目录

1. 概述

本文描述了 GGHK 集群管理工具的部署、使用、卸载和常见问题处理，以帮助用户快速掌握使用 GGHK 进行 Opengauss 数据库集群管理。

【约定】文档中出现在命令中的斜体字内容需要根据实际运行环境进行替换。

2. GGHK 部署

GGHK 部署包括三个模块的部署：GGHC 部署、GGHS 部署和 FLOATIP 部署。以下章节将对其分别描述。

2.1. GGHC 模块部署

2.1.1. 准备部署环境

首先，准备一台 GGHC 主机，建议主机至少达到配置：2CPU、2G MEM、2GB 以上剩余磁盘空间，百兆网卡。

然后，主机上安装 centos7.6，python3.6 以及 paramiko 库，具体安装和配置方法不在文本描述。

之后，在主机上创建 gghc 用户并设置密码。

此外，需要为 Opengauss 集群规划一个浮动 IP，该浮动 IP 必须可以通过外网访问，并且预先配置到 Opengauss 的 pg_hba.conf 文件访问许可列表中。该浮动 IP 之后还将绑定到 Opengauss 主机 postgresql.conf 文件中 listen_addresses 配置项 IP 所属的网卡上。

2.1.2. 建立与 OMM 的 SSH 信任关系

GGHC 需要调用 GGHS 模块的 python 脚本代码实现对 Opengauss 集群管理，GGHS 将部署在 Opengauss 的 omm 账户下，所以 GGHC 需要与 omm 建立 SSH 信任关系，这样 GGHC 就可以通过 SSH 连接调用部署在 omm 账户下的 GGHS 模块脚本。

操作步骤如下：

1) 生成密钥对

以 gghc 账户登录 GGHC 主机，执行 ssh-keygen -t rsa 生成密钥对。

```
$ ssh-keygen -t rsa
```

请按提示输入信息或直接回车生成密钥对，默认情况下，密钥将生成在$HOME/.ssh 目录下。进入目录可以看到两个文件：id_rsa 和 id_rsa.pub。前者是私钥后者是公钥。

2) 将公钥传送给 omm

以 gghc 账户登录 GGHC 主机，使用如下命令将 gghc 公钥传送给 Opengauss 集群的 omm。

```
$ ssh-copy-id omm@<ip>
```

实际操作时<ip>使用 Opengauss 主机 listen_address 的 ip 地址进行替代，因为 Opengauss 集群存在一主多备多个主机，所以上述命令需要针对每个主机各执行一次。

执行命令过程，需要输入 omm 账户的密码进行认证确认，否则建立 SSH 信任关系将失败。

信任关系建立后，可以在 omm 的$HOME/.ssh/authorized_keys 文件的底部查看到 gghc 账户的公钥字符串。

3) 检验 SSH 信任关系是否成功建立

以 gghc 账户登录 GGHC 主机，使用如下命令检验信任关系是否建立成功。

```
$ssh omm@<ip>
```

实际操作时<ip>使用 Opengauss 主机 listen_address 的 ip 地址进行替代。如果回车后可以直接进入远端的 omm 账户，则说明 SSH 信任关系建立成功。如果系统提示输入 omm 密码，则表示建立不成功，需检测错误原因，并删除本次的生成内容，重新创建。

2.1.3. 部署 GGHC 代码

1) 部署代码

下载 gghk-1.0.0-centos-7.6.tar.gz 文件，解压出其中的 gghc.tar 文件，以 binary 方式上传到 GGHC 主机的 gghc 账户 HOME 目录，然后以 gghc 账户登录 GGHC 主机，解压上传包。

```
$ tar xvf gghc.tar
```

解压后，可以看到 HOME 目录有类似如下的内容：

```
$ ls
bin  conf  gghc  gghc.tar
```

其中：

bin 目录存放系统可执行文件，目前只有脚本文件 gghc.sh。

conf 目录存放系统配置文件，配置文件名为 config.xml，里面的数据需要根据实际环境进

行修改。运行时该目录下还可能会生成一个 status.xml 文件，用于记录集群管理时因为节点故障或网段不通而未能完成的操作和数据，以便在节点故障恢复后继续完成处理。

gghc 为模块的代码路径，存放模块所有 python 代码，其中 GGHC.py 为主执行文件

运行运行后，还将在 HOME 目录下创建 log 目录，用于存放系统运行中产生的 log。系统有两个 log 文件: system.log 记录系统重要的处理信息；state.log 记录检测到的 opengauss 集群的状态数据，状态每次变化都会产生一条记录。

说明：也可以下载 gghc 的源码直接上传部署，但是上传必须选择 ASCII 方式上传。上传后保持上述目录结构，并确保 GGHC.py 具有可执行权限。

2)　修改配置

以 gghc 账户登录 GGHC 主机，进入 conf 目录，打开 config.xml 文件，完成相关配置。

相关配置如下，请根据实际环境进行修改：

```
<!-- 数据库集群监听 IP,与 DB 集群安装配置文件一致，不含有浮动 IP -->
<db_listen_addresses>10.88.50.111,10.88.50.165,10.88.50.152</db_listen_addresses>
<db_listen_port>26000</db_listen_port>
<db_datanode_path> /opt/software/install/data/db1 </db_datanode_path>
<db_user>omm</db_user>
<float_ip>10.88.50.254</float_ip>
<!-- 浮动 IP 的网卡标签名 -->
<floatip_eth>enp0s3:1</floatip_eth>
<!--gghc 端的 IP 地址 -->
<gghc_connect_ip>10.88.51.164 </gghc_connect_ip>
<!-- ssh 认证私钥文件路径-->
<gghc_private_key_file>/home/gghc/.ssh/id_rsa</gghc_private_key_file>
<gghs_agent_path>gghs/agent.py</gghs_agent_path>
<state_check_period> 10 </state_check_period>
<ssh_timeout> 60 </ssh_timeout>
<!-- 支持 CRITICAL，ERROR，WARNING，INFO，DEBUG 基本配置 -->
<log_level>DEBUG</log_level>
```

配置项说明：

db_listen_addresses	Opengauss 数据库集群 listen_address 的 IP 列表，其顺序与在 Opengauss 上使用 gs_om -t status --detail 命令查看到的顺序一致
db_listen_port	Opengauss 数据库集群对外服务端口号
db_datanode_path	Opengauss 数据库集群数据库节点的数据存储目录
db_user	Opengauss 数据库管理员账户
float_ip	为 Opengauss 数据库集群规范的浮动 IP

floatip_eth	绑定浮动 IP 时使用的网卡标签名
gghc_connect_ip	gghc 进程进行 Opengauss 数据库集群检查时所用的 IP 地址
gghc_private_key_file	gghc 账户 rsa 私钥文件全路径
gghs_agent_path	gghc 远程代理脚本在 omm 账户中相对于其 HOME 的相对路径
state_check_period	gghc 健康检查周期（单位秒）
ssh_timeout	gghc 通过 SSH 连接发起远程访问的超时时间（单位秒）
log_level	gghc 运行期的日志输出级别（不支持动态修改）

2.2. GGHS 模块部署

下载 gghk-1.0.0-centos-7.6.tar.gz 文件，解压出其中的 gghs.tar 文件，以 binary 方式上传到 Opengauss 的每一个主机的 omm 账户 HOME 目录，然后以 omm 登录集群主机，直接解压代码包。

```
$ tar xvf gghs.tar
```

解压后，可以看到 HOME 目录有类似如下的内容：

```
$ ls
gghs gghs.tar
$ cd gghs
$ ls -al
总用量 12
drwx------ 2 omm dbgrp    22 1 月  27 10:39 .
drwx------ 4 omm dbgrp   135 1 月  27 10:39 ..
-rwx------ 1 omm dbgrp 10970 1 月  26 18:38 agent.py
```

注意：请检测 agent.py 具有可执行权限，如果没有请执行 chmod u+x agent.py 增加可执行权限。

说明：也可以下载 gghs 的源码直接上传部署，但是上传必须选择 ASCII 方式上传。上传后保持上述目录结构，并确保 agent.py 具有可执行权限。

2.3. FLOATIP 模块部署

1) 部署代码文件

下载 gghk-1.0.0-centos-7.6.tar.gz 文件，解压出其中的 floatip.sh 文件。

打开 floatip.sh 文件，修改文件上部的三行变量值，使其与实际环境一致，如下：

```
# Opengauss 监听地址所属网卡
netcard='enp0s3'
# 浮动 IP 网卡标签名称
float_card_label='enp0s3:1'
```

```
# 浮动 IP 地址
floatip='10.88.50.254'
```

保存修改后，以 ASCII 方式上传到 Opengauss 的每一个主机的/usr/bin 目录下。

2) 修改文件权限

以 root 账户身份登录 Opengauss 每个主机，修改/usr/bin/floatip.sh 文件权限为最小使用权限：

```
# chmod 500 /usr/bin/floatip.sh
# ls  -al /usr/bin/floatip.sh
-r-x------ 1 root root 994 1 月  27 15:36 /usr/bin/floatip.sh
```

3) 增加 omm 对 floatip.sh 的 sudo 执行权限

以 root 账户身份登录 Opengauss 每个主机，执行 visudo 命令，打开/etc/sudoers 文件。

```
# visudo
```

说明：直接执行 visudo 命令，无需指定文件名。

在文件中找到下面位置：

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
```

然后在下面增加类似下一行配置，并保存退出：

```
omm  gauss1=(root)NOPASSWD:/usr/bin/floatip.sh
```

说明：

omm 是执行用户名，表示后面配置的命令 omm 可以 sudo 方式执行。

gauss1 是主机名，需要用实际环境主机名进行替换，表示后面的配置命令只能在本机执行，不能远程登录执行。

(root)表示必须以 root 用户身份执行，即必须 sudo 执行。

NOPASSWD 表示 sudo 执行时不需要输入 omm 密码。

/usr/bin/floatip.sh 是执行文件的全路径。

4) 检查是否 floatip 成功配置

以 omm 账户登录 Opengauss 的每个主机节点。执行如下命令：

```
$ sudo floatip.sh up
```

```
success
$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.88.50.111  netmask 255.255.255.0  broadcast 10.88.50.255
        inet6 fe80::fb9d:5293:dbe1:4654  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:67:55:a3  txqueuelen 1000  (Ethernet)
        RX packets 23206  bytes 2241854 (2.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5827  bytes 1008757 (985.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.88.50.254  netmask 255.255.255.0  broadcast 0.0.0.0
        ether 08:00:27:67:55:a3  txqueuelen 1000  (Ethernet)
$ sudo floatip.sh down
success
```

如果配置成功，执行 sudo floatip.sh up 命令时不需要输入 omm 密码即可获得 success 结果，并且在执行随后的 ifconfig 命令，可以浮动 IP 出现在虚拟网卡 enp0s3:1 上，该虚拟网卡即为先前配置在 floatip.sh 中“浮动 IP 网卡标签名称”。

说明：请记得最后执行 sudo floatip.sh down 取消浮动 ip。

3. GGHK 启动

在启动 GGHK 之前，请检查以上 GGHC、GGHS、FLOATIP 模块全部部署成功，并且配置无误，而且 Opengauss 集群 Standby 节点没有启动浮动 IP。然后检查 Opengauss 集群，要求其已经启动，且处于 Normal 运行状态。

GGHK 启动只涉及 gghc 的启动。以 gghc 账户登录 GGHC 主机，进入 HOME 目录。

执行如下命名启动 GGHC 进程：

```
$ cd $HOME/bin
$ gghc.sh start
GGHC.py started, pid is 5257
please use command 'tail -F $HOME/start.log' to get more detail
```

进程启动后，将在屏幕打印上述内容，告诉用户可以通过 tail -F $HOME/start.log 查看启动日志，并随后在$HOME 下创建 log 目录，存放运行日志，包括 system.log 和 state.log。

系统运行后，gghc 还可能会在$HOME/conf 目录下生成一个 status.xml 状态文件。这是因为主备切换后，Opengauss 需要执行 gs_om -t refreshconf 命令重新刷新集群主备关系，而且 gghc 也需要将曾经增加到故障主节点 postgresql.conf 文件中 listen_addresses 的浮动 ip

清除掉。若这时主机故障或网卡不通，则无法处理，因此必须把这些未完成操作记录在一个状态文件中，并在每轮健康检测时，检测主机或网卡是否故障恢复，若恢复继续完成之前未完成操作。

4. GGHK 卸载

卸载 GGHK 之前，请先停止 gghc 进程。以 gghc 账户登录 GGHC 主机，进行如下操作：

```
$ cd $HOME/bin
$ gghc.sh stop
```

GGHK 卸载同样包括三个模块卸载：GGHC 模块、GGHS 模块、FLOATIP 模块，下面分别进行描述。

4.1. 卸载 GGHC 模块

以 gghc 账户登录 GGHC 主机，进入 HOME 目录，删除 gghc 账户下相关文件。命令如下：

```
$rm -rf bin conf gghc log
```

4.2. 卸载 GGHS 模块

首先，以 omm 账户登录 Opengauss 各个主机节点，进入 HOME 目录，执行如下命名删除 gghs 模块。

```
$ rm -rf gghs
```

然后，打开$HOME/.ssh/authorized_keys

```
$ vi $HOME/.ssh/authorized_keys
```

找到之前建立 SSH 信任关系 gghc 公钥数据行，删除后保存。

4.3. 卸载 FLOATIP 模块

首先，以 root 账户登录 Opengauss 各个主机节点，执行如下命令，删除 floatip.sh 文件。

```
$ rm /usr/bin/floatip.sh
```

然后执行 visudo 命令，打开/etc/sudoers 文件。

```
$ visudo
```

找到之前在文件中配置的 omm 数据行，进行删除，并保存。

5. 人工维护 Opengauss 集群

如果需要人工维护 Opengauss 集群，请先停止 GGHC 进程。

需要注意，进行人工维护时，如果集群使用 GGHK 接管过，正常情况下会在集群 Primary 节点 postgresql.conf 文件的 listen_addresses 配置项增加浮动 IP。如果此时主机的浮动 IP 没有生效，则拉起数据库进程将失败，因为数据库进程启动是需要绑定 listen_addresses 中的全部 IP，如果该 IP 没有生效，则绑定操作失败，进程无法启动。

解决方法有二：

方法一，删除 listen_addresses 配置项中的浮动 IP

方法二，执行 sudo floatip.sh up 启动浮动 IP，然后再拉起进程。

同样道理，如果 Standby 节点曾经做过 Primary 节点，gghc 也会在其 listen_addresses 配置项增加浮动 IP，并在其切换为 Standby 后将其删除。但是也会因为主机故障或网卡故障，使得 gghc 无法通过 SSH 连接这些主机，导致浮动 IP 继续残留。对于无法删除情况，gghc 会在本地的$HOME/conf/status.xml 文件进行记录，并在每一轮监控检测时检测该文件，对未完成操作继续处理。因此正常情况下，只要 gghc 能够通过 SSH 连接这些主机，就能删除 Standby 主机 listen_addresses 配置项的浮动 IP。如果一直无法 SSH 连接上，则一直保留，因此人工维护 Standby 节点，发现无法将其拉起时，也需要检测 postgresql.conf 文件的 listen_addresses 配置项是否存在浮动 IP，如果存在，将其删除。

待 Opengauss 集群人工维护完毕，能够正常拉起到 Normal 状态后，可以继续让 GGHK 接管。接管前，请先检查 GGHC 主机中 gghc 账户下，$HOME/conf/目录中是否存在 status.xml 文件，如果存在，请将其删除，然后启动 gghc 进程。

6. FAQ

6.1. GGHC 启动时日志报告设置浮动 IP 失败而退出

6.1.1. 问题描述

启动 GGHC 进程后，system.log 报告配置不了浮动 IP，GGHC 自动停止运行。

system.log 日志打印如下：

```
2021-01-27 17:50:33,693 - INFO:[checker.py(221)] Find primary node has no float ip 10.88.50.254 on network card enp0s3:1 or not become effective in postgresql.com
2021-01-27 17:51:33,783 - CRITICAL:[sshclient.py(50)] The network card may be failure in the connection of nodeId: 1 at sending 'PRIMARY_ADD_FLOATIP'.
Traceback (most recent call last):
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/channel.py", line 699, in recv
    out = self.in_buffer.read(nbytes, self.timeout)
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/buffered_pipe.py", line 164, in read
    raise PipeTimeout()
paramiko.buffered_pipe.PipeTimeout

During handling of the above exception, another exception occurred:
```

```
Traceback (most recent call last):
  File "/home/lizhenfeng/gghc/sshclient.py", line 44, in execute
    for info in stdout.readlines():
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/file.py", line 349, in readlines
    line = self.readline()
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/file.py", line 291, in readline
    new_data = self._read(n)
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/channel.py", line 1361, in _read
    return self.channel.recv(size)
  File "/usr/local/python3/lib/python3.6/site-packages/paramiko/channel.py", line 701, in recv
    raise socket.timeout()
socket.timeout

2021-01-27 17:51:33,784 - CRITICAL:[checker.py(227)] set float ip on primary node 1 failed, system exited
```

6.1.2. 解决办法

以 root 账户登录 Opengauss 集群主机节点。使用 visudo 命令打开/etc/sudoers 文件。查看是否正确配置下面数据行：

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
omm  gauss1=(root)NOPASSWD:/usr/bin/floatip.sh
```

说明：

omm 是执行用户名，表示后面配置的命令 omm 可以 sudo 方式执行。

gauss1 是主机名，需要用实际环境主机名进行替换，表示后面的配置命令只能在本机执行，不能远程登录执行。

(root)表示必须以 root 用户身份执行，即必须 sudo 执行。

NOPASSWD 表示 sudo 执行时不需要输入 omm 密码。

/usr/bin/floatip.sh 是执行文件的全路径。

如果不存在，则加上该数据行。

然后，检查/etc/sudoers 文件的权限是否如下所示。

```
# ls -al /etc/sudoers
-r--r----- 1 root root 4482 1月  27 11:02 /etc/sudoers
```

如果不一致，请按如下方法进行修改：

```
# chmod 440 /etc/sudoers
```

操作完成后重启 gghc 进程，即可以正常运行!

6.2.GGHC 启动时日志报告数据库集群状态为 Unavailable 而退出

6.2.1. 问题描述

启动 GGHK 后，system.log 日志打印

```
ERROR:[checker.py(540)] At system start, DB cluster state is 'Unavailable;(1,testnode1,10.100.80.***,P,Down,Manually stopped,);(2,testnode2,10.100.80.***,S,Standby,Need repair,Connecting);(3,testnode3,10.100.80.***,S,Standby,Need repair,Disconnected).'
It should be Normal or Degraded and can successfully acquire db nodes' listen_addresses, so system exits..
File "/home/gghc//gghc/checker.py", line 558, in check
    exit(-1)
  File "/usr/local/python3/lib/python3.6/_sitebuiltins.py", line 26, in __call__
    raise SystemExit(code)
SystemExit: -1。
```

6.2.2. 解决办法

正常启动 gghc 的前提是集群状态为 Normal or Degraded，并且能够通过在 Opengauss 集群节点执行“gs_guc check -D /opt/software/install/data/db1 -c "listen_addresses" -N all”命令获得 Opengauss 的 listen_addresses，否则会退出。因为 gghc 系统启动时需要获取 Openguass 集群的有关信息并进行校验，以便后续进行正确的集群管理。

6.3.omm 账户登录执行 sudo floatip.sh up 失败

6.3.1. 问题描述

以 omm 账户登录 Opengauss 集群节点，执行命令 sudo floatip.sh up/down，出现错误提示如下：

```
$ sudo floatip.sh up/down
sudo: floatip.sh：找不到命令
```

6.3.2. 解决办法

出现上述错误的原因在于脚本没有可执行权限。切换到 root 用户下，进入/usr/bin 目录，给 floatip.sh 可执行权限

```
# chmod u+x floatip.sh
```

6.4. GGHK 进行集群管理后再也无法手工拉起集群节点

6.4.1. 问题描述

使用 GGHK 进行 Opengauss 集群系统管理测试，使用一段时间后关闭 gghc 进程，并关闭 Opengauss 集群。几天后，重新启动 Opengauss 集群失败，而且通过 gs_ctl 命令拉起主节点也无法成功。

6.4.2. 解决办法

出现上述问题的原因是：GGHC 为 Opengauss 主节点的 postgresql.conf 文件 listen_addressess 配置项中增加了浮动 IP。因为 Opengauss 必须在进程启动时绑定 listen_addressess 中的浮动 IP，若该浮动 IP 未启动，则绑定操作失败，数据库进程无法启动。

因此，可以使用两种方法进行解决。

方法一：

删除主节点 listen_addressess 中的浮动 IP，然后拉起 Opengauss 集群。

方法二：

继续保留主节点 listen_addressess 中的浮动 IP，并启动浮动 IP，然后拉起 Opengauss 集群。

```
# sudo floatip.sh up
# gs_om -t start
```

说明：如果 standby 节点不能正常拉起，人工也不能拉起，也请检查其 postgresql.conf 文件中是否存在 floatip。如果存在请先删除，然后再拉起进程。

6.5.GGHC 启动失败，提示 failed at querying cluster listen ip

6.4.3. 问题描述

GGHC 无法启动 floatip，检查 system.log 日志发现报错 Get SSH Client failed at querying cluster listen ip。

详细 log 信息如下：

GGHC 启动报错，提示：

```
2021-07-20 15:20:34,295 - CRITICAL:[checker.py(77)] gets ssh connection for node 1 failed, it needs manual support
2021-07-20 15:20:34,295 - CRITICAL:[sshclient.py(51)] Traceback (most recent call last):
File "/home/gghc/gghc/sshclient.py", line 42, in connect
key = paramiko.RSAKey.from_private_key_file(self.private_key_file)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 206, in from_private_key_file
key = cls(filename=filename, password=password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 48, in init
self._from_private_key_file(filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 169, in _from_private_key_file
data = self._read_private_key_file('RSA', filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 279, in _read_private_key_file
data = self._read_private_key(tag, f, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 289, in _read_private_key
raise SSHException('not a valid ' + tag + ' private key file')
paramiko.ssh_exception.SSHException: not a valid RSA private key file

2021-07-20 15:20:34,295 - CRITICAL:[checker.py(77)] gets ssh connection for node 2 failed, it needs manual support
2021-07-20 15:20:34,295 - CRITICAL:[sshclient.py(51)] Traceback (most recent call last):
File "/home/gghc/gghc/sshclient.py", line 42, in connect
key = paramiko.RSAKey.from_private_key_file(self.private_key_file)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 206, in from_private_key_file
key = cls(filename=filename, password=password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 48, in init
self._from_private_key_file(filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 169, in
```

```
_from_private_key_file
data = self._read_private_key_file('RSA', filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 279, in _read_private_key_file
data = self._read_private_key(tag, f, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 289, in _read_private_key
raise SSHException('not a valid ' + tag + ' private key file')
paramiko.ssh_exception.SSHException: not a valid RSA private key file

2021-07-20 15:20:34,295 - CRITICAL:[checker.py(77)] gets ssh connection for node 3 failed, it needs manual support
2021-07-20 15:20:34,295 - INFO:[checker.py(88)] finished to init ssh clients, the result is: [False, False, False]
2021-07-20 15:20:34,296 - CRITICAL:[sshclient.py(51)] Traceback (most recent call last):
File "/home/gghc/gghc/sshclient.py", line 42, in connect
key = paramiko.RSAKey.from_private_key_file(self.private_key_file)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 206, in from_private_key_file
key = cls(filename=filename, password=password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 48, in init
self._from_private_key_file(filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/rsakey.py", line 169, in _from_private_key_file
data = self._read_private_key_file('RSA', filename, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 279, in _read_private_key_file
data = self._read_private_key(tag, f, password)
File "/usr/lib/python3.7/site-packages/paramiko/pkey.py", line 289, in _read_private_key
raise SSHException('not a valid ' + tag + ' private key file')
paramiko.ssh_exception.SSHException: not a valid RSA private key file

2021-07-20 15:20:34,296 - CRITICAL:[checker.py(77)] gets ssh connection for node 1 failed, it needs manual support
2021-07-20 15:20:34,296 - ERROR:[checker.py(110)] Get SSH Client failed at querying cluster listen ip
```

6.4.4. 解放方法

分析日志可以发现，问题原因如下：

问题应该在这里：

```
raise SSHException('not a valid ' + tag + ' private key file')
paramiko.ssh_exception.SSHException: not a valid RSA private key file
```

也就是说，是 GGHC 上生成的 ssh 密钥文件有问题。

原因是 openssh 产生的 RSA 文件以"BEGIN OPENSSH PRIVATE KEY"开始，而 paramiko 需要的是以"BEGIN RSA PRIVATE KEY"开始。

改用如下方式产生 ssh RSA 密钥对文件即可：

```
$ ssh-keygen -m PEM -t rsa
```