http://bbs.aliyun.com/read/179606.html?spm=5176.7189909.0.0.WhWVdB

http://bbs.aliyun.com/read/180225.html?spm=5176.7189909.0.0.S2X1h0

http://bbs.aliyun.com/read/182036.html?spm=5176.7189909.0.0.5AS2wV

[**[业务架构实践]**](http://bbs.aliyun.com/thread/248.html?type=1040)**应该如何使用阿里云？基础篇**

**前言**

大家好。我们是成都瑞小博科技 (www.rippletek.com)。

自从去年10月在阿里云购买了第一台ECS并部署服务后，到现在已过去了一年。在这一年间，随着业务的扩张和用户规模的增长。我们的ECS数量从1台增长为20台，并开通了SLB, RDS, OSS, CDN,OCS, SLS, MQS等多项业务。在这一年中，通过工单系统得到了阿里云的技术支持团队和开发团队的大量帮助和指导。在"应该如何使用阿里云"这个问题上，也积累了一些心得和经验，希望和大家分享，抛砖引玉。

**初探ECS**

使用阿里云的第一步， 当然是购买一台ECS(后面称为ecs1)。这台ECS应该是什么样的配置呢？一般情况下，我们建议的最低配置是单核，2G内存。如果只有1G的内存，运行阿里云的一键lamp安装脚本可能会oom。操作系统，如果是从头开始建站，建议使用Ubuntu 14.04. 相比CentOS 6.5, Ubuntu的内核版本更高，能玩的花样也更多(比如docker)。带宽的话，1M其实就够了。无论如何，不要超过5M。根据阿里云的带宽阶梯计价公式，1M带宽的月单价是21元，2M比1M多23(即44元，后面类推)，3M比2M多25， 4M比3M多27， 5M比4M多29， 而一旦超过5M， 每多1M带宽，价格就多100元! 所以，带宽越高，每M的单价也越高。同时，对于需要高可用的服务，根本不可能通过ECS的外网带宽开出去。所以，购买1M带宽，获得一个外网IP，就足够了。至于数据盘，一般情况下，你不会需要它。ECS购买完成后，你会收到阿里云的短信通知，发给你root密码。**用那个密码登入ECS后，应该做的第一件事是加入authorized\_key并删除root密码**:

[复制代码](javascript:)

1. mkdir .ssh
2. chmod 700 .ssh
3. vi .ssh/authorized\_keys #加入自己的id\_rsa.pub
4. chmod 600 .ssh/authorized\_keys
5. passwd -d root

这样，可以在根本上杜绝root密码被猜中的可能性。

**初探SLB**

第二步，购买一个公网的负载均衡器SLB(后面称为slb1)。如果你的域名已申请好，可以把域名指向slb1了。**为了保证服务的高可用性，服务一定要通过SLB开出去。**设想一下，如果你的服务是走单台ECS出去，某天你需要修改一下apache的配置，修改完毕后，service httpd restart却发现httpd进程没有起来 (由于httpd.conf里有个小小的笔误..)。事实上，这个时候你就已经停服了！这也是为什么前面建议不必购买高带宽ECS的原因。

为了避免出现上面提到的那种尴尬的情况，SLB的后端服务器至少要有两台ECS并配置好health check。 尽管我们现在只有ecs1，如果上面的服务环境已经部署ok, 就先把它加入到slb1的后端服务器组里，然后在浏览器里输入我们的域名开始测试吧！然后，我们还需要另一台ECS。

**再探ECS**

第三步，再购买一台ECS（后面称为ecs2）。这一次，如果我们的业务代码本身不用从internet上获取数据的话，我们可以试试内网ECS，即0M带宽的机器。这样的机器没有带宽费用，会便宜一些，更重要的是，它没有公网IP，所以也更安全。下面问题来了：

* 它没有公网IP，我如何ssh上去？
* 它无法访问internet，我如何使用yum, apt这类包管理器来搭建业务环境？

对于第一个问题，有两个解决办法：

1. 先ssh到有公网IP的ecs1上，然后再ssh ecs2的内网IP，搞定
2. 也可以通过一些配置，实现内外网无感知的直接ssh上去，方法如下：

* 在ecs1上安装nc工具，并修改/etc/hosts, 加入 ecs2 <ecs2内网ip地址>的解析
* 修改本地终端的.ssh/config, 加入配置

[复制代码](javascript:)

* 1. [li][li][li][li][li][li][li][li][li]Host ecs2[/li][li]    ProxyCommand ssh root@ecs1 exec nc %h %p[/li][/li][/li][/li][/li][/li][/li][/li][/li]

在本地终端的/etc/hosts中加入 ecs1 <ecs1外网ip地址>的解析

* 然后，ssh root@ecs2 即可直接登录ecs2

对于第二个问题，也有两个解决办法:

1. 在ecs1上架设如privoxy一类的http代理，然后在ecs2的/etc/bashrc文件中加入这行代码 export http\_proxy="http://ecs1:<proxy\_port>"，然后ecs2就可以走ecs1的代理使用curl, yum等工具像外网ECS一样工作了。
2. 给ecs1的系统盘做个磁盘快照，并建立一个自定义镜像，然后**对ecs2使用“替换系统盘”的操作把系统盘替换为我们刚刚创建的自定义镜像就可以立即获得和ecs1完全一样的系统环境**！

我们一般是把上面两个办法结合起来使用，快速搭建环境。在ecs2环境ok后，也将它加入slb的后端服务器组。至此，我们消除了ecs1的单点，再也不用为重启web server这种事情纠结了。

但是ecs2并没有公网IP，如何才能访问它以确认服务OK? 有两种办法:

1. ssh tunnel. 在测试机器test和ecs2分别建立ssh tunnel到ecs1的同一端口对接起来.
2. 建一个测试用的公网SLB作为代理网关，里面只放ecs2一个节点。用公网SLB的IP就能访问ecs2了。显然，第2种办法要简单得多，所以我们一般用这种办法 :)

在系统设计上，方便性和安全性存在冲突，内网ECS亦是一例。在[进阶篇](http://bbs.aliyun.com/read/180225.html?spm=5176.7189909.0.0.p4fEUg) 中，我们会进一步讨论内网ECS的优缺点。

**分布式计算的几个基本概念**

在继续我们后面的云端飞行旅程之前，我先介绍几个分布式计算的基本概念。阿里云平台提供的几个基础服务正是为了解决这几个问题才开放出来的。

1. Single Point Failure (SPF), 单点故障。在一个服务系统中，因为一个节点的故障，导致整个系统不可用。例如上面提到的网站域名指向ecs1的IP，但ecs1上的web server restart失败的情形
2. Redundancy，冗余。为了解决SPF，在系统的关键路径上，必须存在两个或更多的选择。即当一个节点失效后，仍有其他的节点可以继续提供服务。
3. Stateless Node，无状态节点。为了让多个计算节点具有完全一致的行为，单个节点上面一定不能有自己独有的状态信息。
4. Load Balance, 负载均衡。多个计算节点共同承担系统负载。

SLB这一工具已经解决了1,2,4这三点。那么，我们如何从ECS上抽离状态信息以实现无状态节点呢？为此，阿里云提供了另外两个工具：**RDS抽离动态数据和OSS抽离静态数据将ECS变为无状态的计算节点**。可以把RDS当成一个超大型的Mysql数据库集群，OSS当成一个超大型的文件存储集群。

**RDS和OSS**

我们先讨论RDS。虽然阿里云提供了lamp的一键安装包sh-1.3，但事实上，本地的mysql服务是不需要的。RDS的可靠性，易用性和可扩展性都比自己搭建的mysql服务器要好很多。如果你仍在使用自己的mysql数据库，强烈建议迁移到RDS。

再说说OSS。ECS的本地磁盘上只应存放业务代码和配置，20G的系统盘完全够用，这就是为什么前面不建议购买数据盘的原因。用于下载或用户上传的数据文件都可以存到OSS中。

对于一个基本的建站业务而言， 将动态数据存RDS，静态文件存OSS，就实现了ECS的无状态。在这样的框架下，可以方便的对系统扩容:

1. 当访问量变大，现有计算资源紧张时，对ECS进行横向(增加ECS的节点个数)或纵向(增加单个节点的CPU核心和内存)扩展
2. 如果数据库访问出现瓶颈，升级RDS实例至更高配额

**CDN和OCS**

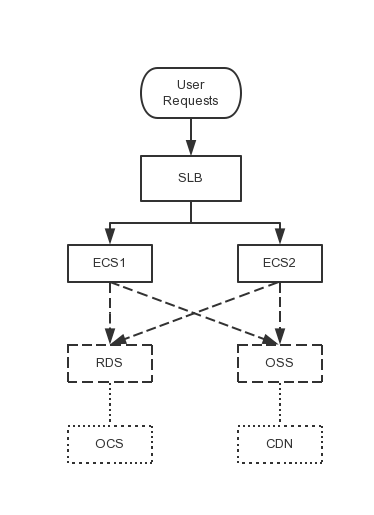
后面，很自然的可以开通CDN为OSS的访问加速。这一点，从定价上不难看出OSS+CDN是阿里云希望用户使用的方式。OSS的流量是0.8元/GB,而CDN是0.4. OSS通过CDN出去，不仅访问速度更快，还能降低成本。

如果RDS中有一部分数据的读取频率比更新频率高很多，可以通过将这部分数据缓存入OCS，在提升系统响应速度的同时缓解RDS的读取压力。RDS+OCS业务代码的实现逻辑:

1. 如果要从RDS的表table0中读取主键为id的数据项，先尝试从OCS中读取key="table0\_id"的数据，如果命中，则使用从缓存中读取的数据，不用再访问RDS
2. 如果缓存不命中，访问RDS读出数据v，以key="table0\_id", value=v写入OCS
3. 当table0中的id数据项被变更或删除的同时，删除OCS中key="table0\_id"的数据

小结

综上，我们介绍了如何使用阿里云提供的ECS,SLB,RDS,OSS,CDN和OCS这6个业务来构建我们自己的高可用业务网站系统。它的基本结构如下图所示：



几点说明:

1. 通过外网SLB的IP向外提供服务，用户请求通过SLB分发到后端的几个ECS来计算处理
2. 利用RDS和OSS来抽离状态数据，使得ECS成为无状态的计算节点
3. 可以利用OCS缓存RDS的数据， CDN缓存OSS的数据，提高访问速度

至此，我们的业务系统设计暂告一段落。然而，我们的云计算旅程才刚刚开始，后续更多精彩内容请见 [应该如何使用阿里云？(进阶篇)](http://bbs.aliyun.com/read/180225.html?spm=5176.7189909.0.0.lCjLc8)

[**[业务架构实践]**](http://bbs.aliyun.com/thread/248.html?type=1040)**应该如何使用阿里云进阶篇**

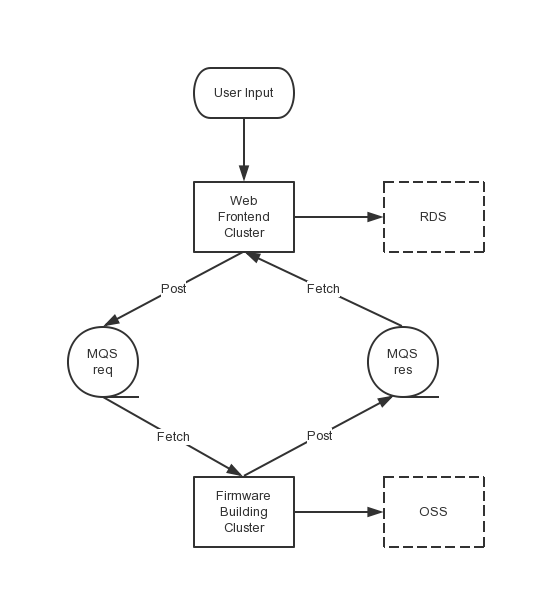
**前情提要及概述**

在[基础篇](http://bbs.aliyun.com/read/179606.html?spm=5176.7189909.0.0.qFN0Du)中，我们介绍了如何利用阿里云的几个基础服务: ECS, SLB, RDS, OSS, CDN和OCS来构建一个高可用的业务网站系统。在本篇中，我们将进一步介绍上面这些基础工具，以及如何从单业务系统拓展到多业务系统，和日常开发和运维的一些常用技巧。

**多业务系统之间的交互手段**

我们从几个具体的case说起吧 :)

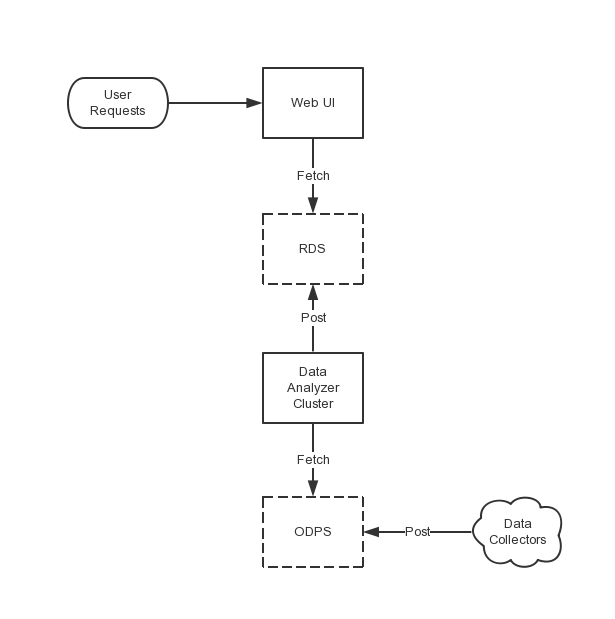
case #1 利用消息队列实现的固件定制系统



如上图所示，先是由一个web前端的表单系统将用户的输入参数组装为一条消息放入固件生成的请求队列中，后面由固件构建集群取得用户的配置参数，生成固件后存入OSS，再将生成的结果放入结果消息对列中，前端系统获取结果后更新RDS的状态。

我们可以利用阿里云提供的MQS服务非常方便的在两个业务系统之间实现用消息队列异步传输小尺寸数据。如果尺寸较大，亦可以通过往RDS/OCS或OSS中写入临时数据，用消息队列传输key或url来解决。

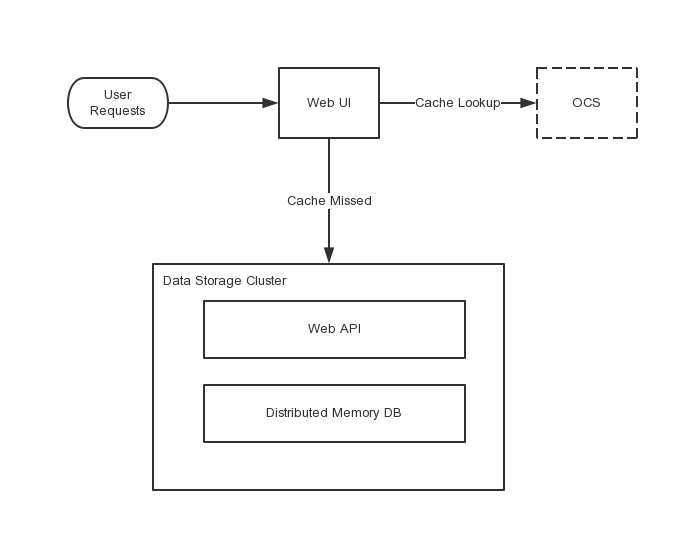
case #2 利用ODPS/RDS实现的非实时数据分析系统



ODPS是一个阿里云上的接口类似Hadoop + Hive的数据分析系统。我们可以部署多个数据收集节点，将数据存入ODPS。再搭建一个数据分析集群，定期对写入ODPS的新数据进行提取，然后将提取的结果放入RDS。最后由一个web前端系统读取RDS中的数据生成报表呈现给最终用户。

这种reader/writer的模式是两个业务系统之间通信的常见方式，只要是两个系统之间可以共享的系统资源，都可以通过一端写入，另一端读出来实现通信。而在阿里云的系统设计中，同一账户下的ECS可以共享RDS，OCS, MQS, OSS, ODPS这些资源。我们应当针对不同的应用场景选择合适的资源类型加以利用。

case #3 利用OCS/内存数据库构建实时数据分析系统



web前端系统接受用户查询请求后，先查找OCS是否有缓存的结果，如果cache命中则直接返回结果。如果cache不命中，调用后端数据存储集群的web api. web api负责查询分布式内存数据库并计算分析后返回统计结果。前端系统拿到查询结果后，用查询参数hash出一个key，将查询结果作为value存入OCS中。注意：该系统的实时性在很大程度上取决于OCS缓存的expiration. 应依据业务特点设置合适的expiration值。如果对结果的实时性要求很高并且后端数据存储集群的计算性能有充分安全边际的情况下，也可以移除OCS，每次都重新提取数据计算结果。

同样，web api系统也务必消除单点。在这里，我们可以使用内网SLB解决。

**再探SLB**

在[基础篇](http://bbs.aliyun.com/read/179606.html?spm=5176.7189909.0.0.qFN0Du)中我们介绍过外网SLB，并强调一切对外服务一定要通过外网SLB开放出去以消除单点。同样的，所有的系统内部通信web api也一定要通过内网SLB访问。内网SLB和外网SLB的区别在于内网SLB只有内网IP，没有外网IP，所以无法从internet上访问到。另外，内网SLB没有实例费和流量费，所以一定要多多的用起来！

SLB有两种监听转发方式， TCP和HTTP，一般情况下，web服务都采用http的转发方式，使用cookie来保持会话，这样即使在应用中使用本地文件来保存session，也不会成为问题。唯一的例外是给网站开启CDN时，为了消除CDN回源的单点，自然的，我们不能用单台ECS来回源，应考虑使用SLB，然而，如果使用HTTP + cookie保持会话，CDN会由于页面带了cookie而拒绝缓存。这种情况下，只能使用无会话保持的HTTP转发或TCP转发。

**小结**

至此我们展示了几个多业务系统的具体架构。云计算的组件组合方式可以是多种多样的。然而，在构建高可用系统这个问题上，有几个基本原则可以参考:

1. 务必消除单点。(如果读到这里您仍不理解什么是单点，请再看一遍[基础篇](http://bbs.aliyun.com/read/179606.html?spm=5176.7189909.0.0.CzZl9o) )
2. 尽量使用阿里云提供的系统服务，不要自己用ECS进行重复建设。例如，使用RDS，而不是自己搭建mysql集群; 使用OCS， 而不是自己搭建memcached集群；使用OSS， 而不是自己搭建文件服务器；使用ODPS， 而不是自己搭建Hadoop + Hive; 使用MQS, 而不是自己架RabbitMQ... 一是可以大幅降低投入，二是尽可能的把高可用问题交给阿里云的运维团队，而不是自己的运维团队解决，会有更佳的效果。
3. 设计异常恢复机制。任何系统都有可能会出现各种异常，阿里云也不例外。例如: ECS的宕机迁移会使ECS实例重启，最好在系统启动时即自动启动服务；RDS和SLB都有可能出现闪断的情况，需要自动重连；甚至云服务节点间的内网通信也有可能中断，导致内网SLB失效以及分布式数据库brain-split这类对服务质量有很大影响的问题，所幸在这一年里我们只遇到过一次这样的故障。
4. 安全平稳的线上代码变更。不管基础系统架构多么完美稳定，如果运行在上面的业务代码剧烈震荡，系统的可用性也还是个问题。所以在本篇的最后，我再介绍一下

**RippleTek的上线流程**

我们的每一个服务都是通过至少一个外网SLB开放出去的。在每个SLB的后面，至少有两个主服务节点SRV1和SRV2。另外，还有一个线上引流测试节点pilot。当需要线上代码进行变更的时候，先将pilot下线 (使用slb api将它从slb中移除，或者在控制台中将它的权重设为0. 这时你可能会问两个问题: 1) 把节点的权重设为0和直接将节点从slb中移除的区别是什么? 2) 为什么不用slb api将它的权重设为0? 这两个问题的答案请见下方的Q&A)，在pilot中的代码或配置变更不会对线上服务产生影响。同时，我们有一个测试专用的SLB\_TEST，后端服务器就只有一个pilot，自己测试就用SLB\_TEST的IP来做。功能测试OK后，把pilot重新加入线上服务的SLB，导入5%流量，持续观察日志5分钟看是否有异常情况。如无异则一小时后再观察。如一切正常，就将变更部署到该业务的全部主服务器上。部署后密切观察线上日志和监控，如有异常，先回退变更，再结合异常日志调查原因。该流程的有效性在很大程度上依赖于pilot的环境配置和SRV1/2的一致性，务必保证它们的环境是完全一样的!

到这里我们的旅程也暂告一段落了。稍事休息，更多精彩和乐趣将在 [应该如何使用阿里云？(高级篇)](http://bbs.aliyun.com/read/182036.html?displayMode=1) 中呈现。 敬请期待。

**Q&A**

* 把节点的权重设为0和直接将节点从slb中移除的区别是什么?

把节点A的权重设为0可以让SLB不再转发新的请求到A，但已经调度到这个服务器的连接继续保持，直到这些连接全部结束。而直接将节点A从slb中移除会导致已经调度到这个服务器的连接中断，可能会对服务质量有细微影响。

* 为什么不用slb api将它的权重设为0?

目前最新的slb api v20140515 尚未提供后端分发节点的权重设置。

[**[业务架构实践]**](http://bbs.aliyun.com/thread/248.html?type=1040)**应该如何使用阿里云？高级篇**

前情提要

在[基础篇](http://bbs.aliyun.com/read/179606.html?spm=5176.7189909.0.0.jFm8jC) 和[进阶篇](http://bbs.aliyun.com/read/180225.html?spm=5176.7189909.0.0.PwkG26) 中，我们先介绍了ECS, SLB, RDS, OCS等基础云计算工具，并给出了一个简单的高可用的业务网站的系统模型，然后我们通过分析几个具体的更复杂的多业务系统架构，总结了一些使用阿里云计算平台的一些基本原则。

**概述**

在本篇中，我们将讨论在保证系统可用性的基础上，如何快速扩容，如何提升计算资源的利用率的高级技巧。

这些技巧主要是依靠一种名为Docker的服务容器实现的。RippleTek的大部分线上业务，也是运行于Docker之中。Docker将是我们讨论的一大重点。

**再探ECS**

经测试，Ubuntu 14.04 64-bit可以稳定运行docker v1.3。CentOS 6.5 64-bit只能运行docker v1.1并且内核会报aufs相关的很多错误信息，无法正常使用。虽然没有逐个测试所有的Linux Distribution, 但我们推荐在Ubuntu 14.04 64-bit上使用docker. 对于一个全新的ubuntu CentOS 6.5 64-bit系统，用下面几行命令就可以安装好docker:

[复制代码](javascript:)

1. # apt-get update
2. # apt-get -y install curl
3. # curl -sSL https://get.docker.com/ubuntu/ | sh

为了保证系统安全，请单独再创建一个用户来运行docker，在这里我们把这个用户称为docker\_user

[复制代码](javascript:)

1. # adduser docker\_user
2. # passwd -d docker\_user #use authorized\_key only
3. # usermod -a -G docker docker\_user #add docker\_user to docker group

下面以一个运行于CentOS系统上的网站为例，描述它的docker化过程:

docker安装完毕后，就切换到docker\_user用户，pull一个Cent OS的image下来

[复制代码](javascript:)

1. $ docker pull registry.mirrors.aliyuncs.com/library/centos

运行这个docker image就相当于启动了一个CentOS的虚拟机:

[复制代码](javascript:)

1. $ docker run -t -i registry.mirrors.aliyuncs.com/library/centos /bin/bash

下面我们利用新开的这个docker的root shell打通ssh接入:

[复制代码](javascript:)

1. # /usr/sbin/sshd-keygen
2. # vi /bin/init\_docker.sh
3. init\_docker.sh的内容:
4. #!/bin/sh
5. /usr/sbin/sshd -D
6. # chmod +x /bin/init\_docker.sh
7. # adduser docker\_user  #增加一个同名user，同时，要注意它的uid也需与前面在ecs shell中创建的docker\_user一致 (使用id命令确认)
8. # #切换至docker\_user， 并加入你的key至~/.ssh/authorized\_keys以完成ssh认证
9. # #为了便于后面安装环境，在/etc/sudoer文件中将docker\_user配置为无密码sudo

在ecs中shell中提交这个container以保存当前的docker image.

[复制代码](javascript:)

1. $ docker ps #查看container id
2. $ docker commit <container id> my\_centos

然后我们就可以退出docker shell了。回到ecs shell，这时，docker ps发现刚才的那个docker已经不见了。那是因为它是用interactive mode启动的，shell退出后，它就stop了。可以用docker ps -a命令找出它的container id, 然后docker rm <container id>删除之。接下来，我们利用刚刚保存的image启动一个可以ssh接入的docker:

[复制代码](javascript:)

1. docker run -p 20022:22 -d -u root my\_centos /bin/init\_docker.sh

由于在启动命令行里指定一个host:20022 -> docker:22的端口映射，我们用ssh连接ECS的20022端口就能登陆进入docker了:

[复制代码](javascript:)

1. ssh -p20022 docker\_user@ecs

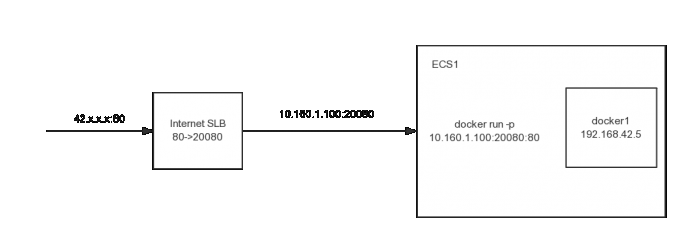
登入后下载并运行阿里云的LAMP一键安装脚本，搭建好网站的运行环境。通过scp部署网站代码。于是，我们的网站就在docker里运行起来了。下面问题来了:

1）docker是一个在192.168网段的内网VM，我们怎么把服务开出去?

2）折腾了这么半天，这个docker做的事情和我们没有用docker之前的ecs完全一样，都是提供一个网站服务，这到底有什么意义?

**三探SLB**

要解决问题1), 我们需要我们的老朋友SLB， SLB除了能负载均衡外，还有个很重要的功能 - 端口映射 (SLB前后端的端口号可以是不一样的!)。docker自身带端口映射，可以把docker中的服务端口映射到ecs的端口上，再通过SLB的端口二次映射就把服务开出去了。图示如下:



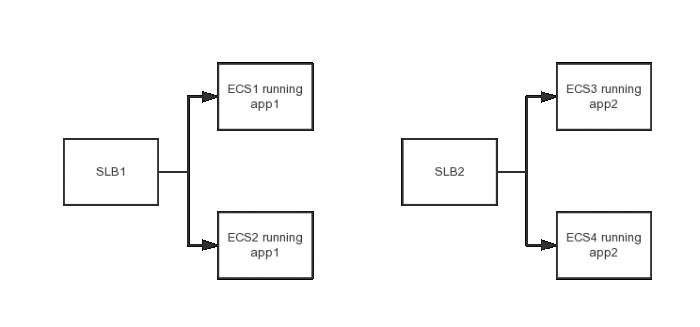
说明如下：

1. 由于我们的docker\_user不具有root权限，所以只能映射在ecs的高端口上，上图给出的例子中使用的是20080端口
2. SLB转发只使用内网IP，所以我们在启动docker就只把docker的服务端口(例子中为http端口80)映射在内网IP(docker run -p参数中指定)
3. SLB配置为80->20080的映射。这样，最终向42.x.x.x这个公网SLB发起的http请求经SLB转发至10.169.1.100的20080端口，再经docker进程的端口映射来到docker1的web服务器中

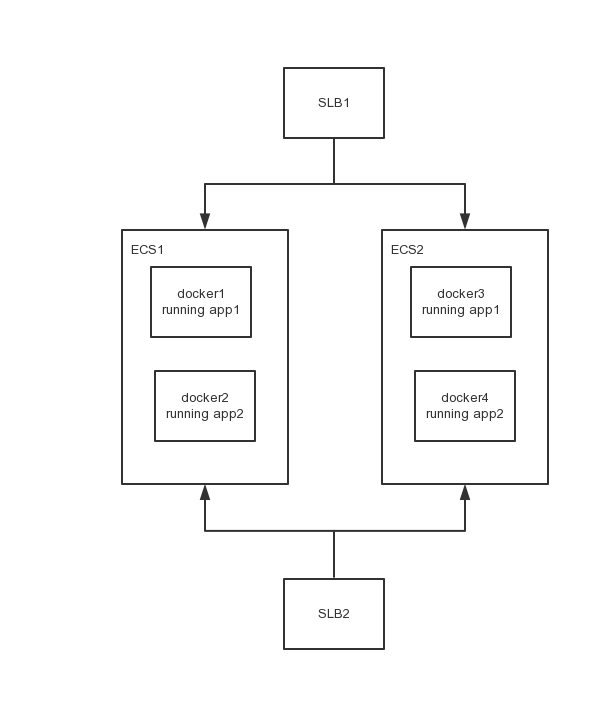
**计算节点**

前面我们介绍的计算模型中，都是将ecs作为单个计算节点来使用的。把服务docker化的意义在于可以快速方便的在一个ecs节点上运行多个服务，即将以前的单个计算节点拓展为多个计算节点。这是快速扩容和提升计算资源利用率的关键所在。docker化的服务资源前后对比图示:

docker化前:



docker化后:



将服务docker化后，我们可以很容易的在一个ecs节点上运行多个docker节点，于是，可以使用较少的ecs承载同等数量的业务并且消除单点。

**节点复制**

docker化前: 通过自定义磁盘镜像将一个ECS的系统盘快速复制到一个新的ECS上，需要重启ECS才能使用

docker化后: 从自己的private docker images repo中pull一个线上服务的镜像然后跑起来即可。可以通过一个自己编写的脚本一键秒起。

**结语**

我们在阿里云平台的使用经验分享至此就告一段落了。感谢大家这两个多月的持续关注和支持。这个系列帖子今后还会不断更新，结合阿里云的最新动态把我们总结出的best practice分享出来。同时，也希望各位如果发现帖子中有任何bug，能通过回复给我们提出来，让我们一起持续改进这个系列吧。

这一年多以来，我感到基于云计算平台的服务架构和以往传统的IDC托管服务相比，是两个完完全全不同的游戏。有很多以前很多不能想象或是不可能实现的事，现在都能轻松愉快的以很低的成本实现。云计算使得硬件和软件之间的界限，从来没有像今天这么模糊过，计算资源获取的便捷和计算的自由度亦到达了一个前所未有的高度。

* 