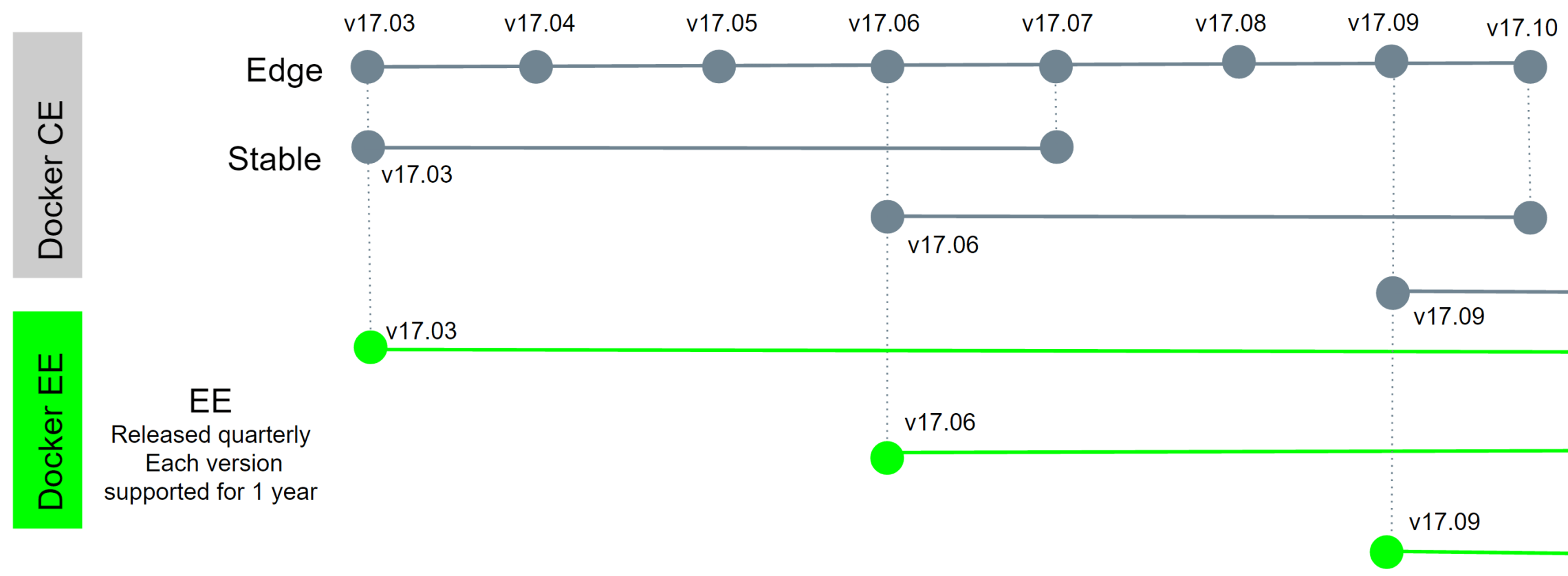
Docker从1.13.x版本开始，版本分为企业版EE和社区版CE，版本号也改为按照时间线来发布，比如17.03就是2017年3月，有点类似于ubuntu的版本发布方式。

企业版自然会提供一些额外的服务，当然肯定也是收费的。企业版说明https://blog.docker.com/2017/03/docker-enterprise-edition/

社区版分为stable和edge两种发布方式。

stable版本是季度发布方式，比如17.03, 17.06, 17.09

edge版本是月份发布方式， 比如17.03, 17.04......



Docker的linux发行版的软件仓库从以前的https://apt.dockerproject.org和https://yum.dockerproject.org变更为目前的https://download.docker.com, 软件包名字改为docker-ce和docker-ee。

https://download.docker.com/linux/static/stable/x86\_64/

# [Nginx发布1.9.0版本，新增支持TCP代理和负载均衡的stream模块](http://blog.csdn.net/libaineu2004/article/details/47780819)

昨天在公司微信群，CTO分享了这个消息，对运维来说以后基于TCP协议的后端业务的高可用又多了一个新的选择，实在是棒极了！

一直以来，Nginx 并不支持tcp协议，所以后台的一些基于TCP的业务就只能通过其他高可用负载软件来完成了，比如Haproxy。

这算是一个[nginx](http://zhangge.net/tag/nginx/)比较明显的缺憾。不过，在1.90发布后这个认知将得到改写：

|  |  |
| --- | --- |
| 2015-04-28 | [nginx-1.9.0](http://zhangge.net/go/?url=http://nginx.org/en/download.html) mainline version has been released, with the stream module for generic TCP proxying and load balancing.  [nginx](http://zhangge.net/tag/nginx/)-1.9.0 已发布，该版本增加了 stream 模块用于一般的 TCP 代理和负载均衡。 |

The ngx\_stream\_core\_module module is available since version 1.9.0. This module is not built by default, it should be enabled with the --with-stream configuration parameter.

ngx\_stream\_core\_module 这个模块在1.90版本后将被启用。但是并不会默认安装，需要在编译时通过指定 --with-stream 参数来激活这个模块。

其他改进包括：

* Change: 删除过时的 aio 和 rtsig 事件处理方法
* Feature: 可在 upstream 块中使用 "zone" 指令
* Feature: 流模块，支持 TCP 代理和负载均衡
* Feature: ngx\_http\_memcached\_module 支持字节范围
* Feature: Windows 版本支持使用共享内存，带随机化地址空间布局.
* Feature: "error\_log" 指令可在 mail 和 server 级别
* Bugfix: the "proxy\_protocol" parameter of the "listen" directive did not work if not specified in the first "listen" directive for a listen socket.

所以，我们如果需要用到这个功能，就需要加上 --with-stream 参数重新编译[nginx](http://zhangge.net/tag/nginx/)。对于已在线上运行的nginx，你可能要用到平滑升级来避免线上的服务被中断，可以参考张戈以前分享的教程：

《[Nginx在线服务状态下平滑升级或新增模块的详细操作记录](http://zhangge.net/4856.html)》

最后贴一下官方分享的stream模块的简单配置demo：

<http://nginx.org/en/docs/stream/ngx_stream_core_module.html>

worker\_processes auto; error\_log /var/log/nginx/error.log info; events {     worker\_connections  1024; }  stream {     upstream backend {         hash $remote\_addr consistent;         server backend1.example.com:12345 weight=5;         server 127.0.0.1:12345            max\_fails=3 fail\_timeout=30s;         server unix:/tmp/backend3;     }      server {         listen 12345;         proxy\_connect\_timeout 1s;         proxy\_timeout 3s;         proxy\_pass backend;     }      server {         listen [::1]:12345;         proxy\_pass unix:/tmp/stream.socket;     } }

和http模块类似，简单明了。相信熟悉 nginx 的朋友很容易的就能完成一个 nginx 下的 TCP 负载均衡集群配置。

由于工作繁忙，实在是心有余而力不足。还好最近公司给我招了个小鲜肉来做运维助理，等空下来了，我再去测一测这个 Nginx 的 TCP 代理和负载均衡功能。到时候再来博客分享一二，敬请期待！

下面测试nginx代理TCP协议的配置。

realserver : 10.134.241.68

nginx :10.134.72.166

客户端:10.129.157.168

TCP监听端口:2014

一、配置nginx

看官网上的文档 <http://nginx.org/en/docs/stream/ngx_stream_core_module.html>

nginx1.90对TCP协议的代理并不是默认开启的，需要在编译的时候配置 --with-stream 参数：

[img]http://images.cnitblog.com/blog2015/450613/201505/071746123452724.png[/img]

nginx.config文件参照官网：

stream {

upstream cloudsocket {

hash $remote\_addr consistent;

server 10.134.241.68:2014 weight=5 max\_fails=3 fail\_timeout=30s;

}

server {

listen 2014;

proxy\_connect\_timeout 1s;

proxy\_timeout 3s;

proxy\_pass cloudsocket;

}

}

启动nginx，发现nginx已经开始监听2014端口了



二、测试客户端连realserver

在客户端通过telnet连接realserver的2014端口：





在realserver上查看网络连接：



可以正常连接

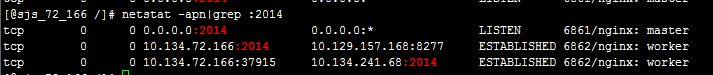
三、测试客户端连接nginx

在客户端通过telnet连接nginx所在服务器的2014端口





在nginx机器上查看网络连接



在realserver上查看网络连接



可以注意到nginx是给做了一个TCP连接的中转。

**说白了就是client和nginx有一个tcp长连接，nginx和realserver有一个tcp长连接，但是client和realserver之间并没有tcp长连接，仅由nginx服务器负责数据中转。**

[**浅谈Nginx负载均衡原理与实现**](http://blog.csdn.net/u012517198/article/details/50910739)

1 负载均衡

先来简单了解一下什么是负载均衡，单从字面上的意思来理解就可以解释N台服务器平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。那么负载均衡的前提就是要有多台服务器才能实现，也就是两台以上即可。

2 测试环境

由于没有服务器，所以本次测试直接host指定域名，然后在VMware里安装了三台CentOS。

测试域名  ：a.com

A服务器IP ：192.168.5.149 （主）

B服务器IP ：192.168.5.27

C服务器IP ：192.168.5.126

部署思路

A服务器做为主服务器，域名直接解析到A服务器（192.168.5.149）上，由A服务器负载均衡到B服务器（192.168.5.27）与C服务器（192.168.5.126）上。

域名解析

由于不是真实环境，域名就随便使用一个a.com用作测试，所以a.com的解析只能在hosts文件设置。

打开：C:WindowsSystem32driversetchosts

在末尾添加

192.168.5.149    a.com

保存退出，然后启动命令模式ping下看看是否已设置成功

从截图上看已成功将a.com解析到192.168.5.149IP

A服务器[nginx](http://www.php100.com/list-195/).conf设置

打开nginx.conf，文件位置在nginx安装目录的conf目录下。

在http段加入以下代码

u[ps](http://www.php100.com/fw/photo.html)tream a.com {

      server  192.168.5.126:80;

      server  192.168.5.27:80;

}

server{

    listen 80;

    server\_name a.com;

    location / {

        proxy\_pass         http://a.com;

        proxy\_set\_header   Host             $host;

        proxy\_set\_header   X-Real-IP        $remote\_addr;

        proxy\_set\_header   X-Forwarded-For  $proxy\_add\_x\_forwarded\_for;

    }

}

保存重启nginx

B、C服务器nginx.conf设置

打开nginx.confi，在http段加入以下代码

server{

    listen 80;

    server\_name a.com;

    index index.html;

    root /data0/htdocs/www;

}

保存重启nginx

测试

当访问a.com的时候，为了区分是转向哪台服务器处理我分别在B、C服务器下写一个不同内容的index.html文件，以作区分。

打开浏览器访问a.com结果，刷新会发现所有的请求均分别被主服务器（192.168.5.149）分配到B服务器（192.168.5.27）与C服务器（192.168.5.126）上，实现了负载均衡效果。

B服务器处理页面

C服务器处理页面

假如其中一台服务器宕机会怎样？

当某台服务器宕机了，是否会影响访问呢？

我们先来看看实例，根据以上例子，假设C服务器192.168.5.126这台机子宕机了（由于无法模拟宕机，所以我就把C服务器关机）然后再来访问看看。

访问结果：

我们发现，虽然C服务器（192.168.5.126）宕机了，但不影响网站访问。这样，就不会担心在负载均衡模式下因为某台机子宕机而拖累整个站点了。

如果b.com也要设置负载均衡怎么办？

很简单，跟a.com设置一样。如下：

假设b.com的主服务器IP是192.168.5.149，负载均衡到192.168.5.150和192.168.5.151机器上

现将域名b.com解析到192.168.5.149IP上。

在主服务器(192.168.5.149)的nginx.conf加入以下代码：

upstream b.com {

      server  192.168.5.150:80;

      server  192.168.5.151:80;

}

server{

    listen 80;

    server\_name b.com;

    location / {

        proxy\_pass         http://b.com;

        proxy\_set\_header   Host             $host;

        proxy\_set\_header   X-Real-IP        $remote\_addr;

        proxy\_set\_header   X-Forwarded-For  $proxy\_add\_x\_forwarded\_for;

    }

}

保存重启nginx

在192.168.5.150与192.168.5.151机器上设置nginx，打开nginx.conf在末尾添加以下代码：

server{

    listen 80;

    server\_name b.com;

    index index.html;

    root /data0/htdocs/www;

}

保存重启nginx

完成以后步骤后即可实现b.com的负载均衡配置。

主服务器不能提供服务吗？

以上例子中，我们都是应用到了主服务器负载均衡到其它服务器上，那么主服务器本身能不能也加在服务器列表中，这样就不会白白浪费拿一台服务器纯当做转发功能，而是也参与到提供服务中来。

如以上案例三台服务器：

A服务器IP ：192.168.5.149 （主）

B服务器IP ：192.168.5.27

C服务器IP ：192.168.5.126

我们把域名解析到A服务器，然后由A服务器转发到B服务器与C服务器，那么A服务器只做一个转发功能，现在我们让A服务器也提供站点服务。

我们先来分析一下，如果添加主服务器到upstream中，那么可能会有以下两种情况发生：

1、主服务器转发到了其它IP上，其它IP服务器正常处理；

2、主服务器转发到了自己IP上，然后又进到主服务器分配IP那里，假如一直分配到本机，则会造成一个死循环。

怎么解决这个问题呢？因为80端口已经用来监听负载均衡的处理，那么本服务器上就不能再使用80端口来处理a.com的访问请求，得用一个新的。于是我们把主服务器的nginx.conf加入以下一段代码：

server{

    listen 8080;

    server\_name a.com;

    index index.html;

    root /data0/htdocs/www;

}

重启nginx，在浏览器输入a.com:8080试试看能不能访问。结果可以正常访问

既然能正常访问，那么我们就可以把主服务器添加到upstream中，但是端口要改一下，如下代码：

upstream a.com {

      server  192.168.5.126:80;

      server  192.168.5.27:80;

      server  127.0.0.1:8080;

}

由于这里可以添加主服务器IP192.168.5.149或者127.0.0.1均可以，都表示访问自己。

重启Nginx，然后再来访问a.com看看会不会分配到主服务器上。

主服务器也能正常加入服务了。

最后

一、负载均衡不是nginx独有，著名鼎鼎的[apache](http://www.php100.com/list-121/)也有，但性能可能不如nginx。

二、多台服务器提供服务，但域名只解析到主服务器，而真正的服务器IP不会被ping下即可获得，增加一定安全性。

三、upstream里的IP不一定是内网，外网IP也可以。不过经典的案例是，局域网中某台IP暴露在外网下，域名直接解析到此IP。然后又这台主服务器转发到内网服务器IP中。

四、某台服务器宕机、不会影响网站正常运行，Nginx不会把请求转发到已宕机的IP上