# SFO部署手册

# 一、环境准备

1.一个完整的openstack-swift集群，可以是all in one的集群（即PACO各个角色都安装在一台物理主机上）；

2.集群节点需要配置recon功能，该功能在swift的配置文件中启用；

3.集群节点需要配置StatsD服务器并在StatsD服务上将数据发送到Kafka集群，StatsD服务配置方法在openstack官方文档中有介绍，将StatsD数据发送到Kafka集群的方法可以参考我们的手册；

4.一个Zookeeper+Kafka集群，创建两个主题，主题名写入agent配置文件中，如果公司有相关集群，只需要保证网络联通的情况下直接创建主题，如果存储部署在私网中，需要自行搭建，kafka集群主要用于集群节点与管理中心数据通信，如果不想使用，可修改agent的数据传输模块和控制中心的数据消费模块，这两个模块与前端界面展示效果独立。

5.一个Mysql数据库，需要创建keystone和sfo两个数据库，keystone用于集群认证，可用tempauth替换（但如果需要用到本软件的集群管理功能，请使用keystone，否则部分功能将不可用）；

6.如果需要集群平均响应时间功能,还需要配置ELK集群,将proxy日志导入ELK集群,创建索引规则为：集群名-swift-proxy-时间（时间格式为YYYY.MM.DD），集群名写入conf配置文件中，多个集群用“，”隔开；

7.集群管理功能会依赖yum 源，如果需要一键部署集群、安装服务等，请先配置好yum源，配置时请选择合适的openstack-swift版本。

# 二、Agent部署

登陆到需要监控的openstack-swift集群节点，下载sfo源码后解压到/app/sfo-agent，删除目录下的sfo\_server和sfo\_web目录(agent不需要用到server和web相关代码)：

|  |
| --- |
| # su - root  # mkdir /app  # cd /app  # wget sfo地址 （或者通过其他方式上传sfo源码包，我的为sfo.zip）  # unzip -d /app/sfo-agent/ sfo.zip  # mkdir –p /var/log/sfo/  # cd /app/sfo-agent  # rm –rf sfo\_server sfo\_web  # cd scripts/  # yum install -y dos2unix  # dos2unix \*  # chmod +x \*  # cd ..  # tar -zxvf venv.tar.gz  # vim sfo.conf  修改如下项：  [cluster]  swift\_cluster\_name = default  proxy\_port = 8080 (修改为你配置的集群端口)  account\_port = 6012 (修改为你配置的集群端口)  container\_port = 6011 (修改为你配置的集群端口)  object\_port = 6010 (修改为你配置的集群端口)  host\_ip\_prefix = 127.0.0 （这个根据集群配置bind ip修改）  node\_ip\_prefix = 192.168 （这个根据集群配置复制ip修改）  statsD\_ip = 192.168 （配置集群的stastd服务器ip前缀）  process\_workers = 1  thread\_workers = 5  [kafka]  kafka\_sys\_topic = sfo-cluster-info （配置kafka主题名称）  kafka\_statd\_topic = sfo-statsd-info （配置kafka主题名称）  kafka\_sys\_group = test-consumer-group  zookeeper\_server = 192.168.2.4:2181（根据zookeeper集群修改，多个用“，”逗号隔开）  [log]  conf = /logging.conf  [refreshs]  host\_refresh = 300 （数据采集间隔，单位均为秒）  node\_refresh = 15  mem\_refresh = 15  cpu\_refresh = 15  disk\_refresh = 15  net\_refresh = 120  rep\_refresh = 1800  heart\_refresh = 10  mon\_refresh = 15  [agents]  sys\_agent\_pfile = /var/run/sfo/sysagent.pid  swift\_agent\_pfile = /var/run/sfo/swiftagent.pid  data\_agent\_pfile = /var/run/sfo/datagent.pid  server\_agnet\_pfile = /var/run/sfo/serveragent.pid  heart\_agnet\_pfile = /var/run/sfo/beatheart.pid  elklog\_agnet\_pfile = /var/run/sfo/elklogagent.pid  # yum install librdkafka-devel –y  # /app/sfo-agent/venv/bin/python /app/sfo-agent/sfo\_agent/sfo\_agents.py all start |

# 三、Datahandler部署

登陆事先准备好的中心处理机，要求能与数据库和zookeeper服务器网络互通。

|  |
| --- |
| # su - root  # mkdir –p /app  # mkdir –p /var/log/sfo  # cd /app  # wget sfo地址 （或者通过其他方式上传sfo源码包，我的为sfo.zip）  # unzip -d sfo-agent sfo.zip  # cd sfo-agent  # rm –rf sfo\_server sfo\_web sfo\_agent  # yum install librdkafka-devel –y  # vim sfo.conf  [cluster]  swift\_cluster\_name = default  ip = 0.0.0.0  port = 41717  proxy\_port = 8080  account\_port = 6202 (修改为你配置的集群端口)  container\_port = 6201 (修改为你配置的集群端口)  object\_port = 6200 (修改为你配置的集群端口)  host\_ip\_prefix = 127.0.0 （这个根据集群配置bind ip修改）  node\_ip\_prefix = 192.168 （这个根据集群配置复制ip修改）  statsD\_ip = 192.168  temp\_file = /tmp/sfo/  sfo\_server\_temp\_file = /app/sfo/sfo\_temp/  elk\_index\_name=oss-1,oss-2 （根据实际情况配置集群名，多个用“，”号隔开）  elk\_server\_url=http://elk.xxx.com:9200/ （elasticsearch查询地址）  process\_workers = 1  thread\_workers = 5  [swift]  swift\_auth\_url = http://192.168.2.3:8080/auth/v1.0/  swift\_auth\_version = 3  storage\_url = http://192.168.1.2:8080/v1/  swauth\_url = http://127.0.0.1:8080/auth/v2  swift\_user = admin  swift\_password = password  [kafka]  kafka\_sys\_topic = sfo-cluster-info （kafka主题名，切记与agent保持一致）  kafka\_statd\_topic = sfo-statsd-info  kafka\_sys\_group = test-consumer-group  zookeeper\_server = 192.168.2.4:2181 （根据zookeeper集群修改，多个用“，”逗号隔开）  [mysqldb]  mysql\_host = 192.168.2.4  mysql\_port = 3306  mysql\_dbname = sfo  mysql\_user = sfo  mysql\_passwd = Sfo@123456  [log]  conf = /logging.conf  [refreshs]  host\_refresh = 300 （数据采集时间间隔，单位均为秒）  node\_refresh = 15  mem\_refresh = 15  cpu\_refresh = 15  disk\_refresh = 15  net\_refresh = 120  rep\_refresh = 1800  heart\_refresh = 10  mon\_refresh = 15  [agents]  sys\_agent\_pfile = /var/run/sfo/sysagent.pid  swift\_agent\_pfile = /var/run/sfo/swiftagent.pid  data\_agent\_pfile = /var/run/sfo/datagent.pid  server\_agnet\_pfile = /var/run/sfo/serveragent.pid  heart\_agnet\_pfile = /var/run/sfo/beatheart.pid  elklog\_agnet\_pfile = /var/run/sfo/elklogagent.pid  # tar -zxvf venv.tar.gz  # /app/sfo-agent/venv/bin/python /app/sfo-agent/sfo\_datahandler/sfo\_data2db.py start |

# 四、SFO部署

## 1. SFO数据库初始化

1) 连接mysql数据库，创建database: sfo

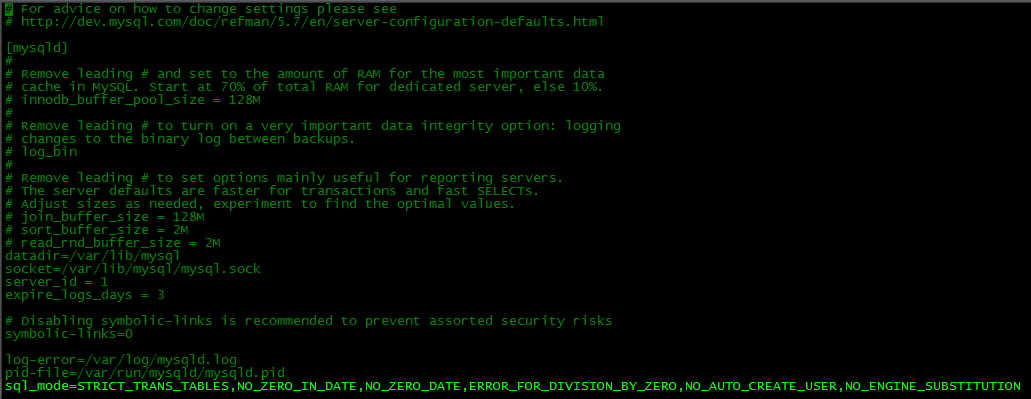
2) 先执行db/ sfo-table.sql脚本

3) 再执行db/ sfo-sfo\_configure\_info.sql脚本

4) 再执行db/ sfo-sfo\_cluster\_configure\_info.sql脚本

注：Mysql5.7版本不支持group by语句,需要在/etc/my.cnf内修改配置,添加sql\_mode=STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION

如图所示：



## 2. SFO web server端部署

SFO web server通过uwsgi对外提供服务，首先我们需要安装uwsgi软件

1. 由于通过yum安装的uwsgi存在功能不全（例如不支持websocket）的风险，我们采取手动安装uwsgi软件：
2. 先决条件

* 安装了python
* 安装了python-devel
* 安装了pip
* 其它依赖库（这些库主要是为了启动websocket），<c>中编译（make）完后应该包含以下字段：

|  |
| --- |
| pcre = True  ifaddrs = True  ssl = True  zlib = True  routing = True  ucontext = True  xml = libxml2 |

这些依赖库可以直接yum安装

|  |
| --- |
| yum install pcre-devel.x86\_64 -y  yum install openssl-devel -y  yum install zlib-devel.x86\_64 -y  yum install libxml2-devel -y |

1. 获取uwsgi最新的源码包： uwsgi-2.0.17.1.tar.gz(uwsgi-lts.tar.gz)
2. 安装uwsgi

|  |
| --- |
| tar -zxf uwsgi-2.0.17.1.tar.gz  cd uwsgi-2.0.17.1  make #先make一下是为了看一下编译uwsgi是否包含了websocket所需的一些依赖库  python setup.py install #如果依赖包没问题，就可以安装了 |

1. 创建 /usr/lib/systemd/system/uwsgi.service 文件，内容为：

[Unit]

Description=uWSGI Emperor Service

After=syslog.target

[Service]

EnvironmentFile=-/etc/sysconfig/uwsgi

ExecStartPre=/bin/mkdir -p /run/uwsgi

ExecStartPre=/bin/chown uwsgi:uwsgi /run/uwsgi

ExecStart=/usr/bin/uwsgi --ini /etc/uwsgi.ini

ExecReload=/bin/kill -HUP $MAINPID

KillSignal=SIGINT

Restart=always

Type=notify

StandardError=syslog

NotifyAccess=all

[Install]

WantedBy=multi-user.target

1. 新建 /etc/uwsgi.d/ 文件夹， /etc/uwsgi.ini 文件

[uwsgi]

uid = root

gid = root

socket = /var/run/uwsgi/uwsgi.socket

pidfile = /var/run/uwsgi/uwsgi.pid

emperor = /etc/uwsgi.d

#emperor-tyrant = true

master = true

autoload = true

log-date = true

logto = /var/log/uwsgi.log

1. 配置sfo服务
2. /etc/uwsgi.d/sfo.ini

[uwsgi]

#sfo工作模块，根据实际修改

chdir = /app/sfo/

wsgi-file = SFO.py

uwsgi-socket = 0.0.0.0:8001

buffer-size = 65535

#sfo虚环境，根据实际修改

virtualenv = /root/venv

master = false

enable-threads = true

processes = 4

threads = 2

thunder-lock = true

plugins = python

callable = app

1. /etc/uwsgi.ini/sfo-socket.ini

[uwsgi]

gevent=1000

gevent-monkey-patch=true

http=:5555

http=:5556

http=:5557

http=:5558

http-websockets=true

master=true

#sfo工作模块，根据实际修改

chdir=/app/sfo/

#sfo虚环境，根据实际修改

virtualenv=/root/venv

buffer-size=65535

thunder-lock=true

vacuum=true

wsgi-file=SFO\_SOCKET.py

callable=app

1. 解压运行的虚环境，启动uwsgi服务

|  |
| --- |
| tar -zxvf /app/sfo/venv.tar.gz -C /app/sfo/  systemctl start uwsgi.service |

## 3. SFO web前端打包

sfo web前端需要依赖nodejs以及npm，nodejs版本为v8.11.3

1) 进入sfo\_web目录，安装相关依赖包；

|  |
| --- |
| npm install  #注：如果速度过慢，可通过以下操作解决，但不建议使用cnpm，会有bug  npm install --registry=https://registry.npm.taobao.org  #如果安装sass失败，可通过以下操作安装：  npm i node-sass --sass\_binary\_site=https://npm.taobao.org/mirrors/node-sass/ |

2) 修改sfo\_web/config/prod.env.js,

假设前端的域名是<http://www.example.com>

修改BASE\_API例：[http:// www.example.com](http://sfo-dev.sf-express.com)

修改SOCKET\_BASE\_URL例：[http:// www.example.com /socket.io](http://sfo-dev.sf-express.com/socket.io)



3) 打包

|  |
| --- |
| npm run build |

1. 打包成功后，会在sfo\_web目录下生成一个dist文件夹

## 4. 配置Nginx

说明：nginx的版本至少为1.4以上， 以支持websocket

1. 修改/etc/nginx/nginx.conf文件，增加upstream sfo-admin以及upstream socketio\_nodes部分，重启Nginx服务:

|  |
| --- |
| upstream sfo-admin {  server 127.0.0.1:8001 max\_fails=3 fail\_timeout=30s;  }  upstream socketio\_nodes {  server 127.0.0.1:5555;  server 127.0.0.1:5556;  server 127.0.0.1:5557;  server 127.0.0.1:5558;  } |

1. 在conf.d目录下创建sfo.conf文件，内容如下：

|  |
| --- |
| map $http\_upgrade $connection\_upgrade {  default upgrade;  '' close;  }  server {  listen \*:80;  server\_name \*. example.com;  access\_log /var/log/nginx/sfo.access.log main;  error\_log /var/log/nginx/sfo.error.log;  # 路径根据实际情况配置，为sfo\_web/dist的实际绝对路径  root /app/sfo/sfo\_web/dist; #build打包成功后的dis文件路径  index index.html;  add\_header Access-Control-Allow-Origin http:// www.example.com;  #http://sfo.stg.sf-express.com与sfo\_web/config/prod.env.js内BASE\_API一致  add\_header Access-Control-Allow-Headers Authorization, Content-Type, Accept, Origin, User-Agent, DNT, Cache-Control, X-Mx-ReqToken, X-Data-Type, X-Auth-Token, X-Requested-With;  add\_header Access-Control-Allow-Methods GET, POST, OPTIONS, HEAD, PUT, DELETE;  add\_header Access-Control-Allow-Credentials true;  location / {  # 路径根据实际情况配置，为sfo\_web/dist的实际绝对路径  root /app/sfo/sfo\_web/dist; #build打包成功后的dis文件路径  try\_files $uri $uri/ /index.html last;  index index.html;  }  location ^~ /api {  uwsgi\_pass sfo-admin;  uwsgi\_send\_timeout 600;  uwsgi\_connect\_timeout 600;  uwsgi\_read\_timeout 600;  include uwsgi\_params;  # 路径根据实际情况配置  uwsgi\_param UWSGI\_PYHOME /app/sfo/venv;  uwsgi\_param UWSGI\_CHDIR /app/sfo;  uwsgi\_param UWSGI\_SCRIPT SFO:app;  uwsgi\_param SCRIPT\_NAME "";  }  location /socket.io {  proxy\_pass http://socketio\_nodes/socket.io;  proxy\_http\_version 1.1;  proxy\_buffering off;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection "Upgrade";  }  }  server {  listen 80;  server\_name \_;  access\_log /var/log/nginx/nginx.log main;  error\_log /var/log/nginx/nginx.err;  return 400;  } |

1. 创建日志目录, 重启nginx服务

|  |
| --- |
| mkdir -p /var/log/sfo/ |