# Spinnaker高可用安装构思

由于官方没有提供高可用的方案。官方直接建议由K8S进行调度，但是由于网络原因往往无法安装成功。生产环境如果是单点方案，死掉一个以后，恢复的时间由于网络原因时间会非常长。着急上线或者部署的话会出现很大的问题。

Spinnaker所有的配置文件由minio进行存储。Minio高可用的话，可以解决存储的问题。但是还是不能解决spinnaker挂掉一个然后不影响使用的问题

基本构思是，先安装一个minio集群，然后将另外一套spinnaker直接连接相同的k8s，相同的minio，然后达到高可用的方案。

国内安装spinnaker需要解决翻墙的问题、存储的问题、服务恢复的问题、使用IP登录的问题、等等。

# 准备环境

直接安装在主机上，spinnaker只支持ubuntu。选择了较为新的ubuntu18.04.

Ubuntu翻墙方案。采用shadowsocks+ privoxy的方案来分别使用http+https的代理方式

下载文件

wget https://bootstrap.pypa.io/get-pip.py --no-check-certificate

执行安装

sudo python get-pip.py

sudo pip install <https://github.com/shadowsocks/shadowsocks/archive/master.zip>

配置shadowsocks.json

sudo vi /etc/shadowsocks.json

{

"server":"这里翻墙的服务器",

"server\_port":10367,

"local\_address":"127.0.0.1",

"local\_port":1080,

"password":"这里填翻墙的密码.",

"timeout":300,

"method":"rc4-md5",

"fast\_open":false,

"workers": 1

}

这个是自己的服务器。后期估计随时有可能断线，需要搭建ssl服务器

运行 sslocal

sudo nohup sslocal -c /etc/shadowsocks.json &>> /var/log/sslocal.log &

安装完了shadowsocks以后 需要使用代理链接

apt-get install privoxy -y

echo "forward-socks5 / 127.0.0.1:1080 ." >>/etc/privoxy/config

systemctl restart privoxy

配置wget代理下载

vi /etc/wgetrc

添加

https\_proxy = http://127.0.0.1:8118/

http\_proxy = http://127.0.0.1:8118/

ftp\_proxy = http://127.0.0.1:8118/

设置http/https代理

export http\_proxy=http://127.0.0.1:8118 # privoxy默认监听端口为8118

export https\_proxy=<http://127.0.0.1:8118>

测试是否成功

curl [www.google.com](http://www.google.com)

# 安装hal

curl -O <https://raw.githubusercontent.com/spinnaker/halyard/master/install/debian/InstallHalyard.sh>

下载完文件后，脚本默认使用curl方式下载文件，由于网络原因经常无法下载或断开连接，修改脚本中的curl变更成wget的方式

Curl -O <https://storage.googleapis.com/$gcs_bucket_and_file>

变成

Wget <https://storage.googleapis.com/$gcs_bucket_and_file>

sudo apt-get install software-properties-common -y

sudo apt-get install openjdk-8-jdk -y

sudo bash [InstallHalyard.sh](https://raw.githubusercontent.com/spinnaker/halyard/master/install/debian/InstallHalyard.sh)

安装完成，不能安装的话 多重复几遍。

安装完成以后设置翻墙

sudo vi /opt/halyard/bin/halyard

DEFAULT\_JVM\_OPTS='"-Djava.security.egd=file:/dev/./urandom" "-Dspring.config.location=/opt/spinnaker/config/" "-DsocksProxyHost=127.0.0.1 -DsocksProxyPort=8118"'

重启hal服务

hal shutdown

hal

# 三、安装minio集群

以上安装了hal官方建议的管理程序。接下来安装minio集群。这边有四台服务器，来组建一个minio的最小集群方案，官方建议最小的是4个minio节点，minio的端口跟spinnaker的端口冲突了。这里修改为9001。

下载minio的二进制包

curl -O https://dl.minio.io/server/minio/release/linux-amd64/minio

sudo chmod +x minio

sudo mv minio /usr/local/bin

安装完成以后，配置systemd管理方式

vim /lib/systemd/system/minio.service

[Unit]

Description=Minio service

Documentation=https://docs.minio.io/

[Service]

WorkingDirectory=/opt/minio/

ExecStart=/opt/minio/run.sh

Restart=on-failure

RestartSec=5

[Install]

WantedBy=multi-user.target

chmod +x /lib/systemd/system/minio.service

编辑minio的启动脚本

vim /opt/minio/run.sh

#!/bin/bash

export MINIO\_ACCESS\_KEY=minio

export MINIO\_SECRET\_KEY=minio123

minio server --address 172.16.2.109:9001 --config-dir /etc/minio \

http://172.16.2.81:9001/data/minio/ http://172.16.2.109:9001/data/minio/ \

http://172.16.2.82:9001/data/minio/ <http://172.16.2.112:9001/data/minio/>

chmod +x /opt/minio/run.sh

这个脚本注意每个机器上面的IP地址不对，需要根据实际IP进行变动，每台机器上面放一个 。

启动minio

systemctl start minio

将minio添加到开机启动

systemctl enable minio

# 四、开始安装spinnaker

设置存储minio

export MINIO\_SECRET\_KEY=“minio123“

export ENDPOINT=“http://172.16.2.109:9001“

export MINIO\_ACCESS\_KEY=“minio“

echo $MINIO\_SECRET\_KEY | hal config storage s3 edit --endpoint $ENDPOINT \

--access-key-id $MINIO\_ACCESS\_KEY \

--secret-access-key # will be read on STDIN to avoid polluting your

# ~/.bash\_history with a secret

mkdir ~/.hal/default/profiles

echo "spinnaker.s3.versioning: false" >> ~/.hal/default/profiles/front50-local.yml

hal config storage edit --type s3

查看版本

hal version list

选择版本

hal config version edit --version 1.9.3

安装版本

sudo hal deploy apply

顺利的话，多试几遍是能成功的，不顺利的话，就试更多遍

# 五、配置与K8S的连接

将kubeconfig文件放到主机的/home/ubuntu/.kube/config这个位置，然后

hal config provider kubernetes enable

hal config provider kubernetes account add my-k8s-v2-account \

--provider-version v2 \

--context $(kubectl config current-context)

hal config features edit --artifacts true

sudo hal deploy apply

接着查看clouddriver是否启动正常。

# 六、安装另外一台高可用的节点

这台的基本构思就是使用主服务器的配置在副节点上回放。然后将文件都改成本地的配置，让他们基于minio进行数据的同步。这样就可以完成高可用的方案。

首先需要准备几个文件哈

1. 主机器的/home/ubuntu/.hal/config 复制到另外一台机器上相同的目录。
2. 在另外一台机器上面复现翻墙的工具，安装hal与上面方法相同。副机器不需要配置与K8S的连接，完全由主机器配置。
3. 完成以上步骤的话，sudo hal deploy apply 等待完成以后，查看是否OK

## 七、配置IP访问spinnaker

由于官方只提供了基于域名的访问方式，公司的环境所有人配上DNS，难度略大。所以使用IP加端口的方式访问。需要修改以下文件。

修改apache2-deck的配置文件，配置转发IP

vim /opt/deck/html/settings.js

var gateHost = 'http://172.16.2.112:8084';

修改echo的服务与deck的对接地址

vi /opt/spinnaker/config/echo.yml

spinnaker.baseUrl: <http://172.16.2.112:9000>

修改spinnaker的配置文件，分别修改gate与deck

修改并添加一下内容，其中

deck:

overrideBaseUrl: http://172.16.2.112:9000

baseUrl: <http://172.16.2.112:9000>

gate:

overrideBaseUrl: http://172.16.2.112:8084

baseUrl: <http://172.16.2.112:8084>

修改apache服务启动绑定ip

vim /etc/apache2/ports.conf

vim /etc/apache2/sites-enabled/spinnaker.conf

修改上面两个文件中的host 从localhost to 0.0.0.0

内网用户可以访问

重启apache

systemctl restart apache2

修改spinnaker 的gate服务的host

修改 /opt/spinnaker/config/gate.yml 文件中的 address 为 0.0.0.0

重启gate服务

systemctl restart gate

如果有可能的话 spinnaker整个重启一下。

systemctl restart spinnaker

访问<http://172.16.2.109:9000> 与<http://172.16.2.112:9000>

然后随便建立一个application看是否能与另外一个同步，如果能同步就代表高可用已经完成。