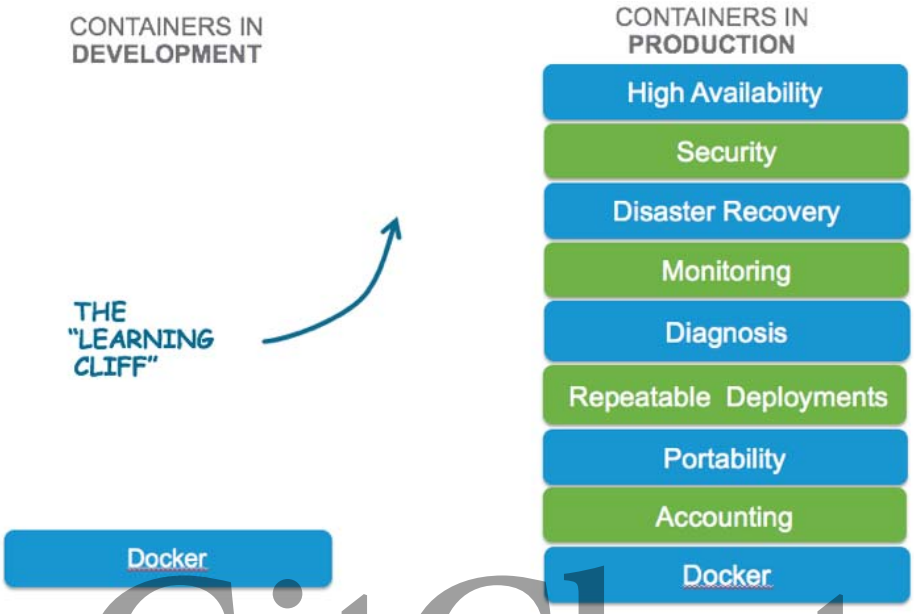
容器化存储技术探秘之旅

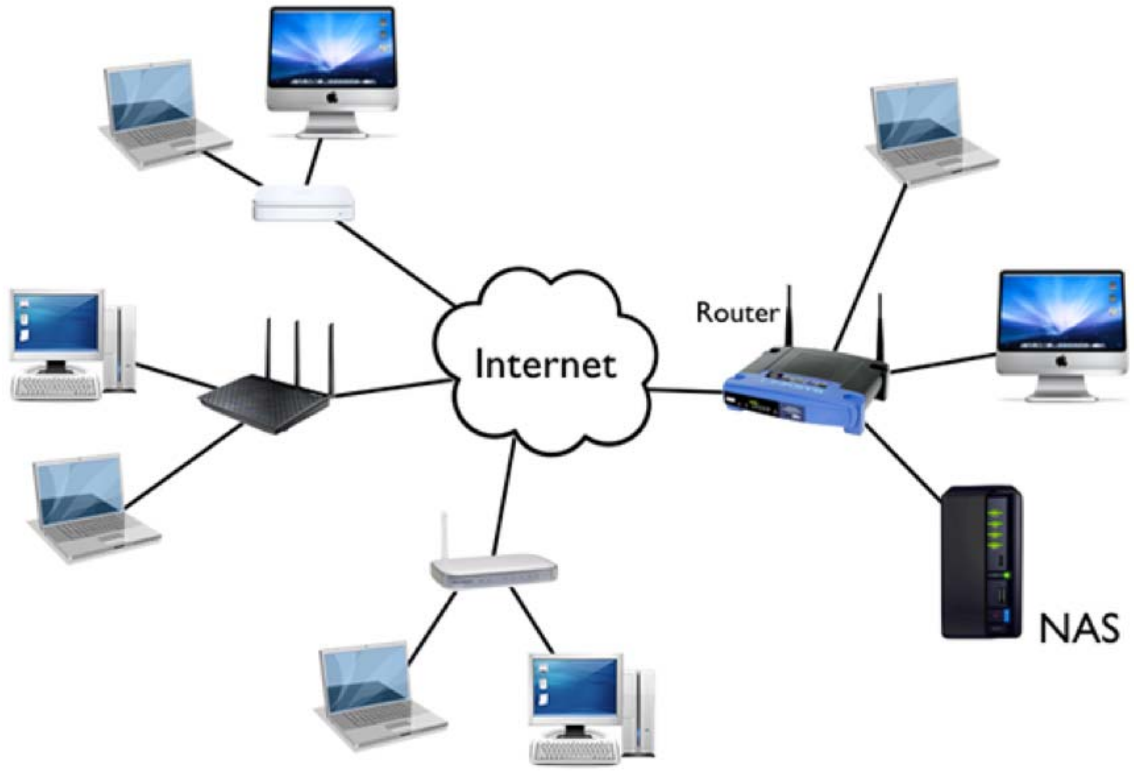
我是从2013年开始接触容器技术的，从容器技术的落地过程中，经常会有一个很难讲清楚的问题，你们容器存储是怎么解决？当时，我个人限于自身存储知识的边界，我能想到的对接容器存储方式就2种：一种是挂盘(NFS)，一种是存储驱动(volume plugin)。并且实话讲，全球范围内容器存储技术也是一个半吊子阶段，能上生产的解决方案没几个。所以国内的容器解决方案大多是依赖传统的存储方案作为背书，通过一些脚手架插件来支撑容器存储的解决方案。如下图所示：



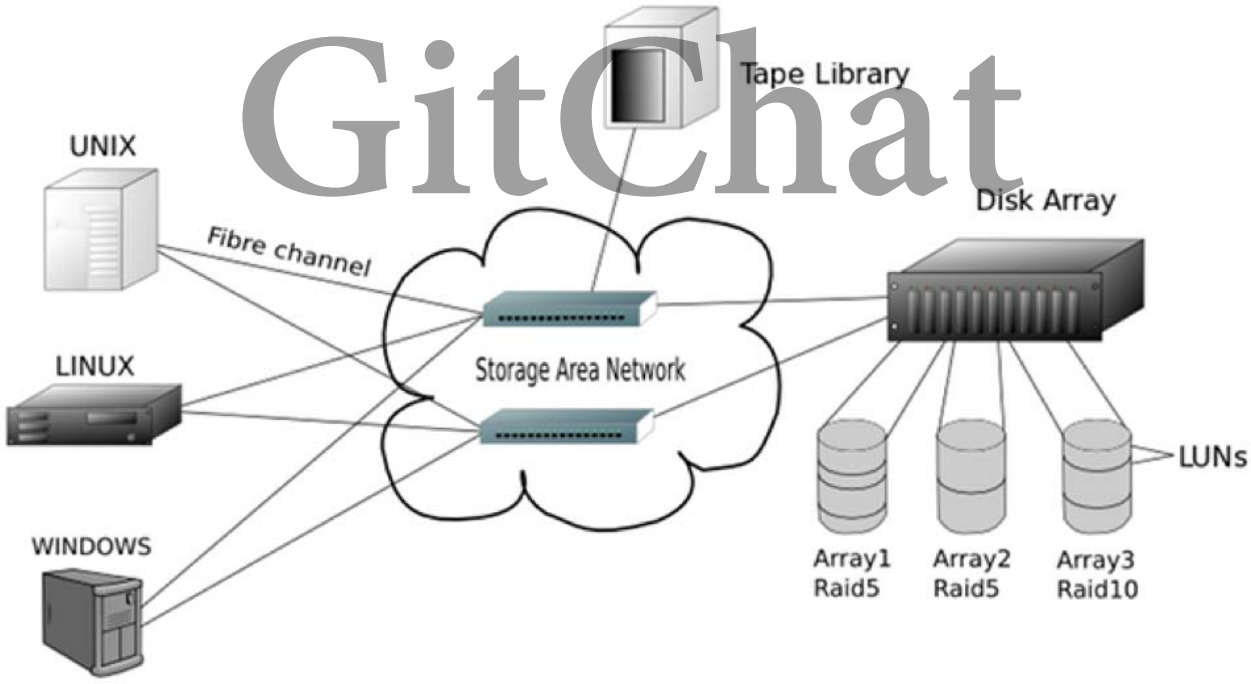
所谓容器从开发者的角度，也就是Docker工具，薄薄的一层，真的要上到生产环境之后，想构建适用于不可变基础架构的生产级别容器环境，就需要套上高可用、安全、灾难恢复、监控、帐户系统、存储一致性等等业务需求的枷锁。在容器使用早期阶段，大家都是用容器来解决无状态应用的。什么是无状态，就是只有守护进程，没有数据落盘一致性的要求，或者说落盘的数据可以初始化重来，比如系统日志类、配置文件类等等。但是在企业级业务领域场景下，大家不可能用这种不靠谱的存储方案来提升IT运营水平。所以，对于企业级环境之下传统应用的容器化迁移就需要考虑如何提供一套靠谱的方案来解决。我们也不用卖关子，当前容器存储关键技术解决之道就是在合适的场景之下选择合适的分布式存储系统(Distributed Storage System)。本文通过一些浅显易懂的分析，解决一些包括我在内对容器存储技术知识结构的梳理，对未来5年之内的容器存储技术建立一套合理的技术选择方法论。

首先，有一个关键存储协议，NFS大家要搞清楚，NFS我一开始也是很忽略的一项关键技术。需要把NFS的缩写展开：Network File System。从维基百科对它的介绍中，Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984…

你可以看到distributed这个词。分布式这个词很关键，可以说容器技术就是构建分布式系统最佳的搭档。基本上容器存储就是需要用分布式存储解决方案才能发挥容器的可移动的效能。另外，NFS是一套协议，不是实现，目前最新版本的协议实现是NFSv4。并且，在Docker这块来讲，有一套现成的驱动方案可以满足NFS的需求，Netshare(http://netshare.containx.io/)。那么好了，行家里手和你谈到网络存储方案，不可能和你扯犊子聊NFS，这是一套协议，不是具体实现。所以大家口头上说的是纳斯盘(NAS，network attached storage), 或者是storage area network (SAN)。



NAS的好处就是通过网络可以让所有的机器都能挂载这个盘。一般的NAS都支持NFS协议。所以你懂的。听说日立的NAS技术很牛逼，就是没见过真家伙，遗憾。

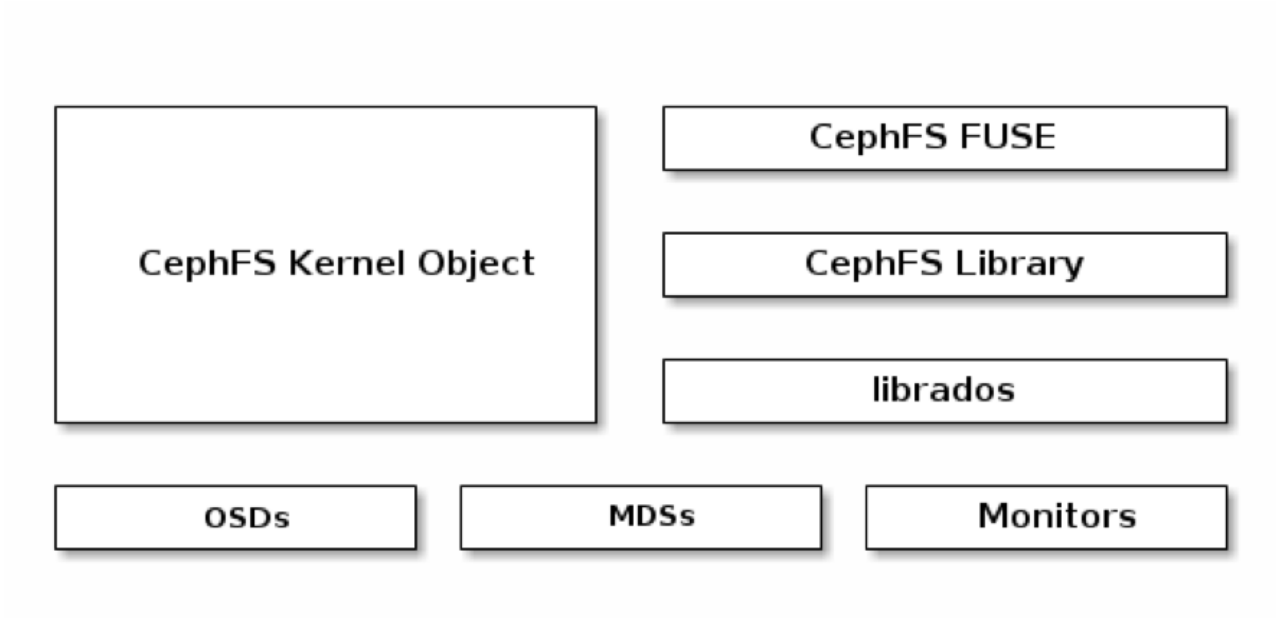


SAN比NAS稍微底层一点，是一套磁盘阵列并通过Fibre Channel和iSCS协议与一套操作系统连接并需要格式化后才能使用它的存储。虽然有点啰嗦，但是希望大家留心一下这点区别。看到这些比较后，你会发现存储盘的IO是单点。一旦业务量上来后，IO瓶颈要么在网络设备上，要么在磁盘访问点上。举个实际的例子，比如搭建一套企业内部镜像中心，存储镜像这一块就是配置一个目录，这个目录可以挂载一块网络盘来解决。你挂上一块SAN盘后，就会遇到一个瓶颈，只要在发布应用的时候，所有请求都会到这块磁盘上拉镜像文件，不忙的时候，一点压力都没有。所以这种请求模式加上这种集中存储的方案，这种方案数据不会丢，但是如果盘阵掉电了咋办，又不能跨IDC管理。但是无论优缺点这么明显，好歹是一套商业上卖的合格的方案。但不是理想的容器存储方案。

SAN比NAS稍微底层一点，是一套磁盘阵列并通过Fibre Channel和iSCS协议与一套操作系统连接并需要格式化后才能使用它的存储。虽然有点啰嗦，但是希望大家留心一下这点区别。看到这些比较后，你会发现存储盘的IO是单点。一旦业务量上来后，IO瓶颈要么在网络设备上，要么在磁盘访问点上。举个实际的例子，比如搭建一套企业内部镜像中心，存储镜像这一块就是配置一个目录，这个目录可以挂载一块网络盘来解决。你挂上一块SAN盘后，就会遇到一个瓶颈，只要在发布应用的时候，所有请求都会到这块磁盘上拉镜像文件，不忙的时候，一点压力都没有。所以这种请求模式加上这种集中

存储的方案，这种方案数据不会丢，但是如果盘阵掉电了咋办，又不能跨IDC管理。但是无论优缺点这么明显，好歹是一套商业上卖的合格的方案。但不是理想的容器存储方案。

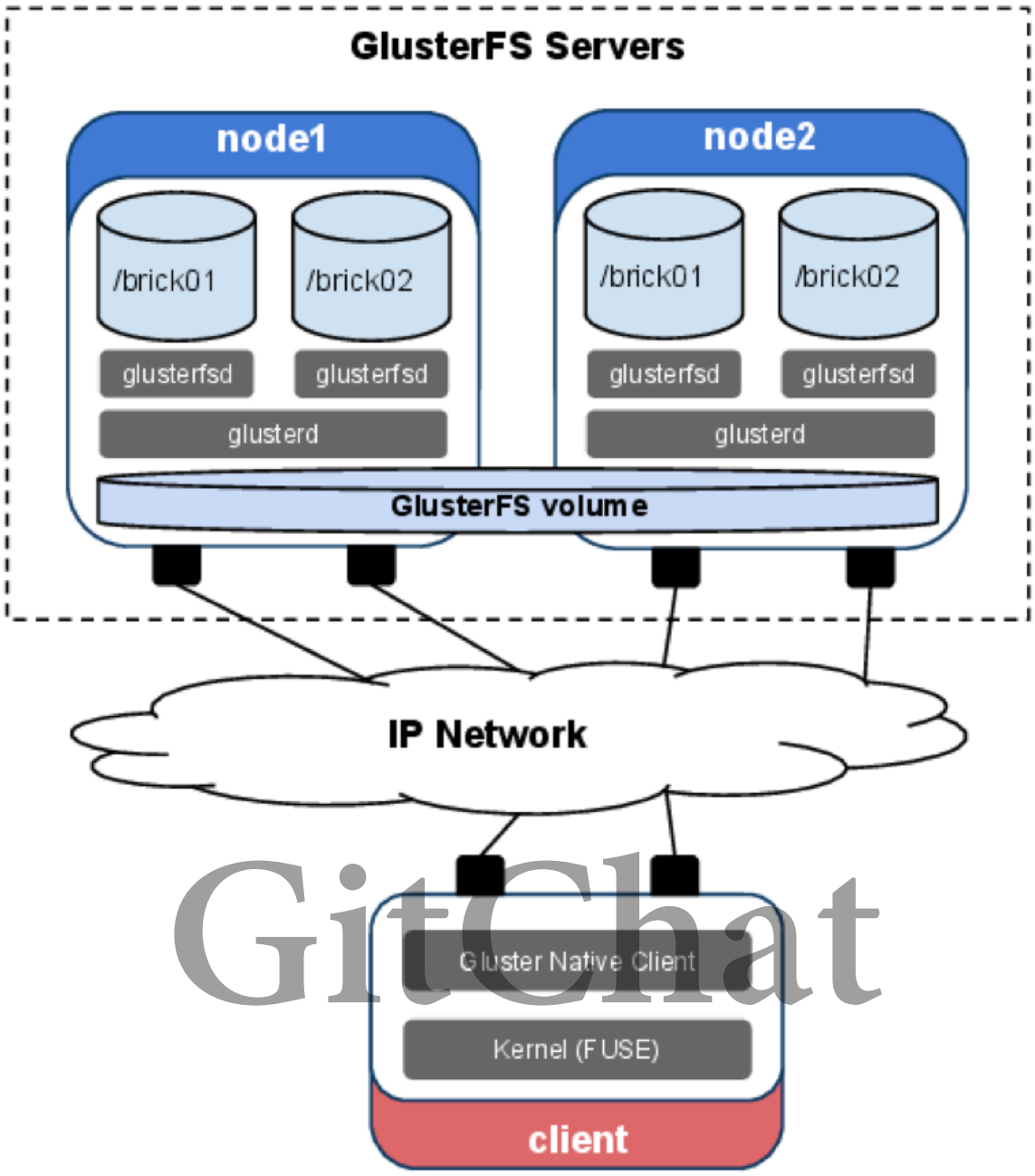
虽然，GlusterFS和CephFS都是比较专业的技术，我通过官方的红帽技术文档也可以看出一些技术特性：



Source:

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/2/html/ceph_file_system_guide_technology_preview/what_is_the_ceph_file_system_cephfs

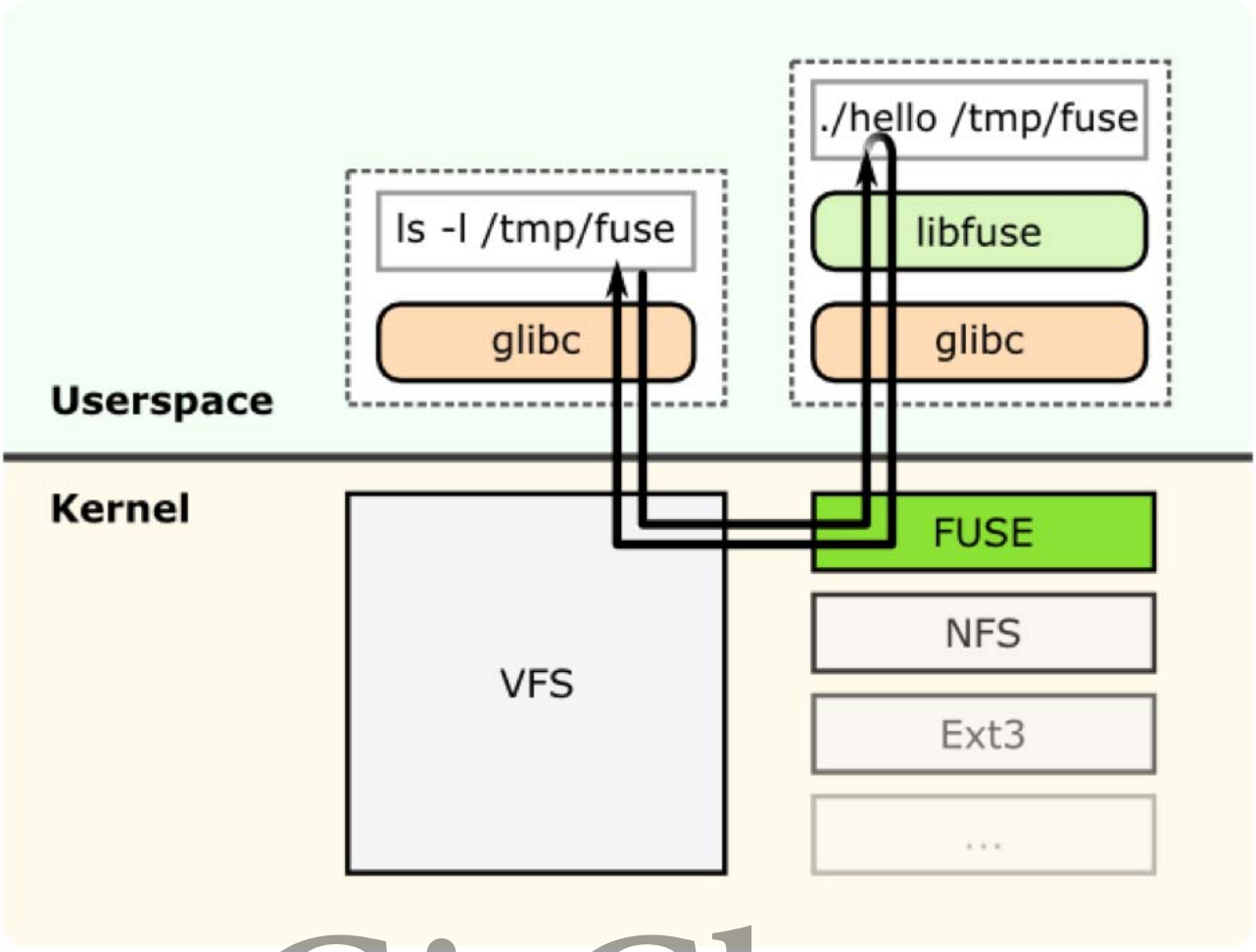
其他我不太懂，我看到了一个FUSE关键字，我们在看看GlusterFS



Source:

http://deliveryimages.acm.org/10.1145/2560000/2555790/11549f1.png

GusterFS架构还是用到了FUSE。那么这里FUSE就成为了我需要普及的技术点了。FUSE缩写对于外行一定是看不明白的，所有用维基百科科普一下，Filesystem in Userspace (FUSE) ，技术图如下：



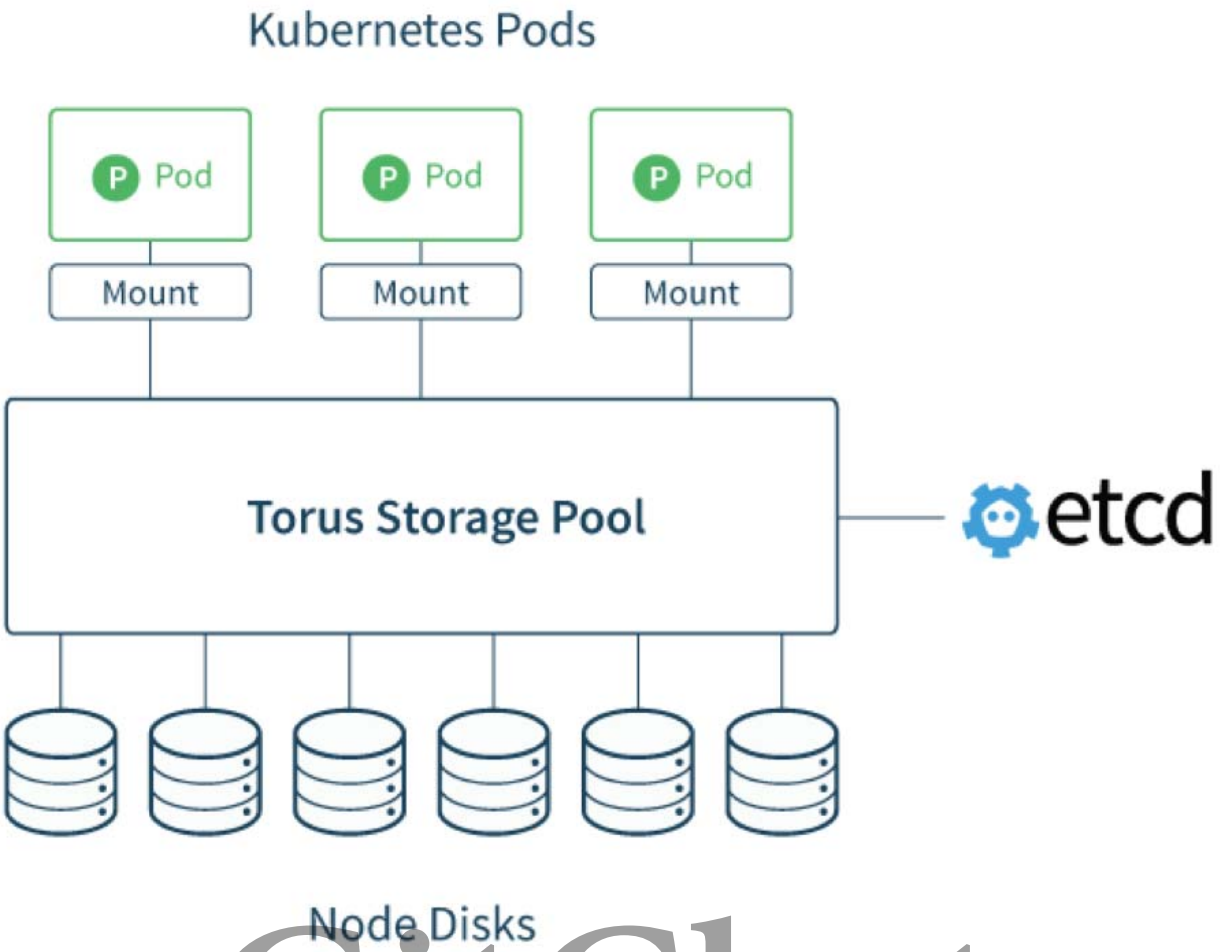
Source:

https://en.wikipedia.org/wiki/Filesystem_in_Userspace#/media/File:FUSE_structure.svg

FUSE是一套软件接口，借助这套接口可以实现在非Kernel层实现独立的文件系统。很厉害的一套接口定义。从维基百科的FUSE网页里看到实现了FUSE的软件中就有GlusterFS。在分析完这两套成熟的文件系统方案之后，大家心里面一定有谱了。当前，成熟的方案GlusterFS支持容器存储是比较理想的，文件已经分散到各个主机了，并且还能容灾。这也是红帽现在主推的容器级别存储解决方案。

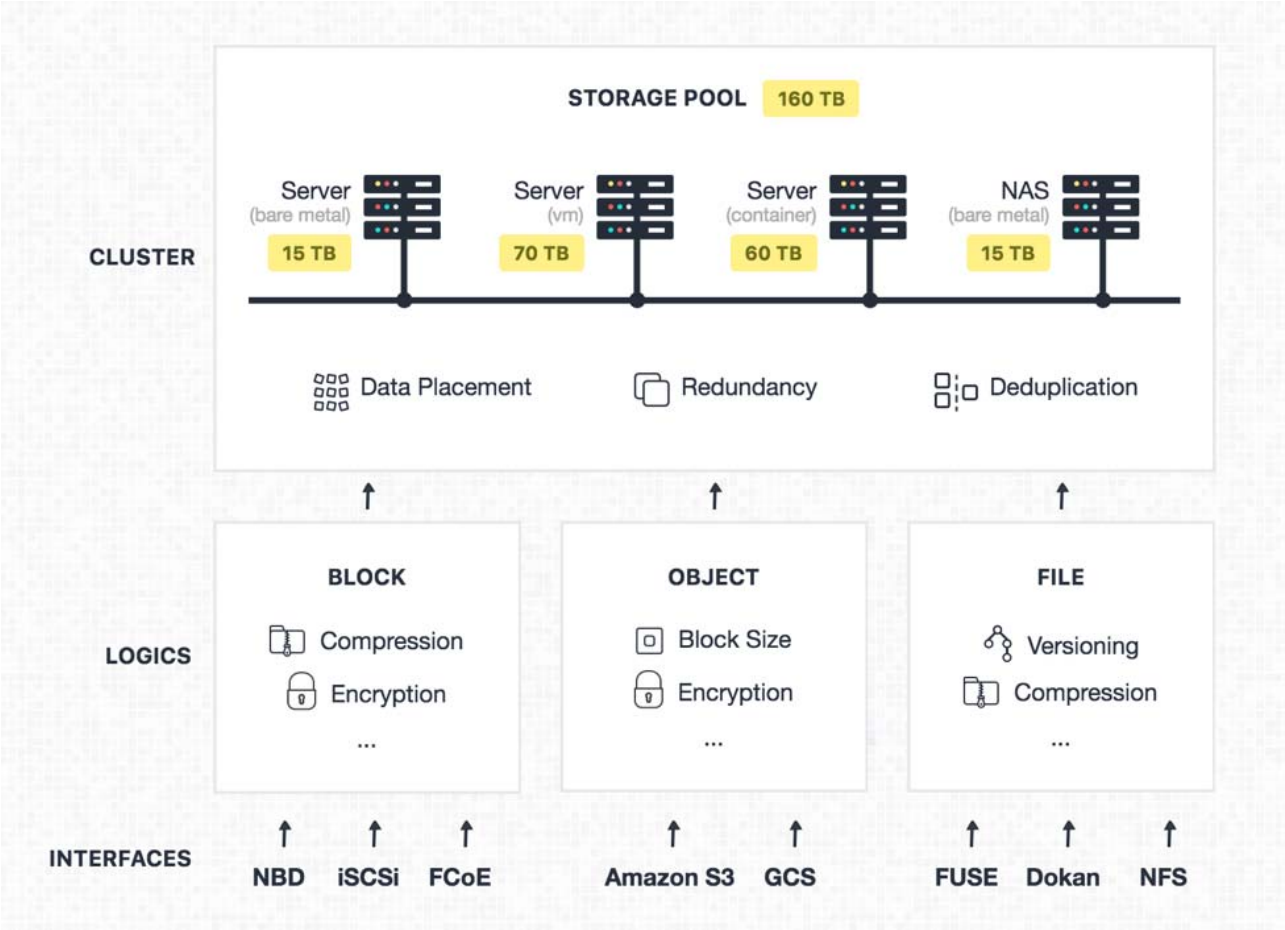
除了这些主流的方案，能折腾的各大帮派肯定有不服气的。所以，我不得不介绍一下当前的存储技术的形式。方便大家做二次创业的时候能用的上。

Torus Distributed Storage是CoreOS发布的容器存储解决方案。CoreOS是做容器OS的，所以不用多想，它肯定现在存储领域杀出一遍天地。代码完全用Go写的，所以我们要看看它的实现套路是：

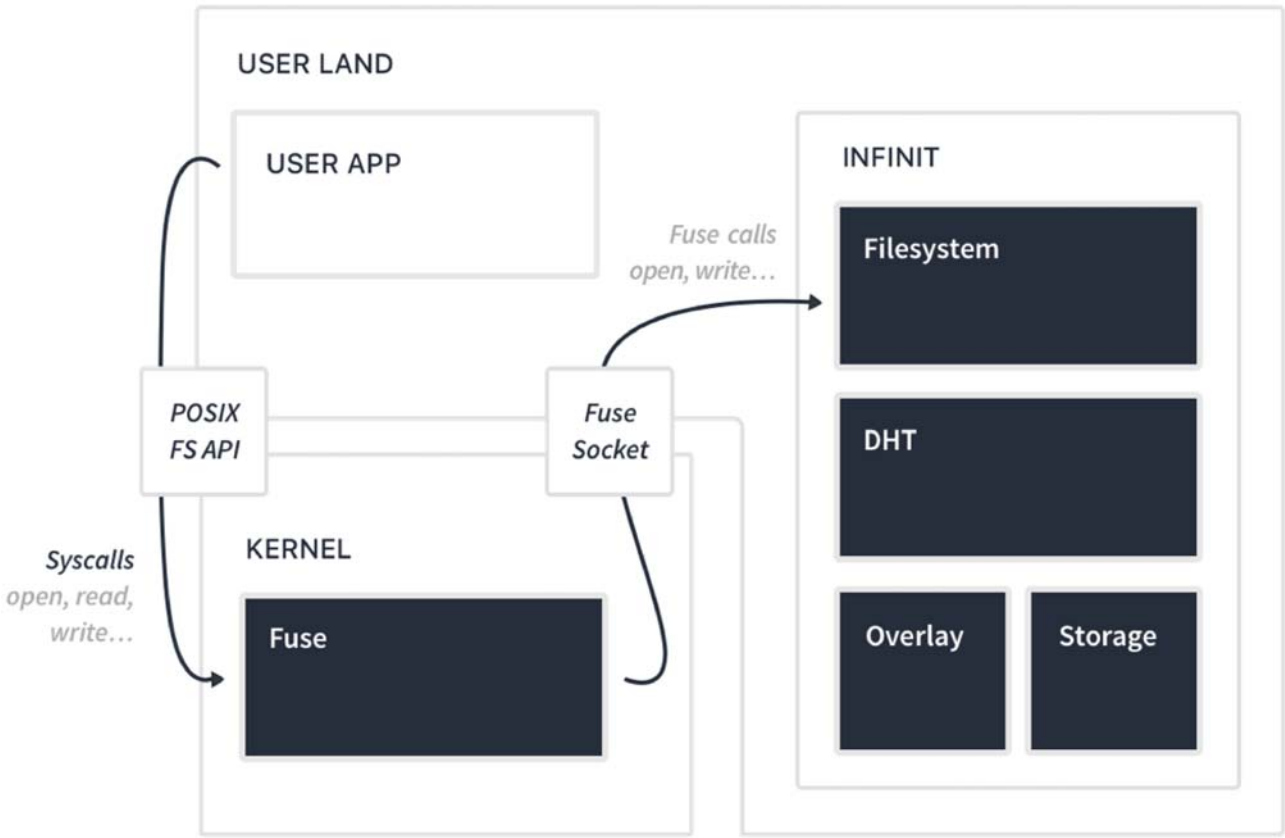


不过开源项目最惨烈的就是搞几个月没有达到预期，就死掉了。CoreOS的开发者很任性，文中提到8个月时间内并没有达到他们期望的效果，这套系统没有在Kubernetes社区占到任何便宜，所以在2017年2月停止了开发，目前正在寻找另外一套开源系统rook(https://github.com/rook/rook)的支持，期望合并社区分支。妈蛋，1千多颗star就这么白白浪费了。这里引出来的rook项目，号称云原生环境下对分布式系统的开源调度器。啥叫云原生啊，其实扯开遮羞布就是Kubernetes么，懂了吧。2016年11月发起的项目，目前为止还是Alpha。它想干啥呢，就是在Kubernetes的基础之上调度Ceph存储块。为什么Ceph做不了，因为没有Kubernetes么。用上容器，不仅仅是文件系统要关心，你还要考虑automating deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management等等。这个十八弯的设计思路确实把我绕晕了。我并不是特别喜欢这种取巧的办法，你要是感兴趣，可以深入研究一下。

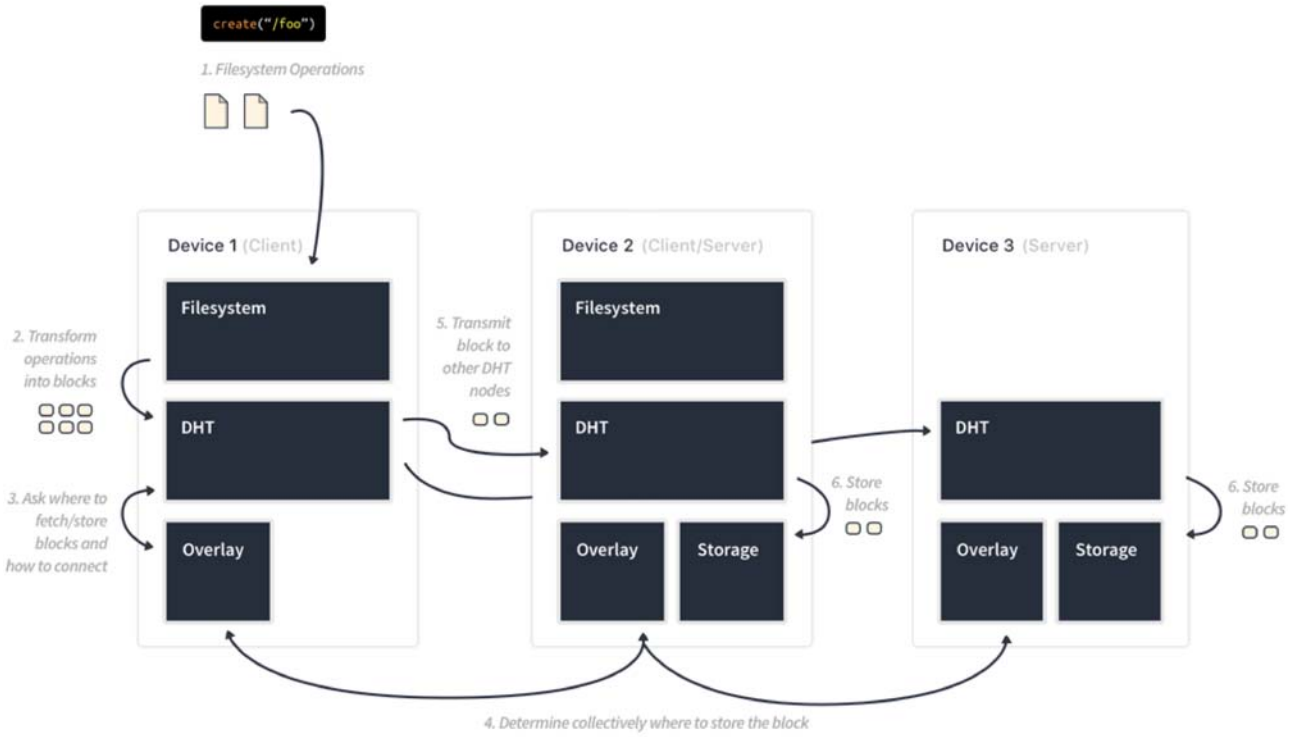
不过开源项目最惨烈的就是搞几个月没有达到预期，就死掉了。CoreOS的开发者很任性，文中提到8个月时间内并没有达到他们期望的效果，这套系统没有在Kubernetes社区占到任何便宜，所以在2017年2月停止了开发，目前正在寻找另外一套开源系统rook(https://github.com/rook/rook)的支持，期望合并社区分支。妈蛋，1千多颗star就这么白白浪费了。这里引出来的rook项目，号称云原生环境下对分布式系统的开源调度器。啥叫云原生啊，其实扯开遮羞布就是Kubernetes么，懂了吧。2016年11月发起的项目，目前为止还是Alpha。它想干啥呢，就是在Kubernetes的基础之上调度Ceph存储块。为什么Ceph做不了，因为没有Kubernetes么。用上容器，不仅仅是文件系统要关心，你还要考虑automating deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management等等。这个十八弯的设计思路确实把我绕晕了。我并不是特别喜欢这种取巧的办法，你要是感兴趣，可以深入研究一下。



支持多重文件接口，成为万能存储Hub。然后把块，对象，文件三个类别的逻辑都提供了企业级别的加密，压缩，版本控制的特性支持。在此基础之上在实现各种你需要的存储场景，不管是单机的磁盘，虚拟机上的磁盘，容器盘，还是网络盘，全面支持。以至于我以为自己到了乌托邦的设计中了。好奇心害死人，怎么做到的。Infinit号称开源软件，但是能看到的代码非常有限，先给个图吧：



怎么个意思呢。用infinit客户端在用户系统里面创建一套基于DHT协议的文件系统，同时维护一套overlay网络，相当于种子群，并且支持传统的存储系统。用户应用看到的就是应用目录。和Dropbox网盘相比呢，主要是不需要完全克隆整个目录的文件，infinit是即需即用，有P2P协议么。行家懂的明白，这个好是好，如何做好副本算法的冗余调度才是关键。



当然，这套系统最牛的地方是用到了DHT，外加一套闭环的Overlay网络形成了一套近乎完美的存储管理平台。

因为这帮喊着开源但还没有开源的infinit开发者，幸好在get-started页面放出了命令行工具(https://infinit.sh/get-started)，得以让我们在不费吹灰之力之下就可以玩转起来这套系统。

第一步，Fuse驱动需要安装上。允许非root帐号可以访问FUSE接口。

```
echo "user_allow_other" >> /etc/fuse.conf
```

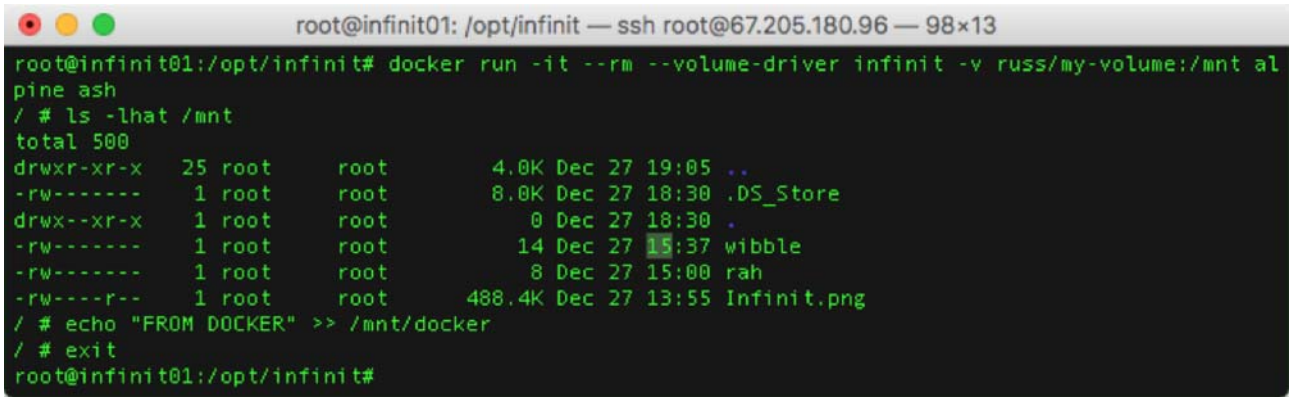

第二步，启动docker volume plugin服务进程

```
infinit-daemon --start --as russ --docker-user root
```

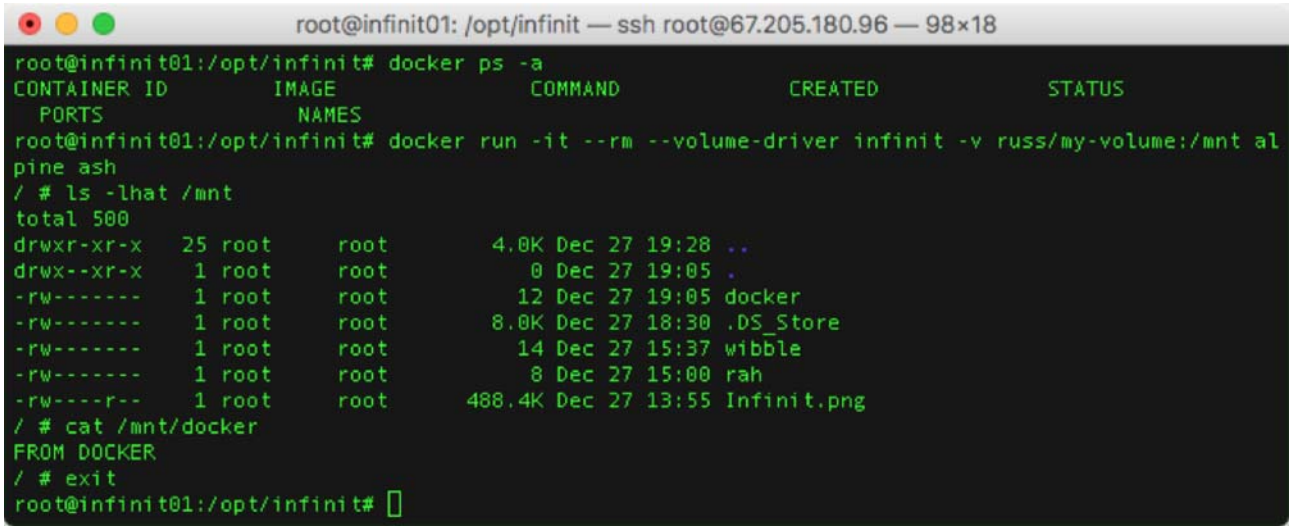
第三步，启动容器挂载infinit存储:

```
docker run -it --rm --volume-driver infinit -v russ/my-volume:/mnt alpine ash
```

看下面的截图，是写入一行信息FROM DOCKER到文件/mnt/docker中：



在启动一个容器，读取这个文件看到FROM DOCKER信息。



Infinit厉害的地方是把复杂的分布式存储配置变成了单机版容器存储的体验。也难怪Docker公司收购它并在做孵化整合。

从这次容器存储技术的归档梳理之后，我发现当前容器存储的趋势分为两大类型，一种是做存储编排的，合理的调度各种存储资源。另外一种是提供一套P2P的文件分布式系统，支撑容器存储的分布式需求。我觉得infinit这种的超前尝试确实有它的趋势，DHT+Overlay network + Hashtable，如果想玩更深入一点，可以尝试手工打造一套自己的存储系统，其实也没有多复杂。