# 配置管理平台部署文档

环境:

CentOS 6/7 x64

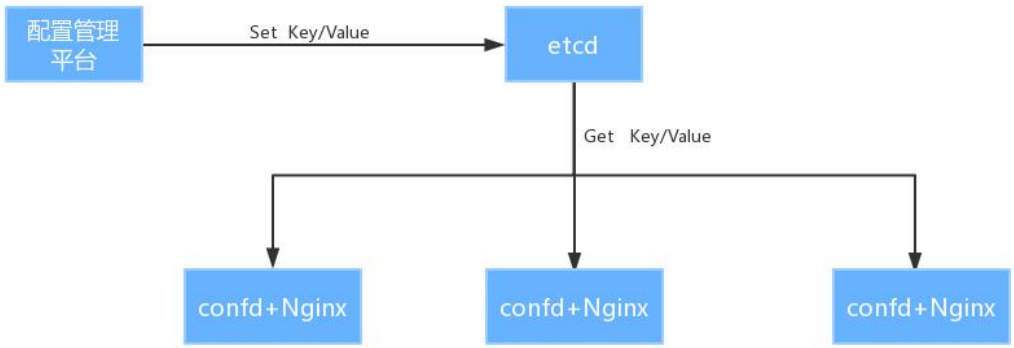
Python:  2.7.6

Etcd：  3.2.18

Confd:  0.16.0

Nginx: 1.12.1

## 一. 拓扑图:




## 二. 涉及软件

**etcd**：分布式 KV 存储系统，一般用于共享配置和服务注册与发现。是 CoreOS 公司发起的一个开源项目。 ETCD 存储格式类似于文件系统，以根"/"开始下面一级级目录，最后一个是 Key，一个 key 对应一个 Value。

**etcd 集群**：使用 Raft 协议保证每个节点数据一致，由多个节点对外提供服务。这里只用单台。

**confd**：管理本地应用配置文件，使用 etcd 或 consul 存储的数据渲染模板，还支持 redis、zookeeper 等。

confd 有一个 watch 功能，通过 HTTP API 定期监测对应的 etcd 中目录变化，获取最新的 Value，然后渲染模板

**Nginx**： Nginx 是一款轻量级的 Web 服务器/反向代理服务器以及电子邮件代理服务器，并在一个 BSD-like 协议下发行。由俄罗斯的程序设计师 Igor Sysoev 所开发，供俄国大型的入口网站及搜索引擎 Rambler 使用。其特点是占有内存少，并发能力强，事实上 nginx 的并发能力确实在同类型的网页服务器中表现较好。

# 三．软件部署

环境说明:　建议使用 Cento7.X　X64

**1) 安装 etcd(这里安装的单机,集群环境根据自己的需求选取)**

# yum install etcd -y

# sed -i　's/localhost/0.0.0.0/g'　/etc/etcd/etcd.conf　　#配置监听地址

# systemctl　start　etcd　&&　systemctl　enable　etcd　#启动服务设置开机动

```
[root@mail ~]# netstat  -tunlp |grep etcd
tcp        0      0 127.0.0.1:2380          0.0.0.0:*               LISTEN      123016/etcd
tcp6       0      0 :::2379                 :::*                    LISTEN      123016/etcd
```

**2) 安装 nginx**

#cd　/usr/local/src

#wget　http://nginx.org/download/nginx-1.12.1.tar.gz

#git clone https://github.com/yaoweibin/nginx_upstream_check_module.git

#tar　-zxvf　nginx-1.12.1.tar.gz

#cd nginx-1.12.1

#patch　-p1 </usr/local/src/nginx_upstream_check_module/check_1.12.1+.patch

#./configure　--prefix=/usr/local/nginx

--add-module=/usr/local/src/nginx_upstream_check_module/

make && make install

#mkdir　/usr/local/nginx/conf/vhost/

```
#user  nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

include   vhost/*.conf;
}
```

Nginx 主配置文件修改为这个样子,增加 include 目录配置

**3) 安装 confd**

下载地址 https://github.com/kelseyhightower/confd/releases
下载完毕丢到系统里面
# cp confd  /usr/bin/confd
# which  confd
/usr/bin/confd

**4) 创建配置文件目录**

# mkdir -p /etc/confd/{conf.d,templates}
conf.d  **# 资源模板，下面文件必须以 toml 后缀**
templates  **# 配置文件模板，下面文件必须以 tmpl 后缀**

**5) 创建 confd 配置文件**

# vi /etc/confd/conf.d/app01.conf.toml

[template]
src = "app01.conf.tmpl"  # 默认在/etc/confd/templates 目录下
dest = "/usr/local/nginx/conf/vhost/app01.conf"  **#要更新的配置文件**
keys = [
  "/Shopping",  **#监测的 key**
]
reload_cmd ="/usr/local/nginx/sbin/nginx -s reload"  **#最后执行的命令**

**6) 创建 confd 模板**

```
# vi    /etc/confd/templates/app01.conf.tmpl
upstream {{getv "/Shopping/nginx/cluster1/proxy_name"}} {
      {{range getvs "/Shopping/nginx/cluster1/upstream/*"}}
            server {{.}};
      {{end}}

   check interval=5000 rise=1 fall=5 timeout=4000 type=http;
   check_http_send "HEAD / HTTP/1.0\r\n\r\n";
   check_http_expect_alive http_2xx http_3xx;

}

server {
    server_name      {{range getvs "/Shopping/nginx/cluster1/server_name/*"}} {{.}}
{{end}};
    location / {
          proxy_pass              http://{{getv    "/Shopping/nginx/cluster1/proxy_name"}};
          proxy_redirect off;
          proxy_set_header Host $host;
          proxy_set_header X-Real-IP $remote_addr;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
     }
       location /status {
                    check_status;
                    access_log      off;
               }
}
```

**7) 启动 confd 并设置开机启动**

开机启动脚本会随文档附带

拷贝至/etc/init.d/confd ,只需要更改 etcd 的连接地址即可

```
. /etc/init.d/functions
confd_start() {
    nohup   /usr/bin/confd   -watch -backend etcd -node http://192.168.0.221:2379   >/var/log/confd.log   2>&1  &
}

confd_stop() {
          /usr/bin/pgrep  confd |xargs  kill  -9   >/dev/null    2>&1

}
```

#/etc/init.d/confd  start  && chkconfig  --add  confd  && chkconfig  confd on

```
[root@mail ~]# ps -aux |grep confd
root       7919  0.2  0.5  25188 21120 ?        Sl   13:52   0:00 /usr/bin/confd -watch -backend etcd -node http://192.168.0.221:2379
root       7943  0.0  0.0 112664   968 pts/4    S+   13:53   0:00 grep --color=auto confd
```

# 四．配置平台部署

**1) Github 克隆平台代码安装平台依赖**

```
# git    clone    https://github.com/1032231418/Conf_Web.git
# cd Conf_Web/ospweb/
#virtualenv      env                  #建议创建一个沙盒环境跑该平台
# source    env/bin/activate         #使用沙盒环境
# pip install -r requirement.txt     #安装相关软件
```

**2) 创建数据库并将表刷入数据库**

```
# vi opsweb/settings.py      #这里数据库信息改为自己的数据库信息
                DATABASES = {
                        'default': {
                            'ENGINE': 'django.db.backends.mysql',
                            'NAME': 'confd',
                            'HOST': '192.168.8.114',
                            'USER': 'root',
                            'PASSWORD': '123456',
                            'PORT': 3306,
                        }
                }
# python manage.py      migrate                #提交迁移文件至数据库,将表刷入数据库
```

```
(env) [root@mail ospweb]# python manage.py   migrate
Operations to perform:
  Apply all migrations: admin, auth, confd, contenttypes, dashboard, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying dashboard.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying confd.0001_initial... OK
  Applying confd.0002_auto_20180611_1832... OK
  Applying confd.0003_auto_20180612_1148... OK
  Applying confd.0004_auto_20180612_1155... OK
  Applying confd.0005_auto_20180612_1535... OK
  Applying confd.0006_auto_20180612_1552... OK
  Applying confd.0007_vhosts_confd_vhosts_status... OK
  Applying sessions.0001_initial... OK
```

3) 创建超级管理员账号

# python manage.py　　createsuperuser


4) 运行平台

**# python manage.py　runserver 0:8000**

访问地址就是 http://ip/8000　　账号密码就是上一步创建的超级管理员账号密码

5) 登录平台为 nginx 创建 key/value

例子:　Shopping 平台为例

项目创建:

1. 创建商城项目　/Shopping
2. 创建商城项目里面的 /Shopping/nginx　nginx 服务
3. 创建 nginx 集群目录　/Shopping/nginx/cluster1
4. 给我们的商城 nginx 集群 1 项目创建配置文件

域名 和 节点名称可能是多个，这里我们需要创建目录 /Shopping/nginx/cluster1/server_name 和 /Shopping/nginx/cluster1/upstream

| 项目名称 | 项目路径 | 操作 |
| --- | --- | --- |
| 商城 | /Shopping | 删除 |
| 商城_nginx | /Shopping/nginx | 删除 |
| 商城_nginx_集群1 | /Shopping/nginx/cluster1 | 删除 |
| 商城_nginx_集群1_upstream | /Shopping/nginx/cluster1/upstream | 删除 |
| 商城_nginx_集群1_server_name | /Shopping/nginx/cluster1/server_name | 删除 |

5. 给集群创建节点，和域名

配置创建:

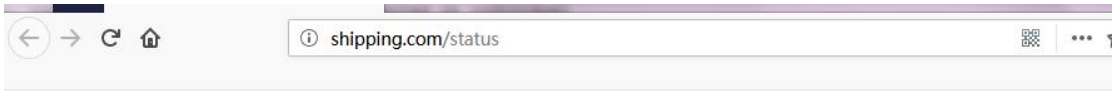
1. 反向代理　　/Shopping/nginx/cluster1/proxy_name
2. 绑定一个域名　　/Shopping/nginx/cluster1/server_name/1
3. 创建一个集群节点 /Shopping/nginx/cluster1/upstream/web1

| 项目名称 | 键 | 值 | 操作 |
| --- | --- | --- | --- |
| 商城_nginx_集群1_server_name | /Shopping/nginx/cluster1/server_name/1 | shipping.com | 编辑 关闭 删除 |
| 商城_nginx_集群1 | /Shopping/nginx/cluster1/proxy_name | shipping | 编辑 关闭 删除 |
| 商城_nginx_集群1_upstream | /Shopping/nginx/cluster1/upstream/web1 | 192.168.0.214:80 | 编辑 关闭 删除 |

```
upstream shipping {

        server 192.168.0.214:80;


  check interval=5000 rise=1 fall=5 timeout=4000 type=http;
  check_http_send "HEAD / HTTP/1.0\r\n\r\n";
  check_http_expect_alive http_2xx http_3xx;

}

server {
   server_name    shipping.com ;
   location / {
       proxy_pass        http://shipping;
       proxy_redirect off;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
       proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
      location /status {
                check_status;
                access_log   off;
            }
}
```

生成的配置文件

通过 hosts 文件我们可以查看节点状态




# Nginx http upstream check status

## Check upstream server number: 1, generation: 95

| Index | Upstream | Name | Status | Rise counts | Fall counts | Check type | Check port |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | shipping | 192.168.0.214:80 | down | 0 | 28 | http | 0 |