CeTune User Guide

Revision <2.1>

Last Print Date: 11/11/2016 - 3:14 PM

Primary Author(s): **Chendi, Xue Chendi.xue@intel.com**

1. Table of Contents

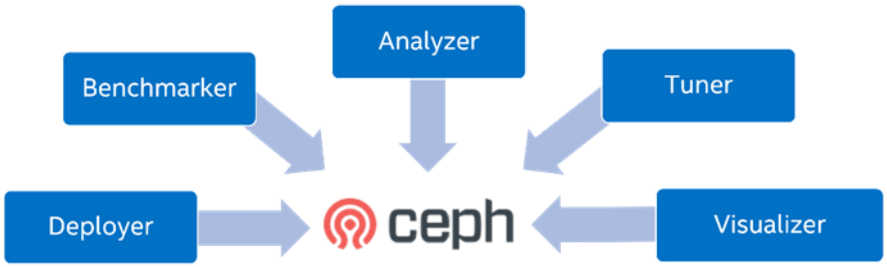
1. Introduction of CeTune

1.1 What is CeTune

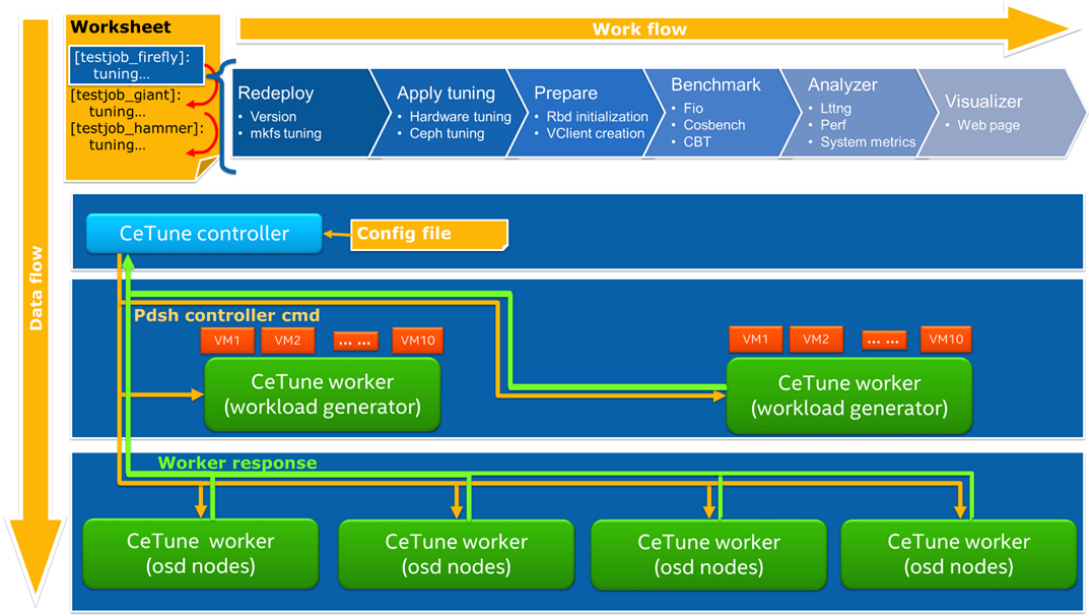
CeTune is a python-based framework, designed and implemented to profile and tune ceph cluster performance. This is the v1 version of CeTune, which is rewrite from the original shell-based Automation kit (cephperf).

In CeTune, we designed it to be kicked off by one click from deploy ceph cluster to benchmark and profile ceph and show performance report with HTML.

To reach that goal, CeTune comprises five distinct components. And using two config files to drive all these five components.



1.2 Terms definition

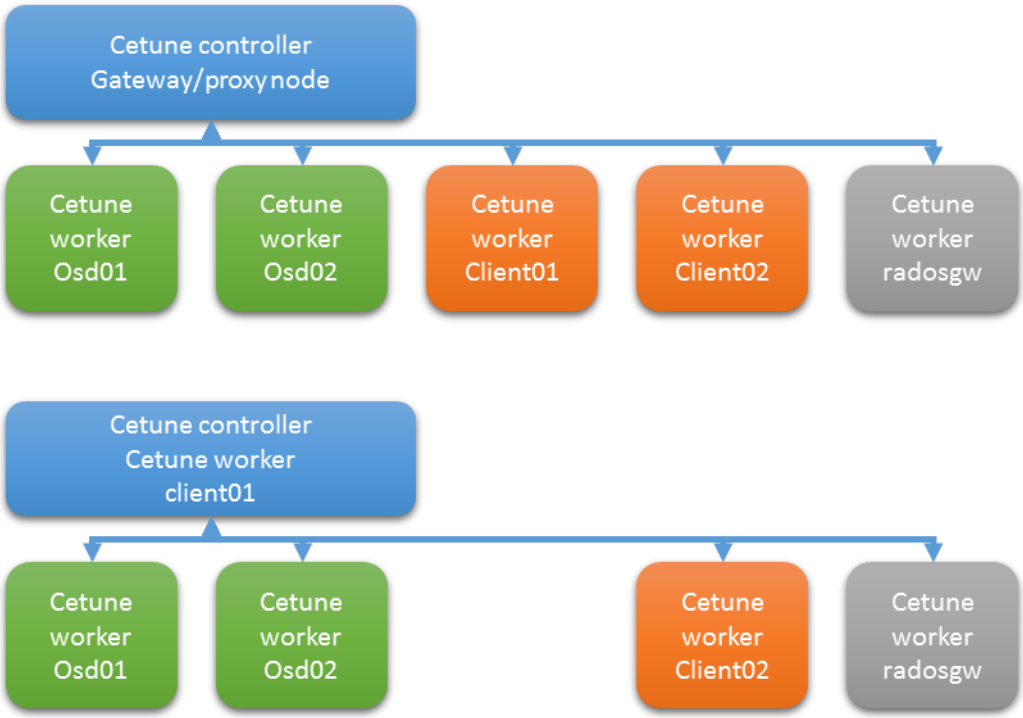


[CeTune Controller] : Is the node who is responsible to communicate with all other nodes(nodes like ceph osd, rgw, client, cosbench workers, etc.). Controller is the only node need to be able to record all other nodes ip address into hosts file and be able to auto ssh to these nodes without asking for password each time.

Controller node can be any node in ceph cluster or even an individual node. We recommend to put CeTune controller at one of your client node. (Our test is on this scenario, so this will be more stable and bugless.)

[CeTune Worker] : Are nodes participate in the benchmark, ceph osd nodes, monitor nodes, radosgw nodes, client, cosbench controllers and workers all can be seen as CeTune worker. These nodes receive job from CeTune controller by ssh and reply with the system return values.

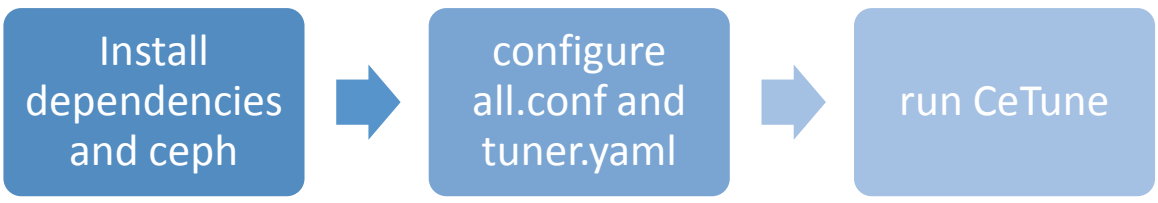
Two scenarios are both workable under CeTune View:



1.3 How to use CeTune

CeTune is being able to help engineers doing deployment, benchmarking, vm setting up and other works like autossh, disk_partition.

We aim to make using CeTune as simple as possible, with only three major steps, you can run CeTune to do a whole test from zero ceph in cluster to complete the ceph performance report.



2. Preparations (install dependencies and ceph)

1.4 build ceph

Refer to Ceph Document: http://docs.ceph.com/docs/jewel/install/build-ceph/

```
##git ceph:
git clone --recursive https://github.com/ceph/ceph.git
##ceph version set:
cd ceph
git tag
git checkout v10.2.0
##build prerequisites:
./install-deps.sh
##build ceph:
./autogen.sh
./configure
##get cpus nums
cat /proc/cpuinfo | grep pro | wc -l
make -j cpus*2
make install
```

1.5 Install ceph-deploy and dependencies

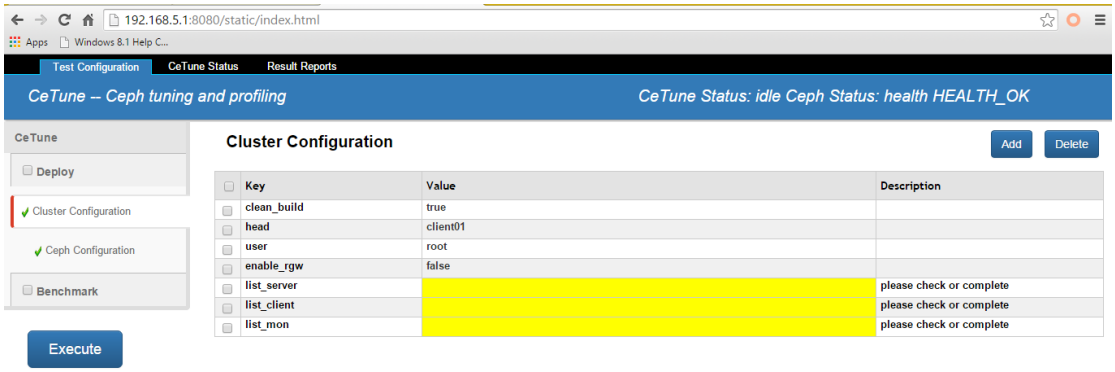
Ceph-deploy will be called from CeTune scripts to install and deploy ceph, so before using CeTune, make sure the ceph-deploy is installed.

```
[CeTune_controller]
sudo apt-get install git
git clone https://github.com/01org/CeTune.git
cd /CeTune/deploy/
python controller_dependencies_install.py
[CeTune_worker]
sudo apt-get install git
git clone https://github.com/01org/CeTune.git
cd /CeTune/deploy/
python worker_dependencies_install.py
```

1.6 Install and deploy CeTune webui

```
[CeTune_controller]
##install webpy python module
git clone https://github.com/webpy/webpy.git
cd webpy
python setup.py install
##run CeTune webui
cd {your-path}/CeTune/webui/
Python webui.py
##you will see below output
root@client01:/CeTune/webui# python webui.py
http://0.0.0.0:8080/
```

then, you can access this node:8080 by browser



For the first time login CeTune WebUI, you will need to fill out some configuration to describe your cluster, all these configuration will stored as all.conf and tuner.yaml files under CeTune/conf/ dir.

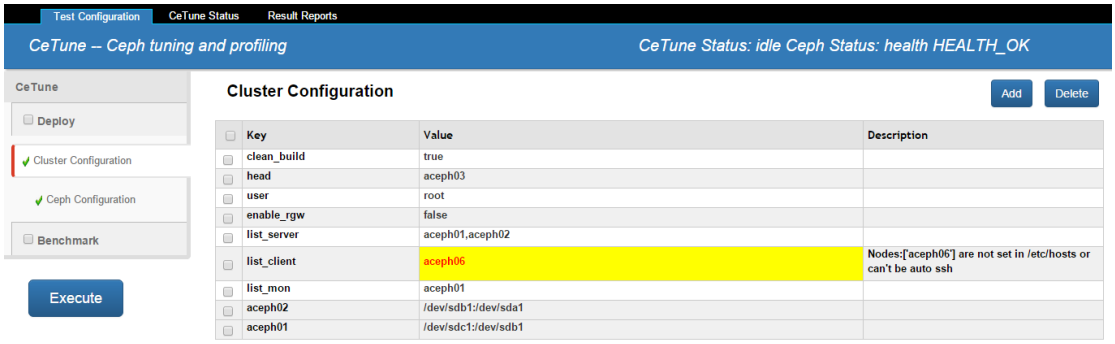
1.7 Prepare Cluster

CeTune Web UI provides a new way to prepare your cluster, you can configure on the WebUI at the same time, WebUI can help to check if you miss some preparation on the cluster.

Basically, the MUST DO preparations are:

a) Enable the auto ssh to all ceph nodes.

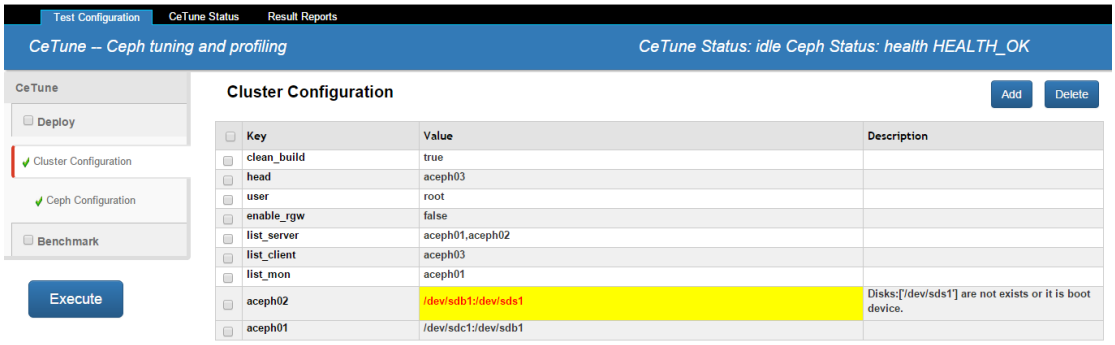
Which means the node you fill into the CeTune WebUI should also be auto ssh by CeTune Controller, or you will see below warning.




```
apt-get install expect
#configure auto ssh by below command
cd deploy/prepare-scripts; ./configure_autossh.sh ${host} ${ssh_password}
must do: make sure the local host 127.0.0.1 is also being auto-sshed
./configure_autossh.sh 127.0.0.1 ${ssh_password}
```

b) make sure you had already do the partition to the device name you input into CeTune WebUI.

When you fill in the list_server slot, CeTune will create new configuration line whose key name is the osd node name, you should input osd_journal pair to these lines.



If the device not exists, or being a boot device, CeTune WebUI will give you warning like above.
CeTune provides a partition script, please follow below to do partition.

```
[CeTune_controller]
# Fill all nodes list in list_server with desired osd_journal pair in conf/all.conf
# you can only use cli not webui
vim conf/all.conf
#config each osd node with osd_device and journal_device
osd_node1=${osd_device}:${journal_device}
example:
aceph01=/dev/sda1:/dev/sdb1,/dev/sdd1:/dev/sdb2,/dev/sde1:/dev/sdb3…
# how many partitions on one osd device and the size
# CeTune has script to do partitioning
# set the osd_partition_size to “ ”, CeTune will use whole disk space as one partition
osd_partition_count=1
osd_partition_size=2000G
# how many partitions on one journal device and the size
journal_partition_count=5
journal_partition_size=60G

cd deploy/prepare-scripts;
bash list_disk_partition.sh -l  #To check current osd and journal device partition
bash list_disk_partition.sh -w  #apply all.conf to osd and journal device
```

c) vm setup if necessary

If you plan to do rbd test inside VM, you need to prepare VM before list them into CeTune, all CeTune will warn you on these VM nodes not existing, Just refer to 3. VM setup

3. VM setup

If you plan to run qemufio benchmark on ceph cluster, CeTune also provides some scripts to setup VM, skip this chapter if you not plan to run fio in VM.

1.8 Set up hypervisor bridge network.

a) Edit /etc/network/interfaces file.

Add a new ‘br0’ network who bridge_ports to a physical nic (In the example, it points to ‘eth0’), and copy the original physical nic(eth0) address, network, gateway settings to br0. Example is like below, ‘eth0’ should be replaced by the nic can connect to network in your own system.

```
[Ubuntu]
auto eth0
iface eth0 inet manual

auto br0
iface br0 inet static
address 192.168.5.31
netmask 255.255.0.0
gateway 192.168.2.200
bridge_ports eth0
bridge_stp off
bridge_fd 0
bridge_maxwait 0

[CentOS]
[root@client03 ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
TYPE=Bridge
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.5.31
NETMASK=255.255.0.0
DELAY=0
GATEWAY=192.168.2.200
[root@client03 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:1e:67:92:3c:2d
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
USERCTL=no
```

b) Restart the network service.

```
[root@client01 ~]# ifdown eth0
[root@client01 ~]# ifup eth0
Possible Error:
Ignoring unknown interface br0=br0.
Solution:
```

```
[root@client01 ~]# apt-get/yum install bridge-utils
```

c) Check the network setting

```
# System route to 192.168.0.0 by br0
root@ceph-client1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.2.200   0.0.0.0         UG    0      0        0 br0
172.16.96.0     *               255.255.240.0   U     0      0        0 eth2
172.16.96.0     *               255.255.240.0   U     0      0        0 eth5
192.168.0.0     *               255.255.0.0     U     0      0        0 br0
192.168.122.0   *               255.255.255.0   U     0      0        0 virbr0
```

1.9 Prepare VMs

d) Use the scripts in vm-scripts to prepare vm

```
[CeTune_controller]
Set conf/all.conf to create vm image
vim conf/all.conf
list_vclient=vclient01,vclient02,vclient03,vclient04…
cpuset_start=0  #when pin vm vcpu to hypervisor cpu, will start from cpu 0
vm_num_per_client=40  #after create 40 vms, will re-pin vcpu to hypervisor cpu 0
img_path_dir=/mnt/images  #output created vm image folder
ip_prefix=192.168.5 #ip_prefix and ip_fix specify the vm ip,
ip_fix=161          #in below case, vm ip start from 192.168.5.161
vm_image_locate_server=10.239.158.45    #the remote_dir of tmp_vclient.image

cd vm-scripts/
bash prepare-vm.sh
scp –r vmxml/ ${img_path_dir_mnt} ${client_node}  #scp vclient.xml and vclient.img to clients
```

1.10 Create VMs

```
virsh create {xml-file-path}
```

Tip: check the file /etc/hosts in one of clients(the one which has dir CeTune), make sure it had recorded all IP addresses of VMs. After the creating of VMs, please guarantee that all the VMs can be logged on with SSH.

```
[example]
for i in `seq 1 4`; do virsh create vmxml/vclient${i}.xml;done
Domain vclient01 created from vmxml/vclient01.xml
Domain vclient02 created from vmxml/vclient02.xml
Domain vclient03 created from vmxml/vclient03.xml
Domain vclient04 created from vmxml/vclient04.xml

root@ceph-client1:~/CeTune/vm-scripts# virsh list
 Id    Name                           State
```

```
----------------------------------------------------
 19    vclient01                      running
 20    vclient02                      running
 21    vclient03                      running
 23    vclient04                      running
```

1.11 Rbd volume creation

Below is the instruction mostly used for deploy ceph with cephx authentication.

e) Create rbd volume

This script will read the current rbd volume number and vclient number in all.conf, then create proper number of rbd volumes to ensure each vm has one rbd volume. So if you has enough rbd volume for vclient, this script will do nothing.

```
bash create-volume.sh create_rbd
```

f) Create rbd_disk.xml

```
#create rbd volume xml with cephx authentication
bash create-volume.sh create_disk_xml
If you use CephX, pls make sure the secret.xml locates in vm-scripts
1) secret.xml exists, continue with cephx
2) help to generate secret.xml first than create volume
3) continue with none auth
#? 2
Secret b9e48882-7485-4312-ac5b-4148409d59e1 created
Secret value set

cat vdbs/vclient01.xml
<disk type='network' device='disk'>
    <driver name='qemu' type='raw' cache='none'/>
    <auth username='admin'>
        <secret type='ceph' uuid='b9e48882-7485-4312-ac5b-4148409d59e1'/>
    </auth>
    <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a' />
    <target dev='vdb' bus='virtio'/>
    <serial>009ad738-1a2e-4d9c-bf22-1993c8c67ade</serial>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</disk>

#create rbd volume xml with none authentication
bash create-volume.sh create_disk_xml
If you use CephX, pls make sure the secret.xml locates in vm-scripts
1) secret.xml exists, continue with cephx
2) help to generate secret.xml first than create volume
3) continue with none auth
#? 3
```

```
cat vdbs/vclient01.xml
<disk type='network' device='disk'>
    <driver name='qemu' type='raw' cache='none'/>
    <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a' />
    <target dev='vdb' bus='virtio'/>
    <serial>009ad738-1a2e-4d9c-bf22-1993c8c67ade</serial>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</disk>
```

g) rbd volume attach

```
virsh attach-device ${vclient} ${vclient.xml}
```

```
[example]
for i in `seq 1 4`; do virsh attach-device vclient0${i} vdbs/vclient${i}.xml; done
Device attached successfully
Device attached successfully
Device attached successfully
Device attached successfully

#check the rbd volume is attached to vm successfully
for i in `seq 1 4`; do echo vclient0${i}; virsh dumpxml vclient0${i} | grep rbd; done
vclient01
     <source protocol='rbd' name='rbd/volume-72a2a44b-ab14-4447-a49c-2bf7a80bde8a'/>
vclient02
     <source protocol='rbd' name='rbd/volume-a856d868-a065-4127-8b7b-c0d1dfb06e77'/>
vclient03
     <source protocol='rbd' name='rbd/volume-ad0c107a-8d8a-47eb-b558-23ad7ee61967'/>
vclient04
     <source protocol='rbd' name='rbd/volume-db060f87-08df-4c8c-ba6d-6f7a94d64531'/>
```

4. Ceph installation and deployment

1.12 Ceph installation

Current CeTune installs ceph by ceph-deploy and only support installing rbd as client.

a) Check the network connection

Please edit the files /etc/apt/apt.conf, /etc/environment and /etc/wgetrc to make sure that the proxy of all node in your cluster are available, including client and server, and if your system is CentOS, you need to edit your file /etc/yum.conf to make sure your proxy is available.

```
[CeTune_worker]
Check apt-get / yum is able to connect to the repository
Wget ceph.com #make sure wget is able to connect to ceph.com
```

Add all Ceph node include osd, mon, client into [CeTune controller]'s hosts file

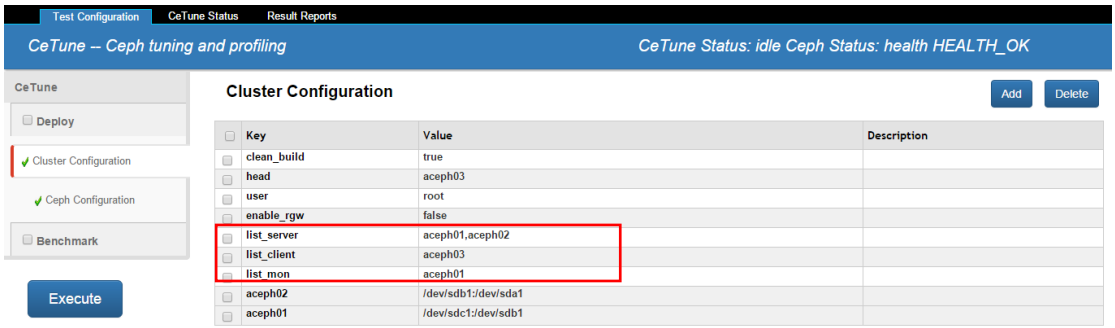
If your cluster using different nic driver as cluster network and public network, remember to add all nic driver ip into /etc/hosts.

```
[CeTune_controller]
make sure all your osd, mon, rbd client can be connected through the controller node.
Or you can use script in CeTune help you to do the autossh setting.

apt-get/yum install expect
Cd deploy/prepare-scripts; ./configure_autossh.sh ${host} ${ssh_password}
```

b) Configurate server list in all.conf

Fill in 1)list_server, 2)list_client, 3)list_mon, so CeTune will install ceph pkg to these nodes.



c) Install ceph

There are two interface to install ceph, by CeTune CLI or CeTune WebUI

Below is how install ceph by CeTune CLI, by CeTune WebUI, you can just forward to

```
[CeTune_controller]
cd deploy
python run_deploy.py install_binary --version hammer
```

d) Purge ceph

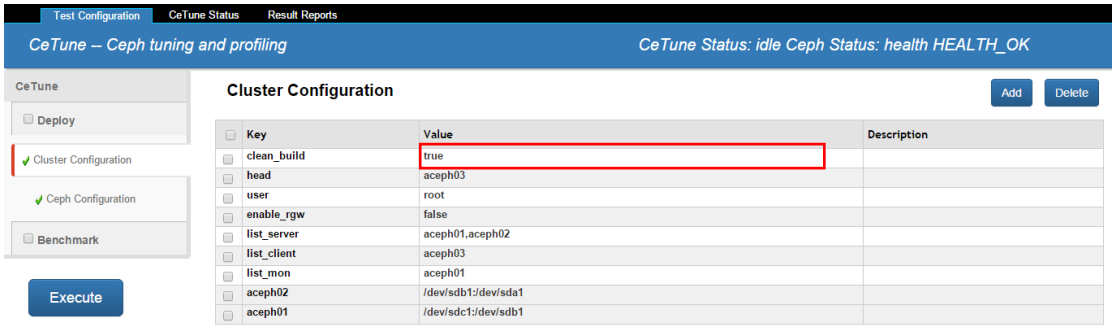
```
[CeTune_controller]
cd deploy
python run_deploy.py uninstall_binary
```

You can see ceph -v at the end if you succeeded in installing ceph

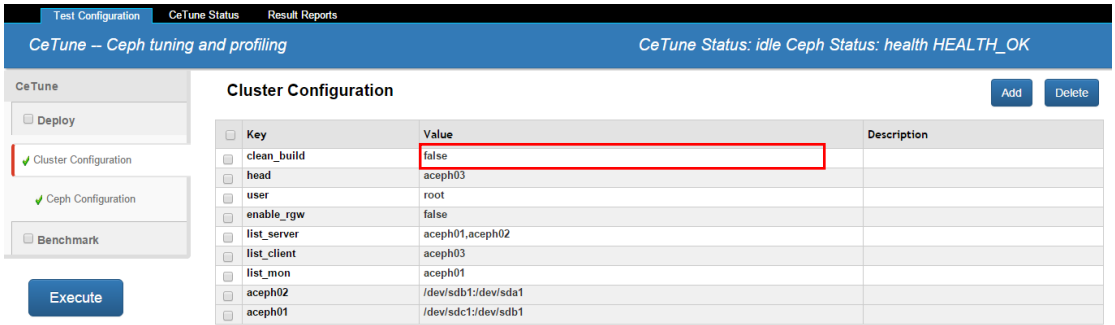
1.13 Ceph deployment

To deploy ceph cluster by CeTune UI, there are two mode: 1) clean_build 2)non_clean_build

Clean_build : this mode is for first time ceph deployment or you want to destroy your ceph cluster and build a new one, please set clean_build as true as below

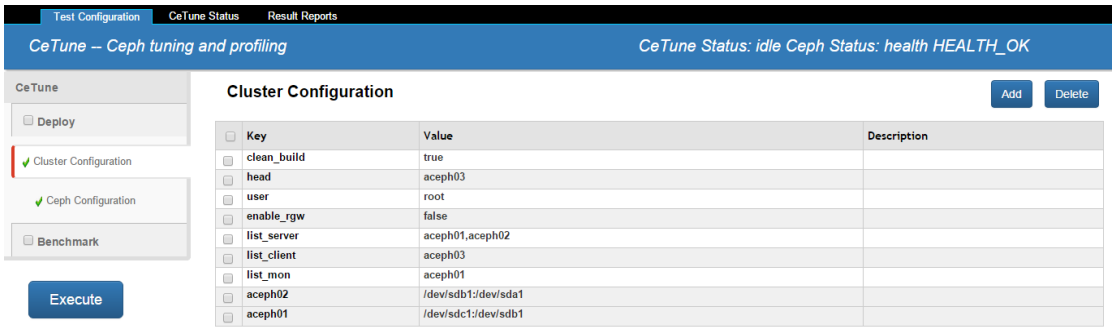


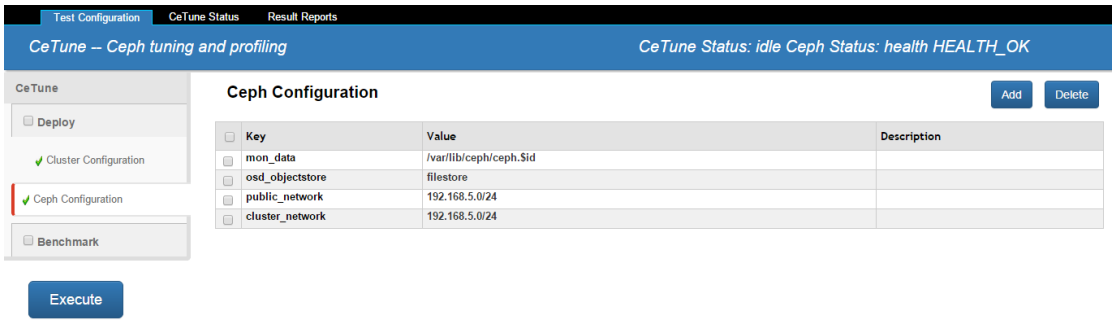
Non_clean_build: cetune try not to destroy your current cluster, and add osd, radosgw for the cluster.



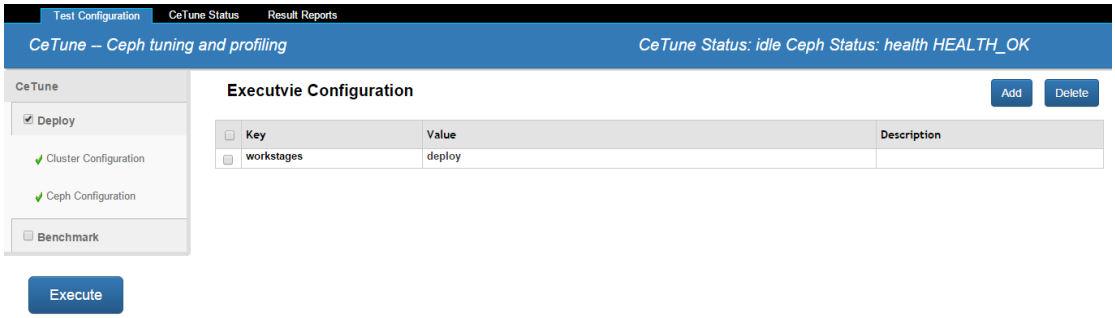
1.14 Clean build a new ceph cluster

a) Done configuring cluster configuration and ceph configuration



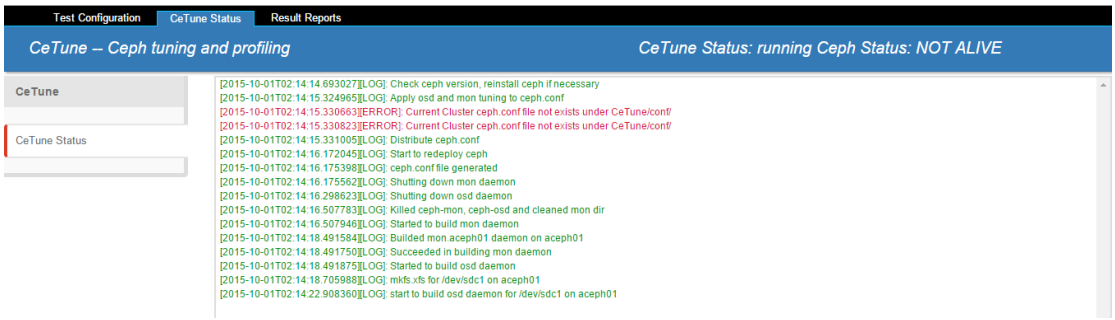


b) Click on Deploy

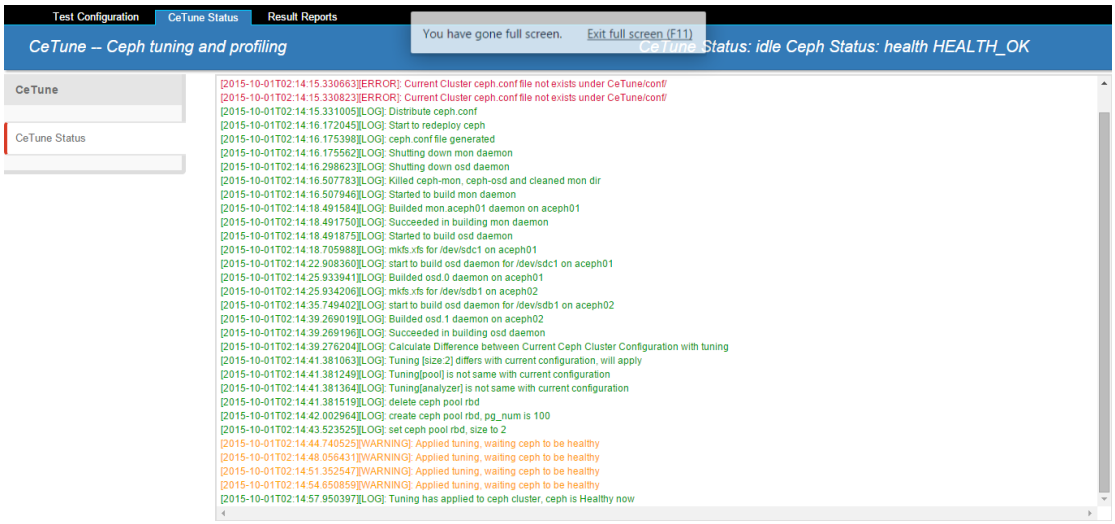


c) Click 'Execute'

After click, CeTune will redirect to CeTune Status tab, and Status here.



After deploying mon, osd, CeTune will apply rbd tuning to ceph cluster, then waiting ceph health to be OK.



Then ceph cluster is deployed.

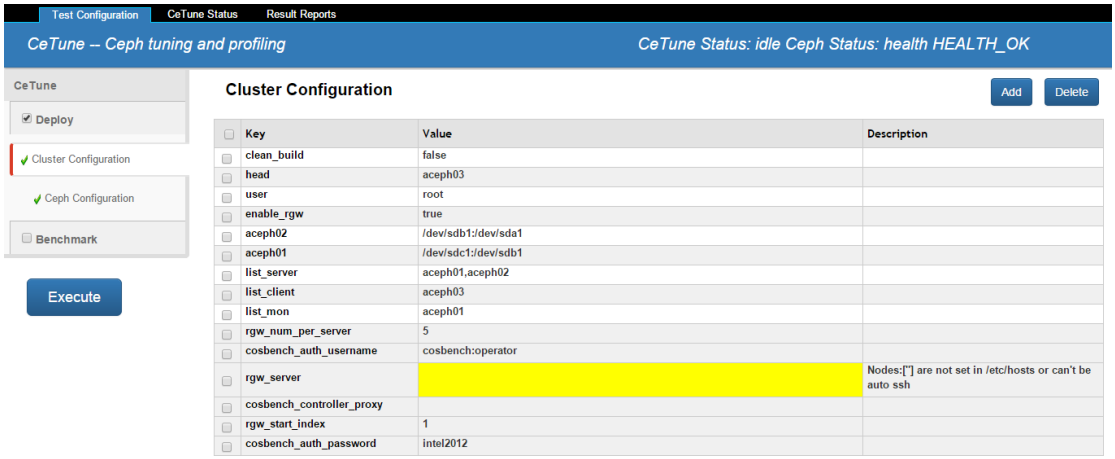
1.15 Non clean build to add radosgw

a) Clean_build to false

b) Enable 'enable_rgw' to true

After enable rgw, you need to fill the adding configuration lines.

cosbench_controller_proxy: leave this line blank, if you need to http_proxy to access radosgw



c) Click ‘Execute’

1.16 Non clean build to add osd

a) Clean_build to false

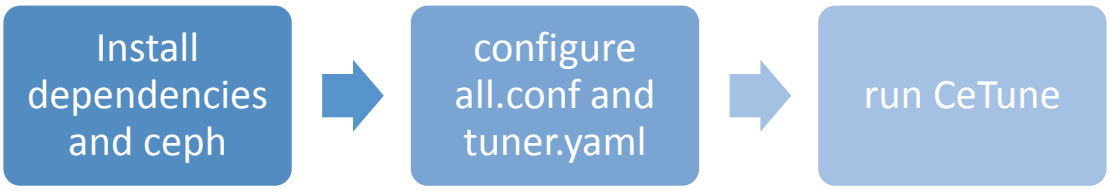
b) Add new osd node or device in cetune webui

c) Click ‘Execute’

5. Benchmark

1.17 Basic Execution Flow

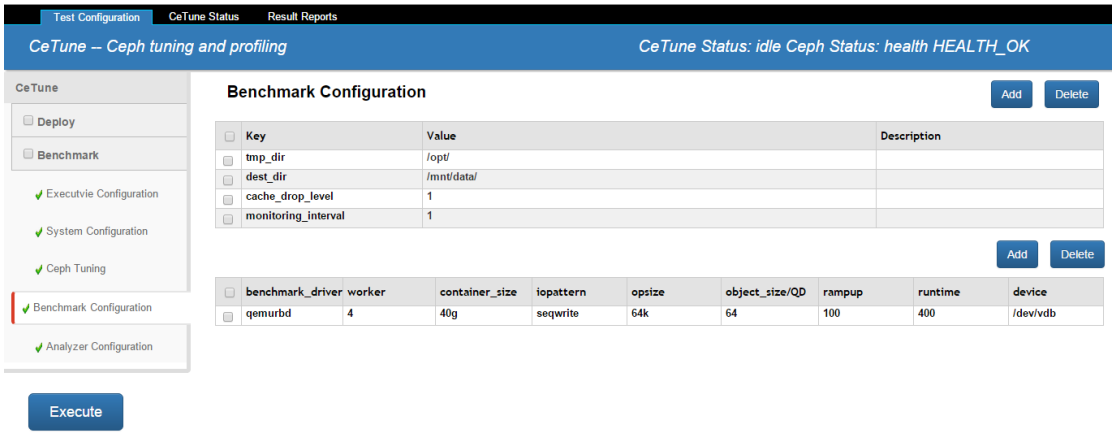
As mentioned at chapter 2, the basic execution flow to use CeTune is
(1)install, (2)configure, (3)run.
Till now, we had installed ceph and all the dependencies; and create vclient if necessary.
All remained to do is do the configuration and kickoff CeTune.



1.18 Configuration

a) Configure benchmark session in CeTune WebUI

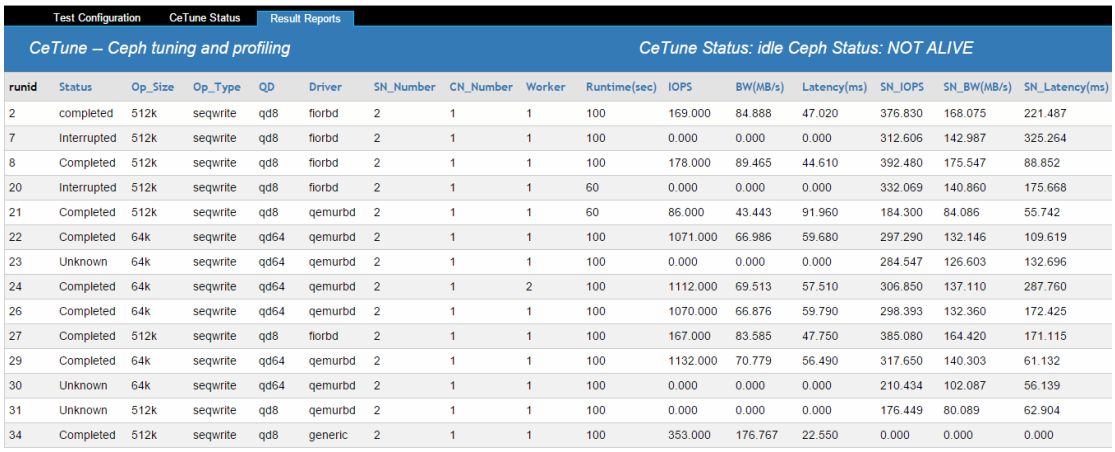
Go through each tab under Benchmark, and specify the testcase you desired under benchmark configuration. Like bolow



b) Click 'Execute'

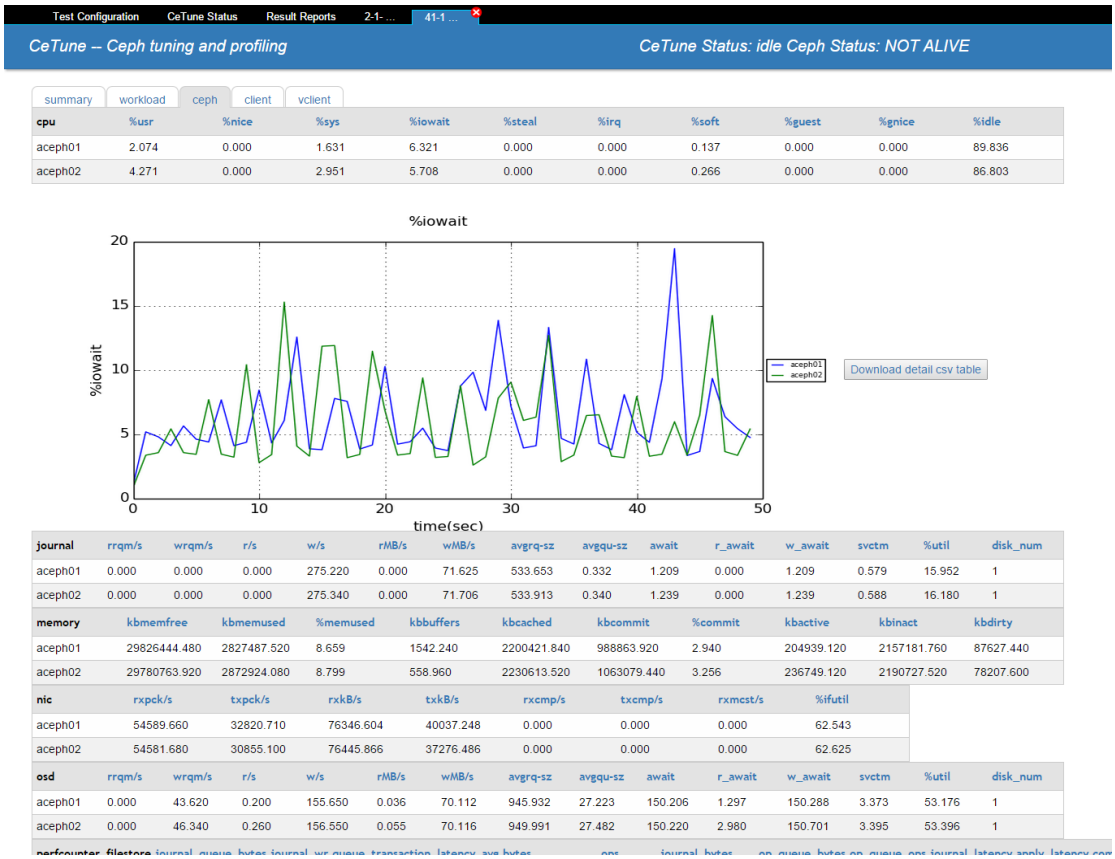
1.19 View Result

a) View CeTune result under 'Result Report'



runid	Status	Op_Size	Op_Type	QD	Driver	SN_Number	CN_Number	Worker	Runtime(sec)	IOPS	BW(MB/s)	Latency(ms)	SN_IOPS	SN_BW(MB/s)	SN_Latency(ms)
2	completed	512k	seqwrite	qd8	fiorbd	2	1	1	100	169.000	84.888	47.020	376.830	168.075	221.487
7	Interrupted	512k	seqwrite	qd8	fiorbd	2	1	1	100	0.000	0.000	0.000	312.606	142.987	325.264
8	Completed	512k	seqwrite	qd8	fiorbd	2	1	1	100	178.000	89.465	44.610	392.480	175.547	88.852
20	Interrupted	512k	seqwrite	qd8	fiorbd	2	1	1	60	0.000	0.000	0.000	332.069	140.860	175.668
21	Completed	512k	seqwrite	qd8	qemurbd	2	1	1	60	86.000	43.443	91.960	184.300	84.086	55.742
22	Completed	64k	seqwrite	qd64	qemurbd	2	1	1	100	1071.000	66.986	59.680	297.290	132.146	109.619
23	Unknown	64k	seqwrite	qd64	qemurbd	2	1	1	100	0.000	0.000	0.000	284.547	126.603	132.696
24	Completed	64k	seqwrite	qd64	qemurbd	2	1	2	100	1112.000	69.513	57.510	306.850	137.110	287.760
26	Completed	64k	seqwrite	qd64	qemurbd	2	1	1	100	1070.000	66.876	59.790	298.393	132.360	172.425
27	Completed	512k	seqwrite	qd8	fiorbd	2	1	1	100	167.000	83.585	47.750	385.080	164.420	171.115
29	Completed	64k	seqwrite	qd64	qemurbd	2	1	1	100	1132.000	70.779	56.490	317.650	140.303	61.132
30	Unknown	64k	seqwrite	qd64	qemurbd	2	1	1	100	0.000	0.000	0.000	210.434	102.087	56.139
31	Unknown	512k	seqwrite	qd8	qemurbd	2	1	1	100	0.000	0.000	0.000	176.449	80.089	62.904
34	Completed	512k	seqwrite	qd8	generic	2	1	1	100	353.000	176.767	22.550	0.000	0.000	0.000

b) Double click one line, you can review its detailed report



cpu	%usr	%nice	%sys	%iowait	%steal	%irq	%soft	%guest	%gnice	%idle
aceph01	2.074	0.000	1.631	6.321	0.000	0.000	0.137	0.000	0.000	89.836
aceph02	4.271	0.000	2.951	5.708	0.000	0.000	0.266	0.000	0.000	86.803

journal	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util	disk_num
aceph01	0.000	0.000	0.000	275.220	0.000	71.625	533.653	0.332	1.209	0.000	1.209	0.579	15.952	1
aceph02	0.000	0.000	0.000	275.340	0.000	71.706	533.913	0.340	1.239	0.000	1.239	0.588	16.180	1

memory	kbmemfree	kbmemused	%memused	kbbuffers	kbcached	kbcommit	%commit	kbactive	kbinact	kbdirty
aceph01	29826444.480	2827487.520	8.659	1542.240	2200421.840	988863.920	2.940	204939.120	2157181.760	87627.440
aceph02	29780763.920	2872924.080	8.799	558.960	2230613.520	1063079.440	3.256	236749.120	2190727.520	78207.600

nic	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcst/s	%ifutil
aceph01	54589.660	32820.710	76346.604	40037.248	0.000	0.000	0.000	62.543
aceph02	54581.680	30855.100	76445.866	37276.486	0.000	0.000	0.000	62.625

osd	rrqm/s	wrqm/s	r/s	w/s	rMB/s	wMB/s	avgrq-sz	avgqu-sz	await	r_await	w_await	svctm	%util	disk_num
aceph01	0.000	43.620	0.200	155.650	0.036	70.112	945.932	27.223	150.206	1.297	150.288	3.373	53.176	1
aceph02	0.000	46.340	0.260	156.550	0.055	70.116	949.991	27.482	150.220	2.980	150.701	3.395	53.396	1

6. Appendix

In the processing of installing ceph you may encounter the follow problems, and here are some tips for solving them.

1.20 Kickoff each module

Cetune modules like ‘deploy’, ‘benchmark’, ‘analyze’, ‘visualize’ can also run separately. Using below command to run

```
# Install ceph package
cd deploy; python run_deploy.py install_binary --version ${version}
# Uninstall ceph package
cd deploy; python run_deploy.py uninstall_binary
# Redeploy ceph cluster
cd deploy; python run_deploy.py redeploy
# Restart ceph cluster
cd deploy; python run_deploy.py restart
# Deploy ceph radosgw
cd deploy; python run_deploy.py deploy_rgw
# Generate ceph conf from tuner.yaml
cd deploy; python run_deploy.py gen_cephconf
```

```
# Send generated ceph conf to all osd and client node.
cd deploy; python run_deploy.py distribute_conf
# Generate test cases for cosbench and fio
cd benchmarking; python run_cases.py –option gen_case
# Run benchmark following the sequence in ../conf/cases.conf
cd benchmarking; python run_cases.py
# Do analyze on one cetune test result
cd analyzer; python analyzer.py –path ${path} process_data
# Do visualize on one cetune analyzed result, need result.json doc under ${path}
```

cd visualizer; python visualizer.py –path {$path} generate_summary_page

1.21 OS type ‘hvm’ unknow

When you creating vclient by virsh and encounter the error below:

error: unknown OS type hvm

error: internal error: no supported architecture for os type 'hvm'

```
[solution]
#Check virsh capabilities fisrt
virsh capabilities | grep os_type
<os_type>hvm</os_type>
#if not get above data, which means kvm is not installed correctly

apt-get/yum install qemu-kvm
modprobe kvm
modprobe kvm_intel
lsmod | grep kvm
kvm-ok
virsh capabilities  #then you can find os type is hvm
```

```
<guest>
  <os_type>hvm</os_type>
  <arch name='x86_64'>
    <wordsize>64</wordsize>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
```

1.22 operation failed: open disk image file failed

```
#Firewall may not being give privilege to virsh
/etc/init.d/apparmor teardown or aa-complain /etc/init.d/libvirt-bin
/etc/init.d/libvirt-bin restart
Re-create vclient
```

1.23 mon.aceph01 low disk space

```
#make sure worker System root directory have 30% disk could use.
```

1.24 How to install fio with rbd engine

```
#Make sure librbd-dev and librados-dev is installed, or you won’t be able to configure fio
with rbdengine
```

```
dpkg -l | grep librbd-dev
ii  librbd-dev   0.80.9-1trusty  amd64  RADOS block device client library (development files)
dpkg -l | grep librados-dev
ii  librados-dev 0.80.9-1trusty  amd64  RADOS distributed object store client library
(development files)
git clone https://github.com/axboe/fio.git
cd fio
./configure
make
make install
fio --enghelp
Available IO engines:
     cpuio
     mmap
     sync
     psync
     vsync
     pvsync
     null
     net
     netsplice
     libaio
     posixaio
     falloc
     e4defrag
     splice
     rbd
     sg
     binject
```

1.25 pgs stuck unclean

```
   root@aceph01:/var/log/ceph# ceph -s
    cluster 2f96f09f-d911-4c69-8236-053e6d15fb11
     health HEALTH_WARN 1024 pgs stuck unclean
     monmap e1: 1 mons at {aceph01=172.16.96.11:6789/0}, election epoch 2, quorum 0 aceph01
     osdmap e70: 16 osds: 16 up, 16 in
      pgmap v136: 1024 pgs, 1 pools, 0 bytes data, 0 objects
            587 MB used, 14896 GB / 14896 GB avail
                1024 active
```

```
[solution]
ceph osd dump | grep pool     #dump the information of pool
```

```
ceph osd tree   #check osd status
#if all above is ok, then
ceph osd pool set rbd size 1
#wait ceph to be healthy, then
ceph osd pool set rbd size ${replica_size}
```