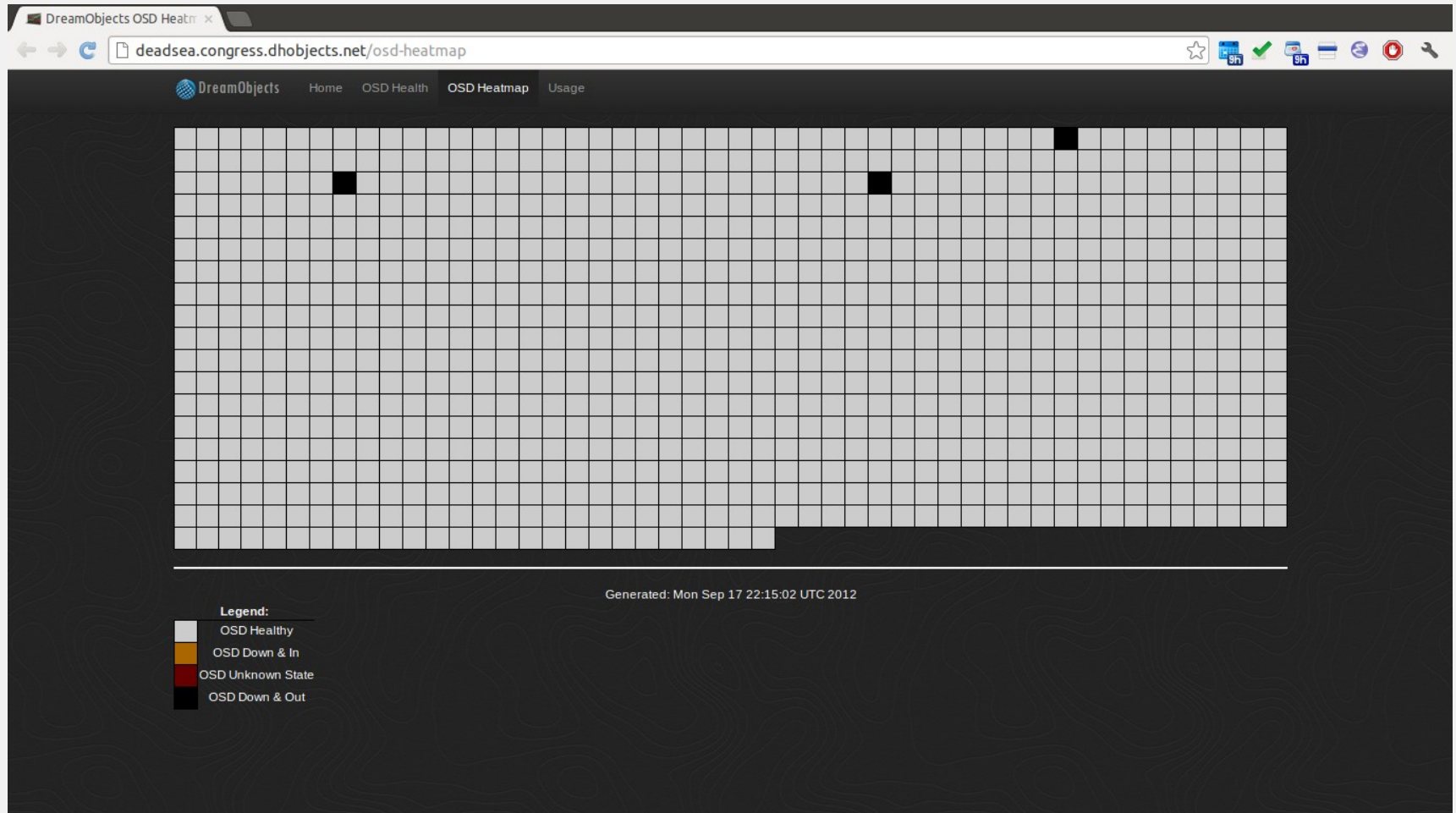
nagios et al





ceph health

DreamObjects





not always a clean fail

things you know to measure are
easy to detect; slow ops etc

for the rest, you need an
imprecise test

smoke test

radosbench

s3-tests for radosgw, lb, apache

working on modularizing teuthology
(our test runner) to make test cases
more self-contained



benchmark

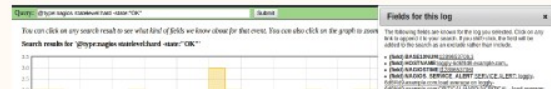
before and after a change



logging

The image is a screenshot of the Logstash website homepage. At the top, the browser's address bar shows "logstash - open source log" and "logstash.net". The website has a green header with navigation links: "home", "docs", "about", "wiki", and "bugs". On the left, there is a large, cartoonish illustration of a log with a face and a mustache. To the right of the log, the word "logstash" is written in a large, bold, black font. Below the logo, there is a paragraph of text: "logstash is a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use (like, for searching). Speaking of searching, logstash comes with a web interface for searching and drilling into all of your logs." Below this text, there are three green buttons: "Download", "Learn", and "Wiki". Further down, there is a large heading: "Ship logs from any source, parse them, get the right timestamp, index them, and search them." At the bottom, there is a paragraph of text: "All your logs from all over your infrastructure in one place - with searching and graphing. Since we can easily parse text-based logs, you can query for more precise things like, all 404 http". On the right side of the bottom section, there is a small screenshot of the Logstash web interface, showing a search bar and a graph.

All your logs from all over your infrastructure in one place - with searching and graphing. Since we can easily parse text-based logs, you can query for more precise things like, all 404 http










kibana


Kibana. Search it, score it, x

← → C

rashidkpc.github.com/Kibana/

☆       

Kibana Home About Installation Support



Kibana

You have logs. Billions of lines of data. You shipped, dated it, parsed it and stored it. Now what do you do with it? **Now you make sense of it.** Kibana helps you do that. Kibana is an alternative browser based interface for [Logstash](#) and [ElasticSearch](#) that allows you to efficiently search, graph, analyze and otherwise make sense of a mountain of logs.

[Getting Started »](#)

Search 🔍

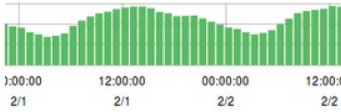
vw* AND NOT agent:Quickcurl AND agent:*

Time	@tags	bytes	agent
02/03 16:32:18	apache-access	3485	*Mozilla/4.0 (comp
02/03 16:32:18	apache-access	2498	*Mozilla/4.0 (comp
02/03 16:32:18	apache-access	76	*Mozilla/5.0 (iPhone
02/03 16:32:18	apache-access	1124	*Mozilla/5.0 (comp

Find one event or one million. Kibana uses

Graph 📊

T16:00:00



Visually analyze trends in log volume to find

Score 📄

L like	652	3.26%	+1.39	Search
r	345	1.73%	-1.05	Search
L like	226	1.13%	-1.03	Search
	258	1.29%	-0.94	Search
	135	0.68%	-0.9	Search

Score, count and trend fields to find patterns

Stream 🎬

AND query:* AND collection:a*x ANI

citycode	path
011	/select
1001	/select
phx	/select
den	/select

Create dashboards from searches to view



structured logging



metrics

DreamObjects



perf counters as metrics

admin socket

```
ceph --admin-daemon /var/run/ceph/ceph-osd.0.asok  
perf dump
```

```
"throttle-msgr_dispatch_throttler-client" : {  
  "get_or_fail_fail" : 0,  
  "get_sum" : 82760,  
  "max" : 104857600,  
  "put" : 2637,  
  "val" : 0,  
  "take" : 0,  
  "get_or_fail_success" : 0,  
  "wait" : {  
    "avgcount" : 0,  
    "sum" : 0  
  },  
  "get" : 2637,  
  "take_sum" : 0,  
  "put_sum" : 82760  
}
```

extract metrics from logs

Logster etc

structured logging will help a lot

collecting metrics

statsd

graphite

bucky

collectd

more integration coming



Conclusion



ceph is hard

so far from status quo

not a lamp stack

not just client-server

ceph is easy

so far from status quo

we can change the product, not
just packaging:

- mon initial members

- uuids

- ceph osd create

- bootstrap keys

high availability is in base design



devops like you mean it

e.g. ceph osd crush

create-or-move will

probably go into C++ once

we're comfortable with it

developers without borders

Thank you

Questions?

Credits

Photo of flames by Kamil Porembiński, CC BY-SA 2.0
<http://www.flickr.com/photos/paszczak000/2802131237/in/photostream/>

Photo of dorodango by Kelly Taylor, CC BY-SA 2.0
http://www.flickr.com/photos/wmshc_kiwitayro/663355489/

Photo of traffic light by David Lofink, CC BY 2.0
<http://www.flickr.com/photos/lofink/4447262258/>