|  |  |
| --- | --- |
| 版本 | 1.1.0 |

# 瀚高LOGO修改-01--20230823

**瀚高高可用集群HGHAC V4**

**用户安装手册**

2023-12

瀚高基础软件股份有限公司

服务热线：400-708-8006

[www.highgo.com](http://www.highgo.com/)

**版本历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修改描述** | **修改日期** | **修改人** | **备注** |
| 1.0 | 手册定稿 | 2023.12 | 杜中国 |  |

**目录**

[1. 前言 5](#_Toc847)

[1.1文档说明 5](#_Toc6619)

[1.2云环境 5](#_Toc9516)

[2. 软件版本 5](#_Toc17489)

[3. 安装前准备 5](#_Toc22986)

[3.1所有节点设置时钟同步 5](#_Toc24084)

[3.2 关闭防火墙 6](#_Toc27332)

[4. 安装部署 6](#_Toc251)

[4.1安装部署数据库 6](#_Toc11953)

[4.2配置启动etcd 6](#_Toc9457)

[4.3配置启动 hghac 8](#_Toc4791)

[5. 安装部署实例 8](#_Toc11209)

[5.1部署规划 8](#_Toc32153)

[5.2 安装前准备 9](#_Toc9350)

[5.2.1所有节点设置时钟同步 9](#_Toc16470)

[5.2.2 关闭防火墙 9](#_Toc18960)

[5.3手动安装部署 10](#_Toc10385)

[5.3.1安装部署数据库 10](#_Toc8306)

[5.3.2 安装部署高可用组件 10](#_Toc11092)

[5.3.2.1 安装DCS/hghac/vip-manager 10](#_Toc27916)

[5.3.2.2 配置启动DCS 10](#_Toc4858)

[5.3.2.2.1 配置启动etcd 10](#_Toc26980)

[5.3.2.2.2 配置启动zookeeper 15](#_Toc12598)

[5.3.2.2.3配置启动consul 16](#_Toc19839)

[5.3.2.3 配置启动 hghac 19](#_Toc24217)

[5.3.2.3.1 配置hghac.yml文件 19](#_Toc30174)

[5.3.2.3.2 启动hghac 22](#_Toc6917)

[5.3.2.3.3 查看hghac集群状态 22](#_Toc19099)

[5.3.2.4 配置启动vip-manager 22](#_Toc32490)

[5.3.2.4.1 配置vip-manager.yml文件 22](#_Toc30777)

[5.3.2.4.2 启动vip-manager 24](#_Toc27462)

[5.3.2.4.3 查看确认vip 24](#_Toc20514)

[5.3.3 设置高可用组件开机自启动 24](#_Toc5815)

[5.3.3.1 etcd开机自启动 24](#_Toc5610)

[5.3.3.1.1 配置文件修改 24](#_Toc18476)

[5.3.3.1.2 设置etcd开机自启动服务 25](#_Toc26572)

[5.3.3.2 zookeeper开机自启动 25](#_Toc21482)

[5.3.3.2.1 配置文件修改 25](#_Toc9219)

[5.3.3.2.2 设置zookeeper开机自启动服务 25](#_Toc22027)

[5.3.3.3 consul开机自启动 26](#_Toc31209)

[5.3.3.3.1 配置文件修改 26](#_Toc15776)

[5.3.3.3.2 设置consul开机自启动服务 28](#_Toc3325)

[5.3.3.4 hghac开机自启动 28](#_Toc5648)

[5.3.3.4.1 配置文件修改 28](#_Toc327)

[5.3.3.4.2 设置hghac开机自启动服务 29](#_Toc10366)

[5.3.3.5 vip-manager开机自启动 29](#_Toc17332)

[5.3.3.5.1 配置文件修改 29](#_Toc7472)

[5.3.3.5.2 设置vip-manager开机自启动服务 29](#_Toc13910)

[5.3.4 hghac初始化数据库并部署流复制的功能说明 29](#_Toc17736)

# 前言

## 1.1文档说明

本文档主要用作瀚高读写分离集群软件的云上安装指导，主要内容包括安装、卸载、快速启动。

## 1.2云环境

云环境如下:

CPU：x86\_64

操作系统：Centos7.6



※注意：开放数据库连接端口5866（数据库默认端口）

云环境中进行操作，与普通linux环境基本相同。

# 软件版本

本手册涵盖的所有组件版本号信息如下：

* hghac: hghac 4.2.1

# 安装前准备

## 3.1所有节点设置时钟同步

确保各节点主机时钟同步。

以ntpdate为例，所有主机安装ntpdate后，执行如下命令完成时钟同步。

ntpdate time.windows.com && hwclock -w

## 3.2 关闭防火墙

systemctl stop firewalld.service

systemctl disable firewalld.service

# 安装部署

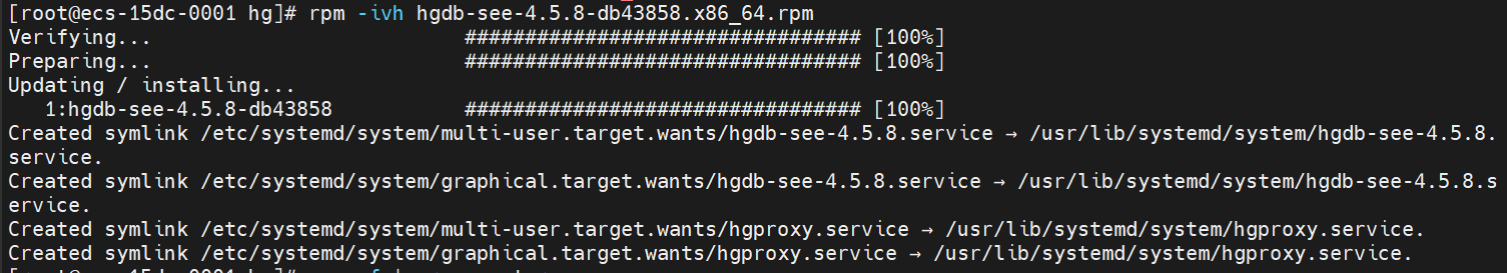
## 4.1安装部署数据库

hghac提供了初始化数据库并部署流复制的功能，用户可选择由hghac来搭建高可用流复制集群，启动hghac之前需安装数据库并设置环境变量。

以安全版数据库4.5.8为例，具体搭建步骤如下：

1. 安装数据库

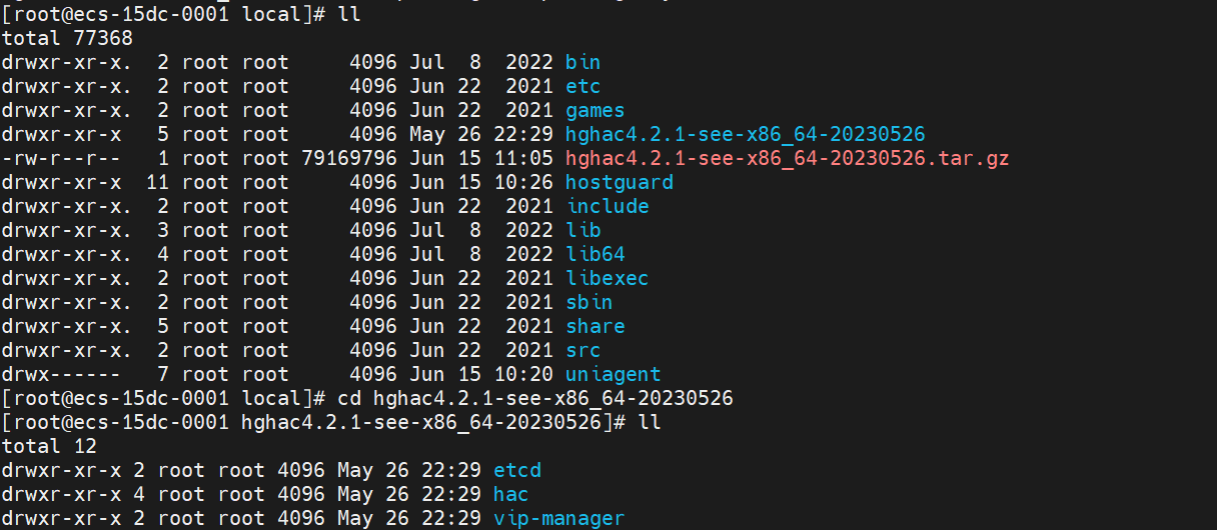
rpm -ivh hgdb-see-4.5.8-db43858.x86\_64.rpm



## 4.2配置启动etcd

1.将hghac.zip压缩包放到/usr/local/目录下解压缩后生成hghac文件夹：

tar -zxvf hghac4.2.1-see-x86\_64-20230526.tar.gz

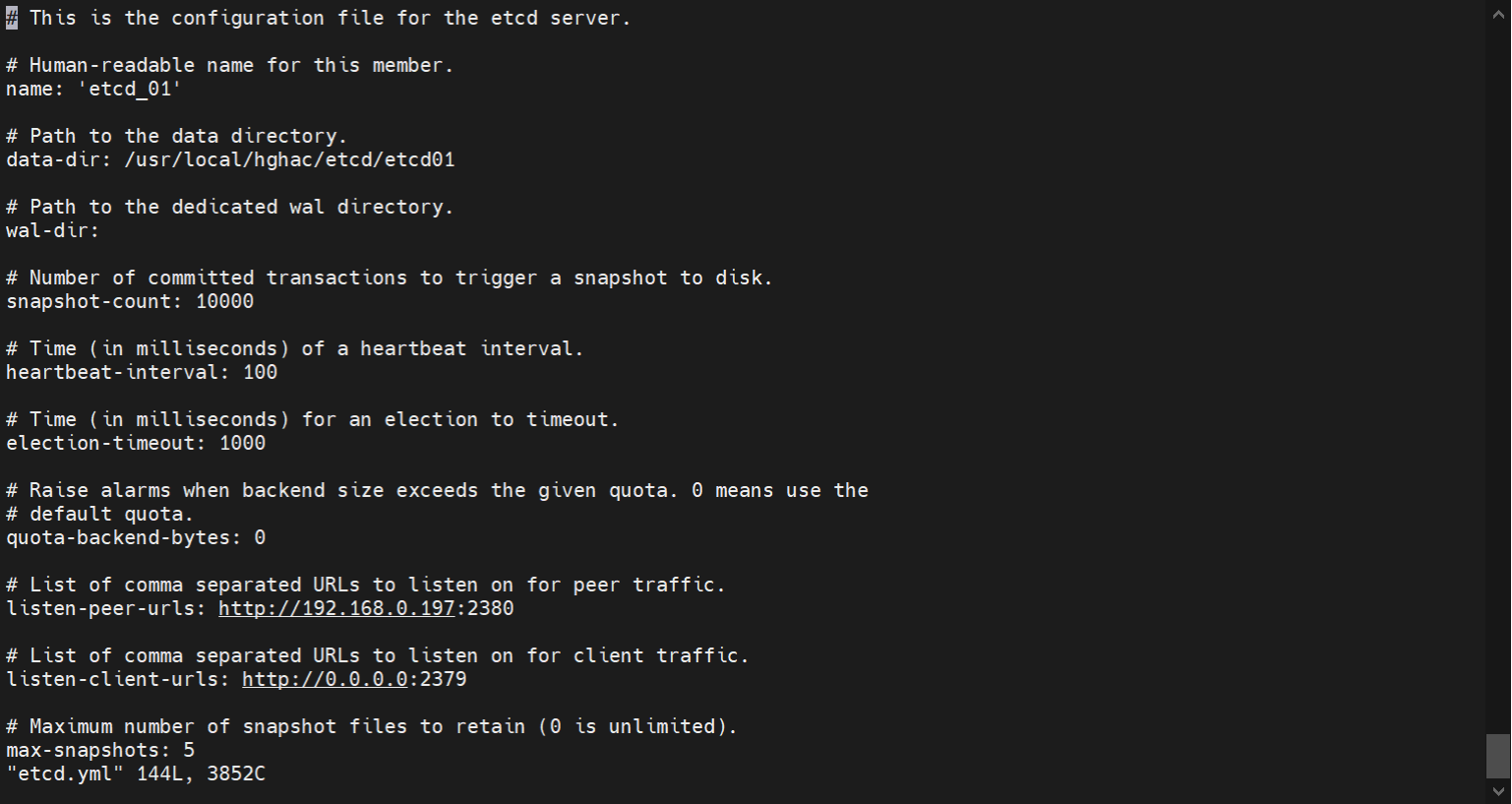


2.配置启动etcd

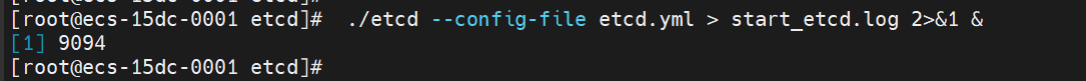
进入etcd文件夹：

cd /usr/local/hghac/etcd

1. ：配置etcd.yml文件



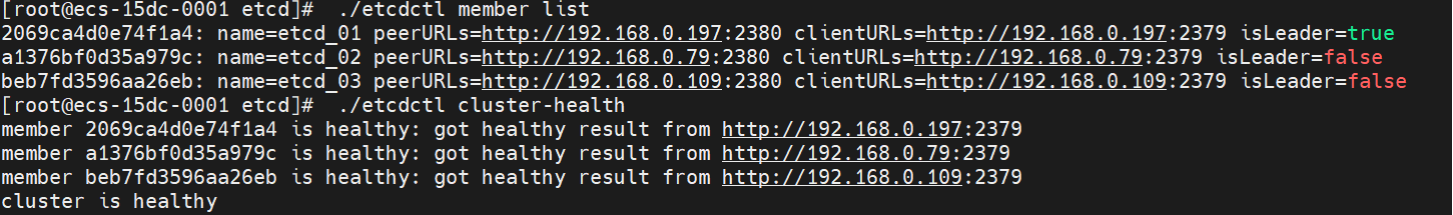
1. ：启动etcd ./etcd --config-file etcd.yml > start\_etcd.log 2>&1 &



（3）：查看etcd集群节点状态信息

./etcdctl member list

./etcdctl cluster-health



## 4.3配置启动 hghac

1.进入hac文件夹：

cd /usr/local/hghac/hac

配置hghac.yml文件

2.启动hghac

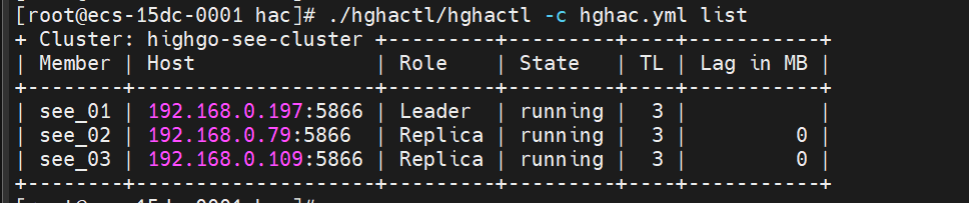
./hghac/hghac hghac.yml > hghac.log1 2>&1 &

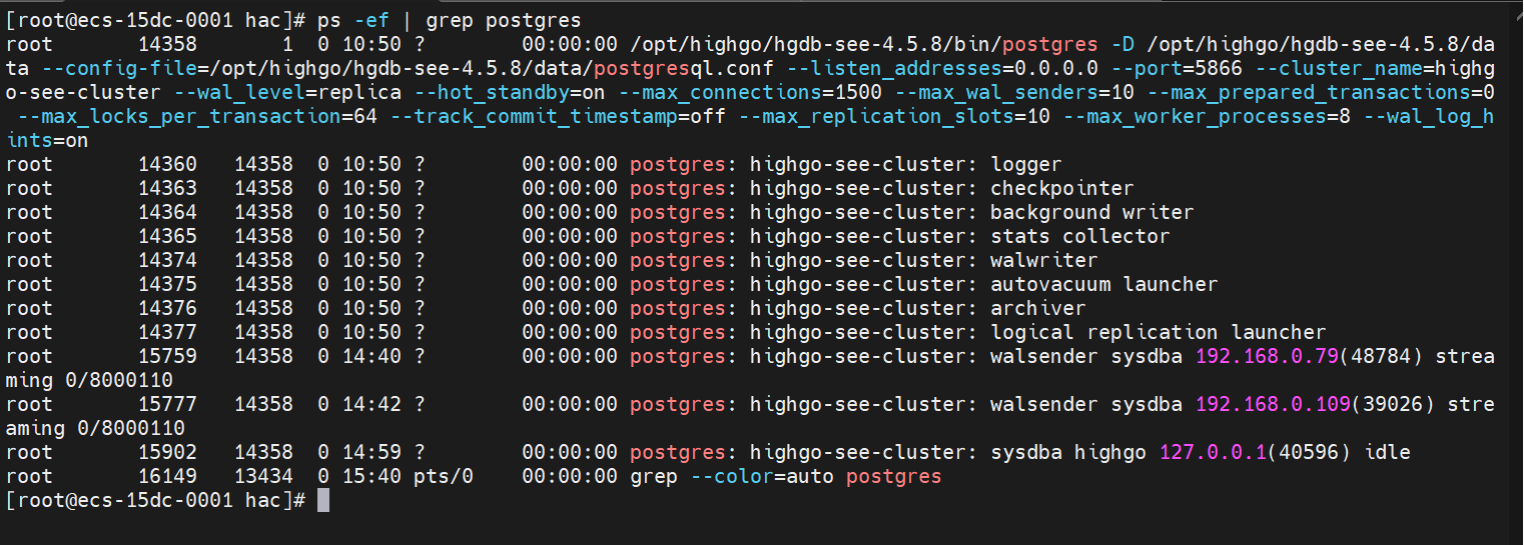
IMG_256

3.查看hghac集群状态

集群中任一节点执行以下命令

./hghactl/hghactl -c hghac.yml list



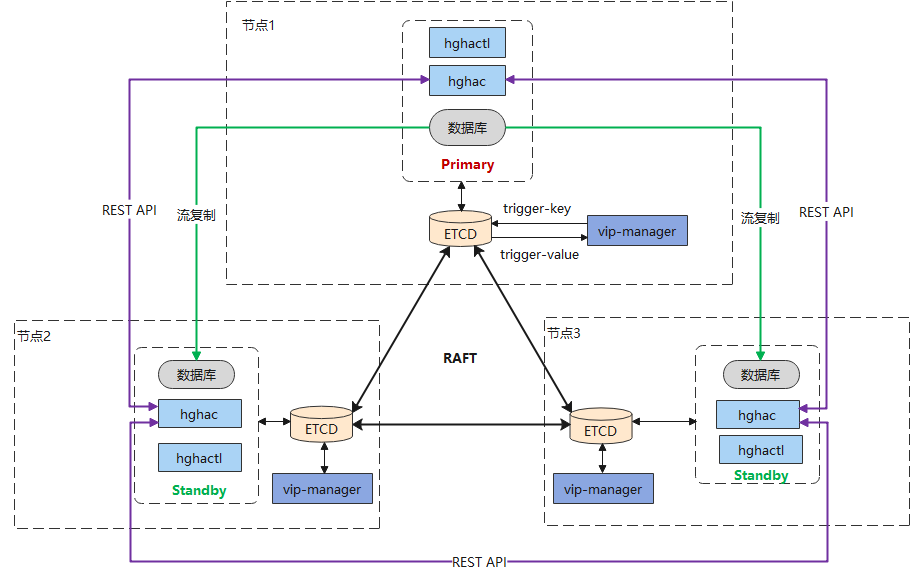


# 安装部署实例

## 5.1部署规划

本实例涵盖的所有组件版本号信息如下：

* hghac: hghac 4.2.1
* DCS: etcd-v3.3.11 / zookeeper-3.8.0
* vip-manager: 1.0.2



| **主机** | **IP** | **组件** | **备注** |
| --- | --- | --- | --- |
| host101 | 192.168.31.101 | Ivorysql、hghac、DCS(etcd/zookeeper/consul)、vip-manager | 主节点 |
| host102 | 192.168.31.102 | Ivorysql、hghac、DCS(etcd/zookeeper/consul)、vip-manager | 备节点1 |
| host103 | 192.168.31.103 | Ivorysql、hghac、DCS(etcd/zookeeper/consul)、vip-manager | 备节点2 |

该部署方案仅作为安装部署的一个例子，实际生产环境建议etcd集群为三个以上且为奇数的节点，不建议部署在同一个机房，尽量部署在不同的机房。

## 5.2 安装前准备

## 5.2.1所有节点设置时钟同步

确保各节点主机时钟同步。

以ntpdate为例，所有主机安装ntpdate后，执行如下命令完成时钟同步。

ntpdate time.windows.com && hwclock -w

## 5.2.2 关闭防火墙

systemctl stop firewalld.service

systemctl disable firewalld.service

## 5.3手动安装部署

## 5.3.1安装部署数据库

数据库的安装部署这里不做过多说明，请参考相应版本的数据库用户安装手册，这里仅特别指出以下几点：

1. hghac提供初始化数据库并部署流复制的功能，但推荐手工搭建流复制，同步异步皆可；
2. 安装部署完毕所有节点须启动数据库。

下文hghac的安装部署即假设流复制已经搭建完毕并正常启动。

**5.3.2 安装部署高可用组件**

**5.3.2.1 安装DCS/hghac/vip-manager**

将hghac.zip压缩包放到/usr/local/目录下解压缩后生成hghac文件夹：

[root@host101 local]# unzip hghac.zip

[root@host101 local]# cd hghac

[root@host101 hghac]# ls

etcd hac jre\_linux jre\_linux.zip vip-manager zookeeper

以下高可用组件的安装部署默认都在此安装目录下操作。

**5.3.2.2 配置启动DCS**

这里提供etcd和zookeeper两种DCS的安装说明，根据需要选择其一即可。

需要注意的是：

目前vip-manager只支持etcd和consul，不支持zookeeper。

**5.3.2.2.1 配置启动etcd**

进入etcd文件夹：

cd /usr/local/hghac/etcd

第一步:配置etcd.yml文件

根据实际情况配置etcd.yml文件，以主节点为例：

# This is the configuration file for the etcd server.

# Human-readable name for this member.

name: 'etcd\_01'

# Path to the data directory.

data-dir: /usr/local/hghac/etcd/etcd01

# Path to the dedicated wal directory.

wal-dir:

# Number of committed transactions to trigger a snapshot to disk.

snapshot-count: 10000

# Time (in milliseconds) of a heartbeat interval.

heartbeat-interval: 100

# Time (in milliseconds) for an election to timeout.

election-timeout: 1000

# Raise alarms when backend size exceeds the given quota. 0 means use the

# default quota.

quota-backend-bytes: 0

# List of comma separated URLs to listen on for peer traffic.

listen-peer-urls: http://192.168.31.101:2380

# List of comma separated URLs to listen on for client traffic.

listen-client-urls: http://0.0.0.0:2379

# Maximum number of snapshot files to retain (0 is unlimited).

max-snapshots: 5

# Maximum number of wal files to retain (0 is unlimited).

max-wals: 5

# Comma-separated white list of origins for CORS (cross-origin resource sharing).

cors:

# List of this member's peer URLs to advertise to the rest of the cluster.

# The URLs needed to be a comma-separated list.

initial-advertise-peer-urls: http://192.168.31.101:2380

# List of this member's client URLs to advertise to the public.

# The URLs needed to be a comma-separated list.

advertise-client-urls: http://192.168.31.101:2379

# Discovery URL used to bootstrap the cluster.

discovery:

# Valid values include 'exit', 'proxy'

discovery-fallback: 'proxy'

# HTTP proxy to use for traffic to discovery service.

discovery-proxy:

# DNS domain used to bootstrap initial cluster.

discovery-srv:

# Initial cluster configuration for bootstrapping.

initial-cluster: etcd\_01=http://192.168.31.101:2380,etcd\_02=http://192.168.31.102:2380,etcd\_03=http://192.168.31.103:2380

# Initial cluster token for the etcd cluster during bootstrap.

initial-cluster-token: 'etcd-cluster-ivorysql'

# Initial cluster state ('new' or 'existing').

initial-cluster-state: 'new'

# Reject reconfiguration requests that would cause quorum loss.

strict-reconfig-check: false

# Accept etcd V2 client requests

enable-v2: true

# Enable runtime profiling data via HTTP server

enable-pprof: true

# Valid values include 'on', 'readonly', 'off'

proxy: 'off'

# Time (in milliseconds) an endpoint will be held in a failed state.

proxy-failure-wait: 5000

# Time (in milliseconds) of the endpoints refresh interval.

proxy-refresh-interval: 30000

# Time (in milliseconds) for a dial to timeout.

proxy-dial-timeout: 1000

# Time (in milliseconds) for a write to timeout.

proxy-write-timeout: 5000

# Time (in milliseconds) for a read to timeout.

proxy-read-timeout: 0

client-transport-security:

# DEPRECATED: Path to the client server TLS CA file.

ca-file:

# Path to the client server TLS cert file.

cert-file:

# Path to the client server TLS key file.

key-file:

# Enable client cert authentication.

client-cert-auth: false

# Path to the client server TLS trusted CA cert file.

trusted-ca-file:

# Client TLS using generated certificates

auto-tls: false

peer-transport-security:

# DEPRECATED: Path to the peer server TLS CA file.

ca-file:

# Path to the peer server TLS cert file.

cert-file:

# Path to the peer server TLS key file.

key-file:

# Enable peer client cert authentication.

peer-client-cert-auth: false

# Path to the peer server TLS trusted CA cert file.

trusted-ca-file:

# Peer TLS using generated certificates.

auto-tls: false

# Enable debug-level logging for etcd.

debug: false

# Specify a particular log level for each etcd package (eg: 'etcdmain=CRITICAL,etcdserver=DEBUG'.

log-package-levels:

# Specify 'stdout' or 'stderr' to skip journald logging even when running under systemd.

log-output: default

# Force to create a new one member cluster.

force-new-cluster: false

第二步：启动etcd

./etcd --config-file etcd.yml > start\_etcd.log 2>&1 &

第三步：查看etcd集群节点状态信息

执行以下命令查看集群节点信息：

[root@host101 etcd]# ./etcdctl member list

3e715d6c6bf37435: name=etcd\_01 peerURLs=http://192.168.31.101:2380 clientURLs=http://192.168.31.101:2379 isLeader=true

a88d215521699e98: name=etcd\_03 peerURLs=http://192.168.31.103:2380 clientURLs=http://192.168.31.103:2379 isLeader=false

f93efbb08dc6751d: name=etcd\_02 peerURLs=http://192.168.31.102:2380 clientURLs=http://192.168.31.102:2379 isLeader=false

[root@host101 etcd]# ./etcdctl cluster-health

member 3e715d6c6bf37435 is healthy: got healthy result from http://192.168.31.101:2379

member a88d215521699e98 is healthy: got healthy result from http://192.168.31.102:2379

member f93efbb08dc6751d is healthy: got healthy result from http://192.168.31.103:2379

cluster is healthy

以上结果说明etcd安装启动成功了。

**5.3.2.2.2 配置启动zookeeper**

第一步：安装jre

解压jre：

unzip jre\_linux.zip

在/etc/profile中添加环境变量路径并source，添加如下：

export JRE\_HOME=/usr/local/hghac/jre\_linux

export CLASSPATH=$JRE\_HOME/lib:$CLASSPATH

export PATH=$JRE\_HOME/bin:$PATH

source /etc/profile

查看java版本，以下说明安装成功：

[root@localhost local]# java -version

openjdk version "1.8.0\_262"

OpenJDK Runtime Environment (build 1.8.0\_262-b10)

OpenJDK 64-Bit Server VM (build 25.262-b10, mixed mode)

版本说明：zookeeper 支持在 Java 1.8 或更高版本中运行（JDK 8 LTS、JDK 11 LTS、JDK 12 - 不支持 Java 9 和 10）

第二步：配置zoo.cfg配置文件

cd /usr/local/hghac/zookeeper/conf

cp zoo\_sample.cfg zoo.cfg

vi zoo.cfg

#修改以下参数：

dataDir=/usr/local/hghac/zookeeper/data

#增加以下内容：

server.0=192.168.31.101:2888:3888

server.1=192.168.31.102:2888:3888

server.2=192.168.31.103:2888:3888

第三步：创建myid文件

创建目录保存myid文件，并且向myid文件中写入内容。

myid中的值和上述zoo.cfg文件中配置的server.\*后面指定的编号是一一对应的。例如，以主节点为例，192.168.31.101这台服务器对应的是server.0,主节点的myid的值就是0：

cd /usr/local/hghac/zookeeper

mkdir data

cd data

echo 0 > myid

第四步：启动zookeeper集群

分别在三个节点上启动zookeeper进程：

cd /usr/local/hghac/zookeeper/bin

[root@host101 bin]# ./zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /usr/local/hghac/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

第五步：查看zookeeper集群节点状态信息

各节点分别执行以下命令查看集群节点状态信息，有一个节点显示为leader，其他两个节点为follower；如果没有启动成功的话则会提示连不上服务not running：

[root@host101 bin]# ./zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /usr/local/hghac/zookeeper/bin/../conf/zoo.cfg

Client port found: 2181. Client address: localhost. Client SSL: false.

Mode: leader

**5.3.2.2.3配置启动consul**

因consul不支持list列表传参与hghac适配时无法实现高可用，DCS选择不建议使用consul。hghac安装包中也不提供consul，如果需要安装consul，请用户单独安装。以下配置、启动步骤仅作为自行安装的参考。

第一步：配置前准备

1、查看版本号确认consul是否安装

[ivorysql@host6 ~]$ /usr/local/hghac/consul/consul -v

Consul v1.13.2

Revision 0e046bbb

Build Date 2022-09-20T20:30:07Z

Protocol 2 spoken by default, understands 2 to 3 (agent will automatically use protocol >2 when speaking to compatible agents)

1. 创建安装启动所需文件夹

mkdir -p /usr/local/hghac/consul/log

mkdir -p /usr/local/hghac/consul/data

mkdir -p /usr/local/hghac/consul/consul.d

第二步：配置启动consul

**1、创建启动脚本**

vi /usr/local/hghac/consul/start.sh

#!/bin/bash

nohup /usr/local/hghac/consul/consul agent -server -bootstrap-expect=3 -data-dir=/usr/local/hghac/consul/data -node=consul-1 -bind=192.168.31.101 -enable-script-checks=true -config-dir=/usr/local/hghac/consul/consul.d -retry-join=192.168.31.101,192.168.31.102,192.168.31.103 -rejoin -ui -log-file=/usr/local/hghac/consul/log/ -client=0.0.0.0 &

**指令详解：**

-advertise：通知展现地址用来改变我们给集群中的其他节点展现的地址，一般情况下-bind地址就是展现地址

-bootstrap：用来控制一个server是否在bootstrap模式，在一个datacenter中只能有一个server处于bootstrap模式，当一个server处于bootstrap模式时，可以自己选举为raft leader。

-bootstrap-expect：在一个datacenter中期望提供的server节点数目，当该值提供的时候，consul一直等到达到指定sever数目的时候才会引导整个集群，该标记不能和bootstrap公用。

-bind：该地址用来在集群内部的通讯，集群内的所有节点到地址都必须是可达的，默认是0.0.0.0。

-client：consul绑定在哪个client地址上，这个地址提供HTTP、DNS、RPC等服务，默认是127.0.0.1。

-config-file：明确的指定要加载哪个配置文件。

-config-dir：配置文件目录，里面所有以.json结尾的文件都会被加载

-data-dir：提供一个目录用来存放agent的状态，所有的agent都需要该目录，该目录必须是稳定的，系统重启后都继续存在。

-dc：该标记控制agent的datacenter的名称，默认是dc1。

-encrypt：指定secret key，使consul在通讯时进行加密，key可以通过consul keygen生成，同一个集群中的节点必须使用相同的key。

-join：加入一个已经启动的agent的ip地址，可以多次指定多个agent的地址。如果consul不能加入任何指定的地址中，则agent会启动失败。默认agent启动时不会加入任何节点。

-retry-join：和join类似，但是允许你在第一次失败后进行尝试。

-retry-interval：两次join之间的时间间隔，默认是30s。

-retry-max：尝试重复join的次数，默认是0，也就是无限次尝试。

-log-level：consul agent启动后显示的日志信息级别。默认是info，可选：trace、debug、info、warn、err。

-node：节点在集群中的名称，在一个集群中必须是唯一的，默认是该节点的主机名。

-protocol：consul使用的协议版本。

-rejoin：使consul忽略先前的离开，在agent再次启动后仍旧尝试加入集群中。也就是说如果不加入这个参数，当前节点一旦退出，下次重启后是不会自动加入到集群中去的，除非是手动触发 consul join xxxx ，所以为了降低重启后对本身服务的影响，这里统一使用 -rejoin参数。

-server：定义agent运行在server模式，每个集群至少有一个server，建议每个集群的server不要超过5个。

-syslog：开启系统日志功能，只在linux/osx上生效。

-ui：启用内置Web UI服务

-ui-dir:提供存放web ui资源的路径，该目录必须是可读的。

-pid-file:提供一个路径来存放pid文件，可以使用该文件进行SIGINT/SIGHUP(关闭/更新)agent。

1. **启动consul server**

把/usr/local/hghac/consul 拷贝到另外2个节点（有就不需拷贝了）

chown -R ivorysql: ivorysql /usr/local/hghac/consul

逐个节点 切换用户启动 su - ivorysql 启动consul

cd /usr/local/hghac/consul

./start.sh

第三步：其他节点加入主节点

其他节点加入主节点，其他2个节点（102,103）分别切换ivorysql用户执行如下

/usr/local/hghac/consul/consul join 192.168.31.101

[ivorysql@host1 ~]$ /usr/local/hghac/consul/consul join 192.168.31.101

Successfully joined cluster by contacting 1 nodes.

第四步：查看consul集群状态

执行以下命令查看集群节点信息：

[ivorysql@host5 consul]$ ./consul operator raft list-peers --http-addr 192.168.31.101:8500

Node ID Address State Voter RaftProtocol

consul-3 4c44d40f-b030-c73d-f6f5-44f1e44c4524 192.168.31.101:8300 follower true 3

consul-1 878ee1e9-2973-cec4-9e52-f885f5b08acf 192.168.31.102:8300 leader true 3

consul-2 14f65e40-8476-c545-fdca-1c1424a8894b 192.168.31.103:8300 follower true 3

查看集群members（里面若有启client也会显示在列表内）

[ivorysql@host5 consul]$ /usr/local/hghac/consul/consul members --http-addr 192.168.31.101:8500

Node Address Status Type Build Protocol DC Partition Segment

consul-1 192.168.31.101:8301 alive server 1.13.2 2 dc1 default <all>

consul-2 192.168.31.102:8301 alive server 1.13.2 2 dc1 default <all>

consul-3 192.168.31.103:8301 alive server 1.13.2 2 dc1 default <all>

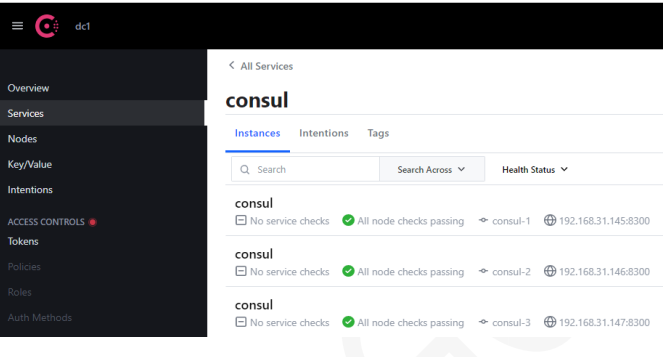
以上结果说明安装配置启动成功了。

第五步：查看url管理界面

服务启动后，即可通过url查看管理界面，默认端口是8500

[http://ip:8500/ui/dc1/services/consul/instances](http://192.168.31.145:8500/ui/dc1/services/consul/instances)

例如 [http://192.168.31.101:8500/ui/dc1/services/consul/instances](http://192.168.31.145:8500/ui/dc1/services/consul/instances)



**5.3.2.3 配置启动 hghac**

进入hac文件夹：

cd /usr/local/hghac/hac

**5.3.2.3.1 配置hghac.yml文件**

各节点需根据实际情况进行配置，这里以主节点为例：

scope: ivorysql-cluster

namespace: /service/

name: ivorysql\_01

restapi:

listen: 192.168.31.101:8008

connect\_address: 192.168.31.101:8008

#根据选择的DCS进行配置

etcd:

hosts: 192.168.31.101:2379,192.168.31.102:2379,192.168.31.103:2379

#zookeeper:

# ['192.168.31.101:2181’,192.168.31.102:2181’,’192.168.31.103:2181’]

#consul

# host: 192.168.31.101:8500 #不支持list

bootstrap:

dcs:

ttl: 30

loop\_wait: 10

retry\_timeout: 10

maximum\_lag\_on\_failover: 1048576

maximum\_lag\_on\_syncnode：

master\_start\_timeout: 300

synchronous\_mode: false

synchronous\_node\_count:

synchronous\_mode\_strict:

postgresql:

use\_pg\_rewind: true

parameters:

listen\_addresses: "\*"

port: 5432

wal\_level: replica

hot\_standby: "on"

wal\_keep\_size: 100

max\_wal\_senders: 10

max\_replication\_slots: 10

wal\_log\_hints: "on"

archive\_mode: "off"

archive\_timeout: 1800s

postgresql: #数据库本地设置

listen: 0.0.0.0:5432

connect\_address: 192.168.31.101:5432

data\_dir: /opt/ivorysql/data

bin\_dir: /opt/ivorysql/bin

config\_dir: /opt/ivorysql/data

authentication:

replication:

username: ivorysql

password: ivorysql

superuser:

username: ivorysql

password: ivorysql

rewind:

username: ivorysql

password: ivorysql

pg\_hba:

- local all all trust

- host all all 0.0.0.0/0 trust

- host all all ::1/128 trust

- local replication all trust

- host replication all 0.0.0.0/0 trust

- host replication all ::1/128 trust

tags:

nofailover: false

noloadbalance: false

clonefrom: false

nosync: false

**参数说明**：

（这里仅对以上配置文件中涉及到的参数进行说明，更多功能相关参数说明请参考用户使用说明书中相关章节。）

|  |  |
| --- | --- |
| scope | 集群名称 |
| namespace | DCS中集群信息的存储位置，默认为“/service”。 |
| name | 当前节点的名称。在同一个集群中，不同节点的名称应不同。 |
| **restapi:** |  |
| connect\_address | 能够访问REST API的ip（或hostname）和port。集群中的所有节点必须能够连接到该地址，所以该地址一定不能是“localhost”或回环地址（如127.0.0.1）。 |
| listen | REST API的监听ip（或hostname）和port。 |
| **etcd:** |  |
| hosts | etcd端点列表。 |
| **dcs:** | **动态配置参数** |
| loop\_wait | 是否使用pg\_rewind，默认为false。 |
| ttl | 获得leader锁的TTL，单位秒。可以将其视为启动自动故障转移前的时长。默认为30秒。 |
| retry\_timeout | DCS和数据库操作重试超时时间，单位秒。当DCS故障或网络故障恢复时间少于该时间时，hghac将不会进行leader降级。默认为10秒。 |
| maximum\_lag\_on\_failover | 备节点可以滞后于主节点的最大字节数，当滞后超过该数值时，该备节点将不能参与leader选举。 |
| maximum\_lag\_on\_syncnode | 同步备节点可以滞后于主节点的最大字节数，当滞后超过该数值时，该同步备节点将被认为是不健康的候选者，并且将被健康的异步备节点所替换。当集群中有多个备节点时，hghac将使用max replica lsn；当集群中只有一个备节点时，hghac将使用主节点当前的wal lsn。 |
| master\_start\_timeout | 允许主节点故障恢复的最长时间，超过此时间将触发自动故障转移。单位秒，默认为300秒。当该参数值设置为0时，一旦检测到主节点故障将立即触发自动故障转移。 |
| synchronous\_mode | 打开同步流复制模式。 |
| synchronous\_node\_count | 集群中同步流复制节点的个数，默认为1。 |
| synchronous\_mode\_strict | 当没有可用的同步备节点时，防止同步流复制被禁用，从而阻止所有客户端对主节点的写操作。 |
| postgresql.use\_pg\_rewind | 是否使用pg\_rewind，默认为false。 |
| postgresql.parameters | 数据库相关设置列表 |
| **tags:** |  |
| nofailover | 值为true或false。控制该节点是否被允许参与leader竞争并成为一个leader。默认为false。 |
| clonefrom | 值为true或false。当设置为true时，其他节点可能优先使用该节点做引导（从该节点做pg\_basebackup）。如果有多个节点的clonefrom标签都设置为true，则会随机选取引导节点。默认为false。 |
| noloadbalance | 值为true或false。如果设置为true，则当执行GET/replica REST API健康检查时，该节点将返回HTTP状态代码503，从而被排除在负载均衡之外。默认为false。 |
| nosync | 值为true或false。当设置为true时，该节点将永远不会被选为同步备节点。 |

**5.3.2.3.2 启动hghac**

各节点分别以Ivorysql权限启动hghac:

./hghac/hghac hghac.yml > hghac.log1 2>&1 &

**5.3.2.3.3 查看hghac集群状态**

集群中任一节点执行以下命令，查询结果如下，说明hghac安装启动成功了，高可用集群搭建成功了：

[root@host101 hghac]# ./hghactl/hghactl -c hghac.yml list

+ Cluster: ivorysql-cluster (7142757175828284941) -+----+-----------+

| Member | Host | Role | State | TL | Lag in MB |

+-------------+----------------+---------+---------+----+-----------+

| ivorysql\_01 | 192.168.31.101 | Leader | running | 1 | |

| ivorysql\_02 | 192.168.31.102 | Replica | running | 1 | 0 |

| ivorysql\_03 | 192.168.31.103 | Replica | running | 1 | 0 |

+-------------+----------------+---------+---------+----+-----------+

**5.3.2.4 配置启动vip-manager**

进入vip-manager文件夹：

cd /usr/local/hghac/vip-manager

**5.3.2.4.1 配置vip-manager.yml文件**

需根据实际情况进行配置，以主节点为例：

interval: 1000

trigger-key: "/service/ivorysql-cluster/leader"

trigger-value: "ivorysql\_01"

ip: 192.168.31.244

netmask: 24

interface: enp0s3

hosting-type: basic

#根据选择的DCS进行配置

dcs-type: etcd

dcs-endpoints:

http://127.0.0.1:2379,http://192.168.31.101:2379,http://192.168.31.102:2379,http://192.168.31.103:2379

#dcs-type: consul

#dcs-endpoints:

#http://192.168.31.101:8500

#etcd-user: "hghac"

#etcd-password: "Julian's secret password"

#etcd-ca-file: "/path/to/etcd/trusted/ca/file"

#etcd-cert-file: "/path/to/etcd/client/cert/file"

#etcd-key-file: "/path/to/etcd/client/key/file"

#consul-token: "Julian's secret token"

retry-num: 2

retry-after: 250

verbose: false

**参数详解：**

* ip 要管理的虚拟IP地址
* netmask 与虚拟IP vip所属的子网相关联的子网掩码。
* interface 运行vip-manager的机器上的本地网络接口。当使用hosting-type=basic（默认）时需要。该界面将添加和删除vip。
* trigger-key 将由vip-manager监控的DCS中的密钥。必须匹配hghac配置中的<namespace>/<scope>/leader。当DCS返回的值等于trigger-value时，vip-manager将确保虚拟IP已注册到这台机器。如果不匹配，vip-manager会确保虚拟IP没有注册到这台机器上。
* trigger-value DCS的触发键应答将被匹配的值。必须匹配hghac配置中的<name>。这通常设置为与这个vip-manager实例相关联的hghac集群成员的名称。默认为机器的主机名。
* hosting-type 要么是basic要么是hetzner。介绍了虚拟IP的管理机制。默认为basic。
* dcs-type vip-manager用于监控触发键的DCS类型。默认为etcd。
* dcs-endpoints 定义到达DCS地址的url。可以使用逗号分隔的列表将多个端点传递给这个参数或环境变量。在配置文件中，可以指定一个列表，示例请参见配置示例。dcs-type=etcd时，默认为http://127.0.0.1:2379； dcs-type=consul时，默认为http://127.0.0.1:8500。
* 同时，用“#”注释掉 etcd-ca-file， etcd-cert-file， etcd-key-file。我们不使用SSL 认证.

**5.3.2.4.2 启动vip-manager**

各节点以root权限分别启动vip-manager:

./vip-manager --config vip-manager.yml

**5.3.2.4.3 查看确认vip**

各节点执行ip addr命令，可以发现，主节点捆绑了vip，备节点没有。

结果如下，说明vip-manager安装启动成功了：

[root@host101 hghac]# ip addr

......

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 08:00:27:e2:1a:77 brd ff:ff:ff:ff:ff:ff

inet 192.168.31.101/24 brd 192.168.31.255 scope global noprefixroute enp0s3

valid\_lft forever preferred\_lft forever

inet 192.168.31.244/24 scope global secondary enp0s3

valid\_lft forever preferred\_lft forever

......

**5.3.3 设置高可用组件开机自启动**

**5.3.3.1 etcd开机自启动**

**5.3.3.1.1 配置文件修改**

根据实际环境，更改文件etcd.service。

etcd.service文件：

[Unit]

Description=Etcd

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

Type=notify

User=ivorysql

Group=ivorysql

WorkingDirectory=/usr/local/hghac/etcd/

ExecStart=/usr/local/hghac/etcd/etcd --config-file=/usr/local/hghac/etcd/etcd.yml

Restart=no

[Install]

WantedBy=multi-user.target

**5.3.3.1.2 设置etcd开机自启动服务**

拷贝etcd.service到目录/usr/lib/systemd/system/：

cp etcd.service /usr/lib/systemd/system/

设置etcd开机自启动功能：

systemctl enable etcd.service

**5.3.3.2 zookeeper开机自启动**

**5.3.3.2.1 配置文件修改**

根据实际环境，更改zookeeper安装目录下的文件zookeeper.service和zookeeper.env。

zookeeper.service文件：

[Unit]

Description=Zookeeper

After=network.target

Wants=network.target

[Service]

Type=forking

User=ivorysql

Group=ivorysql

EnvironmentFile=/usr/local/hghac/zookeeper/zookeeper.env

ExecStart=/usr/local/hghac/zookeeper/bin/zkServer.sh start

Restart=no

[Install]

WantedBy=multi-user.target

zookeeper.env文件：

JRE\_HOME=/usr/local/hghac/jre\_linux

CLASSPATH=$JRE\_HOME/lib:$CLASSPATH

PATH=$JRE\_HOME/bin:/usr/bin/:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/app/idk/bin:$PATH

**5.3.3.2.2 设置zookeeper开机自启动服务**

拷贝zookeeper.service到目录/usr/lib/systemd/system/：

cp zookeeper.service /usr/lib/systemd/system/

设置zookeeper开机自启动功能：

systemctl enable zookeeper.service

**5.3.3.3 consul开机自启动**

**5.3.3.3.1 配置文件修改**

**1.所有节点新建[consul](https://so.csdn.net/so/search?q=consul&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/xwnxwn/article/details/_blank).sevice文件**

在consul.sevice中配置启动命令，其中ExecStart=/usr/local/hghac/consul/consul agent -config-dir /usr/local/hghac/consul/consul.d -enable-script-checks=true -rejoin -ui -log-rotate-bytes=102400000 就是执行命令，/usr/local/hghac/consul/consul为实际安装目录，根据情况修改。

vi /lib/systemd/system/consul.service

[Unit]

Description=Consul

After=network.target

[Service]

User=ivorysql

Group=ivorysql

ExecStart=/usr/local/hghac/consul/consul agent -config-dir /usr/local/hghac/consul/consul.d -enable-script-checks=true -rejoin -ui -log-rotate-bytes=102400000 -log-file=/usr/local/hghac/consul/log/ -client=0.0.0.0

KillSignal=SIGINT

[Install]

WantedBy=multi-user.target

**2. 创建执行命令文件目录**

mkdir -p /usr/local/hghac/consul/consul.d

**3.创建配置文件**

新建命令文件，配置如下(3个节点根据实际情况修改)

vi /usr/local/hghac/consul/consul.d/server.json

{

"datacenter": "dc1",

"addresses":{

"http" : "192.168.31.145",

"dns": "192.168.31.145"

},

"rejoin\_after\_leave":true,

"retry\_join":["192.168.31.101","192.168.31.102","192.168.31.103"],

"data\_dir": "/usr/local/hghac/consul/data",

"log\_level": "INFO",

"node\_name": "consul-1",

"server": true,

"ui": true,

"bootstrap\_expect":3,

"client\_addr": "0.0.0.0",

"bind\_addr": "192.168.31.145",

"verify\_incoming":false,

"verify\_outgoing":false,

"disable\_remote\_exec":false,

"domain":"consul",

"telemetry": {

"statsd\_address": "127.0.0.1:8125"

}

}

备注：可通过/usr/local/hghac/consul/consul agent -config-dir /usr/local/hghac/consul/consul.d -enable-script-checks=true -rejoin -ui -log-rotate-bytes=102400000 这个命令直接执行，测试看是否正常。

**参数详解：**

datacenter ：数据中心名称

node\_name ：节点名称

server ：是否为 Server 模式，false 为 Client 模式

bootstrap\_expect ：此标志提供数据中心中预期的服务器数量。要么不提供此值，要么该值必须与集群中的其他服务器一致。如果提供，Consul 将等待直到指定数量的服务器可用为止，然后引导群集。这样可以自动选举最初的领导者。不能与旧 -bootstrap 标志一起使用。该标志需要 -server 模式。

data\_dir ：Server 节点数据目录

-log-file-将所有 Consul 代理日志消息写入文件

ui ：是否开启 UI 访问

bind\_addr ：Server 绑定的 IP，填本机即可

client\_addr ：作为 Client 绑定的 IP，默认为 127.0.0.1，也就是只有本地服务可以注册，这很明显是不符合需求的

retry\_join ：尝试加入的其他节点，填写集群中的其他节点 ip 即可

raft\_protocol ：Raft 协议版本

rejoin\_after\_leave ：允许重新加入集群

**5.3.3.3.2 设置consul开机自启动服务**

通过服务启动

systemctl start consul

重新加载启动配置

systemctl daemon-reload

设置开机启动

systemctl enable consul

**5.3.3.4 hghac开机自启动**

**5.3.3.4.1 配置文件修改**

根据实际环境，更改hghac安装目录下的文件hghac.env和hghac.service文件。

hghac.env文件:

LD\_LIBRARY\_PATH=/usr/local/ivorysql/lib

此配置用于指定libpq库所在的目录。

hghac.service文件：

[Unit]

Description=hghac

After=syslog.target network.target

[Service]

Type=simple

User=ivorysql

Group=ivorysql

EnvironmentFile=/usr/local/hghac/hac/hghac.env

# Start the hghac process

ExecStart=/usr/local/hghac/hac/hghac /usr/local/hghac/hac/hghac.yml

# Send HUP to reload from hghac.yml

ExecReload=/bin/kill -s HUP $MAINPID

#Stop the db\_ha-agent process,the main process is killed by default

Restart=no

[Install]

WantedBy=multi-user.target

**5.3.3.4.2 设置hghac开机自启动服务**

拷贝hghac.service到目录/usr/lib/systemd/system/

cp hghac.service /usr/lib/systemd/system/

设置hghac开机自启动功能：

systemctl enable hghac.service

**5.3.3.5 vip-manager开机自启动**

**5.3.3.5.1 配置文件修改**

根据实际环境，更改文件vip-manager.service。

vip-manager.service文件：

[Unit]

Description=hghac

After=network.target

After=network-online.target

Wants=network-online.target

[Service]

Type=simple

User=root

Group=root

ExecStart=/usr/local/hghac/vip-manager/vip-manager --config=/usr/local/hghac/vip-manager/vip-manager.yml

Restart=no

[Install]

WantedBy=multi-user.target

**5.3.3.5.2 设置vip-manager开机自启动服务**

拷贝vip-manager.service到目录/usr/lib/systemd/system/：

cp vip-manager.service /usr/lib/systemd/system/

设置vip-manager开机自启动功能：

systemctl enable vip-manager.service

**5.3.4 hghac初始化数据库并部署流复制的功能说明**

hghac提供了初始化数据库并部署流复制的功能，用户可选择由hghac来搭建高可用流复制集群，启动hghac之前需安装数据库并设置环境变量。

以安全版数据库4.5.8为例，具体搭建步骤如下：

1. 安装数据库

rpm -ivh hgdb-see-4.5.8-db43858.x86\_64.rpm

1. 设置数据库的环境变量

安装完毕后会在/opt/highgo/hgdb-see-4.5.8/etc 目录下生成一个名为 hgdbenv.sample 的模板文件，内容如下，执行 source 命令使之生效。

[root@node etc]# cat hgdbenv.sample

#WARNING! This is a template file, please modify it as needed

export HG\_BASE=/opt/highgo

export HGDB\_HOME=/opt/highgo/hgdb-see-4.5.8

export PGPORT=5866

export PGDATABASE=highgo

export PATH=$HGDB\_HOME/bin:$PATH

export PGDATA=$HGDB\_HOME/data

[root@node etc]# source hgdbenv.sample

#检查环境变量是否生效

[root@node etc]# echo $PGDATA

/opt/highgo/hgdb-see-4.5.8/data

将该文件写入系统的环境变量文件中，例如将如下命令写入/root/.bash\_profile：

source /opt/highgo/hgdb-see-4.5.8/etc/hgdbenv.sample

1. 安装hghac

rpm -ivh hghac2.1.4.2-see-20221225-1.el7.x86\_64.rpm

1. 配置启动DCS

按照以上章节说明配置启动DCS，此处省略。

1. 配置hghac.yml

各节点需根据实际情况进行配置，这里以主节点为例。

需要说明的是，安全版数据库initdb之后需通过hg\_sslkeygen.sh 生成证书和key才能启动数据库，因此在hghac.yml配置文件中指定了自定义初始化脚本来代替默认的initdb。

scope: highgo-see-cluster

namespace: /service/

name: see\_01

restapi:

listen: 10.0.127.215:8008

connect\_address: 10.0.127.215:8008

etcd:

hosts: 10.0.127.215:2379,10.0.127.216:2379,10.0.127.217:2379

#zookeeper:

# hosts: ['192.168.31.101:2181', '192.168.31.102:2181', '192.168.31.103:2181']

bootstrap:

dcs:

ttl: 30

loop\_wait: 10

retry\_timeout: 10

maximum\_lag\_on\_failover: 1048576

master\_start\_timeout: 300

synchronous\_mode: false

postgresql:

use\_pg\_rewind: true

parameters:

listen\_addresses: "\*"

port: 5866

wal\_level: replica

hot\_standby: "on"

wal\_keep\_size: 100

max\_wal\_senders: 10

max\_replication\_slots: 10

wal\_log\_hints: "on"

archive\_mode: "off"

archive\_timeout: 1800s

#以下为安全版数据库自定义的bootstrap方法

method: initdb\_keygen

initdb\_keygen:

command: /usr/local/hghac/hac/initdb\_keygen.sh #指定自定义的脚本作为command执行

keep\_existing\_recovery\_conf: True

no\_params: True

#对于其他数据库的初始化，使用默认的initdb即可，示例如下：

 initdb:

  - encoding: UTF8

  - locale: C

  - data-checksums

postgresql:

database: highgo

listen: 0.0.0.0:5866

connect\_address: 10.0.127.215:5866

bin\_dir: /opt/highgo/hgdb-see-4.5.8/bin

data\_dir: /opt/highgo/hgdb-see-4.5.8/data

config\_dir: /opt/highgo/hgdb-see-4.5.8/data

pgpass: /root/.pgpass

authentication:

replication:

username: sysdba

password: Hello@123

rewind:

username: sysdba

password: Hello@123

sysdba:

password: Hello@123

syssso:

password: Hello@123

syssao:

password: Hello@123

parameters:

ssl: 'on'

ssl\_cert\_file: /opt/highgo/hgdb-see-4.5.8/data/server.crt

ssl\_key\_file: /opt/highgo/hgdb-see-4.5.8/data/server.key

pg\_hba:

- local all all sm3

- host all all 0.0.0.0/0 sm3

- host all all ::1/128 sm3

- local replication all sm3

- host replication all 0.0.0.0/0 sm3

- host replication all ::1/128 sm3

tags:

nofailover: false

noloadbalance: false

clonefrom: false

nosync: false

1. 启动hghac

主节点以root权限启动hghac，hghac启动过程中会初始化启动主数据库:

./hghac/hghac hghac.yml > hghac.log1 2>&1 &

主节点启动以后，备节点以root权限启动hghac，hghac启动过程中以pg\_basebackup的方式拉起备库流复制。

1. 查看hghac集群状态，高可用流复制集群搭建成功：

[root@node hac]# ./hghactl/hghactl -c hghac.yml list

+ Cluster: highgo-see-cluster (7181088023437970509) --+-----------+

| Member | Host | Role | State | TL | Lag in MB |

+--------+-------------------+---------+---------+----+-----------+

| see\_01 | 10.0.127.215:5866 | Leader | running | 1 | |

| see\_02 | 10.0.127.216:5866 | Replica | running | 1 | 0 |

| see\_03 | 10.0.127.217:5866 | Replica | running | 1 | 0 |

+--------+-------------------+---------+---------+----+-----------+