

华为认证 Computing 系列教程

HCIP-Computing

计算集群技术实验

指导手册

版本：2.5



华为技术有限公司

版权所有 ©【2026】【华为技术有限公司】。保留一切权利。

未经华为事先书面许可，任何人不得对本资料进行复制、修改、整合、改编、翻译，也不得将本资料或其任何部分或基于本资料的衍生作品提供、出售、转让、再许可给任何他人用于华为许可以外的任何其他用途。

Copyright ©【2026】【Huawei Technologies Co., Ltd.】. All Rights Reserved.

Without prior written permission of Huawei, no one is allowed to reproduce, modify, integrate, adapt, translate this Document, or provide, sell, transfer, sublicense this Document or any part of this Document or its derivative to any third party for any other purpose otherwise permitted by Huawei.

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://e.huawei.com>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，华为公司打造了覆盖ICT领域的认证体系，包含ICT基础设施、应用与开发两类技术认证及项目管理认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

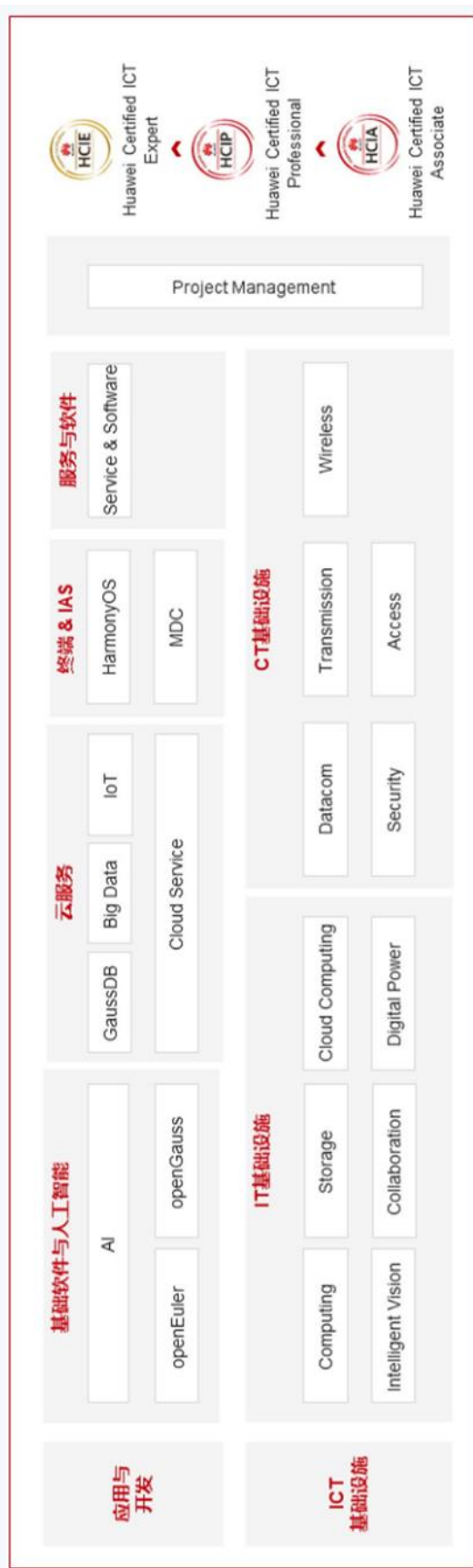
华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-Computing（Huawei Certified ICT Professional-Computing，华为认证计算高级工程师）主要面向希望学习华为计算产品的技术人士。HCIP-Computing认证在内容上涵盖数据中心介绍、计算集群技术、高性能计算和人工智能计算集群解决方案、计算设备日常维护和故障处理等内容。

通过HCIP-Computing V2.5认证，您将了解集群技术产生的背景、当前主要的集群技术以及高性能计算集群、人工智能计算集群对于算存网的典型需求，掌握通用算力集群建设方法，具备熟练进行计算产品维护能力，胜任计算产品运维高级工程师、计算产品售后服务高级工程师等岗位。

华为职业认证全景图

华为职业认证覆盖ICT全领域，致力于提供领先的人才培养体系和认证标准，培养数字化时代的新型ICT人才，构建良性的ICT人才生态。



前言

简介

本书为 HCIP-Computing 认证培训教程，适用于准备参加 HCIP-Computing 考试或者希望了解如何使用开源高可用软件、负载均衡软件搭建集群的学员，以及希望掌握如何在集群中使用开源软件进行高可用和负载均衡配置相关知识的读者。

内容描述

本实验指导书共包含 5 个实验任务，包含如何在 Linux 系统中安装和配置集群控制软件，如 LVS、Nginx、HAProxy 和 Keepalived 等。

- 实验一为 LVS 的安装和配置实验，实验包含配置 LVS 中的 NAT 模式和 DR 模式对集群应用进行负载均衡操作。
- 实验二为 Nginx 反向代理和负载均衡实验，实验包含反向代理配置以及负载均衡不同算法配置和验证。
- 实验三为 HAProxy 配置实验，实验包含如何配置 HAProxy 实现负载均衡。
- 实验四为 Keepalived 高可用实验，实验包含如何配置 Keepalived 对集群应用进行高可用保护。
- 实验五为 Keepalived+LVS 高可用实验，实验通过配置 Keepalived+LVS 实现 Nginx 集群。

读者知识背景

本课程为华为认证基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 具有 Linux 操作系统基础，能熟练使用 Linux 命令行。
- 熟悉 Linux 系统文本编辑功能。
- 掌握集群开源高可用和开源负载均衡软件基础知识。

实验环境说明

本实验环境面向准备 HCIP-Computing 考试的工程师。每套实验环境包括 5 台虚拟机，以下实验内容以华为云弹性服务器 ECS 为例，ECS 购买参考文档：

https://support.huaweicloud.com/qs-ecs/ecs_01_0103.html。如果链接失效，请直接访问华

为云官网 <https://www.huaweicloud.com/>，搜索“自定义购买和使用 Linux ECS”，参考指导文档购买 ECS。备注：只完成本实验，使用按需计费模式费用更低，无需包年/包月。

每套设备在同一个 VPC（本文档中示例网段：192.168.100.0/24）下，每台弹性云服务器规格：

- 规格类型：通用计算型
- 规格名称：x1.2u.4g | 2vCPUs | 4GiB
- 系统盘：通用型 SSD 40GiB
- 操作系统：openEuler 24.03 64bit
- 弹性公网 IP（按流量计费）

逐台购买服务器时，按照如下主机名称和 IP 地址对应关系配置主机名称和手动分配 IP：

主机名称	IP地址	数量
Client	192.168.100.62/24	1台
LVS	192.168.100.61/24	1台
LVS2	192.168.100.66/24	1台
Nginx001	192.168.100.63/24	1台
Nginx002	192.168.100.64/24	1台

目录

前 言	4
简介	4
内容描述	4
读者知识背景	4
实验环境说明	4
1 LVS 集群配置实践	8
1.1 实验场景介绍	8
1.2 实验组网介绍	8
1.3 实验步骤	9
1.3.1 Nginx001 和 Nginx002 配置	9
1.3.2 LVS 服务器创建与配置	10
1.3.3 登录 Client 虚拟机进行验证	11
1.4 思考题	12
2 Nginx 反向代理和负载均衡实践	13
2.1 实验场景介绍	13
2.2 实验组网介绍	13
2.3 实验步骤	13
2.3.1 配置代理服务器	13
2.3.2 在 Nginx 服务器中配置虚拟主机	15
2.3.3 代理服务器修改反向代理配置	18
2.3.4 IP Hash 算法配置实践	19
2.3.5 Generic Hash 算法实践	19
2.3.6 Random 算法实践	21
2.3.7 算法选项配置实践	22
2.4 思考题	24
3 HAProxy 负载均衡实践	25
3.1 实验场景介绍	25
3.2 实验组网介绍	25
3.3 实验步骤	25
3.4 思考题	27

4 Keepalived 实现 Nginx 高可用集群	28
4.1 实验场景介绍	28
4.2 实验组网介绍	28
4.3 实验步骤	28
4.4 思考题	34
5 Keepalived+LVS 高可用集群实践	36
5.1 实验场景介绍	36
5.2 实验组网介绍	36
5.3 实验步骤	36
5.4 思考题	40
6 缩略语	41

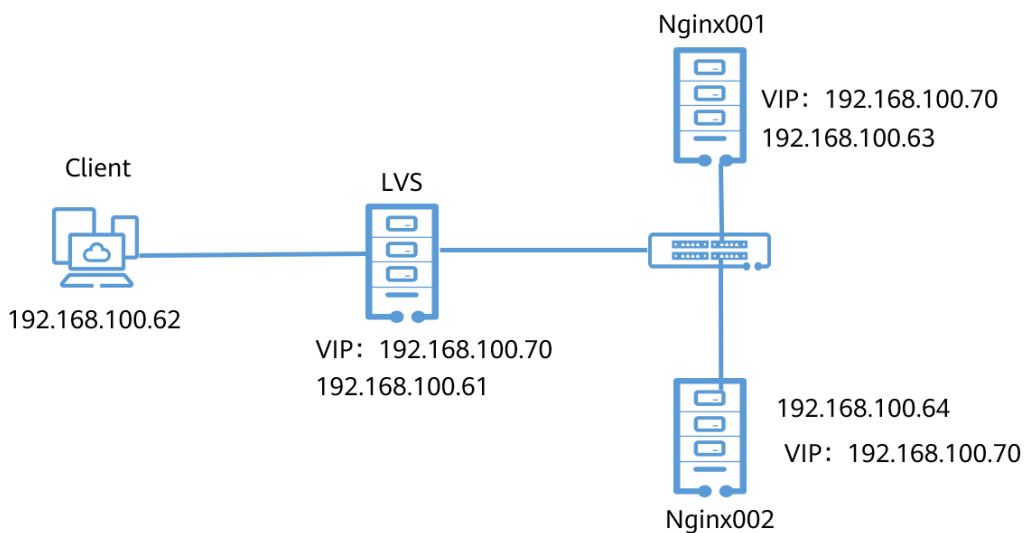
1 LVS 集群配置实践

1.1 实验场景介绍

本实验通过使用 LVS 软件来实现一个集群中 Nginx 服务的负载均衡，在实际工作场景中，集群对外提供服务时所显示的内容是一致的，但为了体现 LVS 负载均衡的算法效果，此处会将两个服务器中的 Nginx 服务显示不同内容，如 Nginx1 显示内容为 “This is 192.168.100.63”，Nginx2 显示内容为 “This is 192.168.100.64”，LVS 使用 DR 模式进行负载均衡。

1.2 实验组网介绍

本实验的设备在同一 VPC 下，内部网络互通，集群内包含的 4 个节点，分别为 Nginx001、Nginx002、LVS 以及 Client。在 DR 模式下，Nginx 服务器和 LVS 服务器 dummy 类型的接口用于承载 VIP。



1.3 实验步骤

1.3.1 Nginx001 和 Nginx002 配置

步骤 1 登录两台 Nginx 服务器，安装 Nginx，命令如下：

```
yum install nginx -y
```

```
Installed:
  gd-2.3.3-4.oe2403.x86_64
  libXpm-3.5.17-2.oe2403.x86_64
  libwebp-1.3.2-3.oe2403.x86_64
  nginx-1:1.24.0-8.oe2403.x86_64
  nginx-filesystem-1:1.24.0-8.oe2403.noarch
  nginx-mod-http-perl-1:1.24.0-8.oe2403.x86_64
  nginx-mod-mail-1:1.24.0-8.oe2403.x86_64
  gperftools-libs-2.15-2.oe2403.x86_64
  libunwind-2:1.7.2-3.oe2403.x86_64
  libxslt-1.1.39-7.oe2403.x86_64
  nginx-all-modules-1:1.24.0-8.oe2403.noarch
  nginx-mod-http-image-filter-1:1.24.0-8.oe2403.x86_64
  nginx-mod-http-xslt-filter-1:1.24.0-8.oe2403.x86_64
  nginx-mod-stream-1:1.24.0-8.oe2403.x86_64

Complete!
```

两台服务器运行 Nginx 服务：

```
systemctl start nginx
systemctl enable nginx
```

```
[root@Nginx001 ~]# systemctl start nginx
[root@Nginx001 ~]# systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[root@Nginx001 ~]#
```

步骤 2 增加 VIP 相关配置，具体命令如下：

```
nmcli con add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 192.168.100.70/32
```

```
[root@Nginx001 ~]# nmcli con add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 192.168.100.70/32
Connection 'dummy-dummy2' (4c1e83a1-4b83-471a-a384-b978648932eb) successfully added.
[root@Nginx001 ~]#
```

修改 arp 内核配置，具体命令如下：

```
cat >>/etc/sysctl.conf <<EOF
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.dummy2.arp_ignore = 1
net.ipv4.conf.dummy2.arp_announce = 2
EOF
```

```
[root@Nginx001 ~]# cat >>/etc/sysctl.conf <<EOF
> net.ipv4.conf.all.arp_ignore = 1
> net.ipv4.conf.all.arp_announce = 2
> net.ipv4.conf.dummy2.arp_ignore = 1
> net.ipv4.conf.dummy2.arp_announce = 2
> EOF
[root@Nginx001 ~]#
```

配置完成后，使用 `sysctl -p` 命令使其生效。

```
sysctl -p | tail -4
```

```
[root@Nginx001 ~]# sysctl -p | tail -4
sysctl: cannot stat /proc/sys/kernel/unknown_nmi_panic: No such file or directory
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.dummy2.arp_ignore = 1
net.ipv4.conf.dummy2.arp_announce = 2
```

完成后重启两台 Nginx 虚拟机，使网络配置生效。

步骤 3 为区分不同 Nginx 服务器的访问内容，修改/usr/share/nginx/html 路径下的 index.html 文件内容，按照组网规划信息配置访问信息。具体命令如下：

Nginx001:

```
mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.bak
echo "This is 192.168.100.63" > /usr/share/nginx/html/index.html
```

```
[root@Nginx001 ~]# mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.bak
[root@Nginx001 ~]# echo "This is 192.168.100.63" > /usr/share/nginx/html/index.html
[root@Nginx001 ~]#
```

Nginx002:

```
mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.bak
echo "This is 192.168.100.64" > /usr/share/nginx/html/index.html
```

```
[root@nginx002 ~]# mv /usr/share/nginx/html/index.html /usr/share/nginx/html/index.bak
echo "This is 192.168.100.64" > /usr/share/nginx/html/index.html
[root@nginx002 ~]#
```

步骤 4 在任意一台虚拟机上使用 curl 命令测试配置结果，命令如下：

```
curl 192.168.100.63
curl 192.168.100.64
```

```
[root@Nginx001 ~]# curl 192.168.100.63
This is 192.168.100.63
[root@Nginx001 ~]# curl 192.168.100.64
This is 192.168.100.64
[root@Nginx001 ~]#
```

1.3.2 LVS 服务器创建与配置

步骤 1 远程登录 LVS 服务器，安装 ipvsadm，并其中添加 VIP 相关配置，具体命令如下：

```
yum install ipvsadm -y
nmcli con add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 192.168.100.70/32
```

```
[root@LVS ~]# nmcli con add type dummy ifname dummy2 ipv4.method manual ipv4.addresses 192.168.100.70/32
Connection 'dummy-dummy2' (dae36d06-82c6-4c7e-994e-2cc433f3c5a7) successfully added.
[root@LVS ~]#
```

步骤 2 运行 ipvsadm，创建 ipvsadm 启动时所需的配置文件：

```
touch /etc/sysconfig/ipvsadm
systemctl enable ipvsadm --now
systemctl start ipvsadm
```

```
systemctl status ipvsadm
```

```
[root@LVS ~]# touch /etc/sysconfig/ipvsadm
[root@LVS ~]# systemctl enable ipvsadm --now
Created symlink /etc/systemd/system/multi-user.target.wants/ipvsadm.service → /usr/lib/systemd/system/ipvsadm.service.
[root@LVS ~]# systemctl start ipvsadm
[root@LVS ~]# systemctl status ipvsadm
● ipvsadm.service - Initialise the Linux Virtual Server
   Loaded: loaded (/usr/lib/systemd/system/ipvsadm.service; enabled; vendor preset: disabled)
   Active: active (exited) since Tue 2024-08-20 18:00:39 CST; 23s ago
     Process: 32735 ExecStart=/bin/bash -c exec /sbin/ipvsadm-restore < /etc/sysconfig/ipvsadm (code=exited, status=0/SUCCESS)
    Main PID: 32735 (code=exited, status=0/SUCCESS)

Aug20 18:00:39 LVS systemd[1]: Starting Initialise the Linux Virtual Server...
Aug20 18:00:39 LVS systemd[1]: Started Initialise the Linux Virtual Server.
[root@LVS ~]# ^C
[root@LVS ~]#
```

步骤 3 启动成功后，通过添加以下配置命令，使得 LVS 通过 RR 轮询调度算法实现负载均衡。

```
ipvsadm -A -t 192.168.100.70:80 -s rr
ipvsadm -a -t 192.168.100.70:80 -r 192.168.100.63
ipvsadm -a -t 192.168.100.70:80 -r 192.168.100.64
```

```
[root@LVS ~]# ipvsadm -A -t 192.168.100.70:80 -s rr
[root@LVS ~]# ipvsadm -a -t 192.168.100.70:80 -r 192.168.100.63
[root@LVS ~]# ipvsadm -a -t 192.168.100.70:80 -r 192.168.100.64
[root@LVS ~]#
```

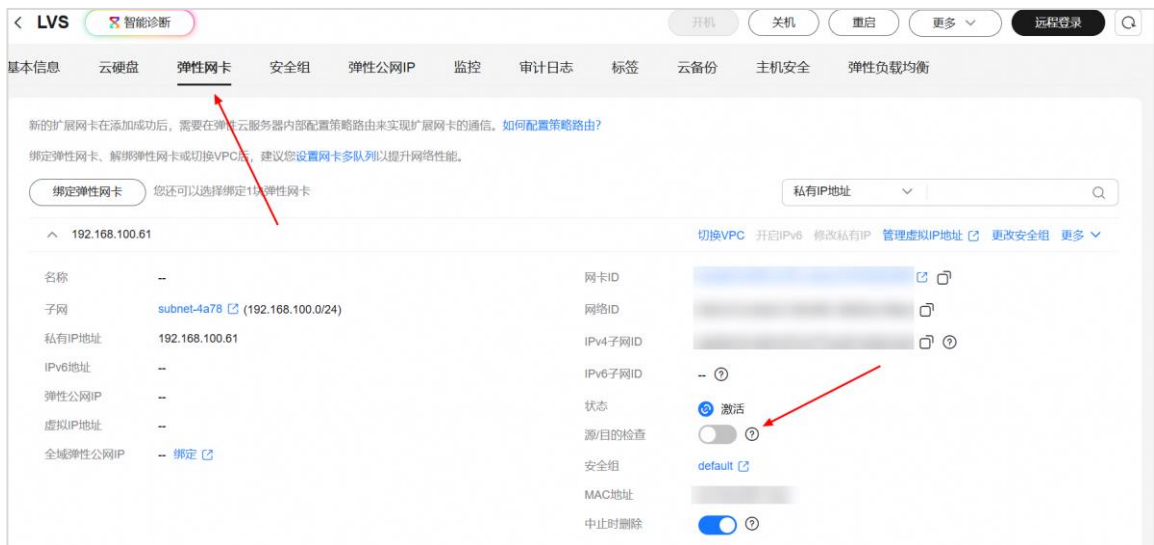
```
ipvsadm -Ln
```

```
[root@LVS ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP   192.168.100.70:80 rr
      -> 192.168.100.63:80          Route    1         0         0
      -> 192.168.100.64:80          Route    1         0         0
[root@LVS ~]#
```

1.3.3 登录 Client 虚拟机进行验证

步骤 1 关闭 LVS、两台 Nginx 服务器的源/目的检查

登录华为云弹性云服务器控制台，分别进入 LVS、LVS2、Nginx001、Nginx002 四台服务器的配置页，选择弹性网卡页签，点击 IP 地址展开配置，关闭“源/目的检查”。



步骤 2 登录 Client 虚拟机，多次通过 LVS VIP 地址访问 web 服务，查看访问结果。

```
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.63
```

从结果可以看出，LVS 服务器以轮询方式分别向 Nginx1 和 Nginx2 转发请求，并将页面返回给客户端。

1.4 思考题

问题：LVS 的轮询调度算法，会考虑服务器性能差异吗？

答案：不会，轮询调度算法按照固定的顺序轮流分配给每台服务器，不考虑服务器性能差异。

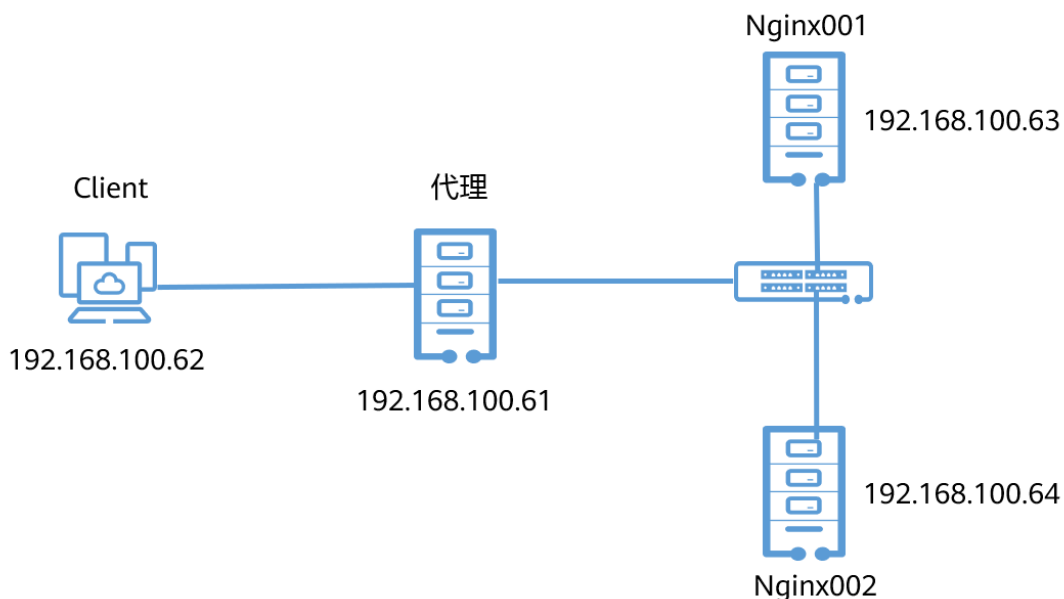
2 Nginx 反向代理和负载均衡实践

2.1 实验场景介绍

本实验通过设置 Nginx 的多种不同的算法来实现后端服务器的工作负载，直观验证不同算法下访问流量的负载分担情况。

2.2 实验组网介绍

Nginx001、Nginx002 和 Client、LVS 服务器复用。具体实验拓扑如下：



2.3 实验步骤

2.3.1 配置代理服务器

步骤 1 登录原先的 LVS 服务器修改为代理服务器，需要将原有 LVS 负载均衡关闭。通过以下命令将前面实验配置的规则进行删除，具体删除命令如下：

```
ipvsadm -D -t 192.168.100.70:80
```

删除完成后，查看规则内容中是否还存在规则：

```
ipvsadm -Ln
```

```
[root@LVS conf.d]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
[root@LVS conf.d]#
```

步骤 2 此时使用 Client 验证 LVS 是否已停止工作

```
[root@Client ~]# curl 192.168.100.70
curl: (7) Failed to connect to 192.168.100.70 port 80: Connection refused
[root@Client ~]# curl 192.168.100.70
curl: (7) Failed to connect to 192.168.100.70 port 80: Connection refused
[root@Client ~]#
```

步骤 3 在代理服务器虚拟机（LVS 虚拟机）中，启动 Nginx 服务

```
yum install nginx -y
systemctl enable nginx
systemctl start nginx
systemctl status nginx
```

```
[root@LVS ~]# systemctl enable nginx
[root@LVS ~]# systemctl start nginx
[root@LVS ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 10:00:00 CST; 8s ago
     Process: 45750 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 45752 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 45754 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
    Main PID: 45755 (nginx)
      Tasks: 3
     Memory: 11.8M
    CGroup: /system.slice/nginx.service
            └─45755 nginx: master process /usr/sbin/nginx
              └─45756 nginx: worker process
                └─45757 nginx: worker process
```

步骤 4 在代理服务器（LVS 虚拟机）的 Nginx 子配置文件目录中，创建代理和负载均衡配置文件 vhost.conf，具体路径为/etc/nginx/conf.d/，并在 vhost.conf 文件下添加如下配置：

```
vim /etc/nginx/conf.d/vhost.conf

upstream 192.168.100.61 {
    server 192.168.100.63:80;
    server 192.168.100.64:80;
}

server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

```
upstream 192.168.100.61 {
    server 192.168.100.63:80;
    server 192.168.100.64:80;
}
server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 5 通过 Client 服务器验证 Nginx 代理服务器功能，如果配置正确，验证结果如下：

```
curl 192.168.100.61:81
```

```
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
```

2.3.2 在 Nginx 服务器中配置虚拟主机

步骤 1 为突出后续负载均衡算法的结果验证，需要分别在两台 Nginx 服务器中添加虚拟主机。首先在 Nginx001 服务器上的/etc/nginx/conf.d/目录中创建一个 static.conf 文件，在文件中添加配置，具体配置如下：

```
cd /etc/nginx/conf.d/
vim static.conf
```

```
[root@Nginx001 ~]# cd /etc/nginx/conf.d/
[root@Nginx001 conf.d]# vim static.conf
```

```
server {
    listen 0.0.0.0:81;
    root /data/Nginx/;
    server_name localhost;
    index index81.html;
}
server {
    listen 0.0.0.0:82;
    root /data/Nginx/;
```

```
server_name localhost;
index index82.html;
}
server{
    listen 0.0.0.0:83;
    root /data/Nginx/;
    server_name localhost;
    index index83.html;
}
```

```
server {
    listen 0.0.0.0:81;
    root /data/Nginx/;
    server_name localhost;
    index index81.html;
}
server {
    listen 0.0.0.0:82;
    root /data/Nginx/;
    server_name localhost;
    index index82.html;
}
server {
    listen 0.0.0.0:83;
    root /data/Nginx/;
    server_name localhost;
    index index83.html;
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 2 在 Nginx002 服务器上重复步骤 1 中的配置，完成 Nginx002 的配置修改

```
cd /etc/nginx/conf.d/
vim static.conf

server {
    listen 0.0.0.0:81;
    root /data/Nginx/;
    server_name localhost;
    index index81.html;
}
server {
    listen 0.0.0.0:82;
    root /data/Nginx/;
    server_name localhost;
    index index82.html;
}
server{
    listen 0.0.0.0:83;
    root /data/Nginx/;
```

```
server_name localhost;
index index83.html;
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 3 创建/data/Nginx/目录，并在其中创建 index81.html、index82.html、index83.html 文件，分别添加不同内容以分别不同端口，如 “hello, {IP}: {端口}”，具体命令如下：

Nginx001:

```
mkdir -p /data/Nginx/
echo "hello,192.168.100.63:81" > /data/Nginx/index81.html
echo "hello,192.168.100.63:82" > /data/Nginx/index82.html
echo "hello,192.168.100.63:83" > /data/Nginx/index83.html
```

```
[root@Nginx001 ~]# mkdir -p /data/Nginx/
[root@Nginx001 ~]# echo "hello,192.168.100.63:81" > /data/Nginx/index81.html
[root@Nginx001 ~]# echo "hello,192.168.100.63:82" > /data/Nginx/index82.html
[root@Nginx001 ~]# echo "hello,192.168.100.63:83" > /data/Nginx/index83.html
[root@Nginx001 ~]#
```

Nginx002:

```
mkdir -p /data/Nginx/
echo "hello,192.168.100.64:81" > /data/Nginx/index81.html
echo "hello,192.168.100.64:82" > /data/Nginx/index82.html
echo "hello,192.168.100.64:83" > /data/Nginx/index83.html
```

```
[root@Nginx ~]# mkdir -p /data/Nginx/
[root@Nginx ~]# echo "hello,192.168.100.64:81" > /data/Nginx/index81.html
[root@Nginx ~]# echo "hello,192.168.100.64:82" > /data/Nginx/index82.html
[root@Nginx ~]# echo "hello,192.168.100.64:83" > /data/Nginx/index83.html
[root@Nginx ~]#
```

步骤 4 完成配置以后，可以通过 Client 虚拟机验证两个 Nginx 服务器的服务端口内容是否正确：

```
[root@Client ~]# curl 192.168.100.63:81
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.63:82
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.63:83
hello,192.168.100.63:83
[root@Client ~]# curl 192.168.100.64:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.64:82
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.64:83
hello,192.168.100.64:83
```

2.3.3 代理服务器修改反向代理配置

步骤 1 在代理服务器（LVS 虚拟机）中的配置文件 vhost.conf 中修改配置，添加虚拟主机到后端主机组中，具体配置如下：

```
vim /etc/nginx/conf.d/vhost.conf

upstream 192.168.100.61 {
    server 192.168.100.63:80;
    server 192.168.100.64:80;
    server 192.168.100.63:81;
    server 192.168.100.63:82;
    server 192.168.100.63:83;
    server 192.168.100.64:81;
    server 192.168.100.64:82;
    server 192.168.100.64:83;
}

server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 2 在 Client 虚拟机验证此时结果如下：

```
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:83
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:83
```

【注】此时通过 curl 命令访问代理 IP 时，代理服务器是按照顺序进行负载均衡的。

2.3.4 IP Hash 算法配置实践

步骤 1 将代理服务器中的 vhost.conf 配置文件内容进行修改，修改算法为 IP Hash，具体如下：

```
upstream 192.168.100.61 {
    ip_hash;
    server 192.168.100.63:80;
    server 192.168.100.64:80;
    server 192.168.100.63:81;
    server 192.168.100.63:82;
    server 192.168.100.63:83;
    server 192.168.100.64:81;
    server 192.168.100.64:82;
    server 192.168.100.64:83;
}
server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 2 算法配置完成重新加载完 Nginx 配置以后，在 Client 端测试，现象如下：

```
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
```

【注】ip_hash 是根据用户请求过来的 ip，然后映射成 hash 值，然后分配到一个特定的服务器里面，可以保证用户的每一次会话都只会发送到同一台服务器上。

2.3.5 Generic Hash 算法实践

步骤 1 在两台 Nginx 服务器的/usr/share/nginx/html 目录创建测试文件 test.txt，命令如下：

Nginx001:

```
echo "192.168.100.63 Generic Hash practice" > /usr/share/nginx/html/test.txt
```

Nginx002:

```
echo "192.168.100.64 Generic Hash practice" > /usr/share/nginx/html/test.txt
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

```
[root@Nginx001 html]# echo "192.168.100.63 Generic Hash practice" > test.txt
[root@Nginx001 html]# cat test.txt
192.168.100.63 Generic Hash practice
[root@Nginx001 html]# nginx -s reload

[root@Nginx html]# echo "192.168.100.64 Generic Hash practice" > test.txt
[root@Nginx html]# cat test.txt
192.168.100.64 Generic Hash practice
[root@Nginx html]# nginx -s reload
```

步骤 2 为体现对比，将代理服务器中的算法调整为默认，完成配置以后，使用命令重新加载 Nginx 服务。

```
upstream 192.168.100.61 {
    server 192.168.100.63:80;
    server 192.168.100.64:80;
    server 192.168.100.63:81;
    server 192.168.100.63:82;
    server 192.168.100.63:83;
    server 192.168.100.64:81;
    server 192.168.100.64:82;
    server 192.168.100.64:83;
}
server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

```
nginx -s reload
```

步骤 3 Client 端多次使用如下命令查看访问结果：

```
curl 192.168.100.61:81/test.txt
```

```
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.64 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.16.1</center>
</body>
```

步骤 4 将代理服务器中的 vhost.conf 配置文件内容进行修改，修改算法为 Generic Hash，具体如下：

```
upstream 192.168.100.61 {
    hash $request_uri;
    server 192.168.100.63:80;
    server 192.168.100.64:80;
}

server {
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 5 Client 端使用上面的命令验证访问结果：

```
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
[root@Client ~]# curl 192.168.100.61:81/test.txt
192.168.100.63 Generic Hash practice
```

2.3.6 Random 算法实践

步骤 1 将代理服务器中的 vhost.conf 配置文件内容进行修改，修改算法为 Random，具体如下：

```
upstream 192.168.100.61 {
    random two;
    server 192.168.100.63:80;
    server 192.168.100.64:80;
    server 192.168.100.63:81;
    server 192.168.100.63:82;
    server 192.168.100.63:83;
    server 192.168.100.64:81;
    server 192.168.100.64:82;
    server 192.168.100.64:83;
}

server {
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 2 Client 端使用如下的命令验证访问结果：

```
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:83
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:81
```

2.3.7 算法选项配置实践

步骤 1 将代理服务器中的 vhost.conf 配置文件内容进行修改，将 Nginx002 的全部虚拟机设置修改为 backup，同时将 Nginx 上的虚拟机设置一定的权重值，具体配置如下：

```
upstream 192.168.100.61 {
    server 192.168.100.63:80 weight=2;
    server 192.168.100.63:81 weight=2;
    server 192.168.100.63:82 weight=3;
    server 192.168.100.63:83;
    server 192.168.100.64:80 backup;
    server 192.168.100.64:81 backup;
    server 192.168.100.64:82 backup;
    server 192.168.100.64:83 backup;
}
server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 2 Client 端使用如下的命令验证访问结果：

```
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.63:83
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.63
```

步骤 3 手动停止 Nginx001 服务器上的 Nginx 服务，然后再去 Client 端使用命令查看访问结果：

```
systemctl stop nginx
systemctl status nginx
```

```
[root@Nginx001 html]# systemctl stop nginx
[root@Nginx001 html]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Wed 2020-08-26 10:00:00 CST; 6s ago
```

```
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:83
[root@Client ~]# curl 192.168.100.61:81
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:82
```

步骤 4 将代理服务器中的 vhost.conf 配置文件内容进行修改，把 Nginx002 中部分虚拟机修改为“down”，具体如下所示：

```
upstream 192.168.100.61 {
    server 192.168.100.63:80 weight=2;
    server 192.168.100.63:81 weight=2;
    server 192.168.100.63:82 weight=3;
    server 192.168.100.63:83;
    server 192.168.100.64:80 down;
    server 192.168.100.64:81 down;
    server 192.168.100.64:82 backup;
    server 192.168.100.64:83 backup;
}
server{
    listen 81;
    server_name localhost;
    location / {
        proxy_pass http://192.168.100.61;
    }
}
```

完成配置以后，使用命令重新加载 Nginx 服务，具体命令如下：

```
nginx -s reload
```

步骤 5 在 Client 虚拟机上通过命令查看此时所访问的结果：

```
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:83
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.61:81
hello,192.168.100.64:83
```

步骤 6 为保证后续实验正常进行，将前面手动停止的 Nginx001 上的服务重新开启

```
systemctl start nginx
```

2.4 思考题

问题：Nginx 默认的负载均衡调度算法是什么？

答案：轮询调度算法。

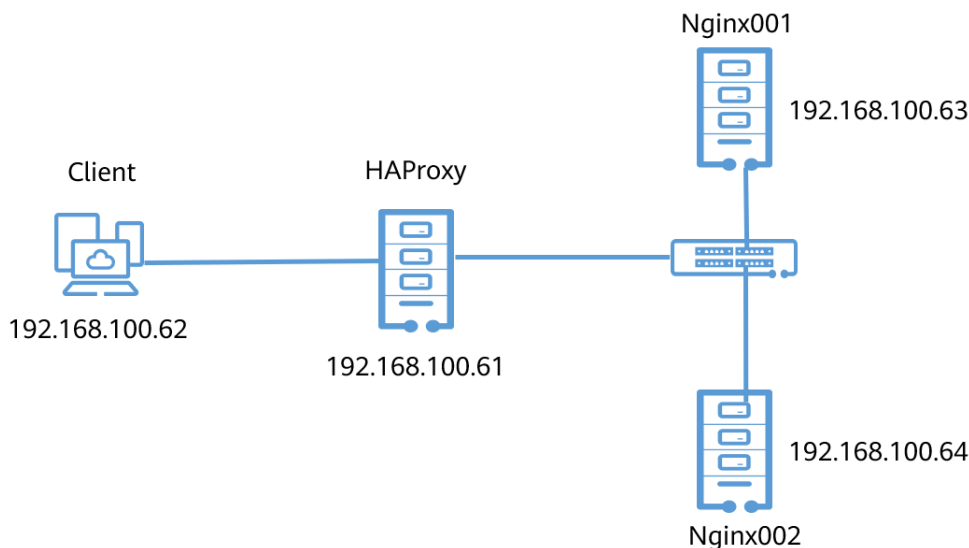
3 HAProxy 负载均衡实践

3.1 实验场景介绍

本实验通过配置 HAProxy 实现高可用功能，从而加强对 HAProxy 所有功能的理解。

3.2 实验组网介绍

Nginx001、Nginx002 和 Client 服务器可以复用，HAProxy 沿用 LVS。具体实验拓扑如下：



3.3 实验步骤

步骤 1 将 LVS 服务器中的 Nginx 代理服务关闭，然后验证代理访问 IP 无法访问到 Nginx 服务器。在后面的实验步骤中，此服务器名称都以 HAProxy 指代。

```
systemctl stop nginx
systemctl status nginx
```

```
[root@LVS ~]# systemctl stop nginx
[root@LVS ~]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Wed 2023-08-23 19:19:19 CST; 19min ago
     Process: 123836 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 123838 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 123840 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
    Main PID: 123841 (code=exited, status=0/SUCCESS)
```

Client 虚拟机侧测试结果如下:

```
[root@Client ~]# curl 192.168.100.61
curl: (7) Failed to connect to 192.168.100.61 port 80: Connection refused
[root@Client ~]# curl 192.168.100.61
curl: (7) Failed to connect to 192.168.100.61 port 80: Connection refused
[root@Client ~]# curl 192.168.100.61:81
curl: (7) Failed to connect to 192.168.100.61 port 81: Connection refused
[root@Client ~]# curl 192.168.100.61:81
curl: (7) Failed to connect to 192.168.100.61 port 81: Connection refused
```

步骤 2 在 HAProxy 服务器中安装 haproxy

```
yum install haproxy -y
```

步骤 3 修改 HAProxy 服务器中配置文件/etc/haproxy/haproxy.cfg, 在修改前建议先备份此文件。本实验只实现 HAProxy 基础的负载均衡功能, 此处仅修改文件中 backend 部分信息, 将其替换为 Nginx001 和 Nginx002 服务器地址即可, 具体配置如下:

```
cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.bak
vim /etc/haproxy/haproxy.cfg
```

```
frontend main
    bind *:80
    default_backend http_back

backend http_back
    balance roundrobin
    server node1 192.168.100.63:81 check
    server node2 192.168.100.63:82 check
    server node3 192.168.100.63:83 check
    server node4 192.168.100.64:81 check
    server node5 192.168.100.64:82 check
    server node6 192.168.100.64:83 check
```

配置完成后, 使用如下命令启动 HAProxy 服务:

```
systemctl start haproxy
systemctl status haproxy
```

```
[root@LVS haproxy]# systemctl start haproxy
[root@LVS haproxy]# systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/usr/lib/systemd/system/haproxy.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-23 19:20:00 CST; 2s ago
     Process: 143892 ExecStartPre=/usr/sbin/haproxy -f $HAPROXY_CONF -c -q (code=exited, status=0/SUCCESS)
    Main PID: 143894 (haproxy)
       Tasks: 3
      Memory: 6.5M
      CGroup: /system.slice/haproxy.service
              └─143894 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
                └─143896 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
```

启动后，在客户端访问 HAProxy 服务器 IP 地址，如果配置正确，访问结果如下所示：

```
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.64:83
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.61
hello,192.168.100.63:83
```

3.4 思考题

问题：HAProxy 默认的负载均衡调度算法是什么？

答案：动态加权轮询。

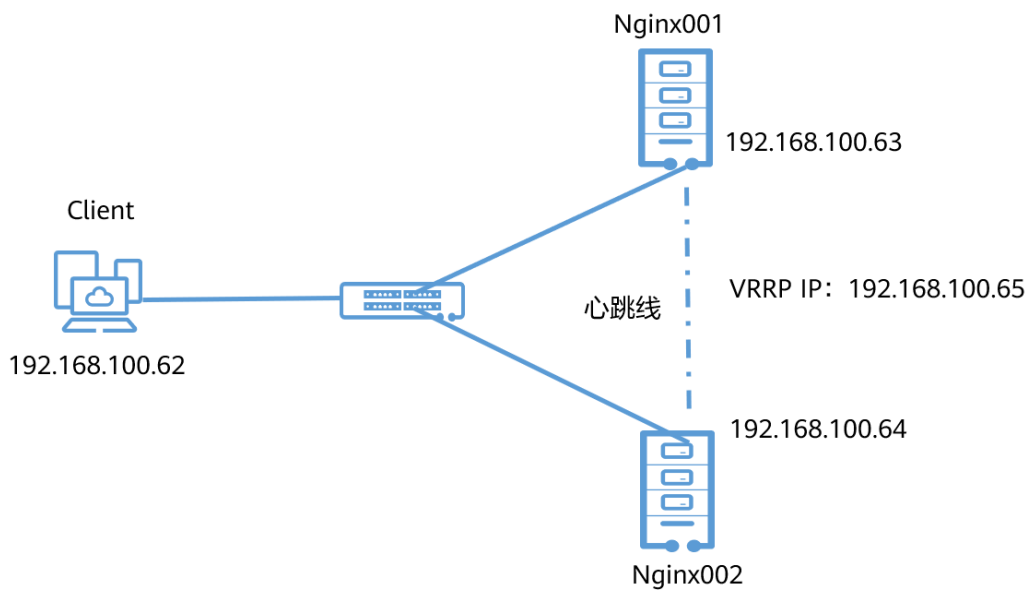
4 Keepalived 实现 Nginx 高可用集群

4.1 实验场景介绍

本实验是通过 keepalived 来实现 Nginx 集群的高可用功能，实现集群中 Nginx 主服务器发生故障以后，可以自动切换到备用服务器上。

4.2 实验组网介绍

Nginx001、Nginx002 和 Client 服务器均可以复用前面实验中的虚拟机， Nginx001 和 Nginx002 之间启动 keepalived 服务，通过浮动 IP 192.168.100.65 对外提供服务，如果其中主服务器发生故障或者 Nginx 服务进程出现故障，业务实现自动切换，具体拓扑如下：



4.3 实验步骤

步骤 1 在两台 Nginx 服务器中安装 keepalived 软件

```
yum install keepalived -y
```

步骤 2 本次实验中将以 Nginx001 设置为主节点，Nginx002 设置为备份节点。在 Nginx001 上创建 keepalived 配置文件（/etc/keepalived/keepalived.conf），具体内容如下：

```
cd /etc/keepalived/  
vim keepalived.conf
```

新建 keepalived.conf 添加内容如下：

```
! Configuration File for keepalived  
  
global_defs {  
    router_id Nginx1  
}  
  
vrrp_instance Nginx {  
    state MASTER  
    interface eth0  
    virtual_router_id 51  
    priority 225  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass hci@123  
    }  
    virtual_ipaddress {  
        192.168.100.65/24  
    }  
}
```

在 Nginx002 上同样新建 keepalived 配置文件（/etc/keepalived/keepalived.conf），具体添加内容如下：

```
! Configuration File for keepalived  
  
global_defs {  
    router_id Nginx2  
}  
  
vrrp_instance Nginx {  
    state BACKUP  
    interface eth0  
    virtual_router_id 51  
    priority 100  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass hci@123  
    }  
    virtual_ipaddress {  
        192.168.100.65/24  
    }  
}
```

```
}
}
```

配置完成以后，分别在 Nginx001 和 Nginx002 服务器上启动 keepalived 服务：

```
systemctl start keepalived
```

步骤 3 查看 Nginx001 上的 IP 信息，可以发现浮动 IP 已经绑定到 eth0 接口上

```
[root@Nginx001 keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:88:eb:13 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.63/24 brd 192.168.100.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet 192.168.100.65/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2a6e:d4ff:fe88:eb13/64 scope link
        valid_lft forever preferred_lft forever
3: dummy2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 92:ca:4a:e5:cd:f4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.70/32 brd 192.168.100.70 scope global noprefixroute dummy2
        valid_lft forever preferred_lft forever
    inet6 fe80::545f:96c4:f810:5247/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

步骤 4 使用 Client 虚拟机进行 Nginx 服务高可用功能测试，访问浮动 IP 查看返回结果：

```
[root@Client ~]# curl 192.168.100.65
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.65
This is 192.168.100.63
[root@Client ~]# curl 192.168.100.65:81
hello,192.168.100.63:81
[root@Client ~]# curl 192.168.100.65:82
hello,192.168.100.63:82
[root@Client ~]# curl 192.168.100.65:83
hello,192.168.100.63:83
```

手动关闭 Nginx001 服务器模拟主服务器宕机，等待几分钟后，查看 Nginx002 上的 IP 信息，看浮动 IP 是否绑定到 eth0 接口上。

```
[root@Nginx keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:88:eb:21 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.64/24 brd 192.168.100.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet 192.168.100.65/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2a6e:d4ff:fe88:eb21/64 scope link
        valid_lft forever preferred_lft forever
3: dummy2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 2e:12:36:08:03:fd brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.70/32 brd 192.168.100.70 scope global noprefixroute dummy2
        valid_lft forever preferred_lft forever
    inet6 fe80::393:4f25:4f46:ef64/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

再次使用 Client 虚拟机进行 Nginx 服务高可用功能测试，访问浮动 IP 查看返回结果：

```
[root@Client ~]# curl 192.168.100.65
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.65:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.65:82
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.65:83
hello,192.168.100.64:83
[root@Client ~]#
```

启动 Nginx001 服务器，恢复业务。

- 步骤 5 前面的实验中 Keepalived 仅能通过主机是否宕机来进行业务切换，如果仅 Nginx 业务出现故障无法切换，需在 Keepalive 中添加健康检查来达到检测 Nginx 是否可用的目的。分别在 Nginx1 和 Nginx2 的 Keepalived 配置文件中添加健康检查的相关配置，具体如下：

```
vim /etc/keepalived/keepalived.conf

vrrp_script nginx_check {
    script "/etc/keepalived/check.sh"
    interval 5
    weight -5
    fail 3
}

vrrp_instance Nginx {
    .....
    track_script {
        nginx_check
    }
}
```

<pre>! Configuration File for keepalived global_defs { router_id Nginx1 } vrrp_script nginx_check { script "/etc/keepalived/check.sh" interval 5 weight -5 fail 3 } vrrp_instance Nginx { state MASTER interface eth0 virtual_router_id 52 priority 225 advert_int 1 authentication { auth_type PASS auth_pass hcip@123 } virtual_ipaddress { 192.168.100.65/24 } track_script { nginx_check } }</pre>	<pre>! Configuration File for keepalived global_defs { router_id Nginx2 } vrrp_script nginx_check { script "/etc/keepalived/check.sh" interval 20 weight -5 fail 3 } vrrp_instance Nginx { state BACKUP interface eth0 virtual_router_id 52 priority 100 advert_int 1 authentication { auth_type PASS auth_pass hcip@123 } virtual_ipaddress { 192.168.100.65/24 } track_script { nginx_check } }</pre>
--	---

步骤 6 在两个 Nginx 服务器上的/etc/keepalived 的目录下创建用于检测 Nginx 服务是否正常的执行脚本 check.sh，具体内容如下：

```
cd /etc/keepalived
vim check.sh

#!/bin/bash
systemctl status nginx|grep "active (running)"> /dev/null
if [ $? -ne 0 ]; then
    systemctl restart nginx &> /dev/null
    sleep 1
    systemctl status nginx|grep "active (running)" > /dev/null
    if [ $? -ne 0 ]; then
        systemctl stop keepalived
    else
        exit
    fi
fi
```

```
#!/bin/bash
systemctl status nginx|grep "active (running)"> /dev/null
if [ $? -ne 0 ]; then
    systemctl restart nginx &> /dev/null
    sleep 1
    systemctl status nginx|grep "active (running)" > /dev/null
    if [ $? -ne 0 ]; then
        systemctl stop keepalived
    else
        exit
    fi
fi
```

步骤 7 为脚本 check.sh 赋权

```
chmod 777 check.sh
```

步骤 8 在 Nginx001 和 Nginx002 节点中重启 keepalived 服务，使配置生效

```
systemctl restart keepalived
```

步骤 9 通过将 Nginx001 服务器中的 nginx.conf 文件（/etc/nginx/目录下）重命名为 nginx.bak，从而使其不能正常启动，然后手动停止 nginx 服务

```
mv /etc/nginx/nginx.conf /etc/nginx/nginx.bak
systemctl stop nginx
```

此时重新启动 nginx 服务会提示报错，模拟 nginx 服务不可用的情况。

```
systemctl start nginx
```

```
[root@Nginx001 keepalived]# systemctl start nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

通过命令查看 keepalived 服务的状态，发现 keepalived 已经停止，IP 已经迁移。

```
systemctl status keepalived
```

```
[root@Nginx keepalived]# systemctl status keepalived
● keepalived.service - LVS and VRRP High Availability Monitor
   Loaded: loaded (/usr/lib/systemd/system/keepalived.service; disabled; vendor preset: disabled)
   Active: inactive (dead)

Sep 09 09:22:28 Nginx keepalived_vrrp[3300281]: (Nginx) removing VIPs.
Sep 09 09:22:28 Nginx keepalived_vrrp[3300281]: (Nginx) Entering BACKUP STATE (init)
Sep 09 09:22:28 Nginx keepalived_vrrp[3300281]: VRRP sockpool: [ifindex(2), family(IPv4), proto(112), unicast(0), fd(11,12)]
Sep 09 09:22:28 Nginx keepalived_vrrp[3300281]: (Nginx) received a 1 auth, expecting 0!
Sep 09 09:22:28 Nginx systemd[1]: Stopping LVS and VRRP High Availability Monitor...
Sep 09 09:22:28 Nginx keepalived[3300280]: Stopping
Sep 09 09:22:28 Nginx keepalived_vrrp[3300281]: Stopped - used 0.000000 user time, 0.001692 system time
Sep 09 09:22:28 Nginx keepalived[3300280]: Stopped Keepalived v2.0.20 (01/22,2020)
Sep 09 09:22:28 Nginx systemd[1]: keepalived.service: Succeeded.
Sep 09 09:22:28 Nginx systemd[1]: Stopped LVS and VRRP High Availability Monitor.
[root@Nginx keepalived]# vim keepalived.conf
```

步骤 10 查看 Nginx002 上的 IP 信息，看浮动 IP 是否绑定到 eth0 接口上

```
[root@Nginx keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:88:eb:21 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.64/24 brd 192.168.100.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet 192.168.100.65/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2a6e:d4ff:fe88:eb21/64 scope link
        valid_lft forever preferred_lft forever
3: dummy2: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 2e:12:36:08:03:fd brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.70/32 brd 192.168.100.70 scope global noprefixroute dummy2
        valid_lft forever preferred_lft forever
    inet6 fe80::393:4f25:4f46:ef64/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

使用 Client 虚拟机进行 Nginx 服务高可用功能测试，访问浮动 IP 查看返回结果：

```
[root@Client ~]# curl 192.168.100.65
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.65:81
hello,192.168.100.64:81
[root@Client ~]# curl 192.168.100.65:82
hello,192.168.100.64:82
[root@Client ~]# curl 192.168.100.65:83
hello,192.168.100.64:83
```

步骤 11 恢复 Nginx001 服务器中的文件，然后重启 nginx 服务，恢复业务

```
mv /etc/nginx/nginx.bak /etc/nginx/nginx.conf
systemctl start nginx
systemctl status nginx
```

```
[root@Nginx001 keepalived]# systemctl start nginx
[root@Nginx001 keepalived]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-08-02 10:00:00 CST; 8s ago
     Process: 18532 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
     Process: 18534 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
     Process: 18536 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
    Main PID: 18537 (nginx)
       Tasks: 3
      Memory: 11.8M
      CGroup: /system.slice/nginx.service
              └─18537 nginx: master process /usr/sbin/nginx
                  └─18538 nginx: worker process
                     18539 nginx: worker process
```

步骤 12 为方便下面实验进行，实验完成后分别在两台 Nginx 服务器上关闭 keepalived 服务。具体命令如下：

```
systemctl stop keepalived
```

4.4 思考题

问题：Keepalived 默认的模式为抢占模式还是非抢占模式？

答案：抢占模式。

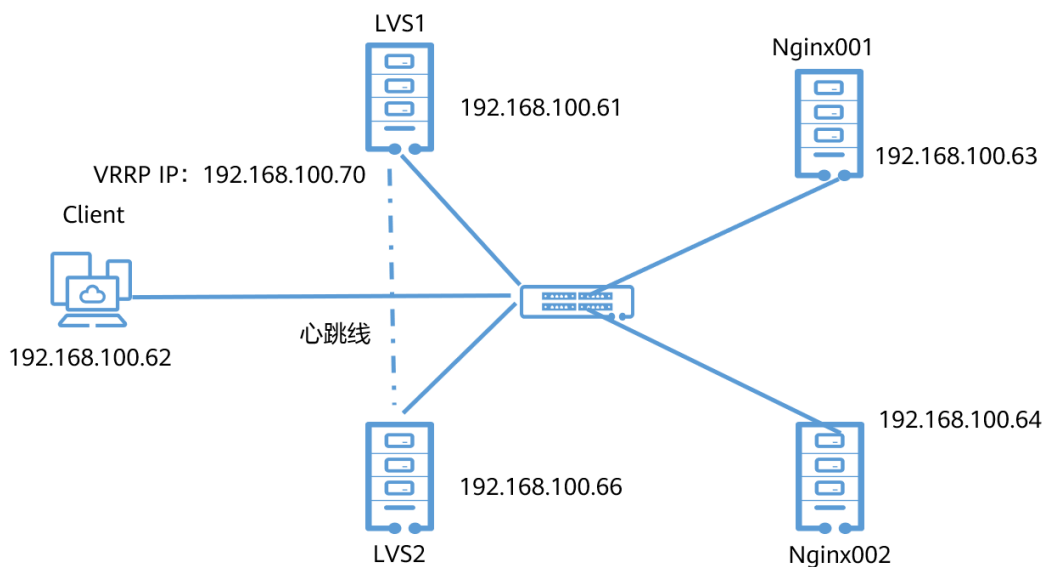
5 Keepalived+LVS 高可用集群实践

5.1 实验场景介绍

本实验将使用 Keepalived 为 LVS 提供高可用配置，同时 LVS 为后端的 Nginx1 和 Nginx2 提供负载均衡，最终通过 Keepalived+LVS 实现高可用、负载均衡的 Nginx 集群。

5.2 实验组网介绍

Nginx001、Nginx002 和 Client 服务器可以复用（Nginx 服务器中的 keepalived 服务需要关闭）。LVS1 沿用 LVS，具体组网拓扑如下：



5.3 实验步骤

步骤 1 LVS1 和 LVS2 上安装 Keepalived 和 ipvsadm

```
yum install keepalived -y
yum install ipvsadm -y
```

在 LVS1 中清除前面实验用到的 VIP：

```
nmcli connection delete dummy-dummy2
```

步骤 2 修改 LVS1 服务器的配置文件（/etc/keepalived/keepalived.conf），修改内容如下所示：

```
vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global_defs {
    router_id Cluster1
}

vrrp_instance Nginx {
    state MASTER
    interface eth0
    mcast_src_ip 192.168.100.61
    virtual_router_id 52
    priority 255
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hci@123
    }
    virtual_ipaddress {
        192.168.100.70/24
    }
}

virtual_server 192.168.100.70 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
    protocol TCP

    real_server 192.168.100.63 80{
        weight 1
        TCP_CHECK{
            connect_timeout 3
            retry 3
            delay_before_retry 3
        }
    }
    real_server 192.168.100.64 80{
        weight 2
        TCP_CHECK{
            connect_timeout 3
            retry 3
        }
    }
}
```

```
        delay_before_retry 3
    }
}
}
```

修改完成后启动 keepalived 服务和 ipvsadm 服务，查看 LVS 相关配置信息。

```
systemctl start keepalived
systemctl enable keepalived
ipvsadm -Ln
```

```
[root@LVS ~]# systemctl start keepalived
[root@LVS ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.100.70:80 rr persistent 50
-> 192.168.100.63:80 Route 1 0 0
-> 192.168.100.64:80 Route 2 0 0
```

步骤 3 修改 LVS2 服务器的配置文件（/etc/keepalived/keepalived.conf），修改内容如下所示：

```
vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global_defs {
    router_id Cluster2
}

vrrp_instance Nginx {
    state BACKUP
    interface eth0
    mcast_src_ip 192.168.100.66
    virtual_router_id 52
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass hcip@123
    }
    virtual_ipaddress {
        192.168.100.70/24
    }
}

virtual_server 192.168.100.70 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    persistence_timeout 50
```

```
protocol TCP

real_server 192.168.100.63 80{
    weight 1
    TCP_CHECK{
        connect_timeout 3
        retry 3
        delay_before_retry 3
    }
}

real_server 192.168.100.64 80{
    weight 2
    TCP_CHECK{
        connect_timeout 3
        retry 3
        delay_before_retry 3
    }
}
}
```

修改完成后启动 keepalived 服务和 ipvsadm 服务，查看 LVS 相关配置信息。

```
systemctl start keepalived
systemctl enable keepalived
ipvsadm -Ln
```

```
[root@LVS2 ~]# systemctl start keepalived
[root@LVS2 ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.100.70:80 rr persistent 50
  -> 192.168.100.63:80            Route    1          0          0
  -> 192.168.100.64:80            Route    2          0          0
```

步骤 4 查看 LVS1 上的 IP 地址信息，可以查看到浮动 IP 地址已绑定

```
[root@LVS ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:88:e5:9f brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.61/24 brd 192.168.100.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet 192.168.100.70/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2a6e:d4ff:fe88:e59f/64 scope link
        valid_lft forever preferred_lft forever
```

步骤 5 在 Client 服务器上进行访问测试，访问 LVS 的浮动 IP 地址，结果如下：

```
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
```

步骤 6 将 LVS1 服务器关机，查看 LVS2 服务上的 IP 地址信息

```
[root@LVS2 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 28:6e:d4:88:e2:aa brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.66/24 brd 192.168.100.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet 192.168.100.70/24 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::2a6e:d4ff:fe88:e2aa/64 scope link
        valid_lft forever preferred_lft forever
```

此时说明集群中的高可用功能生效，故障后的浮动 IP 切换成功。

步骤 7 在 Client 服务器上重新进行访问测试，结果如下：

```
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
[root@Client ~]# curl 192.168.100.70
This is 192.168.100.64
```

此时说明集群浮动 IP 切换，且业务访问不受影响。

5.4 思考题

问题：浮动 IP 是否只能存在于一台设备上？

答案：是的，正常情况下浮动 IP（VIP）只会同时存在于一台设备上。

6 缩略语

缩略语	英文全称	解释
LVS	Linux Virtual Server	基于 IP 层和基于内容请求分发的负载均衡调度解决方法，在 Linux 内核中实现将一组服务器构成一个可伸缩的、高可用网络服务的虚拟服务器，即 Linux 虚拟服务器。
DR	Director Server	调度服务器
VIP	Virtual IP	虚拟 IP

华为认证 Computing 系列教程

HCIP-Computing

计算集群日常运维

实验指导手册

版本：2.5



华为技术有限公司

版权所有 ©【2026】【华为技术有限公司】。保留一切权利。

未经华为事先书面许可，任何人不得对本资料进行复制、修改、整合、改编、翻译，也不得将本资料或其任何部分或基于本资料的衍生作品提供、出售、转让、再许可给任何他人用于华为许可以外的任何其他用途。

Copyright ©【2026】【Huawei Technologies Co., Ltd.】. All Rights Reserved.

Without prior written permission of Huawei, no one is allowed to reproduce, modify, integrate, adapt, translate this Document, or provide, sell, transfer, sublicense this Document or any part of this Document or its derivative to any third party for any other purpose otherwise permitted by Huawei.

商标声明



和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编： 518129

网址： <https://e.huawei.com>

华为认证体系介绍

华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，华为公司打造了覆盖ICT领域的认证体系，包含ICT基础设施、应用与开发两类技术认证及项目管理认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

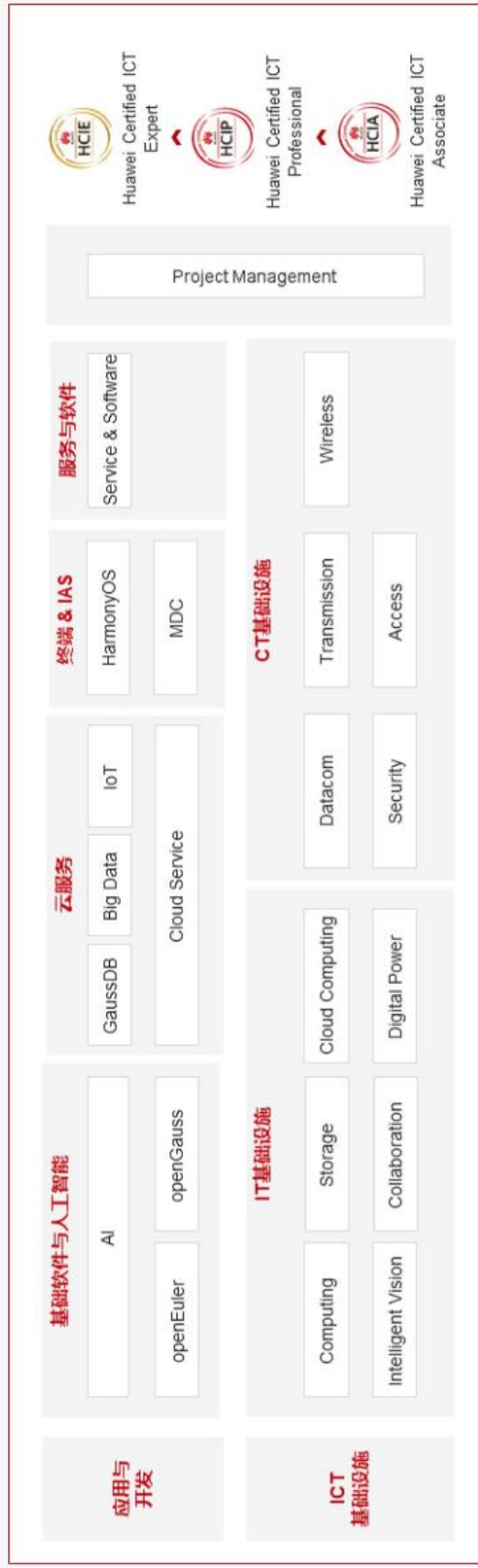
华为认证覆盖ICT全领域，符合ICT融合的技术趋势，致力于提供领先的人才培养体系和认证标准，培养数字化时代新型ICT人才，构建良性ICT人才生态。

HCIP-Computing（Huawei Certified ICT Professional-Computing，华为认证计算高级工程师）主要面向希望学习华为计算产品的技术人士。HCIP-Computing认证在内容上涵盖数据中心介绍、计算集群技术、高性能计算和人工智能计算集群解决方案、计算设备日常维护和故障处理等内容。

通过HCIP-Computing V2.5认证，您将了解集群技术产生的背景、当前主要的集群技术以及高性能计算集群、人工智能计算集群对于算存网的典型需求，掌握通用算力集群建设方法，具备熟练进行计算产品维护能力，胜任计算产品运维高级工程师、计算产品售后服务高级工程师等岗位。

华为职业认证全景图

华为职业认证覆盖ICT全领域，致力于提供领先的人才培养体系和认证标准，培养数字化时代的新型ICT人才，构建良性的ICT人才生态。



简介

本书为 HCIP-Computing 认证培训教程，适用于准备参加 HCIP-Computing 考试的学员或者希望了解服务器日常运维及自动化运维的读者。

内容描述

本实验指导书共包含两个实验：

- 实验一为使用 eComputing ToolKit 工具对服务器进行开局和日常运维管理；
- 实验二为使用 Ansible 工具对服务器进行自动化运维；

读者知识背景

本课程为华为认证基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 具有电脑操作的基础能力，理解硬件、系统和驱动的基本概念。
- 熟悉 Linux 基础命令。

实验设备介绍

设备名称、型号与版本的对应关系如下：

表1-1 实验设备配套关系

设备名称	设备型号	软件版本
TaiShan 200服务器	TaiShan 200 (Model 2280)	/
实验操作桌面	个人电脑	Windows

目录

前 言	3
简介	3
内容描述	3
读者知识背景	3
实验设备介绍	3
1 eComputing ToolKit 运维管理	6
1.1 实验介绍	6
1.1.1 关于本实验	6
1.1.2 实验目的	6
1.1.3 实验所需软件	6
1.2 实验任务配置	6
1.2.1 工具安装	6
1.2.2 设备管理	7
1.2.3 配置对比	9
1.2.4 日志收集	10
1.2.5 电源控制	11
1.2.6 健康检查	11
1.2.7 固件升级	12
1.2.8 一键开局	14
1.2.9 配置验证	17
1.3 思考题	18
2 Ansible 工具运维	19
2.1 实验介绍	19
2.1.1 关于本实验	19
2.1.2 实验目的	19
2.1.3 实验所需软件	19
2.2 实验任务配置	19
2.2.1 Ansible 安装部署	19
2.2.2 Ansible 常用模块命令	20
2.2.3 使用 Ansible 管理服务器	22
2.3 思考题	28



1 eComputing ToolKit 运维管理

1.1 实验介绍

1.1.1 关于本实验

本实验主要介绍使用 eComputing ToolKit 工具对服务器硬件进行日常运维的相关操作。

1.1.2 实验目的

- 掌握 eComputing ToolKit 工具的安装；
- 掌握使用 eComputing ToolKit 工具对服务器进行信息收集的操作方法。

1.1.3 实验所需软件

表1-1 实验所需软件

软件	下载地址
eComputing ToolKit 25.2.0	https://support.huawei.com/enterprise/zh/management-software/ecomputing-toolkit-pid-265916266/software
openEuler-24.03-LTS-SP2	https://repo.openeuler.openatom.cn/openEuler-24.03-LTS-SP2/ISO/aarch64/openEuler-24.03-LTS-SP2-aarch64-dvd.iso

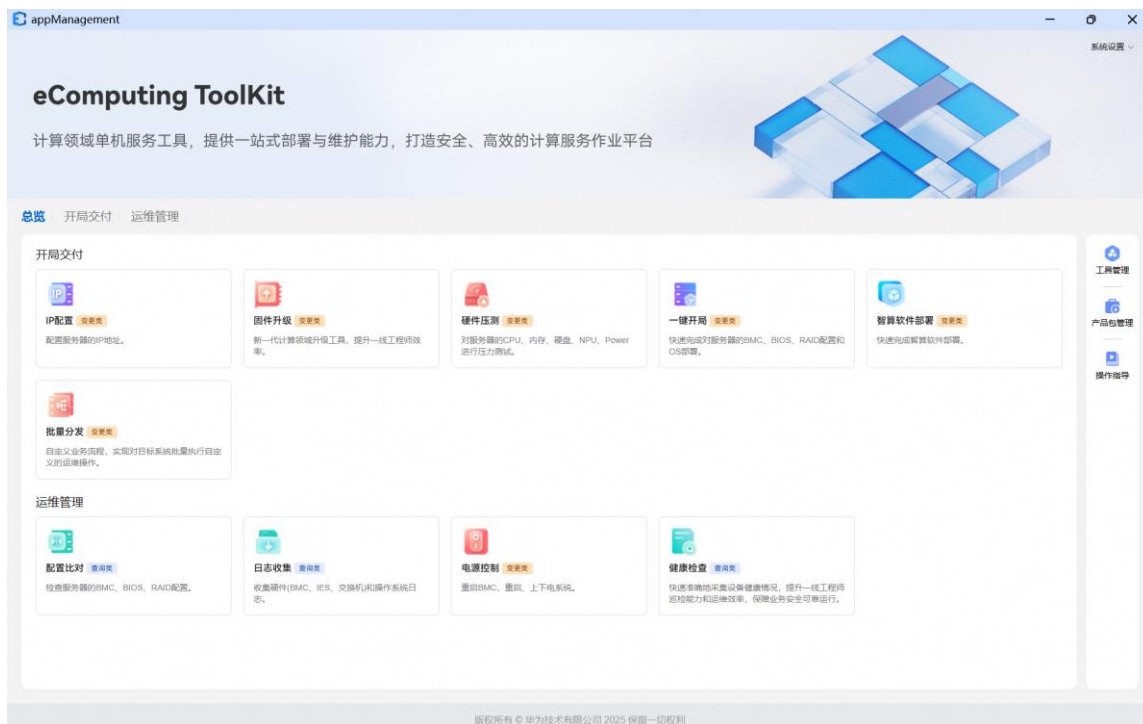
1.2 实验任务配置

1.2.1 工具安装

下载 eComputing_ToolKit_25.2.0_win64.zip 后解压至合法安装路径下，在解压目录下，双击 start.bat 文件，进入工具安装界面。填写数据保存路径等所需参数信息，点击“立即启动”，根据页面提示，选择扫码认证（推荐）或离线授权认证。

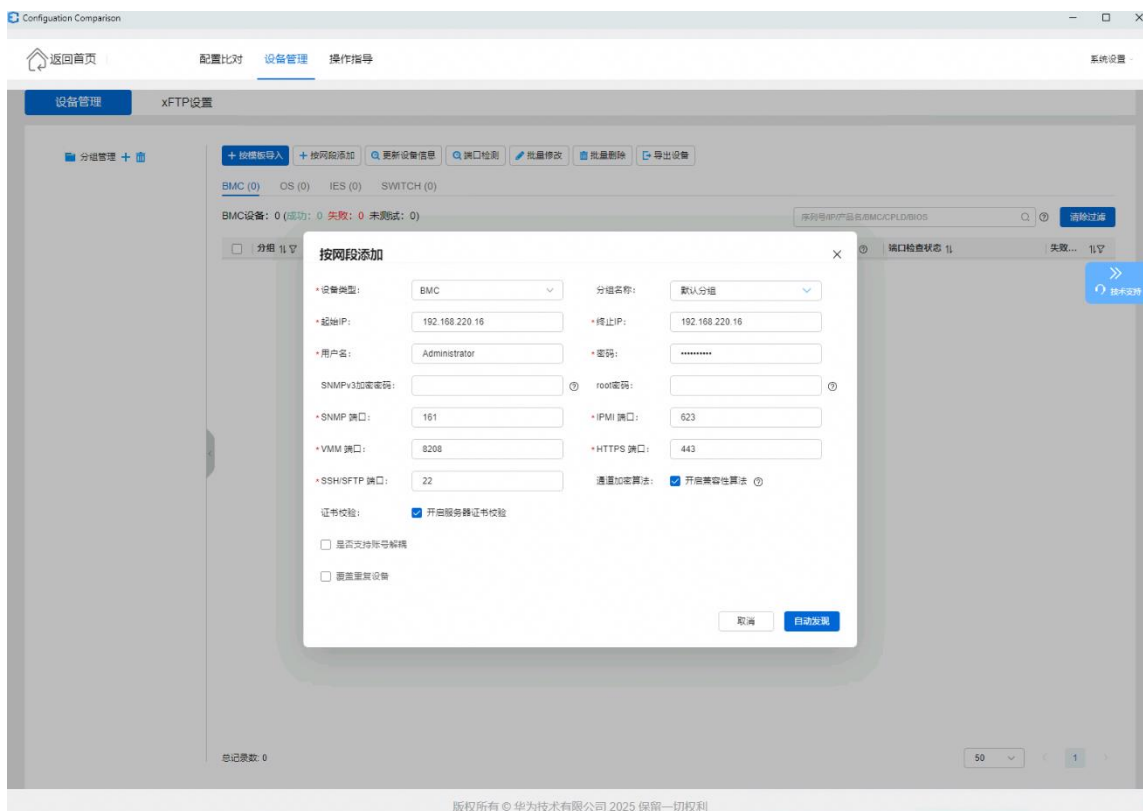
- 1、扫码认证：在登录页面输入 Uniportal 账号及密码，获取认证码。输入认证码，认证成功后自动进入工具首页。
 - 2、离线认证：无法扫码认证场景，使用离线授权认证。根据页面提示填写相关信息后，提交
-

申请，审批通过后，下载离线授权文件至本地，在工具中上传离线授权文件完成认证。



1.2.2 设备管理

打开 eComputing ToolKit，在运维管理界面，启动配置对比，选择设备管理页签，点击“按网段添加”按钮，设备类型选择 BMC，输入服务器 BMC 的 IP 地址和用户名密码，勾选“开启兼容性算法”，点击“自动发现”。



等待设备检查状态成功后，点击“完成”。



设备添加成功。



1.2.3 配置对比

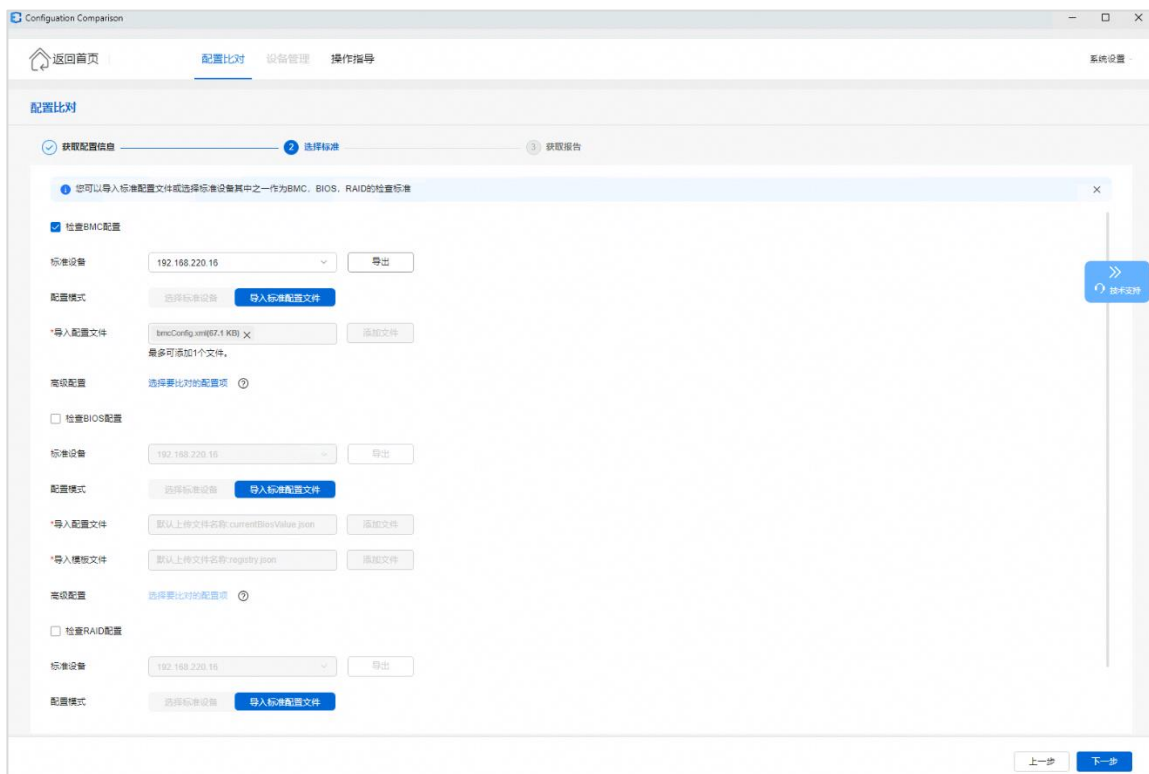
切换到配置对比页签，勾选添加的设备，点击“获取配置信息”，在提示框中点击确定，等待配置文件下载成功。



点击“下一步”，勾选需要检查的配置项，例如 BMC 配置，点击“导出”，打开导出后的配置文件，修改其中一个配置信息，例如将该配置的值由 1 修改为 0：

```
<Attribute Key="Snmp.1.0.0.3" Name="/Snmp/State">0</Attribute>
```

然后点击“添加文件”导入该配置文件，点击“下一步”。



查看配置信息对比汇总表，可以看到一致和不一致的配置数量。



点击“BMC 配置信息差异表”页签，点击设备 IP 左侧下拉列表按钮，查看配置差异明细。



点击页面右下角“完成”，并在弹出窗口点击“确定”，确认完成配置比对。

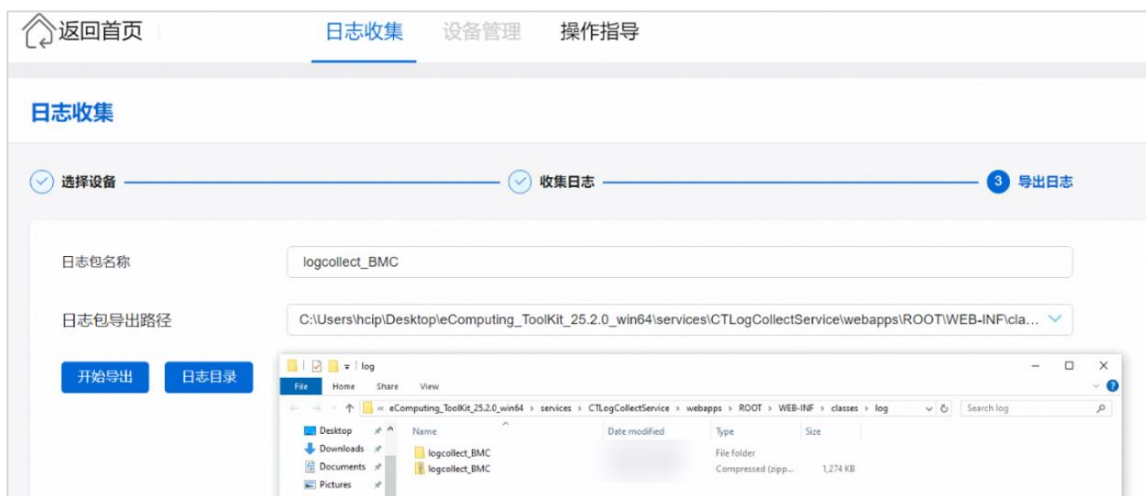
点击页面左上角“返回首页”，点击“离开”，回到 eComputing ToolKit 首页。

1.2.4 日志收集

在 eComputing ToolKit 首页运维管理界面中，启动日志收集，勾选 1.2.2 中已添加的设备，点击“下一步”，启动日志收集。



待收集日志成功后，勾选设备，点击“下一步”，点击“开始导出”，在提示导出成功后，点击“日志目录”，解压日志后可以查看日志。



点击页面右下角“完成”，并在弹出窗口点击“确定”，确认完成日志收集。

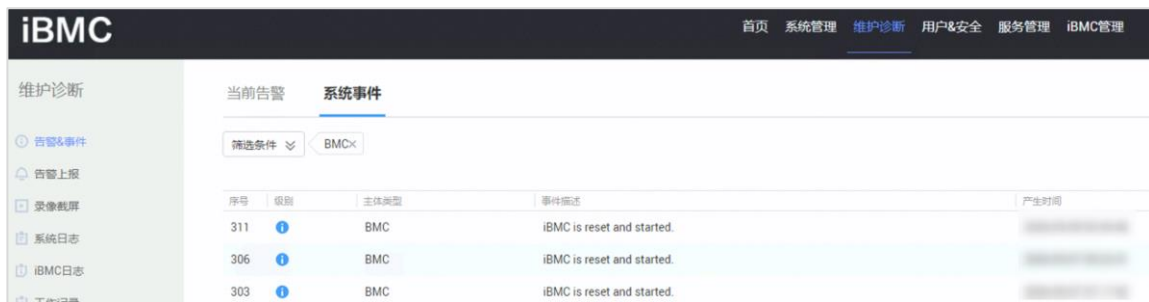
点击页面左上角“返回首页”，点击“离开”，回到 eComputing ToolKit 首页。

1.2.5 电源控制

在 eComputing ToolKit 首页运维管理界面中，启动电源控制，选择操作类型为 BMC 重启，勾选 1.2.2 中添加的设备，点击“电源控制”，在提示框中输入 yes，等待 BMC 重启成功。



登录服务器的 iBMC，在“维护诊断->告警&事件->系统事件”功能中，可以看到 BMC 的重启信息。



点击页面左上角“返回首页”，点击“离开”，回到 eComputing ToolKit 首页。

1.2.6 健康检查

在 eComputing ToolKit 首页运维管理界面中，启动健康检查，勾选 1.2.2 中添加的设备，点击“下一步”，选择场景为常规检查，点击“下一步”，在提示框中点击“确定”，等待健康检查完成。



点击“下一步”，可以生成采集结果和生成报告，将生成的报告解压后，打开 Index.html，可以看到健康检查报告。



点击页面右下角“完成”，并在弹出窗口点击“确定”，确认完成健康检查。

点击页面左上角“返回首页”，点击“离开”，回到 eComputing ToolKit 首页。

1.2.7 固件升级

在进行一键开局实验之前，需要为服务器升级匹配的固件，参考以下一键开局兼容性列表：

<https://support.huawei.com/enterprise/zh/doc/EDOC1100537091/36e40309?idPath=23710424|261608225|261608255|265916266>

一键开局实验以 TaiShan 2280 部署 openEuler-24.03-LTS-SP2 为例，根据兼容性列表，需要将固件升级为 1.20.0 及以上版本：

安装eComputing ToolKit 介绍eComputing ToolKit 使用eComputing ToolKit 配置 固件升级 固件压缩 一键开局 简介 兼容性列表 使用指导					(BMA 2.0 用户指南) 中的描述。 • Smart Provisioning版本为1.13.0及以上支持部署OS。
	openEuler 22.03 LTS SP4	SmartProvisioning	否		部署OS时，不支持同时安装BMA。如需安装BMA，请在OS部署完成后参见《BMA 2.0 用户指南》中的描述。 • Smart Provisioning版本为1.15.0及以上支持部署OS。
	openEuler 22.03 LTS	SmartProvisioning	否		部署OS时，不支持同时安装BMA。如需安装BMA，请在OS部署完成后参见《BMA 2.0 用户指南》中的描述。 • Smart Provisioning版本为1.2.2及以上支持部署OS。
	openEuler 20.03	SmartProvisioning	否		部署OS时，仅Smart Provisioning版本为1.9.0及以上支持同时安装BMA。
	openEuler 20.03 LTS SP4	SmartProvisioning	否		部署OS时，仅Smart Provisioning版本为1.18.0及以上支持同时安装BMA。
	openEuler 24.03 LTS SP1	SmartProvisioning	否		Smart Provisioning版本为1.18.0及以上支持部署OS。
	openEuler 24.03 LTS SP2	SmartProvisioning	否		Smart Provisioning版本为1.20.0及以上支持部署OS。

在 eComputing ToolKit 首页开局交付界面中，启动固件升级，勾选 1.2.2 中添加的设备，点击“查询设备信息”，等待查询成功，点击“下一步”。



点击“导入固件包”，添加文件，需要同时添加上传 SmartProvisioning_1.20.0_aarch64.zip 以及.asc 和.p7s 文件，等待上传成功。



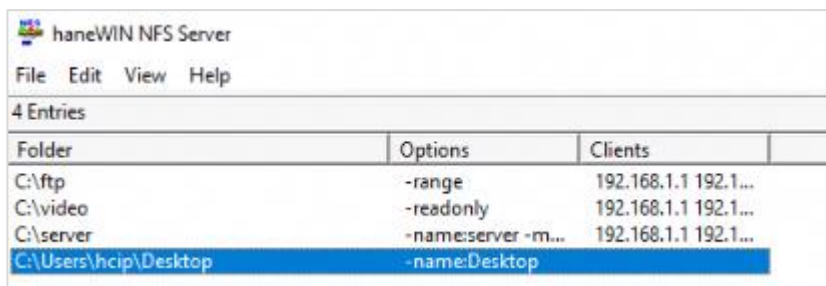
关闭“导入固件包”的弹窗，可以看到 Smart Provisioning 固件匹配成功，点击“下一步”。

Smart Provisioning 固件					
☒ 固件包	固件类型	对应设备上固件	当前版本	目标版本	匹配结果
☒ SmartProvisioning_1.20.0_aarch64.zip	SP	SP	1.21	1.20.0.19	匹配成功

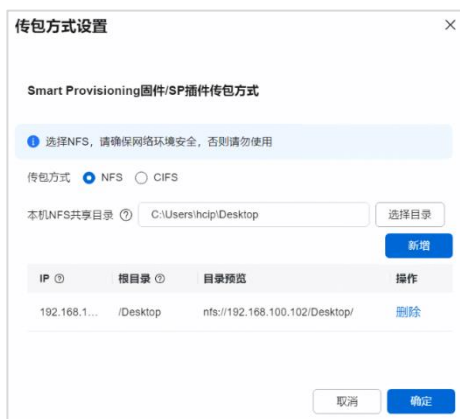
在本地 PC 部署 NFS 服务，并在本地 PC 验证 NFS 服务器是否可用，参考文档：

<https://support.huawei.com/enterprise/zh/doc/EDOC1100537091/61b31912?idPath=23710424|261608225|261608255|265916266>

启动 NFS Server 客户端，在空白处单击右键，选择“Insert”，勾选“-name: Visible name of folder on clients”，Folder 填入 SP 固件包的存放路径，点击“Save”。



在“传包方式设置”弹框中，选择创建的本机 NFS 共享目录，填写根目录，IP 地址为本机 IP，点击“确定”。



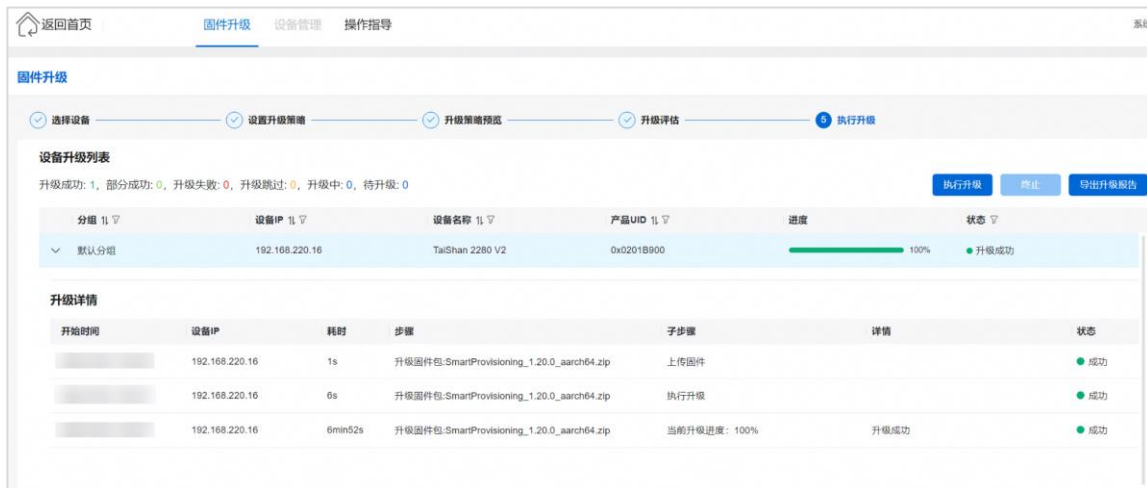
进入升级策略预览步骤，点击“下一步”。



进入升级评估步骤，点击“开始评估”，等待评估通过，点击“下一步”。



进入执行升级步骤，点击“执行升级”，在提示框中点击“确定”，等待升级完成。



*注：若在升级过程中，无当前升级进度直接失败，可能由于 BMC 版本过低，可以尝试升级 BMC。

在升级完成后，点击页面右下角“完成”，在弹出的提示框内，点击“确定”，完成固件升级。

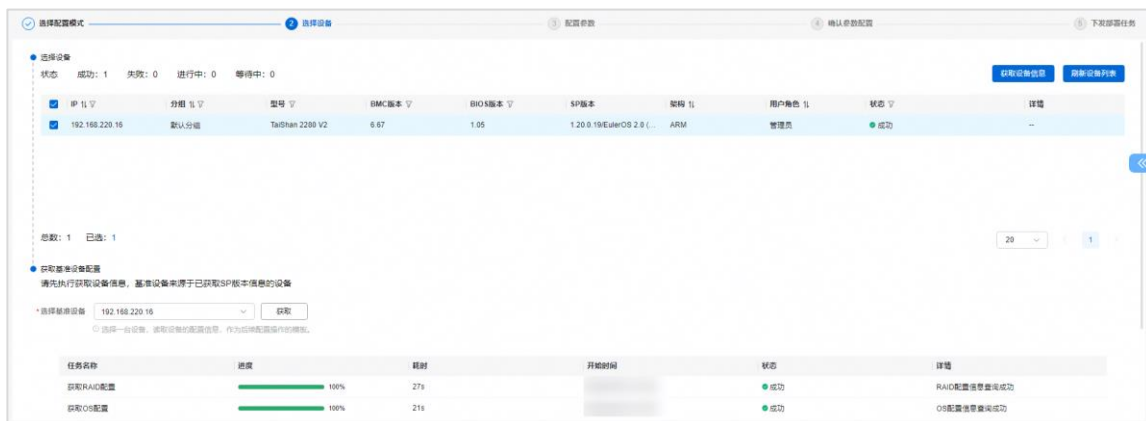
点击页面左上角“返回首页”，点击“离开”，回到 eComputing ToolKit 首页。

1.2.8 一键开局

在 eComputing ToolKit 首页开局交付界面中，启动一键开局，勾选“RAID 配置”“OS 部署”，点击“下一步”，在弹窗中输入“yes”，点击“确定”。

在选择设备界面勾选 1.2.2 中添加的设备，点击“获取设备信息”。

在获取设备信息状态变为“成功”后，在下方选择本设备为基准设备，并点击“获取”，等待 RAID 配置和 OS 配置信息查询成功，点击“下一步”。



在 RAID 配置界面，勾选“清除已有逻辑盘”，并在界面右侧点击“创建”。



在弹出的创建界面中，选择两块物理盘，级别选择“RAID1”，在高级配置中，将启动盘设置为“yes”，并点击“确定”。



初始化访问策略

UnInit

启动盘

yes

在 OS 部署界面，选择系统类型“openEuler”，系统版本“openEuler * aarch64”，iso 挂载方式选择“本地 HTTPS 挂载”，选择系统镜像文件 openEuler-24.03-LTS-SP2-aarch64-dvd，勾选“是否开启 HTTPS 兼容性算法”，点击“确定参数配置”。

RAID配置

OS部署

1. 官方渠道：定制化解非标准来源的镜像文件可能导致部署失败，建议使用从官方渠道下载的标准镜像；2. 部署系统时需要格式化待安装的操作系统的硬盘，若硬盘中有数据，请在安装操作前备份数据。

选择OS系统

系统类型

openEuler

系统版本

openEuler * aarch64

模板名

(NULL)

ISO挂载方式

本地HTTPS挂载

系统镜像文件

C:\Users\hnp\Desktop\New folder (2)\open

是否开启HTTPS兼容性算法

是

确定参数配置

设置OS参数

需先确定选择OS系统的参数配置

在“设置 OS 参数”界面，自定义设置管理员密码，设置时区为 Shanghai，点击配置文件下方的“下载模板”按钮，下载服务器部署专用参数模板。

设置OS参数

用户名

root

管理员密码

确认密码

时区

Asia/Shanghai

语言

English

键盘

us

BIOS启动模式

UEFI

驱动固件包

选择目录

配置文件

下载模板

选择文件

登录待开局服务器 iBMC 界面，在“系统管-系统信息-网络适配器”界面，找到规划的网口并记录 MAC 地址信息。

iBMC

首页 系统管理 维护诊断 用户&安全 服务管理 iBMC管理

系统管理

以太网卡

网卡设备

网卡名称

NIC 1 (TM210)

型号

4*GE

芯片厂商

Huawei

单端口

0x0068

厂商

Huawei

芯片型号

CPU Integration

PCB版本

A

固件白描

CPU1

总线信息

0000:7c:00:0

端口属性

端口	状态	端口类型	介质类型
Port1	连接	物理网口	电口

网络属性

固件版本	-
总线信息	0000:7c:00:0
MAC地址	00:06:5B:9F:66:0B
永久MAC地址	00:06:5B:9F:66:0B

返回服务器部署专用参数模板，在模板中，分别填写如下参数，以下参数均为示例，需根据实际实验环境自行修改：

参数列	参数名称	示例参数
D 列	HostName	HCIP

H 列	Device_Type	Ethernet
J 列	Device_MAC	上一步骤中获取的网口 MAC 地址，例如：D0:C6:5B:9F:66:0B
Q 列	IPv4_AddressOrigin	Static
R 列	IPv4_Address	192.168.101.16
S 列	IPv4_SubnetMask	255.255.255.0
T 列	IPv4_Gateway	192.168.101.1

在配置文件右侧，点击“选择文件”，选择上述步骤填写的参数表，完成配置文件上传。点击界面右下角“下一步”。



The screenshot shows the 'OS配置' (OS Configuration) tab in the RAID configuration tool. It includes sections for '选择OS系统' (Select OS System) and '设置OS参数' (Set OS Parameters). The '选择OS系统' section has dropdowns for '系统类型' (System Type) set to 'openEuler' and '系统版本' (System Version) set to 'openEuler 2.0.0-rc1'. The '设置OS参数' section includes fields for '用户名' (Username) set to 'root', '密码' (Password), '语言' (Language) set to 'English', '时区' (Time Zone) set to 'Asia/Shanghai', 'BIOS启动模式' (BIOS Boot Mode) set to 'UEFI', and '配置文件' (Configuration File) set to 'C:\OSDeployList_Template'. A red box highlights the '选择文件' (Select File) button next to the configuration file field.

在“确认参数配置”页面，确认参数配置无误后，点击“下一步”。

在“下发部署任务”界面，勾选 1.2.2 中添加的设备，点击“执行部署”，等待 OS 部署成功，部署过程需要 0.5~1 小时。



The screenshot shows the '设备列表' (Device List) interface. It displays a table of deployment tasks. The first task, 'Taishan 2280 V2', is highlighted in blue and shows a progress bar at 100%. Below the main table, there is a detailed log for this task, showing various steps like 'RAID配置' (RAID Configuration), '参数校验' (Parameter Check), '系统下电' (System Power Off), 'SP任务检查' (SP Task Check), '上传部署参数' (Upload Deployment Parameters), '新启动和媒体连接' (New Start and Media Connection), '系统上电进入SP' (System Power On to SP), '挂载操作系统镜像文件' (Mount OS Image File), and '部署操作系统' (Deploy OS System). All steps are marked as '成功' (Success).

1.2.9 配置验证

在操作主机中，使用 ping 命令，验证 1.2.8 中安装的操作系统网络配置。

```
C:\Users\hcip>ping 192.168.101.16

Pinging 192.168.101.16 with 32 bytes of data:
Reply from 192.168.101.16: bytes=32 time<1ms TTL=63
Reply from 192.168.101.16: bytes=32 time<1ms TTL=63
Reply from 192.168.101.16: bytes=32 time<1ms TTL=63
Reply from 192.168.101.16: bytes=32 time<1ms TTL=63

Ping statistics for 192.168.101.16:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\hcip>
```

使用 SSH 工具（如 PuTTY、MobaXterm 等），使用 root 账号和在 eComputing Toolkit 或服务部署专用参数模板中配置的密码登录 1.2.8 安装完成的服务器操作系统。

```
Pre-authentication banner message from server:
|
| Authorized users only. All activities may be monitored and reported.
| End of banner message from server
root@192.168.101.16's password:
```

成功登录操作系统，显示的主机名与服务器部署专用参数模板中配置的主机名相同，证明操作系统按照要求安装并配置完成。

```
Authorized users only. All activities may be monitored and reported.
Activate the web console with: systemctl enable --now cockpit.socket

Last login:

Welcome to 6.6.0-98.0.0.103.oe2403sp2.aarch64

System information as of time:

System load: 0.00
Memory used: .5%
Swap used: 0%
Usage On: 4%
IP address: 192.168.101.16
Users online: 2

[root@Taishan ~]#
```

1.3 思考题

问题：eComputing ToolKit 能否对设备进行批量管理。

答案：可以，eComputing ToolKit 可以对设备进行分组管理，并按照分组进行执行任务。

2 Ansible 工具运维

2.1 实验介绍

2.1.1 关于本实验

本实验主要介绍 Ansible 工具的安装，以及如何使用 Ansible 实现服务器、操作系统的运维管理。

2.1.2 实验目的

- 掌握 Ansible 工具的安装方法；
- 掌握使用 Ansible 工具进行服务器、操作系统运维的方法。

2.1.3 实验所需软件

表2-1 实验所需软件

软件	下载地址
openEuler-24.03-LTS-SP2-everything	https://repo.openeuler.openatom.cn/openEuler-24.03-LTS-SP2/ISO/aarch64/openEuler-24.03-LTS-SP2-everything-aarch64-dvd.iso

2.2 实验任务配置

2.2.1 Ansible 安装部署

步骤 1 配置 yum 源（有条件可选）

如果服务器可以访问公网，则跳过本步骤。如果服务器不能访问公网，则需要修改 yum 源，本实验以本地 yum 源为例，将下载好的 openEuler-24.03-LTS-SP2-everything-aarch64-dvd.iso 上传到服务器/root 目录下。

```
[root@localhost ~]# ll
total 23806776
-rw-----. 1 root root 135235 May 9 16:35 anaconda-ks.cfg
-rw-r--r--. 1 root root 24377858048 May 11 06:48 openEuler-24.03-LTS-SP2-everything-aarch64-dvd.iso
-rw-----. 1 root root 134980 May 9 16:35 original-ks.cfg
```

挂载 everything 的 ISO 镜像：

```
# mkdir -p /mnt/openeuler_iso
# mount -o loop /root/openEuler-24.03-LTS-SP2-everything-aarch64-dvd.iso /mnt/openeuler_iso
```

```
[root@localhost ~]# mkdir -p /mnt/openeuler_iso
[root@localhost ~]# mount -o loop /root/openEuler-24.03-LTS-SP2-everything-aarch64-dvd.iso /mnt/openeuler_iso
mount: /mnt/openeuler_iso: WARNING: source write-protected, mounted read-only.
```

配置本地 yum 源：

```
# mv /etc/yum.repos.d/openEuler.repo /etc/yum.repos.d/openEuler.repo.bak
# cat > /etc/yum.repos.d/local.repo <<EOF
[local-everything]
name=OpenEuler Local Everything
baseurl=file:///mnt/openeuler_iso
enabled=1
gpgcheck=0
EOF
```

```
[root@localhost ~]# cat > /etc/yum.repos.d/local.repo <<EOF
> [local-everything]
> name=OpenEuler Local Everything
> baseurl=file:///mnt/openeuler_iso
> enabled=1
> gpgcheck=0
> EOF
```

清理并生成新缓存：

```
# yum clean all
# yum makecache
```

```
[root@localhost ~]# yum clean all
0 files removed
[root@localhost ~]# yum makecache
OpenEuler Local Everything
Last metadata expiration check: 0:00:05 ago on Mon 11 May 2026 07:34:05 AM CST.
Metadata cache created.
```

步骤 2 安装 Ansible

```
# yum -y install ansible
```

步骤 3 检查 Ansible 安装情况

若如图所示，则表示 Ansible 安装成功。

```
# ansible --version
```

```
[root@localhost ~]# ansible --version
ansible 2.9.27
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  executable location = /usr/bin/ansible
  python version = 3.11.6 (main, Jun 18 2025, 23:01:52) [GCC 12.3.1 (openEuler 12.3.1-95.oe2403sp2)]
[root@localhost ~]#
```

2.2.2 Ansible 常用模块命令

步骤 1 在 Ansible 配置文件末尾添加本机地址

```
# vim /etc/ansible/hosts #写入如下信息
[local]
Localhost ansible_connection=local
```

步骤 2 使用如下命令，若返回 pong 则表示命令执行成功

```
# ansible local -m ping
```

```
[root@localhost ~]# ansible local -m ping
[WARNING]: Platform linux on host 127.0.0.1 is using the discovered
interpreter could change this. See https://docs.ansible.com/
127.0.0.1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

步骤 3 命令模块指令

使用 command 模块执行命令：

```
# ansible local -m command -a "yum install libselenium-python3 -y" #输出如下（告警信息请忽略，后续不再提示）：
```

```
127.0.0.1 | CHANGED | rc=0 >>
```

```
Last metadata expiration check: 2:49:22 ago on Tue 12 May 2026 12:48:19 AM CST.
```

```
Dependencies resolved.
```

```
=====
Package                Arch      Version      Repository      Size
=====
Installing:
python3-libselenium    aarch64   3.5-4.oe2403sp2  local-everything 163 k
```

```
Transaction Summary
```

```
=====
Install 1 Package

Total size: 163 k
Installed size: 651 k
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                1/1
  Installing     : python3-libselenium-3.5-4.oe2403sp2.aarch64 1/1
  Running scriptlet: python3-libselenium-3.5-4.oe2403sp2.aarch64 1/1
  Verifying      : python3-libselenium-3.5-4.oe2403sp2.aarch64 1/1

Installed:
python3-libselenium-3.5-4.oe2403sp2.aarch64
```

Complete!

步骤 4 Copy 模块指令

使用 Copy 模块，复制文件：

```
# ansible local -m copy -a "src=/etc/hosts dest=/tmp/hosts"
127.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "checksum": "e56a7671b0028763bd9580db4a8c6067663595cc",
  "dest": "/tmp/hosts",
  "gid": 0,
  "group": "root",
  "md5sum": "34237b3f095501aac5a64024e4ad4b7c",
  "mode": "0644",
  "owner": "root",
  "secontext": "unconfined_u:object_r:admin_home_t:s0",
  "size": 384,
  "src": "/root/.ansible/tmp/ansible-tmp-1778522314.696261-16399-78350794352119/source",
  "state": "file",
  "uid": 0
}
```

步骤 5 File 模块指令

使用 File 模块，给文件添加权限：

```
# ansible local -m file -a "dest=/tmp/hosts mode=777"
127.0.0.1 | CHANGED => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": true,
  "gid": 0,
  "group": "root",
  "mode": "0777",
  "owner": "root",
  "path": "/tmp/hosts",
  "secontext": "unconfined_u:object_r:admin_home_t:s0",
  "size": 384,
  "state": "file",
  "uid": 0
}
```

2.2.3 使用 Ansible 管理服务器

使用 Ansible 的 Playbook 调用 iBMC Redfish 接口实现设备管理。iBMC Redfish 接口参考：

<https://support.huawei.com/enterprise/zh/doc/EDOC1000126991/6974c567?idPath=23710424|251364417|9856629|250697162>

步骤 1 查询服务器系统资源信息

创建 get_system_info.yml 文件，包含 4 个 tasks，先获取系统集合资源信息，确定系统资源的 ID，再通过 system_id 获取指定系统资源信息，最后将所需资源信息打印回显：

```
# cat > get_system_info.yml<<EOF
---
- name: Get TaiShan Server System Info via Redfish
  hosts: local
  gather_facts: no
  vars:
    ibmc_ip: "192.168.220.16"      # 替换为你的 iBMC 管理 IP
    ibmc_user: "Administrator"    # 替换为你的 iBMC 用户名
    ibmc_password: "Huawei12#$"    # 替换为你的 iBMC 密码
    base_uri: "https://{{ ibmc_ip }}/redfish/v1"

  tasks:
    - name: Get System Collection
      uri:
        url: "{{ base_uri }}/Systems"
        method: GET
        user: "{{ ibmc_user }}"
        password: "{{ ibmc_password }}"
        force_basic_auth: yes
        validate_certs: no
        status_code: 200
        register: system_collection

    - name: Extract System ID
      set_fact:
        system_id: "{{ system_collection.json.Members[0]['@odata.id'].split('/')[ -1] }}"

    - name: Get Specific System Details
      uri:
        url: "{{ base_uri }}/Systems/{{ system_id }}"
        method: GET
        user: "{{ ibmc_user }}"
        password: "{{ ibmc_password }}"
        force_basic_auth: yes
        validate_certs: no
        status_code: 200
        register: system_details

    - name: Display Server Info
      debug:
```

```

msg:
  - "Server Model: {{ system_details.json.Model }}"
  - "Serial Number: {{ system_details.json.SerialNumber }}"
  - "Power State: {{ system_details.json.PowerState }}"
  - "Health Status: {{ system_details.json.Status.Health }}"
  - "BIOS Version: {{ system_details.json.BiosVersion }}"
EOF

```

执行 playbook:

```
# ansible-playbook get_system_info.yml
```

```

[root@localhost ~]# ansible-playbook get_system_info.yml
PLAY [Get TaiShan Server System Info via Redfish] *****
TASK [Get System Collection] *****
[WARNING]: Platform linux on host 127.0.0.1 is using the discovered Python interpreter at /usr/bin/python3, but fut
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery
ok: [127.0.0.1]
TASK [Extract System ID] *****
ok: [127.0.0.1]
TASK [Get Specific System Details] *****
ok: [127.0.0.1]
TASK [Display Server Info] *****
ok: [127.0.0.1] => {
  "msg": [
    "Server Model: TaiShan 2280 V2-BAAC",
    "Serial Number: 2102312NFF10KA000181",
    "Power State: On",
    "Health Status: Critical",
    "BIOS Version: 1.05"
  ]
}
PLAY RECAP *****
127.0.0.1 : ok=4  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

步骤 2 查询系统概览信息

创建 get_system_overview.yml 文件，通过系统概览信息，获取 CPU 占用率、内存占用率、设备当前功率、告警数量信息：

```

# cat > get_system_overview.yml<<EOF
---
- name: Get TaiShan Server Overview Info via Redfish
  hosts: local
  gather_facts: no
  vars:
    ibmc_ip: "192.168.220.16"      # 替换为你的 iBMC 管理 IP
    ibmc_user: "Administrator"    # 替换为你的 iBMC 用户名
    ibmc_password: "Huawei12#$"    # 替换为你的 iBMC 密码
    base_uri: "https://{{ ibmc_ip }}/redfish/v1"

  tasks:
    - name: Get System Overview
      uri:
        url: "{{ base_uri }}/SystemOverview"
        method: GET
        user: "{{ ibmc_user }}"
        password: "{{ ibmc_password }}"

```

```

    force_basic_auth: yes
    validate_certs: no
    status_code: 200
    register: system_overview

- name: Display System Overview
  debug:
    msg:
      - "CPUUsagePercent: {{ system_overview.json.Systems[0].CPUUsagePercent }}"
      - "MemoryUsagePercent: {{ system_overview.json.Systems[0].MemoryUsagePercent }}"
      - "PowerConsumedWatts: {{ system_overview.json.Chassis[0].Power.PowerConsumedWatts }}"
      - "CriticalAlarmCount:
{{ system_overview.json.Systems[0].HealthSummary.CriticalAlarmCount }}"
      - "MajorAlarmCount:
{{ system_overview.json.Systems[0].HealthSummary.CriticalAlarmCount }}"
      - "MinorAlarmCount:
{{ system_overview.json.Systems[0].HealthSummary.CriticalAlarmCount }}"
EOF

```

执行 playbook:

```

# ansible-playbook get_system_overview.yml

[root@localhost ~]# ansible-playbook get_system_overview.yml
PLAY [Get TaiShan Server Overview Info via Redfish] *****

TASK [Get System Overview] *****
[WARNING]: Platform linux on host 127.0.0.1 is using the discovered Python interpreter at /usr/bin/python3, but fut
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.
ok: [127.0.0.1]

TASK [Display System Overview] *****
ok: [127.0.0.1] => {
  "msg": [
    "CPUUsagePercent: ",
    "MemoryUsagePercent: ",
    "PowerConsumedWatts: 282",
    "CriticalAlarmCount: 1",
    "MajorAlarmCount: 1",
    "MinorAlarmCount: 1"
  ]
}

PLAY RECAP *****
127.0.0.1 : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

```

步骤 3 查询日志信息

创建 get_ibmc_logs.yml 文件，获取最新一条日志的日志描述：

```

# cat > get_ibmc_logs.yml<<EOF
---
- name: Get TaiShan Server Logs Info via Redfish
  hosts: local
  gather_facts: no
  vars:
    ibmc_ip: "192.168.220.16"      # 替换为你的 iBMC 管理 IP
    ibmc_user: "Administrator"    # 替换为你的 iBMC 用户名
    ibmc_password: "Huawei12#$"    # 替换为你的 iBMC 密码
    base_uri: "https://{{ ibmc_ip }}/redfish/v1"

```

```
tasks:
  - name: Get System Collection
    uri:
      url: "{{ base_uri }}/Systems"
      method: GET
      user: "{{ ibmc_user }}"
      password: "{{ ibmc_password }}"
      force_basic_auth: yes
      validate_certs: no
      status_code: 200
      register: system_collection

  - name: Extract System ID
    set_fact:
      system_id: "{{ system_collection.json.Members[0]['@odata.id'].split('/')[1] }}"

  - name: Get Specific Logs Details
    uri:
      url: "{{ base_uri }}/Systems/{{ system_id }}/LogServices/Log1"
      method: GET
      user: "{{ ibmc_user }}"
      password: "{{ ibmc_password }}"
      force_basic_auth: yes
      validate_certs: no
      status_code: 200
      register: Logs_details

  - name: Display Logs Info
    debug:
      msg:
        - "Message: {{ Logs_details.json.Oem.Huawei.HealthEvent[0].Message }}"

EOF
```

执行 playbook:

```
# ansible-playbook get_ibmc_logs.yml
```

```
[root@localhost ~]# ansible-playbook get_ibmc_logs.yml
PLAY [Get TaiShan Server Logs Info via Redfish] *****

TASK [Get System Collection] *****
[WARNING]: Platform linux on host 127.0.0.1 is using the discovered Python interpreter at /usr/bin/python3, but futu
this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [127.0.0.1]

TASK [Extract System ID] *****
ok: [127.0.0.1]

TASK [Get Specific Logs Details] *****
ok: [127.0.0.1]

TASK [Display Logs Info] *****
ok: [127.0.0.1] => {
  "msg": [
    "Message: The AC/DC input of PSU 2 is lost or out-of-range (SN:2102312HGYHVK9010567, BN:02312HGY)."
```

步骤 4 重启服务器

创建 system_reset.yml 文件，重启服务器：

```
# cat > system_reset.yml<<EOF
---
- name: Reset TaiShan Server via Redfish
  hosts: local
  gather_facts: no
  vars:
    ibmc_ip: "192.168.220.16"      # 替换为你的 iBMC 管理 IP
    ibmc_user: "Administrator"    # 替换为你的 iBMC 用户名
    ibmc_password: "Huawei12#$"    # 替换为你的 iBMC 密码
    base_uri: "https://{{ ibmc_ip }}/redfish/v1"

  tasks:
    - name: Get System Collection
      uri:
        url: "{{ base_uri }}/Systems"
        method: GET
        user: "{{ ibmc_user }}"
        password: "{{ ibmc_password }}"
        force_basic_auth: yes
        validate_certs: no
        status_code: 200
        register: system_collection

    - name: Extract System ID
      set_fact:
        system_id: "{{ system_collection.json.Members[0]['@odata.id'].split('/')[ -1] }}"

    - name: Reset System
      uri:
        url: "{{ base_uri }}/Systems/{{ system_id }}/Actions/ComputerSystem.Reset"
        method: POST
        user: "{{ ibmc_user }}"
        password: "{{ ibmc_password }}"
        force_basic_auth: yes
        validate_certs: no
        status_code: 200
        headers:
          Content-Type: "application/json"
        body: "{{ { 'ResetType': 'ForceRestart' } | to_json }}"
EOF
```

执行 playbook，因本实验 ansible 操控的是本机，因此系统会重启会断连：

```
# ansible-playbook system_reset.yml
```

```
[root@localhost ~]# ansible-playbook system_reset.yml

PLAY [Reset TaiShan Server via Redfish] *****

TASK [Get System Collection] *****
[WARNING]: Platform linux on host 127.0.0.1 is using the discovered
interpreter could change this. See https://docs.ansible.com/ansible
ok: [127.0.0.1]

TASK [Extract System ID] *****
ok: [127.0.0.1]

TASK [Reset System] *****

Network error: Software caused connection abort

Session stopped
- Press <return> to exit tab
- Press R to restart session
- Press S to save terminal output to file
```

2.3 思考题

问题：Ansible 能否管理远端服务器？

答案：可以，Ansible 可以通过在 yml 文件中配置远端服务器的 IP 地址，对远端服务器进行管理。