



























AWS控制台

 **Services** ▾ **Edit** ▾

Amazon Web Services

Compute & Networking <ul style="list-style-type: none"> Direct Connect Dedicated Network Connection to AWS EC2 Virtual Servers in the Cloud Elastic MapReduce Managed Hadoop Framework Route 53 Scalable Domain Name System VPC Isolated Cloud Resources	Database <ul style="list-style-type: none"> DynamoDB Predictable and Scalable NoSQL Data Store ElastiCache In-Memory Cache RDS Managed Relational Database Service Redshift NEW Managed Petabyte-Scale Data Warehouse Service	App Services <ul style="list-style-type: none"> CloudSearch Managed Search Service Elastic Transcoder NEW Easy-to-use Scalable Media Transcoding SES Email Sending Service SNS Push Notification Service SQS Message Queue Service SWF Workflow Service for Coordinating Application Components
Storage & Content Delivery <ul style="list-style-type: none"> CloudFront Global Content Delivery Network Glacier Archive Storage in the Cloud S3 Scalable Storage in the Cloud Storage Gateway Integrates On-Premises IT Environments with Cloud Storage	Deployment & Management <ul style="list-style-type: none"> CloudFormation Templated AWS Resource Creation CloudWatch Resource and Application Monitoring Data Pipeline Orchestration for Data-Driven Workflows Elastic Beanstalk AWS Application Container IAM Secure AWS Access Control OpsWorks NEW DevOps Application Management Service	

1 2 3 4 5 6 7 8 9 10

S3简介

2015年11月20日 16:18

功能:

搭建开源的S3服务器平台，应用到我们的系统上，使客户端可以通过一系列公用的s3客户端工具直接访问，映射fics的文件系统，这样的话可以舍弃单独的win/linux客户端，直接用形如DragonDisk，S3 Browser，cyberduck，Gladinet Cloud Desktop软件通过标准的S3协议访问fics文件系统；用户也可以通过各个语言平台上S3客户端的RPC/API直接访问云上的fics；前者属于一般的用户需求，后者属于给客户

提供SDK型的自定义用户需求。

(1) 创建桶、删除桶

(2) 写入、读取、删除数据元，每个数据元的大小从1byte到5Tbyte

(3) 根据用户密钥，查询桶信息及数据元信息

(4) 选择数据所存储的地区，东京 or 新加坡 or 其他

(5) 数据元的权限设置，可以向指定的用户开放，加密等

(6) 使用基于REST和SOAP接口，提供JAVA，net的SDK

(7) 便于添加其他功能，默认的下载协议是HTTP

(8) 提供AWS管理平台(S3 Console)，对数据进行管理

(9) 在进行存储或者检索数据时，对数据进行校验和计算，验证是否损坏

(10) 去冗余存储，用户可以自定义一些文件，比如缩略图、转码媒体等，使得这些文件的存储冗余副本数目低于标准的冗余副本数目

(11) 数据源的访问日志记录，可用于审核等

(12) 对数据元提供版本控制功能，用户可以恢复到之前的版本

(13) 包括客户端和服务端。

发起请求:

两种方式: 1.通过SDK接口。 2.通过REST接口

请求包头(如下图):

- 1.访问的KEY。
2. 签名。
3. 时间戳。

This section contains information specific to the Amazon S3 REST API.

The examples in this guide use the newer virtual hosted-style method for accessing buckets instead of the path-style. Although the path-style is still supported for legacy applications, we recommend using the virtual-hosted style where applicable. For more information, see [Working with Amazon S3 Buckets](#)

The following example is a virtual hosted-style request that deletes the `puppy.jpg` file from the `mybucket` bucket.

```
DELETE /puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: mybucket.s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: signatureValue
```

The following example is a path-style version of the same request.

```
DELETE /mybucket/puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization: signatureValue
```

协议请求头字段

Common Request Headers

The following table describes headers that can be used by various types of Amazon S3 REST requests.

Header Name	Description
Authorization	The information required for request authentication. For more information, go to The Authentication Header in the <i>Amazon Simple Storage Service Developer Guide</i> . For anonymous requests this header is not required.
Content-Length	Length of the message (without the headers) according to RFC 2616. This header is required for PUTs and operations that load XML, such as logging and ACLs.
Content-Type	The content type of the resource in case the request content in the body. Example: <code>text/plain</code>
Content-MD5	The base64 encoded 128-bit MD5 digest of the message (without the headers) according to RFC 1864. This header can be used as a message integrity check to verify that the data is the same data that was originally sent. Although it is optional, we recommend using the Content-MD5 mechanism as an end-to-end integrity check. For more information about REST request authentication, go to REST Authentication in the <i>Amazon Simple Storage Service Developer Guide</i> .
Date	The current date and time according to the requester. Example: <code>Wed, 01 Mar 2006 12:00:00 GMT</code> . When you specify the <code>Authorization</code> header, you must specify either the <code>x-amz-date</code> or the <code>Date</code> header
Expect	When your application uses 100-continue, it does not send the request body until it receives an acknowledgment. If the message is rejected based on the headers, the body of the message is not sent. This header can be used only if you are sending a body. Valid Values: 100-continue
Host	For path-style requests, the value is <code>s3.amazonaws.com</code> . For virtual-style requests, the value is <code>BucketName.s3.amazonaws.com</code> . For more information, go to Virtual Hosting in the <i>Amazon Simple Storage Service Developer Guide</i> . This header is required for HTTP 1.1 (most toolkits add this header automatically); optional for HTTP/1.0 requests.
x-amz-content-sha256	When using signature version 4 to authenticate request, this header provides a hash of the request payload. For more information see Authenticating Requests by Using the Authorization Header (Com-

s3中最复杂的应该是ACL和policy权限控制

<http://qinxuye.me/article/detail-about-amazon-s3-rest-api/>

来自 <<http://qinxuye.me/article/detail-about-amazon-s3-rest-api/>>

- 模块：
 - buckets
 - object

数据元的生命周期管理：

简单的说就是数据源失效的时间，例子如下：

```
<LifecycleConfiguration>
  <Rule>
    <ID>Example Rule</ID>
    <Prefix>projectdocs/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

数据版本：

在同一个桶中，用户可以拥有具有相同key的两个数据元对象，但是二者的版本号不同。版本是针对桶级别的，一旦设置了，桶下面所有的存储对象都会受到影响。在请求头中添加enabled状态或者put bucket中添加versioning

删除：

如果想要永久性的删除，必须要指定版本号才行。

下图表示，新版本的S3将不支持soap，推荐使用REST API或者亚马逊的SDK API（WSDL针对SOAP）

The location of the latest Amazon S3 WSDL is <http://doc.s3.amazonaws.com/2006-03-01/AmazonS3.wsdl>.

Note

SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs

来自 <<http://blog.csdn.net/anhuidelinger/article/details/9831861#t3>>

REST和SOAP

2015年11月20日 16:37

SOAP: 简单对象访问协议，简单对象访问协议（SOAP）是一种轻量的、简单的、基于 XML 的协议，它被设计成在 WEB 上交换结构化的和固化的信息。 SOAP 可以和现存的许多因特网协议和格式结合使用，包括超文本传输协议（ HTTP）， 简单邮件传输协议（SMTP）， 多用途网际邮件扩充协议（MIME）。它还支持从消息系统到远程过程调用（RPC）等大量的应用程序。

REST: 即REST(Representational State Transfer表述性状态转移)是一种针对网络应用的设计和开发方式，可以降低开发的复杂性，提高系统的可伸缩性。

REST 与SOAP的比较:

- 成熟度

SOAP目前成熟，不同平台，开发语言之间通过SOAP来交互的web service都能够较好的互通。REST相对不太成熟，由于没有类似于SOAP的权威性协议作为规范，REST实现的各种服务风格不一，通用性不强。

- 效率和易用性

SOAP使用门槛高（学习成本高，开发难度大），由于SOAP由于各种需求不断扩充其本身协议的内容，在大并发下性能有所下降。REST 目前大量的Web 2.0网站使用，高效以及简洁易用。这种高效一方面源于其面向资源接口设计以及操作抽象简化了开发者的不良设计，同时也最大限度的利用了Http最初的应用协议设计理念。REST 是一种轻量级的Web Service架构风格，其实现和操作明显比SOAP和XML-RPC更为简洁，可以完全通过HTTP协议实现，还可以利用缓存Cache来提高响应速度，性能、效率和易用性上都优于SOAP协议。

- 安全性

SOAP在安全方面是通过使用XML-Security和XML-Signature两个规范组成了WS-Security来实现安全控制的，当前已经得到了各个厂商的支持，.net ， php ， java 都已经对其有了很好的支持。REST没有任何规范对于安全方面作说明。因此在考虑安全性上，SOAP要高于REST。

总的来说，我认为REST对于资源型服务接口来说很合适，同时特别适合对于效率要求很高，但是对于安全要求不高的场景。而SOAP的成熟性可以给需要提供多开发语言的，对于安全性要求较高的接口设计带来便利。

来自 <<http://quanzhong.iteye.com/blog/1028114>>

总结：简单来说，REST就是用URL和请求的方法作为操作命令，取代soap中复杂的xml交互语法。

S3 Console

2015年11月24日 15:49

s3客户端工具,

S3 bucket策略

2015年11月24日 15:54

策略用来定义访问者的访问权限，有四种策略分别为：
oIAM Policy, oSNS Topic Policy, oSQS Queue Policy
策略用策略语言来编写。如下

```
{
  "Id": "Policy1393570093893",
  "Statement": [
    {
      "Sid": "Stmt1393569661962",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::bucket01/*",
      "Principal": {
        "AWS": [
          "*"
        ]
      }
    }
  ]
}
```

你需要bucket策略授权或者拒绝账户读取和上传文件到你的bucket中

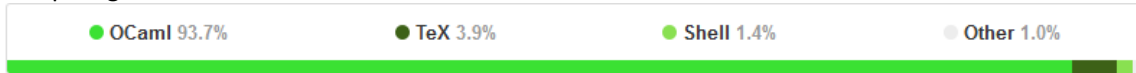
git开源版本

2015年11月24日 16:57

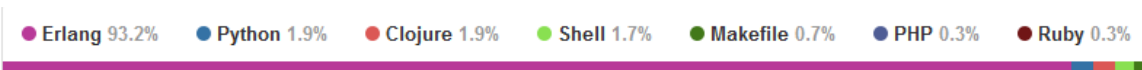
1. <https://github.com/razerbeans/boardwalk> : ruby demo 最近更新在5年前



2. <https://github.com/sx-mirror/libres3>



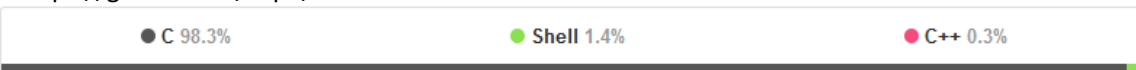
3. https://github.com/basho/riak_cs



4. <https://github.com/eucalyptus/eucalyptus> 更新频繁



5. <https://github.com/ceph/libres3> 2年前更新 客户端



Libre3默认和SX集群配合使用

SX Cluster



SX Cluster is a robust and secure object-storage software for large-scale cloud applications. It is easy to operate and offers native high availability. Your data is transparently distributed and replicated across multiple nodes to eliminate SPoFs.

INSTALL

LEARN MORE

LibreS3



LibreS3 is a high performance daemon which makes SX Cluster fully compatible with AWS S3 APIs.

Set yourself free from Amazon S3 and enjoy all the benefits of SX Cluster: open-source, secure, fast, and scalable.

INSTALL

LEARN MORE

Appendix: SOAP API

Note

SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

This section describes the SOAP API with respect to service, bucket, and object operations. Note that SOAP requests, both authenticated and anonymous, must be sent to Amazon S3 using SSL. Amazon S3 returns an error when you send a SOAP request over HTTP.

1. 在官方文档中发现S3之前的版本支持soap，后续的版本不支持soap，用gsoap等框架实现的soap协议可以放弃。
2. 目前发现S3客户端工具实现的非常多，服务器的实现较少，相对比较热门的libres3是稀有语言Ocaml所写，boardwalk虽然是ruby所写但最近一次更新在5年前，正在查询其它公司使用的实现框架。
3. 可能存在:S3服务端的所有模块概念和当前服务端fics对应原生操作集合的对接封装，后期的工作重点应该全在这之上。
4. 承担的角色为中间服务器，转换fics文件系统接口到S3协议，和onvif的功能一样。
5. 好消息是根据wsdl文件发现，S3协议对外提供的API并不是很多大概65个左右。

Ocaml

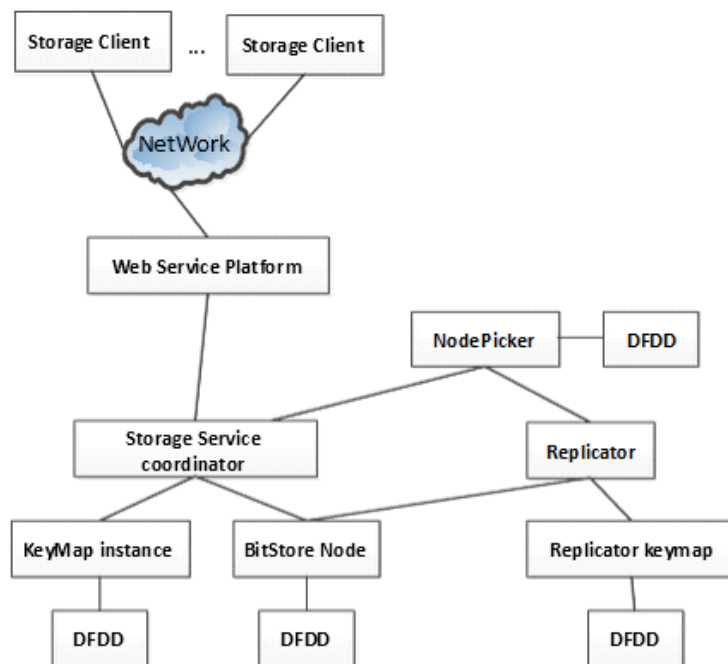
2015年11月25日 12:00

杰出软件产品: Xenserver 虚拟机

语法:

S3框架

2015年11月25日 14:14



如上图所示，S3 存储架构主要是由 keymap+Bitstore 作为基本的存储功能，然后 coordinator 和 NodePicker 充当调度功能，然后 replicator 实现副本的功能，DFDD(discovery, failure,detection daemon)用来检测各个组件的运行状态，web service platform 使用来接收和处理客户端 client 的请求。

MyWebSite.com

Exterior Firewall Hardware or Software Solution to open standard Ports (80, 443)

Web Load Balancer Hardware or Software solution to distribute traffic over web servers

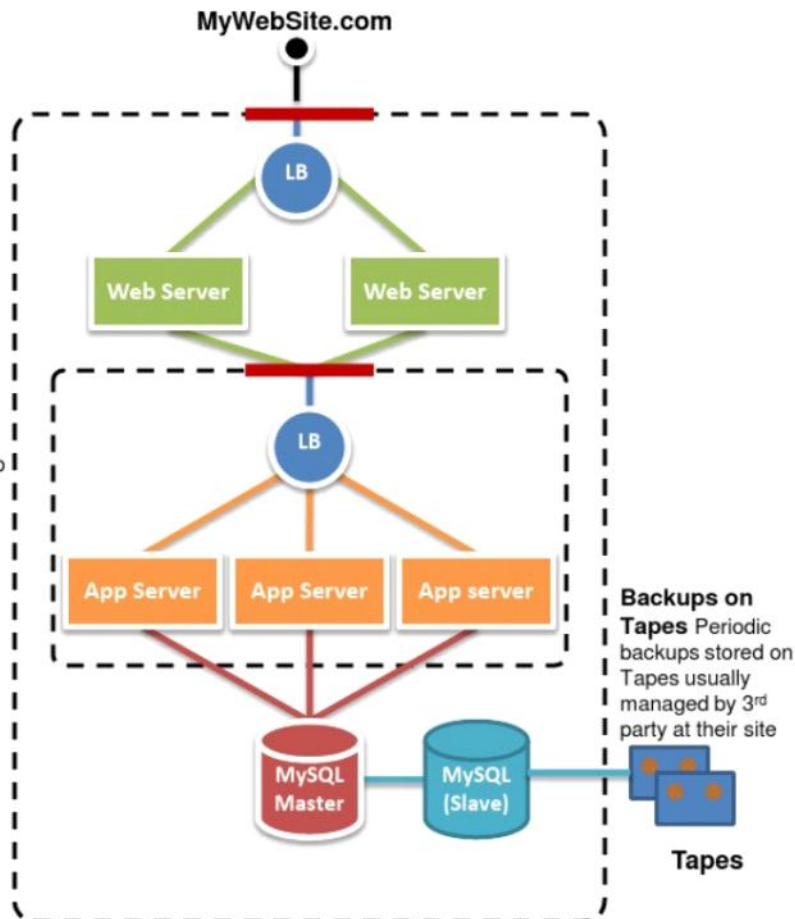
Web Tier Fleet of machines handling HTTP requests.

Backend Firewall Limits access to application tier from web tier

App Load Balancer Hardware or Software solution to spread traffic over app servers

App Server Tier Fleet of machines handling Application specific workloads
Caching server machines can be implemented at this layer

Data Tier Database Server machines with master and local running separately, Network storage for Static objects



Elastic Load Balancer ELB to spread traffic to Web Server Auto-scaling groups

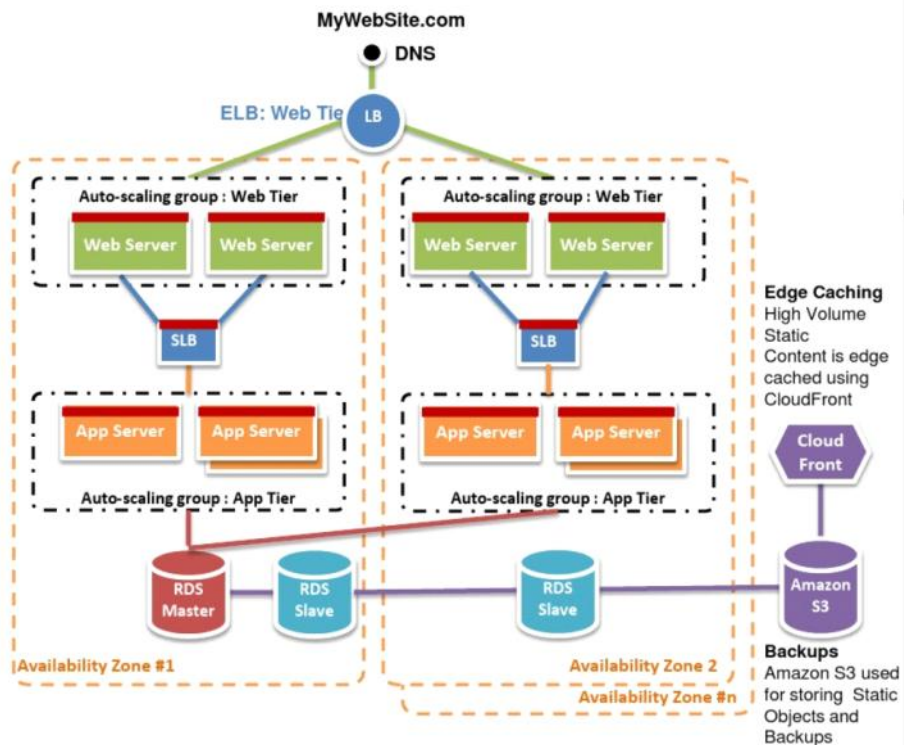
Exterior Firewall no longer needed because EC2 instances are controlled with Security Groups

Auto-scaling Web Tier Group of EC2 instances handling HTTP requests. Backend Firewall no longer needed

App Server Load Balancer Software LB (e.g. HAProxy) on EC2 instance to spread traffic over app server cluster

Auto-scaling App Tier Group of EC2 instances running the actual app. Instances belong to Auto-scaling group. Caching servers instances can be implemented at this layer

DB Tier MySQL RDS DB Instances (master, local slave, x-AZ slave for failover) ; Automated backups to S3 all managed by AWS

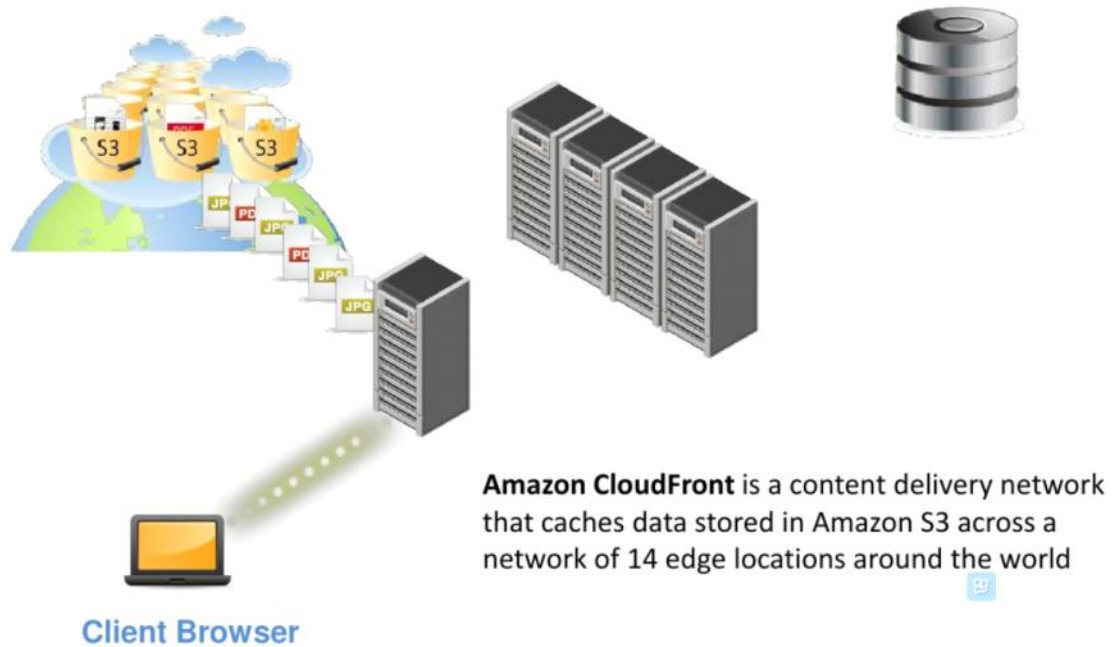


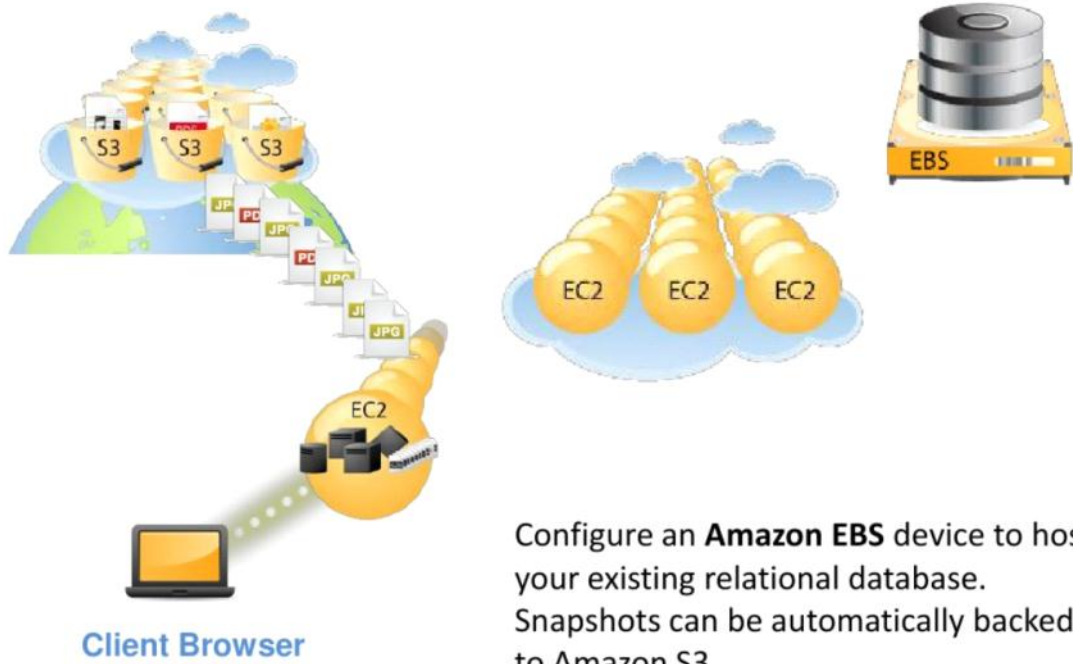
EC2弹性计算云

2015年11月25日 14:56

Elastic Computing Cloud

EC2是一部具有无限计算能力的虚拟计算机，用户能够用来执行一些处理任务。





S3简单存储服务

2015年11月25日 15:04

SQS简单队列服务

2015年11月25日 15:05

Simple DB简单数据库服务

2015年11月25日 15:05

RDS关系数据库服务

2015年11月25日 15:05

Cloud Front 内容推送服务

2015年11月25日 15:06

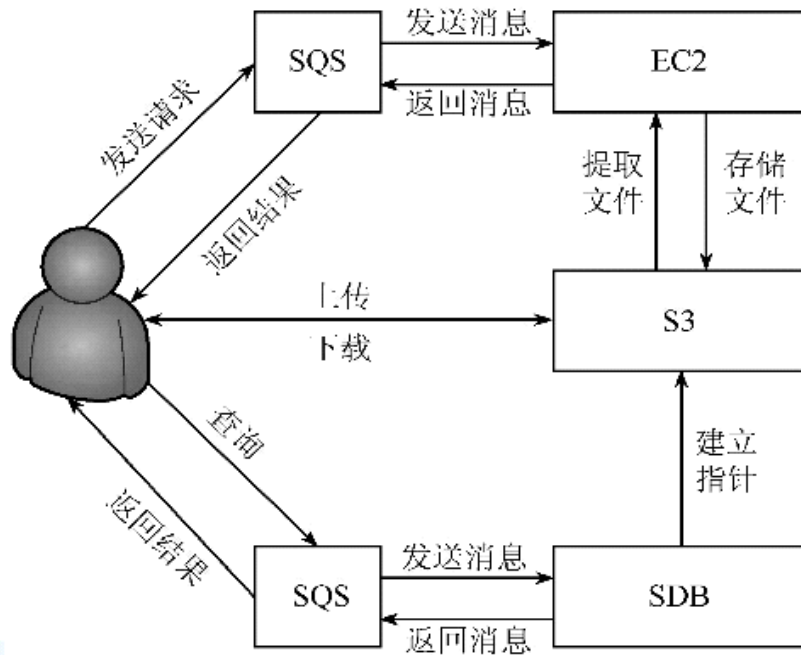
其他Amazon云计算服务

2015年11月25日 15:06

AWS之间的协作

2015年11月25日 15:23

Simple DB和其他AWS的结合使用



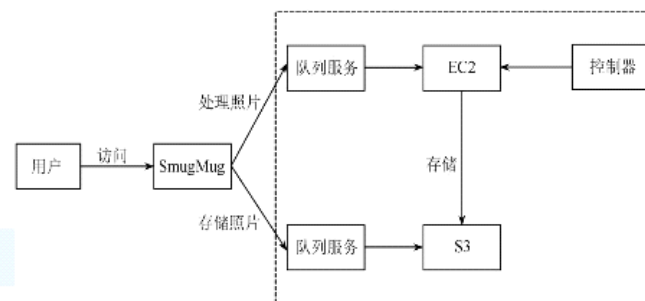
在线照片存储共享网站SmugMug

➤三种照片访问方式

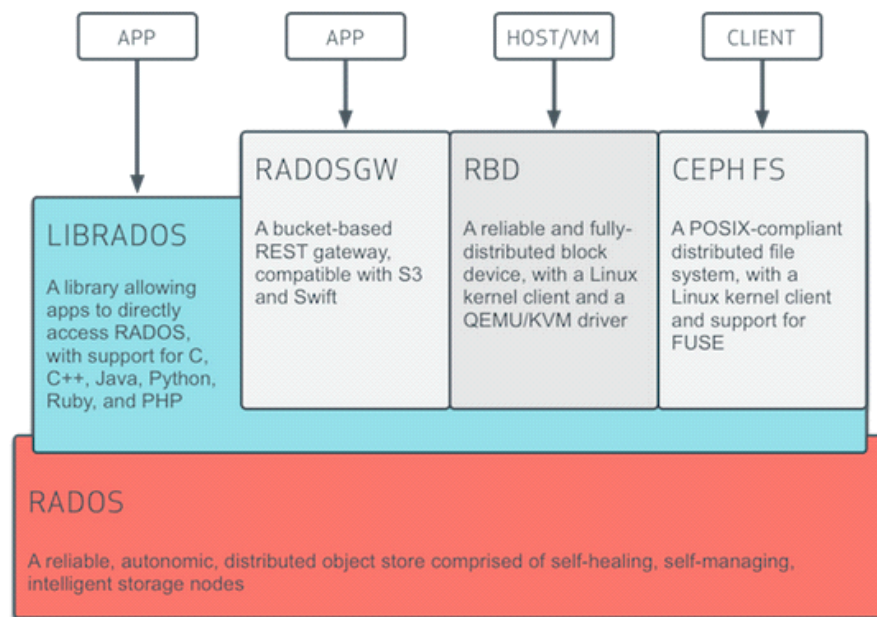
- ☞ (1) SmugMug以代理的身份处理用户访问请求
- ☞ (2) SmugMug对用户访问请求进行重定向
- ☞ (3) 利用有关API直接对存储在S3中的数据进行访问

➤将基础设施部分外包给Amazon

➤SmugMug的系统后台如虚线框所示，主要包括：队列服务和控制器（SmugMug提供），Amazon AWS（包括EC2和S3）



Ceph存储系统的逻辑层次结构如下图所示：



自下而上，可以将Ceph系统分为四个层次：

(1) 基础存储系统RADOS (Reliable, Autonomic, Distributed Object Store, 即可靠的、自动化的、分布式的对象存储)

顾名思义，这一层本身就是一个完整的对象存储系统，所有存储在Ceph系统中的用户数据事实上最终都是由这一层来存储的。而Ceph的高可靠、高可扩展、高性能、高自动化等等特性本质上也是由这一层所提供的。因此，理解RADOS是理解Ceph的基础与关键。

物理上，RADOS由大量的存储设备节点组成，每个节点拥有自己的硬件资源（CPU、内存、硬盘、网络），并运行着操作系统和文件系统。4.2、4.3节将对RADOS进行展开介绍。

(2) 基础库librados

这一层的功能是对RADOS进行抽象和封装，并向上层提供API，以便直接基于RADOS（而不是整个Ceph）进行应用开发。特别要注意的是，RADOS是一个对象存储系统，因此，librados实现的API也只是针对对象存储功能的。

RADOS采用C++开发，所提供的原生librados API包括C和C++两种，其文档参见[2]。物理上，librados和基于其上开发的应用位于同一台机器，因而也被称为本地API。应用调用本机上的librados API，再由后者通过socket与RADOS集群中的节点通信并完成各种操作。

(3) 高层应用接口

这一层包括了三个部分：RADOS GW (RADOS Gateway)、RBD (Reliable Block Device) 和 Ceph FS (Ceph File System)，其作用是在librados库的基础上提供抽象层次更高、更便于应用或客户端使用的上层接口。

其中，RADOS GW是一个提供与Amazon S3和Swift兼容的RESTful API的gateway，以供相应的对象存储应用开发使用。RADOS GW提供的API抽象层次更高，但功能则不如librados强大。因此，开发者应针对自己的需求选择使用。

RBD则提供了一个标准的块设备接口，常用于在虚拟化的场景下为虚拟机创建volume。目前，Red Hat已经将RBD驱动集成在KVM/QEMU中，以提高虚拟机访问性能。

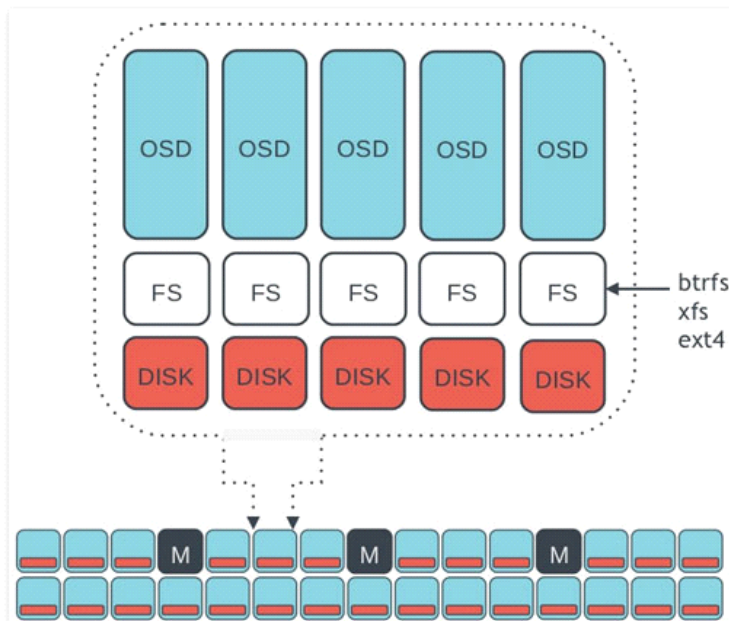
Ceph FS是一个POSIX兼容的分布式文件系统。由于还处在开发状态，因而Ceph官网并不推荐将其用于生产环境中。

(4) 应用层

这一层就是不同场景下对于Ceph各个应用接口的各种应用方式，例如基于librados直接开发的对象存储应用，基于RADOS GW开发的对象存储应用，基于RBD实现的云硬盘等等。

在上文的介绍中，有一个地方可能容易引起困惑：RADOS自身既然已经是一个对象存储系统，并且也可以提供librados API，为何还要再单独开发一个RADOS GW？

理解这个问题，事实上有助于理解RADOS的本质，因此有必要在此加以分析。粗看起来，librados和RADOS GW的区别在于，librados提供的是本地API，而RADOS GW提供的则是RESTful API，二者的编程模型和实际性能不同。而更进一步说，则和这两个不同抽象层次的目标应用场景差异有关。换言之，虽然RADOS和S3、Swift同属分布式对象存储系统，但RADOS提供的功能更为基础、也更为丰富。这一点可以通过对比看出。



Ceph的底层是RADOS，它的意思是“A reliable, autonomous, distributed object storage”。RADOS由两个组件组成：

- OSD：Object Storage Device，提供存储资源。
- Monitor：维护整个Ceph集群的全局状态。

RADOS具有很强的扩展性和可编程性，Ceph基于RADOS开发了Object Storage、Block Storage、FileSystem。Ceph另外两个组件是：

- MDS：用于保存CephFS的元数据。
- RADOS Gateway：对外提供REST接口，兼容S3和Swift的API。

radosgw的实现代码位于src/rgw中

Ceph rados gateway和S3接口测试

RGW:

RGW目前支持直接使用CivetWeb作为WebServer，实现HTTP请求的接受和回复，而不需要配置复杂的FCGI和WebServer了。

RGW业务处理流程:

http request --> apache 转 FastCgi module

FastCgi module --> radosgw 通过socket请求实现(未确定是否有其它方式)

radosgw --> ceph集群 通过socket实现，调用rados接口

rgw需要存储池，在ceph.conf的末尾加入配置项即可使用CivetWeb作为RGW的前端。

CivetWeb启动的radosgw默认将监听7480端口。你可以直接通过访问http://your-host-ip:7480/来访问该RGW对象存储。

你可以通过radosgw-admin命令以管理员的方式访问所启动的RGW，执行例如创建用户等操作。你也可以通过s3cmd命令行工具以用户的方式访问RGW，或者通过s3browser图形界面访问RGW，执行上传/下载文件等操作。

CEPH的对象存储

CEPH的对象存储守护进程，RADOSGW，是一个FASTCGI的服务，提供一个RESTFUL的 HTTP API存储对象和元数据。层顶部CEPH的存储集群与自己的数据格式，并保持自己的用户数据库身份验证

证和访问控制。RADOS网关采用一个统一的命名空间，这意味着你可以使用OPENSTACK的SWIFT兼容的API或亚马逊S3兼容的API。例如，你可以使用S3-COMPTABLE的一个应用程序的API写数据，然后使用SWIFT兼容的API与其他应用程序读取数据。

S3/Swift对象和存储群集对象比较

Ceph的对象存储使用的术语来形容它存储的数据对象。S3和Swift对象Ceph的写入与Ceph的存储集群的对象是不一样的。Ceph的对象存储对象映射到Ceph的存储集群对象。S3和Swift对象不一定对应于以1:1的方式存储在所述存储群集的对象。这是可能的一个S3或Swift对象映射到多个Ceph的对象。

对ceph radosgw的一些理解

Ceph本质上就是一个rados，利用命令rados就可以访问和使用ceph的对象存储，但作为一个真正产品的对象存储服务，通常使用的是Restful api的方式进行访问和使用。而radosgw其实就是这个作用，安装完radosgw以后，就可以使用api来访问和使用ceph的对象存储服务了。首先明白一下架构，radosgw其实名副其实，就是rados的一个网关，作用是对外提供对象存储服务。本质上radosgw（其实也是一个命令）和 rbd(在存储池中创建映像)命令一样，其实是ceph集群的客户端。只不过，radosgw即作为rados（Reliable Autonomic Distributed Object Storage）的客户端，同时又提供http restful接口，作为服务端供用户使用。

Radosgw对用户而言就是一个http restful的应用，因此本质上来讲，对其进行使用就是通过http的方式，但显然每次都要用户构建http访问的url和headers不是一个很方便的方式，因此radosgw兼容了通用的对象存储接口，分别是亚马逊的s3和openstack的swift，这也就是说你可以用swift或者s3的客户端来访问radosgw。

Radosgw包含两个命令行工具，一个是radosgw，这个是用来启动radosgw服务的脚步，是一个二进制文件；另外一个为radosgw-admin，这是用来管理radosgw的账号的一个命令行工具，主要用来创建、查看、修改radosgw的账号信息。

注意，radosgw的账号信息仅仅是对radosgw的用户而言，这个和ceph中的用户不是一个概念。

Radosgw作为ceph集群（rados）的客户端，因此他在ceph中有一个账号，通常叫做client.radosgw.gateway。在启动radosgw这个服务时，会读取ceph.conf中[client.radosgw.gateway]这个section。

Radosgw有自己的账号管理系统，可以使用radosgw-admin来创建和维护账号，之后可以使用curl、swift、s3等方式指明账号名和密码就可以访问，不确定s3怎么样，但如果用swift的话，其实和openstack中类似，先通过账号和密码向radosgw的auth系统获取一个token，然后用这个token再radosgw的存储系统。

```
client.compute
  key: AQAww/ydVVKPODFRAAVoIG4v3vSDtUcUllzGqjobq==
  caps: [mon] allow r
  caps: [osd] allow class-read object_prefix rbd_children, allow rwx pool=volumes
client.images
  key: AQCw/SdVIIIfCJxAAqkkgJiVr7YRsm2xk4T+D9w==
  caps: [mon] allow r
  caps: [osd] allow class-read object_prefix rbd_children, allow rwx pool=images
client.radosgw.cn-master-1
  key: AQAduFGHs3xlFhAAIf7LABfOuqm0HcZrnBNYVw==
  caps: [mon] allow rw
  caps: [osd] allow rwx
```

我们用Radosgw和keystone进行集成，创建对应的endpoint，客户端使用swift接口。操作就和和使用真正的swift+keystone没有任何差别，作为一个用户无法感知到后端究竟真是一个swift还是radosgw。但就附加功能还是有区别的，我们知道swift除了对象存储本身的上传、下载、删除等操作，还有很多其他附加功能，比如object过期设置、tempurl、staticweb等，这些都是通过在proxy节点增加middleware来实现的，radosgw支持swift一些middleware，但并不完全支持。

默认创建的存储池根据功能分，有关于用户的，垃圾收集，索引的，数据主要放在.rgw.buckets里，存储池大小可自定义。

```

root@node1:/data1/logs# radosgw-admin zone get
warning: line 48: 'rgw_dns_name' in section 'client.radosgw.node1' redefined
{ "domain_root": ".rgw",
  "control_pool": ".rgw.control",
  "gc_pool": ".rgw.gc",
  "log_pool": ".log",
  "intent_log_pool": ".intent-log",
  "usage_log_pool": ".usage",
  "user_keys_pool": ".users",
  "user_email_pool": ".users.email",
  "user_swift_pool": ".users.swift",
  "user_uid_pool": ".users.uid",
  "system_key": { "access_key": "",
                  "secret_key": "" },
  "placement_pools": [
    { "key": "default-placement",
      "val": { "index_pool": ".rgw.buckets.index",
               "data_pool": ".rgw.buckets",
               "data_extra_pool": ".rgw.buckets.extra" }}}
root@node1:/data1/logs#

```

<http://docs.ceph.com/docs/v0.80.5/radosgw/federated-config/> 联邦RGW

来自 <<http://www.ithao123.cn/content-8387956.html>>

Radosgw 下面是创建的bucket

```

[root@client1 bin]# ./list-s3buckets.py
my-new-bucket 2015-11-25T11:16:29.000Z
s3-bucket1 2015-11-25T17:08:57.000Z
s3-new-bucket 2015-11-25T17:06:33.000Z
s3test 2015-11-25T16:19:42.000Z
test 2015-11-25T12:23:51.000Z

```

<http://cephnotes.ksperis.com/blog/2014/11/28/placement-pools-on-rados-gw>

创建存储池测试s3实例如上。

问题：多个SGW?多个SGW账号? 用于多个存储池?

S3接口描述 (ceph\doc\radosgw\s3) serviceops.rst 为服务端说明, 其他为客户端的协议交互说明。

S3和Swift3

2015年11月26日 18:24

Swift3:

OpenStack Object Storage (Swift) 是开源的，用来创建可扩展的、冗余的、对象存储（引擎）。 swift使用标准化的服务器存储 PB 级可用数据。但它并不是文件系统（file system），实时的数据存储系统(real-time data storage system)。 swift 看起来更像是一个长期的存储系统（long term storage system），为了获得、调用、更新一些静态的永久性的数据。比如说，适合存储一些类型的数据：虚拟机镜像，图片存储，邮件存储，文档的备份。没有“单点”或者主控节点（master point of control）， swift看起来具有更强的扩展性、冗余和持久性。

S3:

简单存储接口,swift3提供S3型的API。

Libs3

2015年11月27日 10:05

<https://github.com/ceph/libs3> 属于ceph的一个子项目（S3客户端）代码不多结构如下

```
|— inc
|   |— error_parser.h
|   |— libs3.h
|   |— mingw
|   |   |— pthread.h
|   |   |— sys
|   |       |— select.h
|   |       |— utsname.h
|   |— request_context.h
|   |— request.h
|   |— response_headers_handler.h
|   |— simplexml.h
|   |— string_buffer.h
|   |— util.h
|— INSTALL
|— libs3.files
|— libs3.spec
|— LICENSE
|— mswin
|   |— libs3.def
|   |— rmrf.bat
|— README
|— src
|   |— acl.c
|   |— bucket.c
|   |— error_parser.c
|   |— general.c
|   |— mingw_functions.c
|   |— mingw_s3_functions.c
|   |— object.c
|   |— request.c
|   |— request_context.c
|   |— response_headers_handler.c
|   |— s3.c
|   |— service_access_logging.c
|   |— service.c
|   |— simplexml.c
|   |— testsimplexml.c
|   |— util.c
|— tags
|— test
|   |— badxml_01.xml
|   |— goodxml_01.xml
|   |— goodxml_02.xml
|   |— goodxml_03.xml
|   |— test.sh
|— TODO
```

代码中共2个main。一个是测试xml的，一个就是Src/s3.c：对协议中各个接口API的测试。

测试需要S3_ACCESS_KEY_ID, S3_SECRET_ACCESS_KEY
S3_ACCESS_KEY_ID:
S3_SECRET_ACCESS_KEY:
S3_list_service 需要

ceph搭建准备

2015年11月30日 17:05

研发-南通-西昆仑(644657055) 17:04:04

- 你多大的数据量
- 在不在虚拟机我觉得问题不大，反正你做实验

运维-深圳-最爱小鸟(2424658816) 17:04:44

虚拟机的话

将设置

osd pool default size = 1

然后单节点吧

研发-南通-西昆仑(644657055) 17:22:09

```
[client.radosgw.gateway]
host=ceph-2
log_file=/var/log/radosgw/client.radosgw.gateway.log
rgw_frontends=fastcgi,civetweb port=80
keyring=/etc/ceph/ceph.client.radosgw.keyring
```

- civetweb就配置了这个

ceph-成都-亮(546827391) 17:22:39

不用fastcgi

研发-南通-西昆仑(644657055) 17:22:50

- radosgw -c /etc/ceph/ceph.conf -n client.radosgw.gateway --rgw-frontends
- 用这个命令启动

CentOS6.3上部署配置Ceph教程。

一、背景知识

搭建ceph的机器分为两种：client和非client（mds、monitor、osd）。

配置时client只需要在内核编译时选上ceph就行，而其它三种则还需要编译ceph用户态源码（下载地址：<http://ceph.com/download/>），另外osd还要记得安装btrfs文件系统（内核编译作为模块就行）。

内核版本参考：<http://ceph.com/docs/master/install/os-recommendations/#glibc>

二、机器分配

IP	Roles	Hostname	备注
222.31.76.209	client	localhost.localdomain	
222.31.76.178	mds&monitor	ceph_mds	
222.31.76.74	osd	ceph_osd0	
222.31.76.67	osd	ceph_osd1	
222.31.76.235	osd	ceph_osd2	

操作系统：CentOS 6.3

内核版本：linux-3.8.8.tar.xz（stable2013-04-17）

三、编译与配置

- (1) client
- (2) mds/monitor/osd

1. 编译最新版内核3.8.8 (同client)

2. 编译ceph源码

```
#tar -xvf ceph-0.60.tar.gz
#cd ceph-0.60
#./autogen.sh
#./configure --without-tcmalloc --without-libxfs
```

3. 配置ceph

除客户端外, 其它的节点都需一个配置文件ceph.conf, 并需要是完全一样的。这个文件要位于/etc/ceph下面, 如果在./configure时没有修改prefix的话, 则应该是在/usr/local/etc/ceph下。

```
#cp ./src/sample.* /usr/local/etc/ceph/
#mv /usr/local/etc/ceph/sample.ceph.conf /usr/local/etc/ceph/ceph.conf
#mv /usr/local/etc/ceph/sample.fetch_config /usr/local/etc/ceph/fetch_config
#cp ./src/init-ceph /etc/init.d/ceph
#mkdir /var/log/ceph //存放log, 现在ceph自己还不自动建这个目录
```

注:

①部署每台服务器, 主要修改的就是/usr/local/etc/ceph/下的两个文件ceph.conf (ceph集群配置文件) 和 fetch_config (同步脚本, 用于同步各节点的ceph.conf文件, 具体方法是scp远程拷贝, 但我发现没啥用, 所以后来自行写了个脚本)。

②针对osd, 除了要加载btrfs模块, 还要安装btrfs-progs (#yum install btrfs-progs), 这样才有mkfs.btrfs命令。另外就是要在数据节点osd上创建分区或逻辑卷供ceph使用: 可以是磁盘分区 (如 /dev/sda2), 也可以是逻辑卷 (如/dev/mapper/VolGroup-lv_ceph), 只要与配置文件ceph.conf中写的一致即可。

4. 配置网络

① 修改各节点的hostname, 并能够通过hostname来互相访问

参考: <http://soft.chinabyte.com/os/281/11563281.shtml>

修改/etc/sysconfig/network文件以重定义自己的hostname;

修改/etc/hosts文件以标识其他节点的hostname与IP的映射关系;

重启主机后用hostname命令查看验证。

② 各节点能够ssh互相访问而不输入密码

原理就是公私钥机制, 我要访问别人, 那么就要把自己的公钥先发给别人, 对方就能通过我的公钥验证我的身份。

例: 在甲节点执行如下命令

```
#ssh-keygen -d
```

该命令会在“ ~/.ssh” 下面生成几个文件, 这里有用的是id_dsa.pub, 为该节点(甲)的公钥, 然后把里面的内容添加到对方节点(乙) “ ~/.ssh/” 目录下的authorized_keys文件中, 如果没有则创建一个, 这样就从甲节点不需要密码ssh登录到乙上了。

5. 创建文件系统并启动。以下命令在监控节点进行!

```
#mkcephfs -a -c /usr/local/etc/ceph/ceph.conf -mkbtrfs
```

出现下列错误可能需要如下

```
scp: /etc/ceph/ceph.conf: No such file or directory
```

解决: 编写一个脚本, 将配置文件同步到/etc/ceph和/usr/local/etc/ceph目录下 (需手动先建立/etc/ceph目录):

```
[root@ceph_mds ceph]# cat cp_ceph_conf.sh
cp /usr/local/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

```
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd0:/usr/local/etc/ceph/ceph.conf
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd0:/etc/ceph/ceph.conf
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd1:/usr/local/etc/ceph/ceph.conf
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd1:/etc/ceph/ceph.conf
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd2:/usr/local/etc/ceph/ceph.conf
scp /usr/local/etc/ceph/ceph.conf root@ceph_osd2:/etc/ceph/ceph.conf
```

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>

```
mkcephfs -a -c /usr/local/etc/ceph/ceph.conf -mkbtrfs
```

【创建成功】

```
[root@ceph_mds ceph]# mkcephfs -a -c /usr/local/etc/ceph/ceph.conf -mkbtrfs
```

【启动】

```
#/etc/init.d/ceph -a start //必要时先关闭防火墙 (#service iptables stop)
```

```
[root@ceph_mds ceph]# /etc/init.d/ceph -a start
=== mon.0===
Starting Ceph mon.0 on ceph_mds...
starting mon.0 rank 0 at 222.31.76.178:6789/0 mon_data /data/mon0 fsid652b09fb-bbbf-424c-
bd49-8218d75465ba
=== mds.alpha===
Starting Ceph mds.alpha on ceph_mds...
starting mds.alpha at :/0
=== osd.0===
Mounting Btrfs on ceph_osd0:/data/osd0
Scanning for Btrfs filesystems
Starting Ceph osd.0 on ceph_osd0...
starting osd.0 at :/0 osd_data/data/osd0/data/osd.0/journal
=== osd.1===
Mounting Btrfs on ceph_osd1:/data/osd1
Scanning for Btrfs filesystems
Starting Ceph osd.1 on ceph_osd1...
starting osd.1 at :/0 osd_data/data/osd1/data/osd.1/journal
=== osd.2===
Mounting Btrfs on ceph_osd2:/data/osd2
Scanning for Btrfs filesystems
Starting Ceph osd.2 on ceph_osd2...
starting osd.2 at :/0 osd_data/data/osd2/data/osd.2/journal
```

【查看Ceph集群状态】

```
[root@ceph_mds ceph]# ceph -s
health HEALTH_OK
monmap e1: 1 mons at {0=222.31.76.178:6789/0}, election epoch 2, quorum 0 0
osdmap e7: 3 osds:3 up,3in
pgmap v432: 768 pgs:768 active+clean;9518 bytes data, 16876 KB used,293 GB/300 GB avail
mdsmap e4: 1/1/1 up {0=alpha=up:active}
```

```
[root@ceph_mds ceph]# ceph df
GLOBAL:
SIZE AVAIL RAW USED %RAW USED
300M 293M 16876 0
```

```
POOLS:
NAME ID USED %USED OBJECTS
data 0 0 0 0
```



```
metadata 1 9518021
```

```
rbd 2 0 0 0
```

疑问：空间统计有问题吧?! ” ceph -s” 查看是300GB， ” ceph df” 查看是300M。

6. 客户端挂载

```
#mkdir /mnt/ceph
```

```
#mount -t ceph ceph_mds:/ /mnt/ceph
```

遇以下错误：

(1)

```
[root@localhost ~]# mount -t ceph ceph_mds:/ /mnt/ceph/
mount: wrong fs type, bad option, bad superblock on ceph_mds:/,
missing codepage or helper program, or other error
(for several filesystems (e.g. nfs, cifs) you might
need a /sbin/mount.<type> helper program)
In some cases useful info is found in syslog- try
dmesg | tail or so
查看#dmesg
```

```
ceph: Unknown symbol ceph_con_keepalive (err0)
ceph: Unknown symbol ceph_create_client (err 0)
ceph: Unknown symbol ceph_calc_pg_primary (err0)
ceph: Unknown symbol ceph_osdc_release_request (err0)
ceph: Unknown symbol ceph_con_open (err 0)
ceph: Unknown symbol ceph_flags_to_mode (err 0)
ceph: Unknown symbol ceph_msg_last_put (err 0)
ceph: Unknown symbol ceph_caps_for_mode (err 0)
ceph: Unknown symbol ceph_copy_page_vector_to_user (err0)
ceph: Unknown symbol ceph_msg_new (err 0)
ceph: Unknown symbol ceph_msg_type_name (err 0)
ceph: Unknown symbol ceph_pagelist_truncate (err0)
ceph: Unknown symbol ceph_release_page_vector (err0)
ceph: Unknown symbol ceph_check_fsid (err 0)
ceph: Unknown symbol ceph_pagelist_reserve (err0)
ceph: Unknown symbol ceph_pagelist_append (err0)
ceph: Unknown symbol ceph_calc_object_layout (err0)
ceph: Unknown symbol ceph_get_direct_page_vector (err0)
ceph: Unknown symbol ceph_osdc_wait_request (err0)
ceph: Unknown symbol ceph_osdc_new_request (err0)
ceph: Unknown symbol ceph_pagelist_set_cursor (err0)
ceph: Unknown symbol ceph_calc_file_object_mapping (err0)
ceph: Unknown symbol ceph_monc_got_mdsmmap (err0)
ceph: Unknown symbol ceph_osdc_readpages (err 0)
ceph: Unknown symbol ceph_con_send (err 0)
ceph: Unknown symbol ceph_zero_page_vector_range (err0)
ceph: Unknown symbol ceph_osdc_start_request (err0)
ceph: Unknown symbol ceph_compare_options (err0)
ceph: Unknown symbol ceph_msg_dump (err 0)
ceph: Unknown symbol ceph_buffer_new (err 0)
ceph: Unknown symbol ceph_put_page_vector (err0)
ceph: Unknown symbol ceph_pagelist_release (err0)
ceph: Unknown symbol ceph_osdc_sync (err 0)
ceph: Unknown symbol ceph_destroy_client (err 0)
ceph: Unknown symbol ceph_copy_user_to_page_vector (err0)
ceph: Unknown symbol __ceph_open_session (err 0)
```

```
ceph: Unknown symbol ceph_alloc_page_vector (err0)
ceph: Unknown symbol ceph_monc_do_stats (err 0)
ceph: Unknown symbol ceph_monc_validate_auth (err0)
ceph: Unknown symbol ceph_osdc_writespages (err0)
ceph: Unknown symbol ceph_parse_options (err 0)
ceph: Unknown symbol ceph_str_hash (err 0)
ceph: Unknown symbol ceph_pr_addr (err 0)
ceph: Unknown symbol ceph_buffer_release (err 0)
ceph: Unknown symbol ceph_con_init (err 0)
ceph: Unknown symbol ceph_destroy_options (err0)
ceph: Unknown symbol ceph_con_close (err 0)
ceph: Unknown symbol ceph_msgr_flush (err 0)
```

Key type ceph registered

libceph: loaded (mon/osd proto15/24, osdmap5/65/6)

ceph: loaded (mds proto 32)

libceph: parse_ips bad ip 'ceph_mds'

ceph: loaded (mds proto 32)

libceph: parse_ips bad ip 'ceph_mds'

我发现客户端mount命令根本没有ceph类型（无“mount.ceph”），而我们配置的其他节点都有mount.ceph，所以我在ceph客户端上也重新编译了最新版的ceph-0.60。

（2）编译安装ceph-0.60后mount还是报同样的错，查看dmesg

```
#dmesg | tail
```

```
Key type ceph unregistered
```

```
Key type ceph registered
```

```
libceph: loaded (mon/osd proto15/24, osdmap5/65/6)
```

```
ceph: loaded (mds proto 32)
```

```
libceph: parse_ips bad ip 'ceph_mds'
```

```
libceph: no secret set (for auth_x protocol)
```

```
libceph: error -22 on auth protocol 2 init
```

```
libceph: client4102 fsid 652b09fb-bbbf-424c-bd49-8218d75465ba
```

最终查明原因，是因为mount时还需要输入用户名和密钥，具体mount命令为：

```
#mount.ceph ceph_mds:/mnt/ceph -v -o
```

```
name=admin,secret=AQCXnKhRgMltJRAAi0WMqr+atKFPaIV4Aja4hQ==
```

```
[root@localhost ~]# mount.ceph ceph_mds:/mnt/ceph -v -o
```

```
name=admin,secret=AQCXnKhRgMltJRAAi0WMqr+atKFPaIV4Aja4hQ==
```

```
parsing options: name=admin,secret=AQCXnKhRgMltJRAAi0WMqr+atKFPaIV4Aja4hQ==
```

上述命令中的name和secret参数值来自monitor的/etc/ceph/keyring文件：

```
[root@ceph_mds ceph]# cat /etc/ceph/keyring
```

```
[client.admin]
```

```
key = AQCXnKhRgMltJRAAi0WMqr+atKFPaIV4Aja4hQ==
```

注：

1. To mount the Ceph file system you may use the mount command if you know the monitor host IP address(es), or use the mount.ceph utility to resolve the monitor host name(s) into IP address(es) for you.

2. mount参数

-v, -verbose: Verbose mode.

-o, -options opts: Options are specified with a -o flag followed by a comma separated string of options.

3. mount.ceph参考：<http://ceph.com/docs/master/man/8/mount.ceph/>

查看客户端的挂载情况：

```
[root@localhost ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
50G 13G 35G 27%/
tmpfs 2.0G 0.0G 0% /dev/shm
/dev/sda1 477M 48M 405M 11% /boot
/dev/mapper/VolGroup-lv_home
405G 71M 385G 1% /home
222.31.76.178:/ 300G 6.1G 294G 3% /mnt/ceph
```

P. S. 网上说若不想每次输入密钥这么繁琐，可以在配置文件ceph.conf中加入以下字段（并记得同步到其他节点），但我实验发现还是无效，所以暂且采用上述方法挂载使用，有哪位朋友知道我错在哪欢迎指出啊。

```
[mount /]
allow = %everyone
```

【解决】

查看了官网文档<http://ceph.com/docs/master/rados/operations/authentication/>，发现真正要取消挂载时的认证，需要在配置文件[global]下加入以下内容：

对于0.51及以上版本：

```
auth cluster required = none
auth service required = none
auth client required = none
```

对于0.50及以下版本：

```
auth supported = none
```

官方注：If your cluster environment is relatively safe, you can offset the computation expense of running authentication. We do not recommend it. However, it may be easier during setup and/or troubleshooting to temporarily disable authentication.

到此Ceph的安装配置就已全部完成，可以在客户端的/mnt/ceph目录下使用Ceph分布式文件系统。近期我还将进行Ceph的功能性验证测试，测试报告敬请期待！

【参考】

1. 安装过程

ceph的安装过程：<http://blog.csdn.net/wilcke/article/details/8089337>

centos 6.2 64位上安装ceph

0.47.2：<http://blog.csdn.net/frank0712105003/article/details/7631035>

Install Ceph on CentOS 5.5：<http://blog.csdn.net/gurad2008/article/details/6270804>

2. 配置过程

<http://blog.csdn.net/wujievhv2006/article/details/6448168>

<http://blog.csdn.net/polisan/article/details/5624207>

Ceph环境配置文档：<http://wenku.baidu.com/view/a8604b523c1ec5da50e2706b.html>

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>

来自 <<http://linux.it.net.cn/CentOS/course/2015/0110/11660.html>>