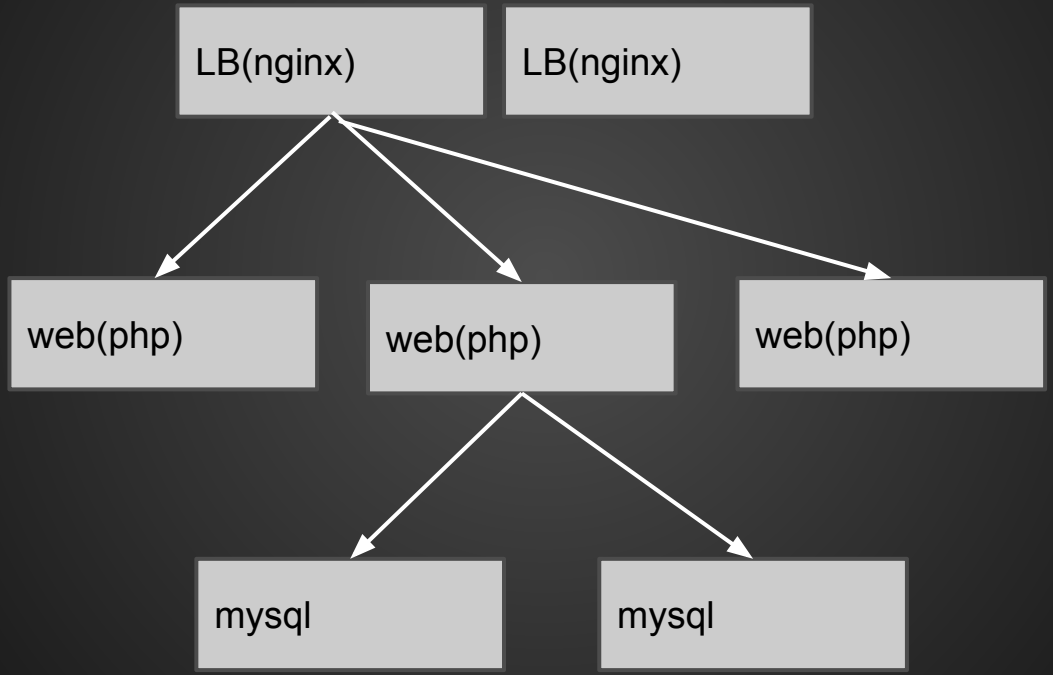
service discovery

docker+etcd+django template

为什么需要服务发现？

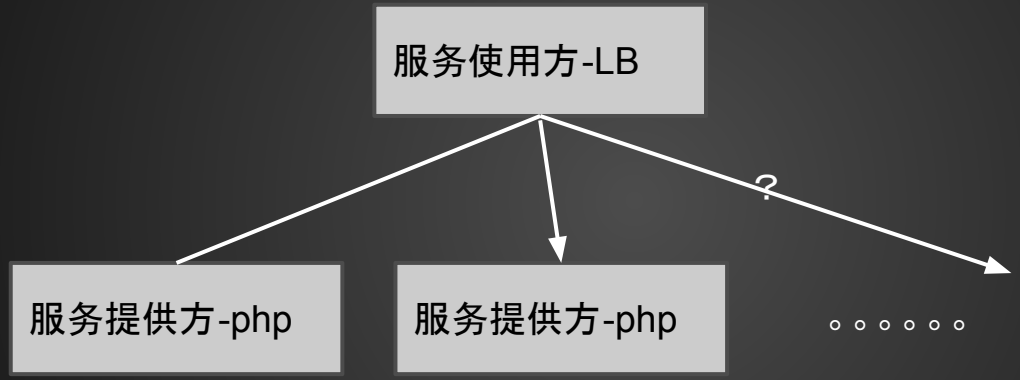
互联网架构



架构特点

- 最上(前)面的服务调用下层的服务
- 一层一层的调用关系
- 上层服务通过配置文件指定如何连接后面服务

简化图



手动档

- 服务调用方有配置文件来决定连接后端
- 后端服务器增加或者减少要手动修改前端的配置文件

自动档

- 后端服务器增加或者减少可以让前端自动的感知到
- 快速大批量部署方便，一次部署几十台到几百台后端服务器，前端迅速感知

服务发现闪亮登场了！

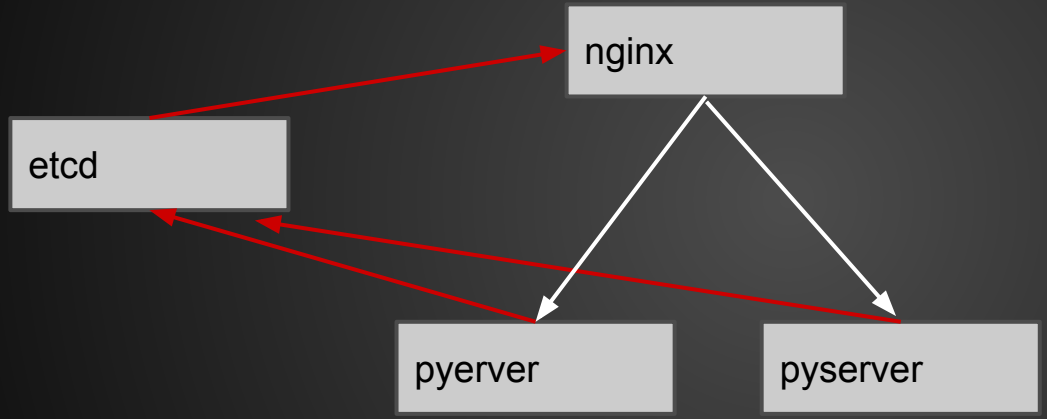
实现原理

- 存在一个中心配置节点
- 服务提供方会持续的向配置节点注册服务
- 服务调用方会持续的读取中心配置节点的配置并修改本机配置，然后reload服务

技术组件选用

- docker，快速部署的不二之选
- nginx，前端服务，服务调用方
- py server，后端服务，服务提供方
- etcd，中心配置管理节点
- django template，从etcd读取的配置参数结合模板，生成最终配置文件
- 服务发现脚本，服务注册脚本

服务发现架构



docker 运行etcd

图森破:

```
docker run -d -p 8001:8001 -p 5001:5001
coreos/etcd -peer-addr ${PUBLIC_IP}:8001 -
addr ${PUBLIC_IP}:5001 -name etcd-node1
```

pyserver dockerfile

```
FROM centos:latest
MAINTAINER lishuai <lishuai860113@gmail.com>
RUN mkdir -p /var/www/cgi-bin
ADD ip.py /var/www/cgi-bin/ip.py
ADD pyserver_reg.sh /usr/bin/pyserver_reg.sh
RUN chmod +x /usr/bin/pyserver_reg.sh /var/www/cgi-bin/ip.py
CMD ["/usr/bin/pyserver_reg.sh"]
```

服务注册pyserver_reg.sh

```
url="http://192.168.1.4:5001/v2/keys/service_dis/"
cd /var/www/
nohup python -m CGIHTTPServer&
while true
do
  echo $ipport
  curl $url$ipport -XPUT -d "ok" -d value=ok -d ttl=5
  sleep 1
done
```

红色部分关键，如果此容器死掉，那么这个etcd配置key会在ttl过期之后消失

构建启动pyserver

构建

docker build -t lishuai/pyserver:v1 .

启动

docker run -d -p 8000:8000 -e ipport=192.168.1.4:8000 lishuai/pyserver:v1

这里会传送环境变量进去给容器内部:“-e ipport=192.168.1.4:8000”

“-p 8000:8000”会把容器的8000端口转向到宿主机上的8000端口

docker 运行nginx

```
RUN yum install -y nginx python-pip
RUN pip install -i http://pypi.douban.com/simple/  django
ADD nginx.conf /etc/nginx/nginx.conf
ADD default.conf /etc/nginx/vhost/default.conf
ADD default.up.conf /etc/nginx/upstreamconf/default.conf
ADD service_dis.sh /usr/bin/service_dis.sh
ADD etcd2nginx.py /usr/bin/etcd2nginx.py
RUN chmod +x /usr/bin/service_dis.sh /usr/bin/etcd2nginx.py
CMD ["/usr/bin/service_dis.sh "]
```

服务发现service_dis.sh

```
while true
do
  tempfile=`mktemp`
  url="http://192.168.1.4:5001/v2/keys/service_dis/"
  services=`curl $url 2>/dev/null|python -m json.tool|grep "/service_dis/" |awk -F'[/|"]' '{print $6}'|tr "\n" ";"|sed -e "s/;$//"`
  etcd2nginx.py $services >$tempfile
  diff_line_count=`diff $tempfile /etc/nginx/upstreamconf/default.conf |wc -l`
  if [[ $diff_line_count -gt 0 ]]
  then
    mv $tempfile /etc/nginx/upstreamconf/default.conf
    /usr/sbin/nginx -s reload
  fi
  pgrep nginx || /usr/sbin/nginx
  sleep 1
done
```

模板生成配置etcd2nginx.py

```
#!/bin/env python
import sys
from django.template import Template
from django.template import Context
from django.conf import settings
settings.configure()
ss=sys.argv[1]
ts="'upstream default {
{% for x in servers %}    server {{x}};
{% endfor %}}'"
t=Template(ts)
a=ss.split(";")
r=t.render(Context({'servers':a}))
print r
```

构建启动nginx

构建

docker build -t lishuai/nginx:v1 .

启动

docker run -d -p 80:80 lishuai/nginx:v1

启动多个后端pyserver

```
for i in `seq 9001 9009`
do
  echo $i
  docker run -d -p $i:8000 -e ipport=192.168.1.4:$i lishuai/pyserver:v1
done
```

验证服务

http://192.168.1.4/cgi-bin/ip.py

可以看到每次刷新都是不同的后端响应

查看nginx配置

```
# cat /var/lib/docker/aufs/mnt/f625dd8821*/etc/nginx/upstreamconf/default.conf
upstream default {
    server 192.168.1.4:9001;
    server 192.168.1.4:9003;
    server 192.168.1.4:9004;
    server 192.168.1.4:9005;
    server 192.168.1.4:9007;
    server 192.168.1.4:9008;
    server 192.168.1.4:9002;
    server 192.168.1.4:9006;
    server 192.168.1.4:9009;
}
```

自动生成配置成功！

Q&A

THX