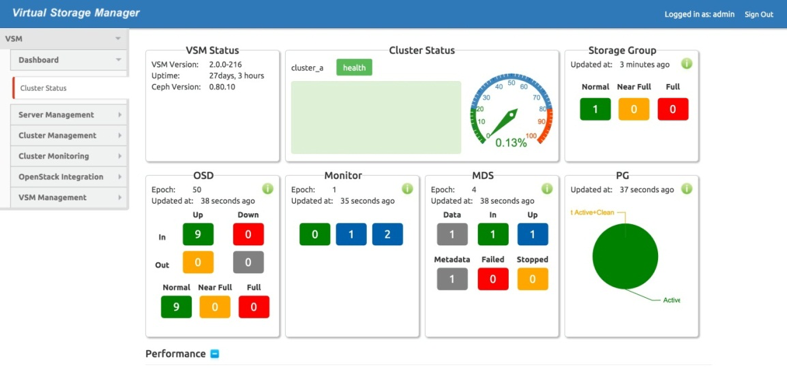
# Ceph开源管理监控平台分析

## 现状

Ceph 的开源管理监控平台有如 VSM（三年前最后更新，read-only），InkScope，Calamari,Suse-enterprise-storage（SUSE收费服务）、CEPH DASHBOARD，prometheus + grafana等；

## VSM（Virtual Storage Manager）



概览图

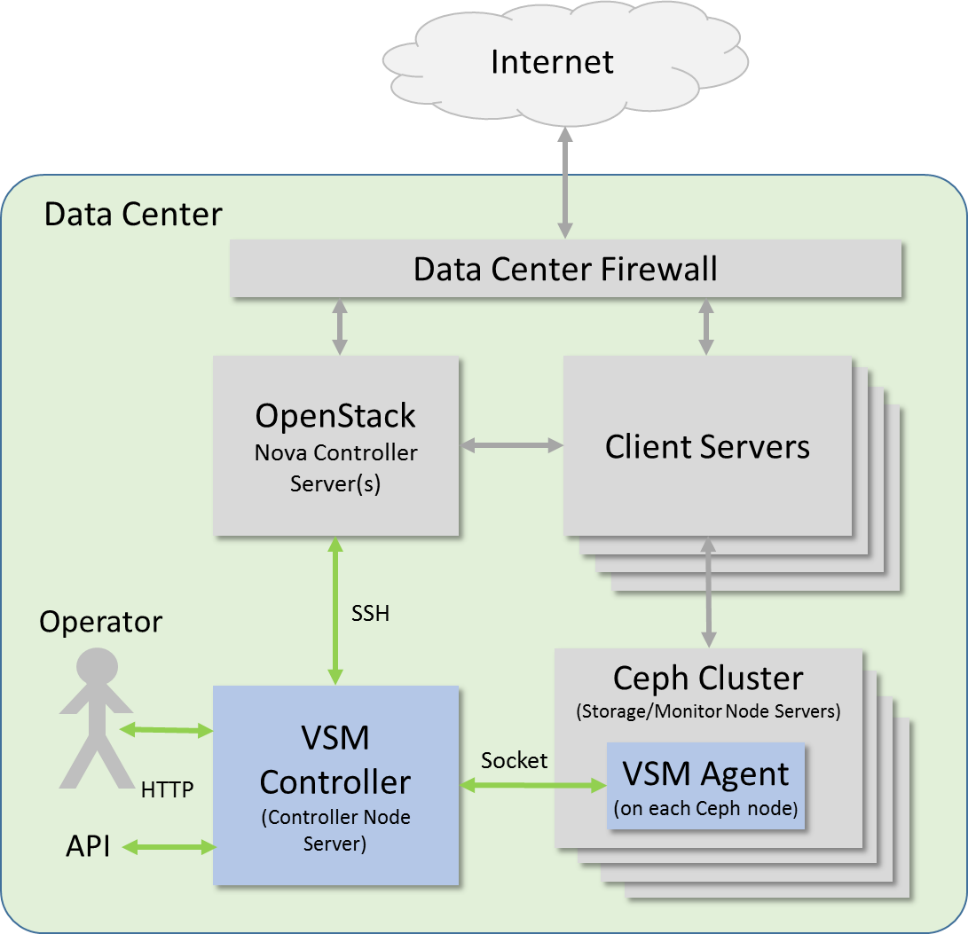
GitHub主页：

<https://github.com/intel/virtual-storage-manager>

### VSM架构

VSM的设计是完全按照OpenStack的架构设计。遵循一定的原则来设计这个架构：

* 容易与OpenStack的模块集成
* 完全与其它的OpenStack模块解耦合
* 易于合并到其他云平台
* 具有高可靠性，大规模数据中心的高可用性
* 易于使用，部署和管理



VSM架构图

VSM分为核心的两个部分，VSM控制节点与VSM存储节点。

#### VSM控制节点

* WebUI – 通过访问VSM REST API用于集群的管理、监控
* REST API – 供vsm client访问
* mariadb, rabbitmq

#### VSM存储节点

* 使用diamond收集ceph节点的监控信息
* vsm-agent工具对ceph节点进行管理

### 组件

* Dashboard（vsm-dashboard）：VSM的webUI界面，用户通过Dashboard来管理与监控ceph集群
* vsmclient（python-vsmclient）：VSM restapi 调用的client
* API（vsm）：VSM的restapi
* scheduler（vsm）：VSM的调度组件
* conductor（vsm）：VSM的数据库操作组件，即所有的数据库操作都是通过conductor来调用mysql
* RabbitMQ：消息中间件，VSM的各个组件相对独立，都是通过发送消息，通过RPC的方式来相互调用
* agent（vsm）：VSM代理服务

#### 组件特点

* 分布式：分开独立部署
* 无状态：各个请求独立，可扩展性强
* RESTFUL
* RPC
* plugin：插件式设计，松耦合

#### 组件代码

* VSM目前最新的发行版本为2.1，VSM的代码组件分为四个：
* vsm-dashboard：VSM的管理与监控web界面
* python-vsmclient：调用restapi的client
* vsm：VSM的核心组件（包括api、scheduler、conductor、agent等）
* vsm-deploy：ceph部署工具

代码基于Python语言，使用了wsgi、django等技术框架

### 支持功能

* 仪表盘：查看vsm、cluster、storage group、OSD、MON、MDS、PG的状态统计信息
  + 可以判断OSD是否正常运作，空间是否满
  + 查看IOPS、latency、bandwidth、CPU实时监控信息(通过diamond实现数据的收集)
  + 可以用来发现ntp延迟的问题
* 所有的宿主节点都需要在安装vsm的时候写在配置文件中
* 添加删除MON/OSD守护进程
* OSD 增删、重启、恢复（N/A）
* osd pool的管理 – 支持cache tier的增删、replicated/EC pool的创建
* StorageGroup的管理 – 添加新的SG，存储资源将以SG为单位进行统计
* 支持Ceph系统的升级功能，通过github下载源码实现
* 将通过ssh配置openstack的控制节点把rbd pool present给cinder
* 管理系统的临界值，将在dashboard中得到体现
* vsm 账户管理（keystone管理账户）

### 优缺点

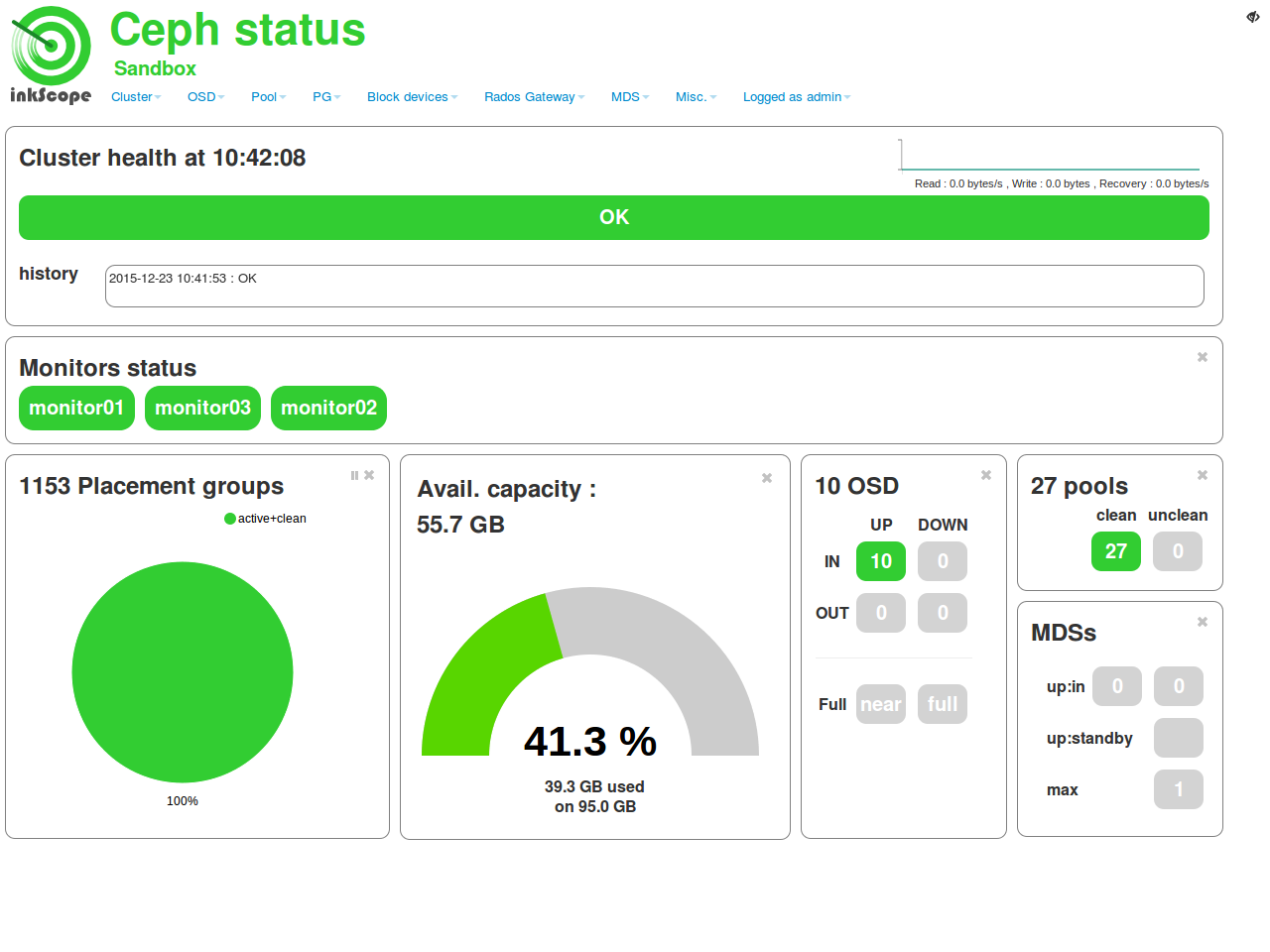
#### 优点：

1. 管理功能完善、充足
2. 界面友好
3. 可以部署Ceph和监控Ceph
4. 与OpenStack一脉传承，设计风格类似（详见架构部分说明）

#### 缺点：

1. 非官方，社区维护，且目前已处于归档状态（read-only）
2. 依赖OpenStack某些包和组件
3. 封装一套自己的rest-api，代码复杂度较高

## Inkscope

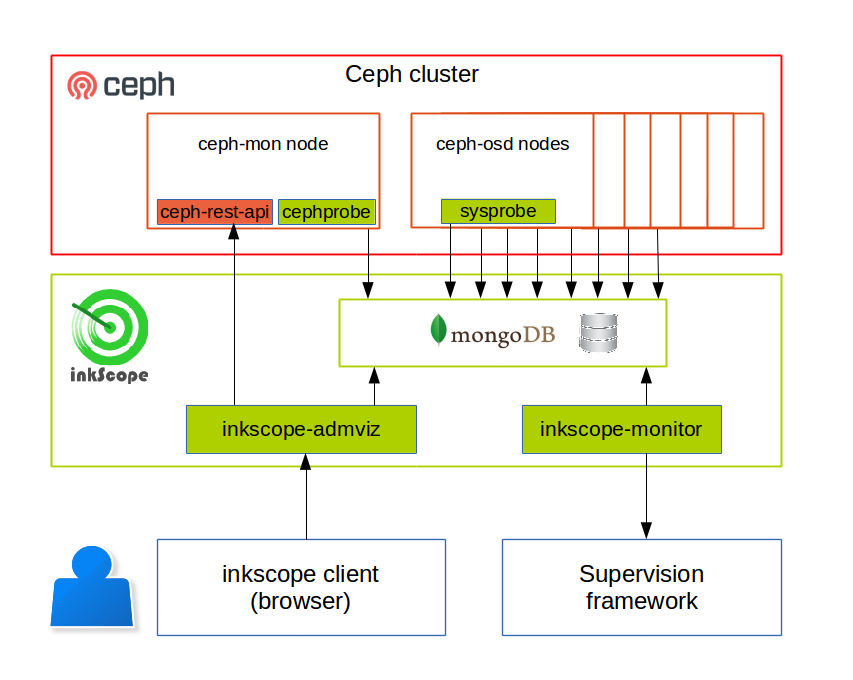


概览图

GitHub主页:

https://github.com/inkscope/inkscope

### Inkscope架构



Inkscope架构图

### 组件

* inkscope-common：包含inkscope的默认配置文件以及其他进程(cephprobe,sysprobe)启动所需的依赖文件，所有相关节点都需要安装。
* inkscope-admviz：包含inkscope的web控制台文件，含接口和界面，仅需要安装一个，该节点（管理节点）上同时需要按安装flask和mongodb
* inkscope-cephrestapi：用于安装启动 ceph rest api 的脚本，仅需要安装在提供api接口的节点上，即mon节点。
* inkscope-cephprobe：用于安装启动 cephprobe 的脚本(整个集群只需一个)，安装在mon节点，脚本主要实现：获取Ceph集群的一些信息，并使用端口（5000）提供服务，将数据存入mongodb数据库中。
* inkscope-sysprobe：安装用于所有mon和osd的sysprobe 所需要脚本，即所有节点均安装，实现获取节点设备资源信息如：CPU、内存、磁盘等等。

### 组件代码

* inkscopeViz：Web客户端
* inkscopeCtrl：inkscope的服务器端，提供了REST API
* inkscopeProbe：收集ceph节点的系统信息，收集到的数据将传输到MongoDB
  + cephprobe：用来或者集群的相关信息和操作的
  + sysprobe：获取节点的磁盘分区等相关信息的
* inkscopeMonitor：对接第三方监控框架

代码基于python语言，使用了wsgi、flask、MongoDB等技术框架

### 支持功能

* 仪表盘：查看cluster、OSD、MON、MDS、PG的状态统计信息可以判断OSD是否正常运作，空间是否满
* 分模块管理OSD、Pool、PG、RBD、RadosGW、MDS
* inkscope账户管理

### 优缺点

#### 优点：

1. 易部署

2. 轻量级

3. 灵活（可以自定义开发功能）

#### 缺点：

1. 监控选项少

2. 缺乏Ceph管理功能

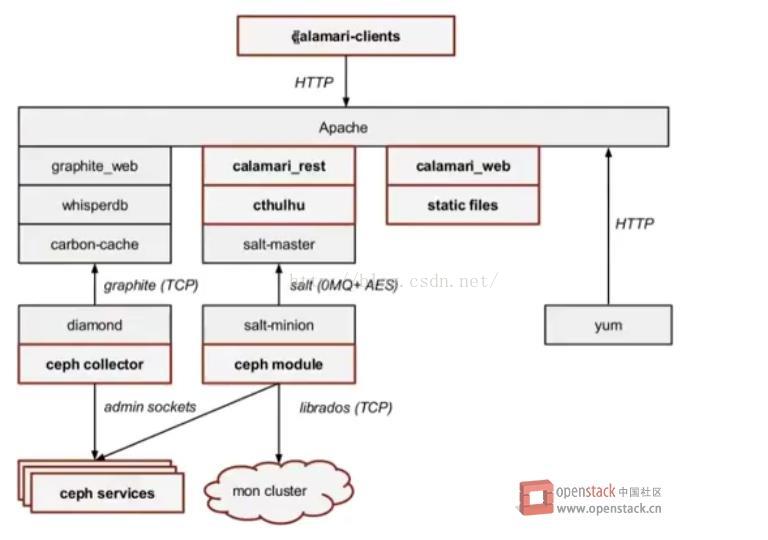
## Calamari



概览图

Github地址：<https://github.com/ceph/calamari>

### 架构



* Calamari 监控平台，使用Apache做服务器
* Calamari-server： 服务端
* Calamari-client： 客户端，包括了web端的UI，可以自己定制
* graphite: 收集数据的存储与展现，提供了接口，可以获取指定间隔的统计量。可以通过服务器地址单独访问 {calamari}/graphite/dashboard。包含在Calamari中
* Salt 服务器基础架构管理平台，具备配置管理、远程执行、监控等功能。
* Salt-master Salt的服务管理端
* Salt-minion Salt的节点端
* diamond 各个节点的数据收集器，收集节点信息发送到服务端。Calamari定制了一个版本。

### 组件

Calamari包含的组件主要有calamari-server；romana；salt-minion；salt-master；diamond。这些模块各自的作用：

calamari-server：这个是提供一个与集群进行交互，并且自己封装了一个自己的API，做集中管理的地方，这个只需要在集群当中的某一台机器上安装，也可以独立安装；

romana：就是原来的calamari-client，是一个web的界面，现在已经更名为romana，这个也是只需要在集群当中的某一台机器上安装，也可以独立安装，需要跟calamari-server安装在一台机器上；

salt-master：是一个管理的工具，可以批量的管理其他的机器，可以对安装了salt-minion的机器进行管理，在集群当中，这个也是跟calamari-server安装在一起的；

salt-minion：是安装在集群的所有节点上的，这个是接收salt-master的指令对集群的机器进行操作，并且反馈一些信息到salt-master上；

diamond：这个是系统的监控信息的收集控件，提供集群的硬件信息的监控和集群的信息的监控，数据是发送到romana的机器上的，是由romana上的carbon来收取数据并存储到机器当中的数据库中。

Graphite不仅是一个企业级的监控工具, 还可以实时绘图。Graphite后端运行一个名为carbon-cache.py的python程序，是高度可扩展的事件驱动的I/O架构的后端进程，负责处理客户端节点上的业务数据，它可以有效地跟大量的客户端通信并且以较低的开销处理大量的业务量。配置文件位于/etc/graphite/carbon.conf

Calamari 使用了Saltstack让Calamari Server和Ceph server node通信。Saltstack是一个开源的自动化运维管理工具，与Chef和Puppet功能类似。Salt-master发送指令给指定的Salt-minion来完成对Cpeh Cluster的管理工作；Salt-minion 在Ceph server node安装后都会从master同步并安装一个ceph.py文件，里面包含Ceph操作的API，它会调用librados或命令行来最终和Ceph Cluster通信。

cthulhu可以理解是Calamari Server的Service层，对上为API提供接口，对下调用Salt-master。但是代码美中不足的是calamari\_rest有些功能直接调用了Salt-master而没有调用cthulhu。calamari\_rest提供Calamari REST API，详细的接口请大家参照官方文档。Ceph的REST API是一种低层次的接口，其中每个URL直接映射到等效的CEPH CLI；Calamari REST API提供了一个更高层次的接口，API的使用者可以习惯的使用GET/POST/PATCH方法来操作对象，而无需知道底层的Ceph的命令；它们之间的主要区别在于，Ceph的REST API的使用者需要非常了解Ceph本身，而Calamari 的REST API更贴近对Ceph资源的描述，所以更加适合给上层的应用程序调用。

supervisord是一个允许用户监控和控制进程数量的系统程序。它可以指定一个服务如何运行。

romana（calamari\_clients）是一个提供web UI的模块，主要为客户端使用Calamari API提供服务，由salt-minion和diamond组成。

Diamond负责收集监控数据，它支持非常多的数据类型和metrics，通过查看源代码，它支持90多种类型的数据；每一个类型的数据都是上图中的一个collector，它除了收集Ceph本身的状态信息，它还可以收集关键的资源使用情况和性能数据，包括CPU，内存，网络，I / O负载和磁盘指标，而且还能收集很多流行软件的性能指标，包括 Hadoop, Mongo, Kafka, MySQL, NetApp, RabbitMQ, Redis, and AWS S3等。Collector都是使用本地的命令行来收集数据，然后报告给Graphite。

romana包括dashboard、login、admin、manage四大模块，构建rpm软件包时，这些模块缺一不可

dashboard是一个javascript的客户端，直接与ceph restful api交互来管理ceph。dashboard包含3个逻辑部分，分别为dashboard、workbench、graphs。

　　dashboard是一个只读的视图，负责展现ceph集群的健康状态

　　workbench是后台OSD和host的虚拟展现，最多限制展现256个OSD

　　graphs是有负责展示图形的graphite和负责在每个节点收集数据的diamond共同展示各种度量数据的视图

login模块用于登录web界面

admin模块用来管理用户和calamari信息的管理工具

manage模块用于管理ceph集群中的各种应用，如OSD管理、pool管理、集群设置和集群日志展现等功能

calamari\_clients是一套用户界面，Calamari Server在安装的过程中会首先创建opt/calamari/webapp目录，并且把webapp/calamari下的manager.py(django 配置)文件考进去， calamari\_web的所有内容到要放到opt/calamari/webapp下面来提供UI的访问页面。

calamari-web包下面的文件提供所有web相关的配置，calamari\_rest和calamari\_clients都要用到。

### 优缺点

#### 优点

1. 轻量级
2. 官方化
3. 界面友好

#### 缺点

1. 不易安装
2. 管理功能滞后
3. 提供的管理功能太少

Calamari为Ceph的运维和管理提供了一个统一的平台，而且用户还可以基于这个平台扩展自己的存储管理产品，但同时也存在着不足和需要改进的地方。

首先，Calamari还不能完成Ceph deploy所实现的部署功能，这是它最大一个不足；Fuel可以完成部署功能，并且可以选择Ceph server的数据盘和日志盘以及定制默认的备份数等，所以Calamari + Fuel可以来实现一个完成的基于Ceph的部署和管理工具。

其次，Calamari提供的管理功能太少，用户无法只使用它来运维一个Ceph环境。

最后，用户可以基于Calamari开发自己的Ceph管理软件，UI部分可以修改calamari\_clients的页面，也可也单独实现一套自己的UI基于calamari\_rest和Graphite\_web，后端的功能的监控部分可以扩展diamond的collector实现，管理Ceph的功能可以扩展rest api，cthulhu，salt等来实现。

### 安装参考

<https://www.gitbook.com/book/zphj1987/calamaribook>

<https://wiki.shileizcc.com/confluence/display/CEPH/Ceph%20Calamari>

# Ceph-Dash

Ceph-Dash 是用 Python 开发的一个Ceph的监控面板，用来监控 Ceph 的运行状态。同时提供 REST API 来访问状态数据。

# 

概览图

Github地址：<https://github.com/Crapworks/ceph-dash>

### 优缺点

#### 优点

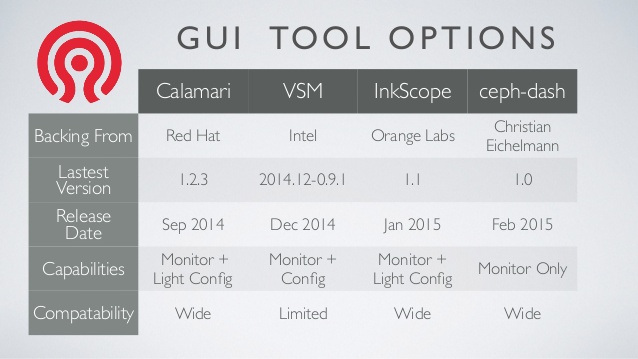
1. 易部署
2. 轻量级
3. 灵活（可以自定义开发功能）

缺点

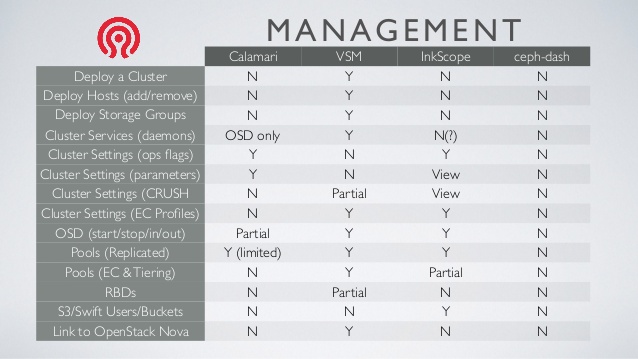
1. 只有监控功能，无管理功能（参考横向对比）

## 横向对比

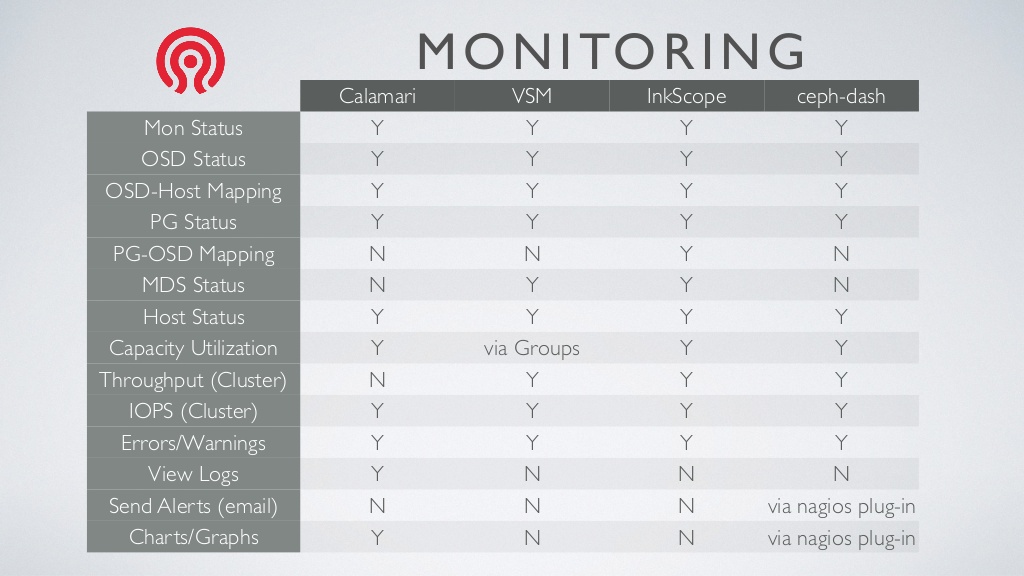
### 背景



### 管理



### 监控



## 一种声音

从 ceph 社区 qq 群看过去，总会有一些运维或者开发询问哪种 ceph 管理平台方便好用，然后就开始对比 inkscope、vsm、calamari。其实这些都不是重点，重点是看看 github 上的这些项目已经 long long ago 不更新代码了，也就是说软件的生命周期走到了尽头，没有更新和扩展。想想群里的兄弟在生产环境上用这些软件，最后是什么结果……况且大部分公司都是一两个码农在搬砖，投入到开发这三个监控平台上也不现实。

大部分生产环境都是用 cli 对 ceph 进行管理，所以生产环境对 ceph 的管理需求不大。在监控上，ustack 之前的文档提过了一套监控方案。建议关注一下https://prometheus.io/项目。前台集成 grafana，运维人员根据自己实际需求，DIY 监控面板，配合后端 exporters 很小的开发量，实现监控任意指标。报警方面 prometheus 也有自己的解决方案。联邦机制的实现可以使监控平台横向扩展。目前很多公司的生产环境都在用此方案。考虑到大规模运营，后续还需要 ELK 等工具的帮助。

<https://my.oschina.net/yangfanlinux/blog/783756>

## 参考资料

* <http://de.slideshare.net/Inktank_Ceph/07-ceph-days-sf2015-paul-evans-static>
* <http://www.hl10502.com/2017/03/30/ceph-web-manage/>
* <https://www.cnblogs.com/luxiaodai/p/10043183.html>