Huawei Cloud EulerOS

常见问题

文档版本 01

发布日期 2024-09-20

华为云计算技术有限公司

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云计算技术有限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，华为云计算技术有限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。

目 录

1 CentOS Linux 停止维护后如何应对?

CentOS官方已计划停止维护CentOS操作系统，华为云上CentOS公共镜像来源于CentOS官方，当CentOS操作系统停止维护后，华为云将会同时停止对该操作系统的支持。本文主要介绍CentOS操作系统停止维护带来的影响，并针对影响提供应对策略。

背景信息

2020年12月08日，CentOS官方宣布了停止维护CentOS Linux的计划，并推出了CentOS Stream项目。更多信息，请参见**CentOS官方公告**。

CentOS 8系统2021年12月31日已停止维护服务，CentOS 7系统将于2024年06月30日停止维护服务。CentOS官方不再提供CentOS 9及后续版本，不再支持新的软件和补丁更新。CentOS用户现有业务随时面临宕机和安全风险，并无法确保及时恢复。

影响

基于CentOS官方的变更计划，对CentOS操作系统的使用者产生的影响如下所述：

- 2021年12月31日以后，CentOS 8的使用者将无法获得包括问题修复和功能更新在内的任何软件维护和支持。
- 2024年06月30日以后，CentOS 7的使用者将无法获得包括问题修复和功能更新在内的任何软件维护和支持。

对于华为云的公共镜像及服务支持存在一定影响：

- 华为云暂不会下线CentOS 8公共镜像，同时已经使用CentOS 8创建的ECS实例运行不会受到影响，但将停止更新镜像。
- 华为云对于CentOS操作系统的服务支持将和CentOS官方日期保持同步。2021年12月31日以后将不再对CentOS 8提供服务支持；对CentOS 7的服务支持将持续至2024年6月30日。

应对策略

为了保障使用CentOS系统的业务正常运行，华为云为您提供替换CentOS操作系统的应对策略。替换CentOS操作系统的方式分为两类，切换操作系统和迁移操作系统。

切换到Debian或Ubuntu具体操作、详见**切换操作系统**。

- 将CentOS操作系统切换为**支持切换的操作系统**。

如果现有的ECS配置（网卡、磁盘、VPN等配置的类型和数量）都不需要改变，仅需要修改ECS的操作系统镜像，并且您的软件和原操作系统耦合度较低，建议使用系统切换。

- 切换到Huawei Cloud EulerOS具体操作，详见**将操作系统切换为HCE**。
- 切换到Debian或Ubuntu具体操作、详见**切换操作系统**。

- 将CentOS操作系统迁移为Huawei Cloud EulerOS操作系统。

  如果现有的ECS配置（网卡、磁盘、VPN等配置的类型和数量）都不需要改变，希望保留操作系统软件的配置参数，可以通过操作系统迁移的方式迁移到Huawei Cloud EulerOS。

  系统迁移详见**将操作系统迁移为HCE**。

系统切换和迁移的区别如下表，请根据需要选择合适的替换方式。

表 1-1 系统切换和迁移的区别

区别	系统切换	系统迁移
数据备份	• 切换操作系统会清除系统盘数据，包括系统盘上的系统分区和所有其它分区。 • 切换操作系统不影响数据盘数据。	• 迁移操作系统不会清除系统盘数据，为避免系统软件的数据丢失，建议将其备份。 • 迁移操作系统不影响数据盘数据。
个性化设置	切换操作系统后，当前操作系统内的个性化设置（如DNS、主机名等）将被重置，需重新配置。	迁移操作系统后，当前操作系统内的个性化设置（如DNS、主机名等）不需重新配置。

表 1-2 支持切换的操作系统

操作系统	概述	适用人群
Huawei Cloud EulerOS	Huawei Cloud EulerOS（简称HCE）是基于openEuler构建的云上操作系统。 HCE打造云原生、高性能、高安全、易迁移等能力，加速用户业务上云，提升用户的应用创新空间，可替代CentOS、EulerOS等公共镜像。	适用于希望使用免费镜像，并延续开源社区镜像使用习惯的个人或企业。
Debian、Ubuntu操作系统	Linux的其他发行版操作系统，不同操作系统在使用习惯和应用兼容性上存在一定差异。	适用于可以自行应对操作系统切换成本的个人或企业。

2 华为云针对 CentOS EOL 有没有迁移方案?

背景信息

CentOS 8系统2021年12月31日已停止维护服务，CentOS 7系统将于2024年06月30日停止维护服务。CentOS不再支持新的软件和补丁更新。CentOS用户现有业务随时面临宕机和安全风险，并无法确保及时恢复。

HCE操作系统从云原生混部竞争力、安全可信、快速迁移、高效运维、专业认证等方面为用户提供专业云服务、解决CentOS停服带来的影响。HCE提供了迁移工具，可将CentOS、EulerOS等操作系统平滑迁移至HCE操作系统。

兼容性评估

华为HCE操作系统已具备完整代替CentOS的技术能力，完全自主可控，并基于openEuler开源社区持续自主演进。南向支持6大类400种板卡，基本覆盖主流计算产品。北向100%兼容主流的应用场景（云原生、存储、数据库、大数据、WEB等）。超过5000种应用软件通过兼容性认证，基本能够替代CentOS的各种部署。

为满足CentOS系列到HCE搬迁的准确性和安全性，请您使用兼容性工具对待迁移软件快速进行扫描，获取评估结果。

对于可兼容的应用软件，迁移过程中并不会修改软件配置，迁移完成后无需重新配置；对于部分不兼容的应用软件，评估报告给出相应的规避策略，请在迁移之后进行相应的适配。

迁移能力评估

HCE已有成熟的搬迁指导，按照分布式集群应用、主备应用、单机应用三种类型对各种应用进行归类，并制定相应的搬迁方案：

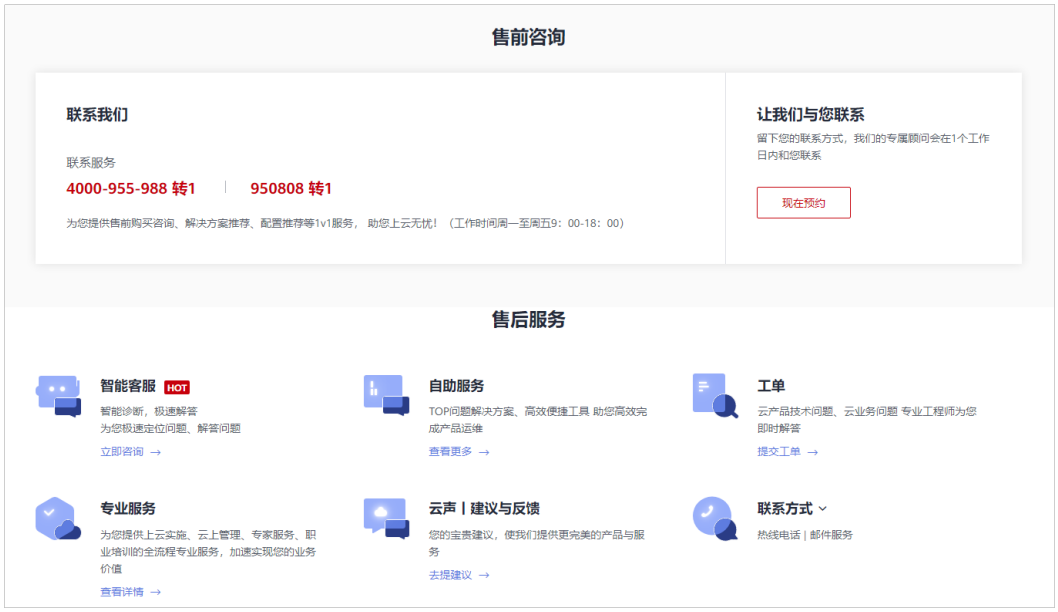
- 针对分布式集群软件，如大数据、分布式存储。CentOS搬迁无需中断业务， 基于分布式软件伸缩扩容机制，HCE实现滚动代替，平滑搬迁。
- 针对主备应用，如数据库等。CentOS搬迁无需中断业务，先备后主，基于主备状态同步机制，平滑搬迁。
- 针对单机应用，CentOS搬迁需中断业务，割接式搬迁，该类搬迁方案和原应用重新部署方式等同，属于成熟方案。

华为云提供**两步切换至HCE操作系统**实践，本实践带您体验从CentOS迁移到Huawei Cloud EulerOS 2.0的完整过程，开发者可直接体验兼容评估和迁移能力评估。

联系我们

华为云专业的服务团队，致力于为您提供专业的售前购买、咨询服务，及完善的售后技术服务，欢迎**联系我们**。

图 2-1 多种技术服务途径



3 如何安装 mlnx 驱动?

本节介绍如何在HCE 2.0操作系统（包含x86架构和Arm架构系统）安装mlnx驱动。

约束与限制

- HCE 2.0的内核为linux 5.10及以上版本。
- CX6网卡驱动为23.10-1.1.9.0-LTS及以上版本。

前提条件

已经安装了5.10或更高内核版本的HCE 2.0系统。

在 x86 架构安装 mlnx 驱动

1. 下载CX6网卡驱动安装包**MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-x86_64.tgz**。
2. 解压驱动安装包并进入工作目录。

   tar -xf MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-x86_64.tgz

   cd MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-x86_64
3. 安装CX6网卡驱动软件。

   ./mlnxofedinstall --basic --without-depcheck --distro OPENEULER22.03 --force --kernel 5.10.0-60.18.0.50.oe2203.x86_64 --kernel-sources /lib/modules/$(uname -r)/build

   说明

   其中，“5.10.0-60.18.0.50.oe2203.x86_64“是官方MLNX_OFED包本身编译时的内核版本。
4. 创建链接。

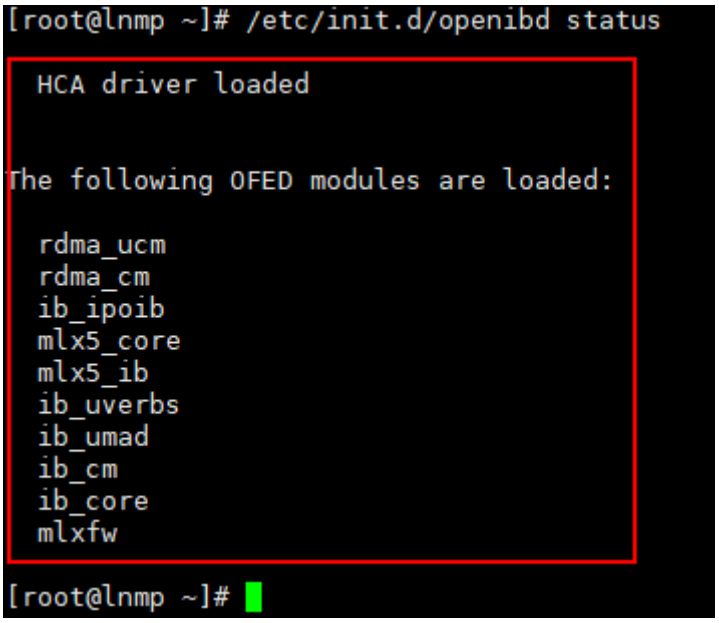
   ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.x86_64/extra/mlnx-ofa_kernel /lib/modules/$(uname -r)/weak-updates/

   ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.x86_64/extra/kernel-mft /lib/modules/$(uname -r)/weak-updates/

   depmod -a
5. 执行**reboot**命令重新系统。

6. 执行**/etc/init.d/openibd status**命令查看驱动安装结果。

   显示如下信息表示驱动安装成功。



在 Arm 架构安装 mlnx 驱动

1. 下载CX6网卡驱动安装包**MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64.tgz**。
2. 解压驱动安装包并进入工作目录。

   tar -xf MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64.tgz

   cd MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64
3. 安装CX6网卡驱动软件。

   ./mlnxofedinstall --basic --without-depcheck --distro OPENEULER22.03 --force --kernel 5.10.0-60.18.0.50.oe2203.aarch64 --kernel-sources /lib/modules/$(uname -r)/build

   说明

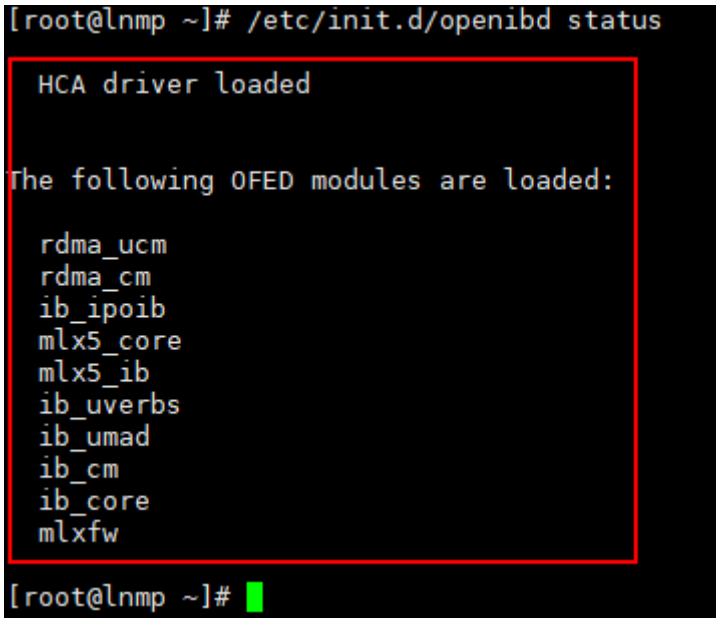
   其中，“5.10.0-60.18.0.50.oe2203.aarch64”是官方MLNX_OFED包本身编译时的内核版本。
4. 执行如下命令创建链接。

   ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.aarch64/extra/mlnx-ofa_kernel /lib/modules/$(uname -r)/weak-updates/

   ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.aarch64/extra/kernel-mft /lib/modules/$(uname -r)/weak-updates/

   depmod -a
5. 执行**reboot**命令重新系统。
6. 执行**/etc/init.d/openibd status**命令查看驱动安装结果。

   显示如下信息表示驱动安装成功。



4 如何开启 HCE 操作系统的 SELinux 功能?

HCE操作系统默认关闭SELinux功能。如果业务需要开启SELinux 功能，请参照本节指导操作。

⚠ 注意

请按照本节指导开启SELinux功能，勿直接通过/etc/selinux/config开启SELinux功能，否则可能会出现无法登录的问题。

操作步骤

1. 打开配置文件/boot/grub2/grub.cfg，删除selinux=0。
2. 执行**touch /.autorelabel**命令。

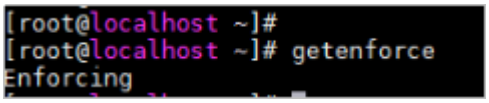
   /.autorelabel文件将触发OS在启动过程中对磁盘上所有文件relabel重新打selinux标签，该过程可能需要持续几分钟。relabel完成后OS将自动重启一次并生效，同时自动删除/.autorelabel文件确保下次不会再重复执行relabel动作。
3. 打开配置文件/etc/selinux/config，设置SELINUX=permissive，并执行**reboot**重启操作系统。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

4. 再次打开配置文件/etc/selinux/config，设置SELINUX=enforcing，并执行**reboot**重启操作系统。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

5. 重启后执行**getenforce**命令查看SElinux状态。
   显示Enforcing表示SElinux已经开启。



5 迁移系统后，如何更改控制台操作系统名称?

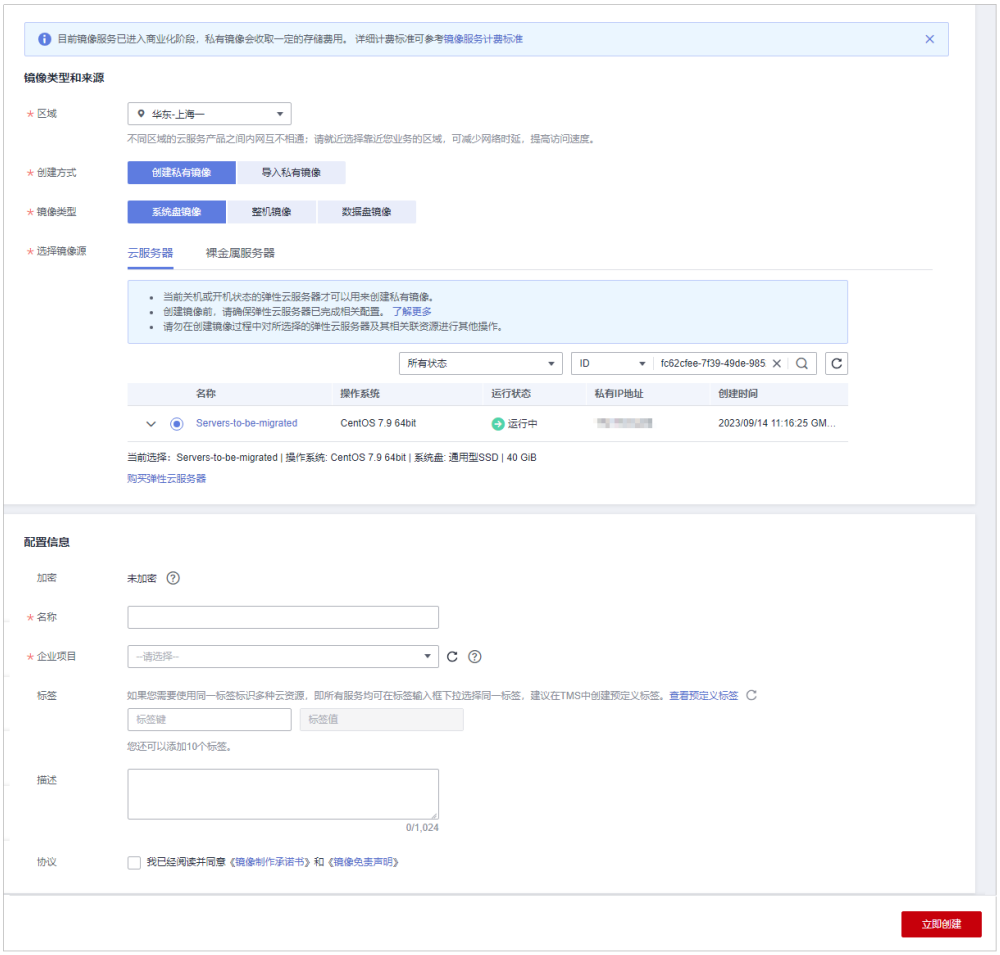
问题背景

原操作系统（例如CentOS 7.9）迁移为Huawei Cloud EulerOS后，控制台仍然显示原操作系统名称CentOS 7.9而不是Huawei Cloud EulerOS。

您可通过创建私有镜像、再切换到此私有镜像的方式，将控制台操作系统名称更改为Huawei Cloud EulerOS。

操作步骤

1. 登录**ECS控制台**。
2. 在待迁移系统的弹性云服务器的“操作”列下，选择“更多 > 镜像 > 创建镜像”。
3. 在“创建私有镜像”页面，配置如下镜像信息。
   - 区域：服务器所在区域，请保持此默认配置。
   - 创建方式：创建私有镜像，请保持此默认配置。
   - 镜像类型：系统盘镜像，请保持此默认配置。
   - 镜像源：迁移系统的弹性云服务器，请保持此默认配置。
   - 名称：填写便于识别的镜像名称。
   - 协议：阅读并勾选协议。



4. 单击“立即创建”创建私有镜像。
5. 确认镜像信息，单击“提交”。
6. 返回弹性云服务器控制台，在待切换操作系统的弹性云服务器的“操作”列下，选择“更多 > 镜像 > 切换操作系统”。
7. 在“切换操作系统”界面配置如下参数。
   - 勾选立即关机。
   - 镜像：选择私有镜像。
   - 登录凭证：选择“使用镜像密码”。

8. 单击“确定”并根据界面提示完成验证。
9. 阅读并勾选声明，单击“确定”。
   系统切换后，控制台系统名称即更改为Huawei Cloud EulerOS。

6 Huawei Cloud EulerOS、openEuler 和 EulerOS 镜像的主要区别是什么?

Huawei Cloud EulerOS、openEuler和EulerOS镜像均为华为自研镜像，主要区别如下**表 1**所示：

表 6-1 Huawei Cloud EulerOS、openEuler 和 EulerOS 镜像的区别与联系

镜像类型	描述
Huawei Cloud EulerOS（简称HCE）	Huawei Cloud EulerOS（简称HCE），是基于openEuler开发的一款商业发行版镜像，可替代CentOS、EulerOS等操作系统，并提供专业的维护保障能力，镜像目前免费对用户使用。 **说明** Huawei Cloud EulerOS 2.0是基于openEuler 22.03 LTS版本构建的云上操作系统。
openEuler	openEuler是一款开源镜像，您可以免费使用，但是不提供商业维护保障能力。openEuler最初由华为研发，但是已经在2021年11月9日正式捐赠给开放原子开源基金会，openEuler的技术支持由开源社区提供。
EulerOS	EulerOS是基于开源技术的企业级Linux操作系统软件，具备高安全性、高可扩展性、高性能等技术特性，能够满足客户IT基础设施和云计算服务等多业务场景需求。 **说明** EulerOS是基于开源操作系统openEuler进行开发的华为内部的操作系统。

7 如何打开内核 wireguard 模块以及安装 wireguard-tools?

> **说明**
>
> wireguard-tools工具来源于社区，如果您在使用中遇到问题，可通过**https://github.com/WireGuard/wireguard-tools/pulls**获取帮助。

打开内核 wireguard 模块

您通过命令**modprobe wireguard**打开内核wireguard模块。

安装 wireguard-tools

步骤1 执行以下命令安装依赖。

dnf install gcc make

步骤2 执行以下命令下载wireguard-tools源码包。

wget https://git.zx2c4.com/wireguard-tools/snapshot/wireguard-tools-1.0.20210914.tar.xz

步骤3 执行以下命令解压上述源码包。

tar -xf wireguard-tools-1.0.20210914.tar.xz

步骤4 进入wireguard-tools-1.0.20210914/src目录，依次执行以下命令编译安装。

make

make install

步骤5 验证安装是否成功。

可以执行**wg -h**和**wg-quick -h**命令验证是否安装成功，如图所示。

```
[root@localhost ~]# wg -h
Usage: wg <cmd> [<args>]

Available subcommands:
  show: Shows the current configuration and device information
  showconf: Shows the current configuration of a given WireGuard interface, for use with `setconf'
  set: Change the current configuration, add peers, remove peers, or change peers
  setconf: Applies a configuration file to a WireGuard interface
  addconf: Appends a configuration file to a WireGuard interface
  syncconf: Synchronizes a configuration file to a WireGuard interface
  genkey: Generates a new private key and writes it to stdout
  genpsk: Generates a new preshared key and writes it to stdout
  pubkey: Reads a private key from stdin and writes a public key to stdout
You may pass `--help' to any of these subcommands to view usage.
[root@localhost ~]# wg-quick -h
Usage: wg-quick [ up | down | save | strip ] [ CONFIG_FILE | INTERFACE ]

  CONFIG_FILE is a configuration file, whose filename is the interface name
  followed by `.conf'. Otherwise, INTERFACE is an interface name, with
  configuration found at /etc/wireguard/INTERFACE.conf. It is to be readable
  by wg(8)'s `setconf' sub-command, with the exception of the following additions
  to the [Interface] section, which are handled by wg-quick:

  - Address: may be specified one or more times and contains one or more
    IP addresses (with an optional CIDR mask) to be set for the interface.
  - DNS: an optional DNS server to use while the device is up.
  - MTU: an optional MTU for the interface; if unspecified, auto-calculated.
  - Table: an optional routing table to which routes will be added; if
    unspecified or `auto', the default table is used. If `off', no routes
    are added.
  - PreUp, PostUp, PreDown, PostDown: script snippets which will be executed
    by bash(1) at the corresponding phases of the link, most commonly used
    to configure DNS. The string `%i' is expanded to INTERFACE.
  - SaveConfig: if set to `true', the configuration is saved from the current
    state of the interface upon shutdown.

See wg-quick(8) for more info and examples.
[root@localhost ~]#
```

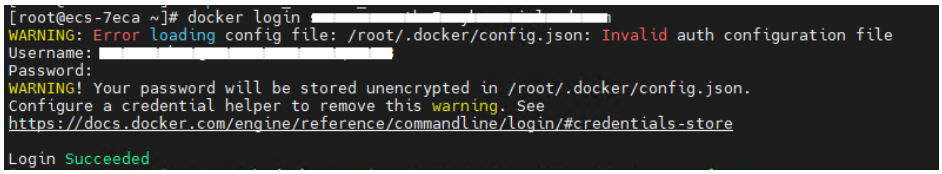
----结束

8 如何将 docker 工具的用户凭证保存方式配置成与社区一致?

问题背景

社区版本的docker工具，使用docker login命令登录成功后，会将用户的用户名、密码等数据以base64的格式保存在用户配置文件，存在较大安全隐患，所以HCE 2.0提供的docker工具，将默认的保存方式改为了加密保存。部分社区工具暂时不支持该安全特性，需要手动将保存方式改为社区的保存方案。

如何将凭证保存方式修改为社区方案

1. 配置环境变量
   ```
   export USE_DECRYPT_AUTH=true
   ```
2. 使用docker login命令重新登录
   ```
   docker login
   ```
   

3. 验证完成后，建议将环境变量配置保存在持久文件中（如~/.bash_profile，/etc/profile等），以便重启后生效
   ```
   echo "export USE_DECRYPT_AUTH=true" >> ~/.bash_profile
   ```

9 OOM 相关参数配置与原因排查

OOM 相关概念

OOM（Out Of Memory，简称OOM）指系统内存已用完，在linux系统中，如果内存用完会导致系统无法正常工作，触发系统panic或者OOM killer。

OOM killer是linux内核的一个机制，该机制会监控那些占用内存过大的进程，尤其是短时间内消耗大量内存的进程，在系统的内存即将不够用时结束这些进程从而保障系统的整体可用性。

OOM 相关参数

表 9-1 OOM 相关参数

参数名称	参数说明	取值	修改方式
panic_on_oom	panic_on_oom参数是控制系统遇到OOM时如何反应的。当系统遇到OOM的时候，通常会有两种选择： • 触发系统panic，可能会出现频繁宕机的情况。 • 选择一个或者几个进程，触发OOM killer，结束选中的进程，释放内存，让系统保持整体可用。	可以通过以下命令查看参数取值： **cat /proc/sys/vm/panic_on_oom**或者 **sysctl -a | grep panic_on_oom** • 值为0：内存不足时，触发OOM killer。 • 值为1：内存不足时，根据具体情况可能发生kernel panic，也可能触发OOM killer。 • 值为2：内存不足时，强制触发系统panic，导致系统重启。 **说明** HCE中参数默认值为1。	例如将参数设置为0，可用以下两种方式： • 临时配置，立即生效，但重启后恢复成默认值。**sysctl -w vm.panic_on_oom=0** • 持久化配置，系统重启仍生效。执行命令**vim /etc/sysctl.conf**，在该文件中添加一行**vm.panic_on_oom =0**，再执行命令**sysctl –p**或重启系统后生效。

参数名称	参数说明	取值	修改方式
oom_kill_allocating_task	当系统选择触发OOM killer，试图结束某些进程时，oom_kill_allocating_task参数会控制选择哪些进程，有以下两种选择： ● 触发OOM的进程。 ● oom_score得分最高的进程。	可以通过以下命令查看参数取值： **cat /proc/sys/vm/oom_kill_allocating_task**或者**sysctl -a | grep oom_kill_allocating_task** ● 值为0：选择oom_score得分最高的进程。 ● 值为非0：选择触发OOM的进程。 **说明** HCE中参数默认值为0。	例如将该参数设置成1，可用以下两种方式： ● 临时配置，立即生效，但重启后恢复成默认值。**sysctl -w vm.oom_kill_allocating_task=1** ● 持久化配置，系统重启仍生效。执行命令**vim /etc/sysctl.conf**，在该文件中添加一行**vm.oom_kill_allocating_task=1**，再执行命令**sysctl –p**或重启系统后生效。
oom_score	指进程的得分，主要有两部分组成： ● 系统打分，主要是根据该进程的内存使用情况由系统自动计算。 ● 用户打分，也就是oom_score_adj，可以自定义。	可以通过调整oom_score_adj的值进而调整一个进程最终的得分。通过以下命令查看参数取值： **cat /proc/*进程id*/oom_score_adj** ● 值为0：不调整oom_score。 ● 值为负值：在实际打分值上减去一个折扣。 ● 值为正值：增加该进程的oom_score。 **说明** oom_score_adj的取值范围是-1000～1000。 若设定成OOM_SCORE_ADJ_MIN或-1000，则表示禁止OOM killer结束该进程。	例如将进程id为2939的进程oom_score_adj参数值设置为1000，可用以下命令： **echo 1000 > /proc/2939/oom_score_adj**

参数名称	参数说明	取值	修改方式
oom_dump_tasks	oom_dump_tasks参数控制OOM发生时是否记录系统的进程信息和OOM killer信息。 例如dump系统中所有的用户空间进程关于内存方面的一些信息，包括：进程标识信息、该进程使用的内存信息、该进程的页表信息等，这些信息有助于了解出现OOM的原因。	可以通过以下命令查看参数取值： **cat /proc/sys/vm/oom_dump_tasks**或者**sysctl -a | grep oom_dump_tasks** • 值为0：OOM发生时不会打印相关信息。 • 值为非0：以下三种情况会调用dump_tasks打印系统中所有task的内存状况。 – 由于OOM导致kernel panic。 – 没有找到需要结束的进程。 – 找到进程并将其结束的时候。 **说明** HCE中参数默认值为1。	例如将该参数设置成0，可用以下两种方式： • 临时配置，立即生效，但重启后恢复成默认值。**sysctl –w vm.oom_dump_tasks=0** • 持久化配置，系统重启仍生效。执行命令**vim /etc/sysctl.conf**，在该文件中添加一行**vm.oom_dump_tasks=0**，再执行命令**sysctl –p**或重启系统后生效。

触发 OOM killer 示例

1. 您可以参考**表9-1**设置HCE系统参数，示例配置如下：

```
[root@localhost ~]# cat /proc/sys/vm/panic_on_oom
0
[root@localhost ~]# cat /proc/sys/vm/oom_kill_allocating_task
0
[root@localhost ~]# cat /proc/sys/vm/oom_dump_tasks
1
```

   – panic_on_oom=0，表示在发生系统OOM的时候触发OOM killer。
   – oom_kill_allocating_task=0，表示触发OOM killer的时候优先选择结束得分高的进程。
   – oom_dump_tasks=1，表示系统发生OOM的时候记录系统的进程信息和OOM killer信息。

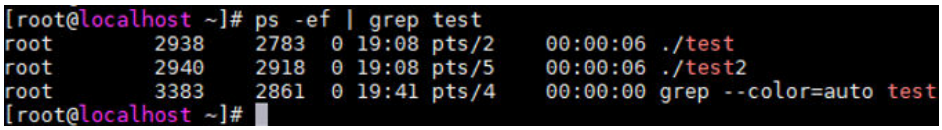
2. 启动测试进程。

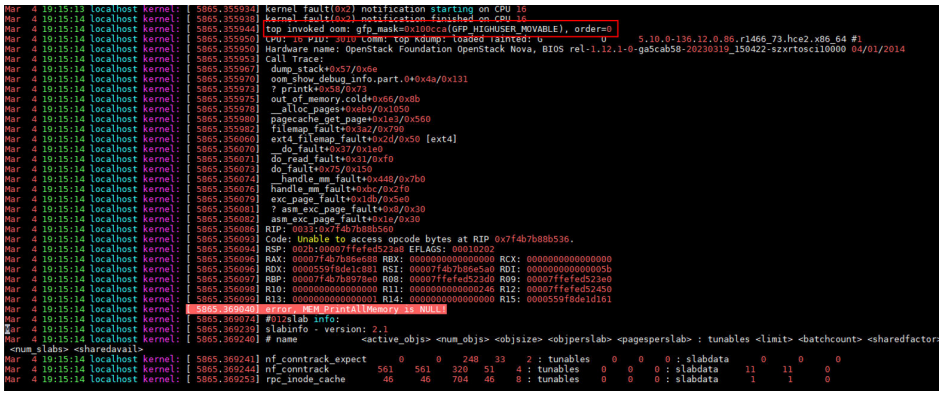
   在系统中同时启动三个相同的测试进程（test、test1、test2），不断申请新的内存，并将test1的oom_score_adj设置成最大的1000，表示OOM killer优先结束该进程，直至内存耗尽触发系统OOM。

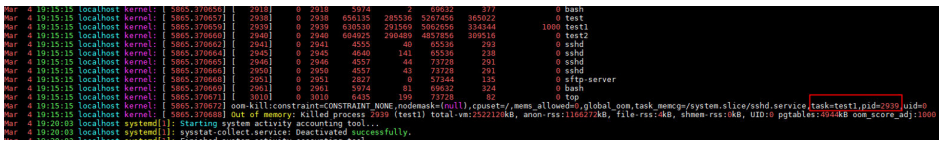
```
[root@localhost ~]# ps -ef | grep test
root        2938    2783  0 19:08 pts/2    00:00:00 ./test
root        2939    2822  0 19:08 pts/3    00:00:00 ./test1
root        2940    2918  0 19:08 pts/5    00:00:00 ./test2
[root@localhost ~]# echo 1000 > /proc/2939/oom_score_adj
[root@localhost ~]# cat /proc/2939/oom_score_adj
1000
```

3. 查看OOM信息。

   经过一段时间后系统发生OOM并触发OOM killer，同时在/var/log/messages中打印系统所有进程的内存等信息并结束了test1进程：







OOM 可能的原因

- cgroup内存不足

  使用的内存超出了cgroup中memory.limit_in_bytes配置的大小，如下示例演示memory.limit_in_bytes配置为80M，使用memhog模拟分配100M，触发OOM，/var/log/messages部分日志如下，可以从日志中看到memhog所在进程（PID: 2021820）使用了81920kB内存，超出了限制，触发了OOM：

  warning|kernel[-]|[2919920.414131] memhog invoked oom-killer: gfp_mask=0xcc0(GFP_KERNEL), order=0, oom_score_adj=0
  info|kernel[-]|[2919920.414220] **memory: usage 81920kB,** limit 81920kB, failcnt 30
  err|kernel[-]|[2919920.414272] Memory cgroup out of memory: Killed process **2021820 (memhog)** total-vm:105048kB, anon-rss:81884kB, file-rss:1544kB, shmem-rss:0kB, UID:0 pgtables:208kB oom_score_adj:0

- 父cgroup内存不足

  在子cgroup中内存仍然足够，但是父cgroup的内存不足，超过了内存限制，如下示例演示父cgroup memory.limit_in_bytes配置为80M，两个子cgroup memory.limit_in_bytes均配置为50M，在两个子cgroup中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下：

  warning|kernel[-]|[2925796.529231] main invoked oom-killer: gfp_mask=0xcc0(GFP_KERNEL), order=0, oom_score_adj=0
  info|kernel[-]|[2925796.529315] **memory: usage 81920kB**, limit 81920kB, failcnt 199
  err|kernel[-]|[2925796.529366] Memory cgroup out of memory: Killed process **3238866 (main)** total-vm:46792kB, anon-rss:44148kB, file-rss:1264kB, shmem-rss:0kB, UID:0 pgtables:124kB oom_score_adj:0

- 系统全局内存不足

  一方面由于OS的空闲内存不足，有程序一直在申请内存，另一方面也无法通过内存回收机制解决内存不足的问题，因此触发了OOM，如下示例演示OS中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下，可以从日志中看到内存节点Node 0的空闲内存（free）已经低于了内存最低水位线（low），触发了OOM：

```
kernel: [ 1475.869152] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|
__GFP_ZERO), order=0
kernel: [ 1477.959960] Node 0 DMA32 free:22324kB min:44676kB low:55844kB high:67012kB
reserved_highatomic:0KB active_anon:174212kB inactive_anon:1539340kB active_file:0kB
inactive_file:64kB unevictable:0kB writepending:0kB present:2080636kB managed:1840628kB
mlocked:0kB pagetables:7536kB bounce:0kB free_pcp:0kB local_pcp:0kB free_cma:0kB
kernel: [ 1477.960064] oom-
kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_mem
cg=/system.slice/sshd.service,task=main,pid=1822,uid=0
kernel: [ 1477.960084] Out of memory: Killed process 1822 (main) total-vm:742748kB, anon-
rss:397884kB, file-rss:4kB, shmem-rss:0kB, UID:0 pgtables:1492kB oom_score_adj:1000
```

- 内存节点（Node）的内存不足

  在NUMA存储模式下，OS会存在多个内存节点，如果程序制定使用特定节点的内存，可能在OS内存充足的情况下触发OOM，如下示例演示在两个内存节点的条件下，使用程序循环在Node 1分配内存，导致Node 1内存不足，但是OS内存足够，/var/log/messages部分日志如下：

  ```
  kernel: [  465.863160] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|
  __GFP_ZERO), order=0
  kernel: [  465.878286] active_anon:218 inactive_anon:202527 isolated_anon:0#012 active_file:5979
  inactive_file:5231 isolated_file:0#012 unevictable:0 dirty:0 writeback:0#012 slab_reclaimable:6164
  slab_unreclaimable:9671#012 mapped:4663 shmem:2556 pagetables:846 bounce:0#012 free:226231
  free_pcp:36 free_cma:0
  kernel: [  465.878292] Node 1 DMA32 free:34068kB min:32016kB low:40020kB high:48024kB
  reserved_highatomic:0KB active_anon:188kB inactive_anon:778076kB active_file:20kB
  inactive_file:40kB unevictable:0kB writepending:0kB present:1048444kB managed:866920kB
  mlocked:0kB pagetables:2752kB bounce:0kB free_pcp:144kB local_pcp:0kB free_cma:0kB
  kernel: [  933.264779] oom-
  kill:constraint=CONSTRAINT_MEMORY_POLICY,nodemask=1,cpuset=/,mems_allowed=0-1,global_oom,
  task_memcg=/system.slice/sshd.service,task=main,pid=1733,uid=0
  kernel: [  465.878438] Out of memory: Killed process 1734 (main) total-vm:239028kB, anon-
  rss:236300kB, file-rss:200kB, shmem-rss:0kB, UID:0 pgtables:504kB oom_score_adj:1000
  ```

- 其他可能原因

  OS在内存分配的过程中，如果伙伴系统的内存不足，则系统会通过OOM Killer释放内存，并将内存提供至伙伴系统。

OOM 问题解决方法

- 从业务进程排查，确认是否有内存泄漏，导致OOM。
- 排查cgroup limit_in_bytes配置是否与业务内存规划匹配，如需要调整，可以手动执行以下命令修改配置参数：

  ```
  echo <value> > /sys/fs/cgroup/memory/<cgroup_name>/memory.limit_in_bytes
  ```

- 如果确认业务需要比较多的内存，建议升级弹性云服务器内存规格。

10 IPVS 报错问题说明

问题背景

IPVS（IP Virtual Server）指IP虚拟服务器，用于负载均衡、网络转发等目的。用户在系统上配置了IPVS虚拟服务器，但未配置真实服务器的情况下，会在VNC上出现错误日志。

问题现象

配置了IPVS虚拟服务器，但未配置真实服务器时，当网络请求发往该虚拟服务器地址后，通过华为云VNC登录的控制台上可以看到类似如下的错误日志。

```
[32264.645949][T268365] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32265.234919][T268366] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32265.954662][T268367] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32266.557032][T268368] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32267.166530][T268369] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32267.725920][T268370] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
```

解决方法

1. 安装ipvsadm。
2. 执行ipvsadm -Ln，查询当前虚拟服务器的配置。找到报错的虚拟服务器对应的表项。

图 10-1 未配置真实服务器

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port[:Subnet] Scheduler Established(Sec.) Flags
  -> RemoteAddress:Port[:Oif]          Forward Weight ActiveConn InActConn VtepAddr:vtepport Vnild Mac
TCP  192.168.1.1:5000  wlc
```

如上图所示没有对应的真实服务器，则说明配置不完整，会引发错误打印。需要排查对应的业务流程是否正确。

图 10-2 已配置真实服务器

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port[:Subnet] Scheduler Established(Sec.) Flags
  -> RemoteAddress:Port[:Oif]          Forward Weight ActiveConn InActConn VtepAddr:vtepport Vnild Mac
TCP  192.168.1.1:5000  wlc
  -> 192.168.1.2:5000            Masq       1      0          0
```

如上图是完整的有真实服务器的配置。

3. 如果希望排除该IPVS错误日志在VNC上对用户操作的干扰，可以按如下的方式处理（选择其中一种即可）
   - 关闭业务发送的网络请求，具体操作需要用户根据自身业务情况来处理。
   - 执行以下命令调整内核printk打印等级。

     echo 3 4 1 7 > /proc/sys/kernel/printk

     说明

     如果临时修改系统配置，建议用户选择适当的时机恢复系统配置。
   - 用华为云的CloudShell方式登录云服务器进行操作。

11 中文环境执行 sulogin 命令终端显示乱码说明

问题背景

使用sulogin命令可以进行单用户登录。sulogin命令目前不支持中文，如果用户将系统语言环境修改为中文，执行sulogin命令时终端会显示乱码。

问题现象

执行export LANG="zh_CN.UTF-8" 修改语言环境为中文后，再执行sulogin终端显示出现乱码，如下图所示：

```
[root@localhost ~]# export LANG="zh_CN.UTF-8"
[root@localhost ~]# sulogin
?? root ?????
(?? Control-D ???):
```

解决方法

执行sulogin命令时，可以临时设置LANG环境变量为英文，比如将LANG设置为en_US.UTF-8：

```
[root@localhost ~]# LANG="en_US.UTF-8" sulogin
Give root password for maintenance
(or press Control-D to continue):
[root@localhost ~]#
```

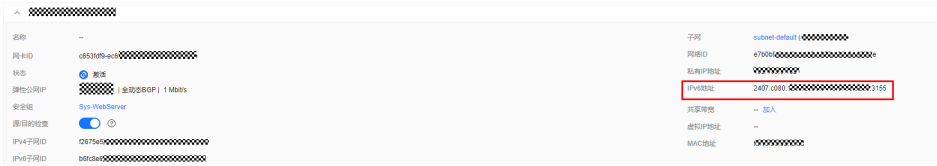
12 ECS 开启 IPv6 后，HCE 系统内无法获取到 IPv6 地址

问题背景

在弹性云服务器ECS控制台上开启云服务器网卡的IPv6功能后，由于未在操作系统内部正确配置IPv6，导致HCE系统内无法获取到IPv6地址。

问题现象

在弹性云服务器ECS控制台上已开启IPv6功能，在详情界面已显示IPv6地址。



但是进入操作系统内部，无法获取到IPv6地址。

解决方法

1. 手动配置dhcp自动获取ipv6地址，如下图在对应网卡配置文件（/etc/sysconfig/network-scripts/ifcfg-ethx）中添加以下参数。

   IPV6INIT="yes"

   DHCPV6C="yes"

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
PERSISTENT_DHCLIENT="yes"
IPV6INIT="yes"
DHCPV6C="yes"
[root@localhost ~]#
```

2. 执行以下命令重启NetworkManager服务即可获取到IPv6地址。

   systemctl restart NetworkManager

```
[root@localhost ~]# systemctl restart NetworkManager
[root@localhost ~]# ip a
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 fa:16:3e:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.xxx/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
       valid_lft 86394sec preferred_lft 86394sec
    inet6 2407:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:3155/128 scope global dynamic noprefixroute
       valid_lft 7195sec preferred_lft 7195sec
    inet6 fe80::f816:3eff:fe9a:f2a8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[root@localhost ~]# _
```

13 如何设置自动注销时间 TMOUT?

操作背景

为了保证系统的安全性，以及减少用户在不使用系统时所造成的资源浪费，在用户离开系统一段时间后，必须对连接进行注销。注销有诸多方法，修改TMOUT变量为其中的解决方案之一。

TMOUT是一个环境变量，它决定了在系统自动注销前所空闲的秒数。因此，在设置了此变量后，若用户在规定时间内没有进行主动活动，则连接将自动断开。若没有设置此变量，或值为0，表示禁用自动注销，用户不会因长时间不活动而断开连接。

操作步骤

1. 执行以下命令查看自动注销时间（即TMOUT的值）。
   ```
   echo $TMOUT
   ```
   显示空白表示没有设置此值。
2. 执行以下命令，设置当前会话下的自动注销时间。如果要长期定义自动注销时间请执行步骤3。
   ```
   export TMOUT=秒数
   ```
3. 长期应用此变量设置自动注销时间。

   方式一

   执行以下命令修改文件/etc/profile（若修改此文件不生效，可再修改/etc/bashrc，修改流程相同），这样可以使所有应用此配置文件的用户均受此自动注销时间的影响。
   ```
   vim /etc/profile
   ```
   或者
   ```
   vim /etc/bashrc
   ```
   添加以下命令至文件末尾。以设置自动注销时间为1200秒为例，实际值可自定义，设置为0禁用注销功能。
   ```
   export TMOUT=1200
   ```



```
    test -r /etc/bashrc
then
    # Bash login shells run only /etc/profile
    # Bash non-login shells run only /etc/bash
    # Check for double sourcing is done in /et
    . /etc/bashrc
fi

export TMOUT=1200
```

保存文件后执行以下命令刷新。

```
source /etc/profile
```

方式二

依次执行以下命令直接修改自动注销时间。

```
sed -i '$a\export TMOUT=1200' /etc/profile
source /etc/profile
```

4. 执行以下命令查看自动注销时间。

```
echo $TMOUT
```

若显示定义的数值，则说明自动注销功能设置成功。

```
[root@localhost ~]# echo $TMOUT
1200
```

