华为认证 Kenpeng 系列教程

HCIA-Kunpeng Application Developer

应用迁移实验指导手册

版本:1.0

华为技术有限公司

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： http://e.huawei.com

华为认证体系介绍

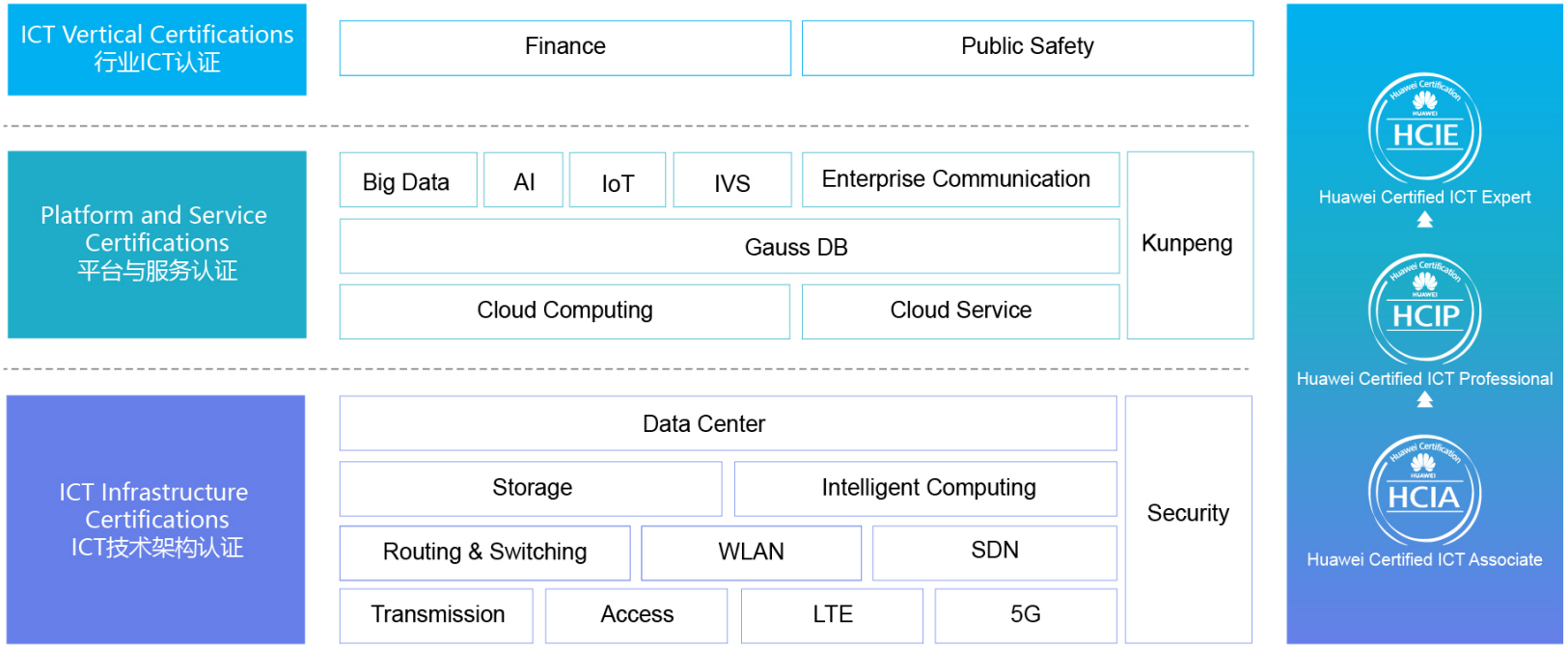
华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的ICT技术架构认证、平台与服务认证、行业ICT认证三类认证，是业界唯一覆盖ICT（Information and Communications Technology 信息通信技术）全技术领域的认证体系。

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。

华为认证HCIA-Kunpeng Application Developer V1.0定位于培养与认证能够应用华为鲲鹏计算平台进行业务部署与迁移，性能测试与调优，常见解决方案设计与规划的工程师。

通过HCIA-Kunpeng Application Developer V1.0认证，您将掌握对鲲鹏计算平台的使用与维护方法，具备对鲲鹏计算平台上的应用进行全生命周期管理的能力，能够胜任鲲鹏平台的应用开发和运维岗位。

Huawei Certification



前 言

简介

在 x86 上的可执行程序需要在基于鲲鹏处理器的平台上运行，需要经过源码的移植修改后方可在鲲鹏云服务器或泰山服务器上运行。x86 平台上所开发的容器镜像也是无法直接运行在鲲鹏平台上的，需要通过鲲鹏平台构建基于鲲鹏的镜像。本实验指导学员如何实现基于鲲鹏平台应用的源码迁移和基于鲲鹏平台的容器迁移。

内容描述

本实验指导包含两个部分，云服务器环境上的应用迁移和容器应用的迁移。

实验一：从云环境准备开始，逐一介绍软件迁移的编译环境准备、源码获取、迁移分析、源码修改和验证。

实验二：从 Docker 安装开始，逐一介绍 Docker 基础镜像的获取、镜像构建和镜像验证。

实验三：删除华为云资源

实验工具

名称	官网下载链接	用途
putty	https://www.putty.org/	远程登录工具
winscp	https://winscp.net/eng/download.php	远程传输工具
Porting Advisor 1.0	https://www.huaweicloud.com/kunpeng/software/portingadvisor.html	代码迁移工具
PortingTest	https://hcia-kunpeng-application-developer.obs.cn-north-1.myhwclouds.com/PortingTest.zip	代码迁移实验的源代码文件

实验资源

- 可选择以下任意一种服务器进行应用迁移实验，本实验以 x86 的弹性云服务器为例

推荐服务器	所需软件版本或规格
基于x86的弹性云服务器	CentOS 7.6|2vCPUs|4GB
基于鲲鹏计算的弹性云服务器	CentOS 7.6|2vCPUs|4GB
RH2288H V5	CentOS 7.8|GCC 4.8
TaiShan 200	CentOS 7.8|GCC 4.8

- 已注册并登陆华为云账号，账号中有一定金额供实验使用。
- 在本地已安装 putty 和 winscp 软件，并获取代码迁移工具和实验源码文件

目录

1 源码迁移实验

1.1 实验介绍

1.1.1 关于本实验

本实验通过一个 C/C++的代码文件为例，实现用 Porting Advisor 工具对代码进行移植分析和修改建议，通过修改代码后，实现代码移植。

1.1.2 实验目的

- 理解 C/C++代码的基本原理。
- 掌握 Porting Advisor 工具进行迁移分析的方法。

1.2 准备虚拟私有云 VPC 环境

在后续创建 ECS 时，选择该步骤中所配置的 VPC、子网和安全组。

步骤 1 登录华为云账号，在虚拟私有云 VPC 页面下，点击“访问控制台”。

https://www.huaweicloud.com/product/vpc.html

步骤 2 在网络控制台界面中，点击“创建虚拟私有云”。

步骤 3　填写如下配置信息，然后点击“立即创建”。

基本信息

- 区域：华北-北京四
- 名称：myvpc
- 网段：默认

子网配置

- 可用区：可用区 1
- 名称：subnet-myvpc
- 子网网段：默认

步骤 4　返回到网络控制台，可看见 VPC 与子网已创建。

步骤 5　在“安全组”页面中找到 Sys-default 安全组。点击“配置规则”。

步骤 6　选择“入方向规则”页签，点击“添加规则”。

步骤 7　协议端口选择“全部放通”，点击“确定”，完成安全组规则的添加。

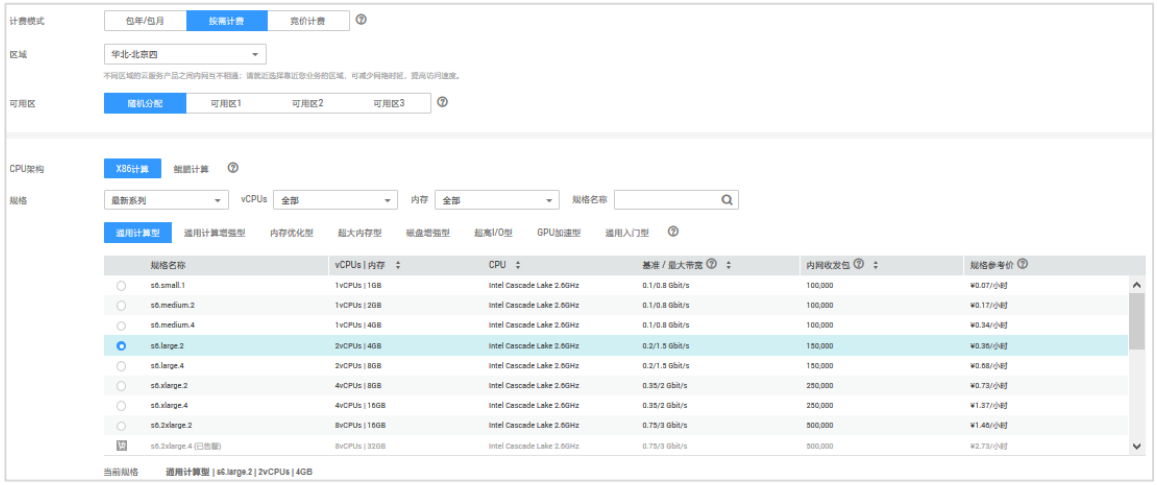
1.3 登录云服务器 ECS

步骤 1　登录华为云账号，在弹性云服务器 ECS 页面下，点击“立即购买”。

https://www.huaweicloud.com/product/ecs.html

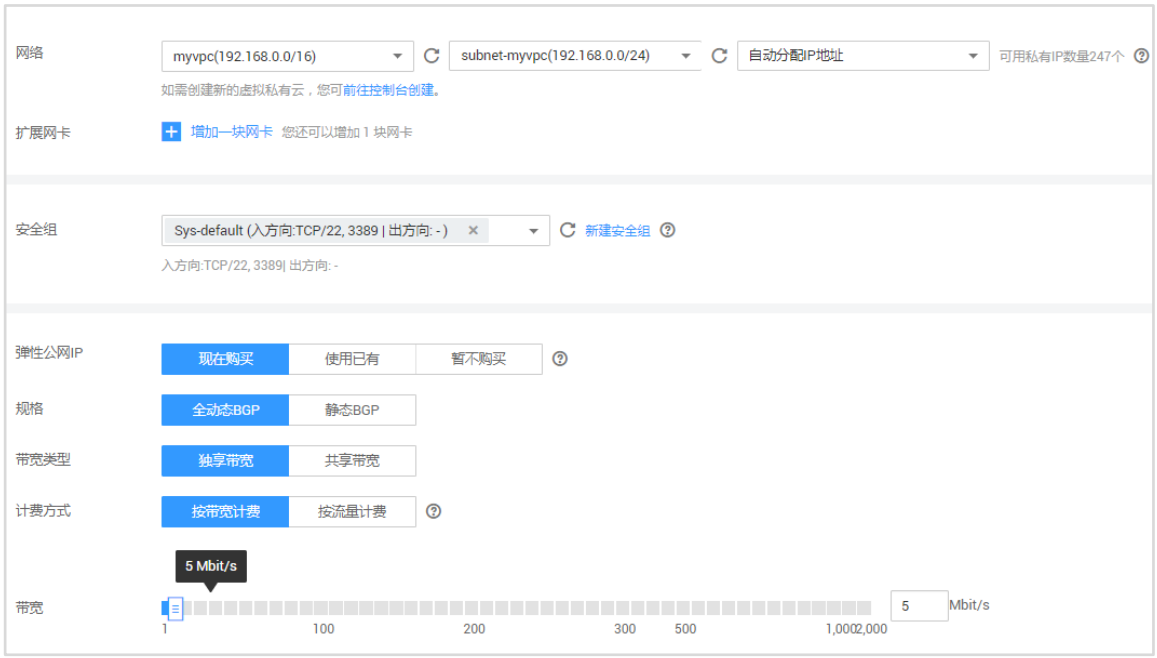
步骤 2　填写如下基础配置信息，然后点击“下一步”。

- 计费模式：按需计费
- 区域：**华北-北京四**
- 可用区：随机分配
- CPU 架构：x86 计算
- 规格：通用计算型 s6.large.2|2vCPUs|4GB
- 镜像：公共镜像 CentOS7.6.64bit
- 系统盘：高 IO|40G



步骤 3　填写如下网络配置信息，然后点击“下一步”。

- 网络：选择已创建的网络和子网，如 myvpc 和 subnet-myvpc
- 安全组：Sys-default
- 弹性公网 IP：现在购买
- 规格：全动态 BGP
- 带宽类型：独享带宽
- 计费方式：按带宽计费
- 带宽：5 Mbit/s



步骤 4　填写如下高级配置信息，然后点击“下一步”。

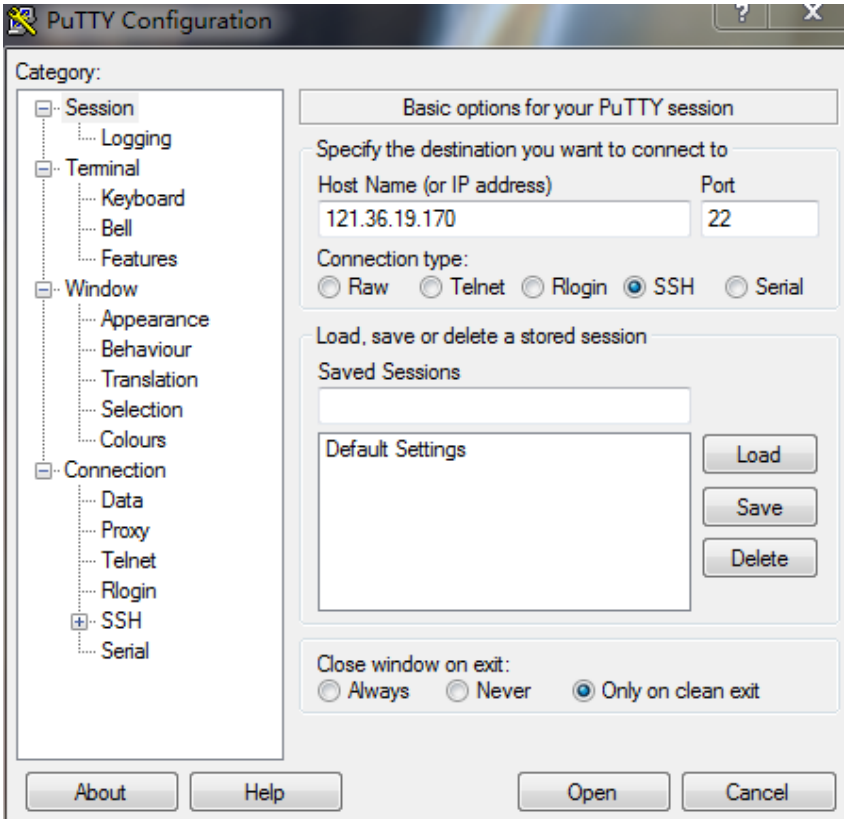
- 云服务器名称：ecs-test
- 登录凭证：密码

- 密码/确认密码：XXXXX（按照密码规则自定义密码）
- 云备份：暂不购买

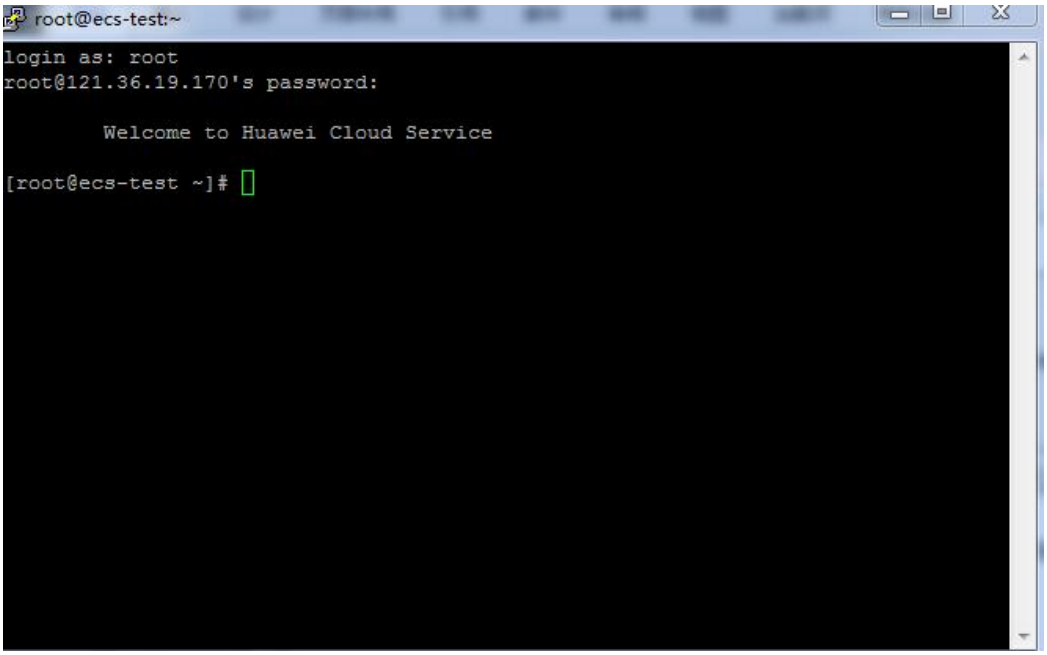
步骤 5　在确认配置界面，勾选“我已经阅读.....”后，点击“立即购买”。

步骤 6　购买完成后，在云服务器控制台可以看到正常运行中的 ECS，记录弹性 IP 地址。

步骤 7　打开 putty，输入弹性 IP 地址后，点击 open。



步骤 8　用 root 用户名，和之前设置的密码登录 ECS。



登录成功后按照以下步骤进行迁移实验。

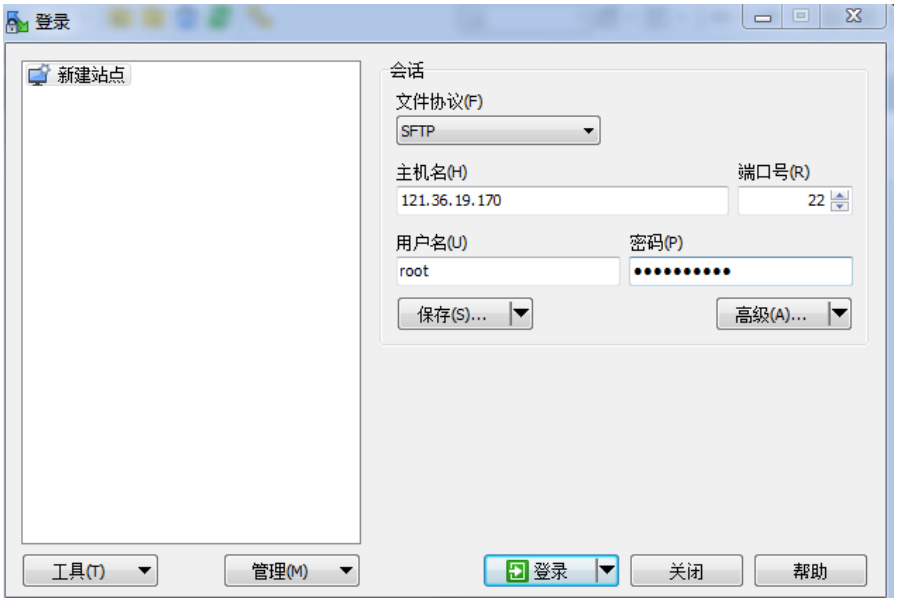
1.4 Porting Advisor 应用迁移操作

1.4.1 安装 Porting Advisor 工具

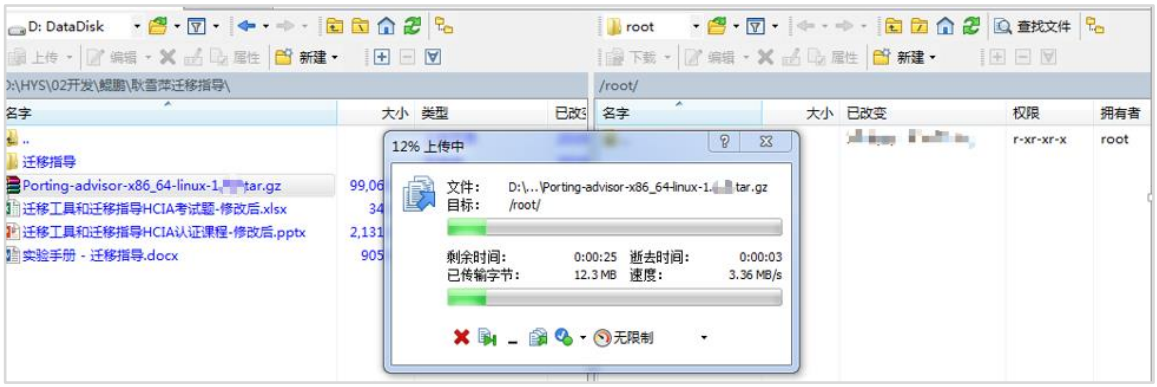
步骤 1 从华为云鲲鹏社区中获取 Porting Advisor 工具，如 Porting-advisor-x86_64-linux-XXX.tar.gz。

https://www.huaweicloud.com/kunpeng/software/portingadvisor.html

步骤 2 打开 WinScp，输入之前记录的 ECS 的 IP 地址，使用 root 用户名，和之前设置的密码登录。



步骤 3 把 Porting Advisor 工具拖到 ECS 中的 root 下。



步骤 4　返回 putty，输入命令“gcc -v”，检查 GCC 环境是否符合要求。

```
[root@ecs-test ~]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr
/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstrap -
-enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib
 --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object -
-enable-linker-build-id --with-linker-hash-style=gnu --enable-languages=c,c++,objc
,obj-c++,java,fortran,ada,go,lto --enable-plugin --enable-initfini-array --disable
-libgcj --with-isl=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-redhat-linu
x/isl-install --with-cloog=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-red
hat-linux/cloog-install --enable-gnu-indirect-function --with-tune=generic --with-
arch_32=x86-64 --build=x86_64-redhat-linux
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC)
[root@ecs-test ~]#
```

步骤 5　执行如下命令，解压文件包。

`tar -zxvf Porting-advisor-XXX.tar.gz`（使用 tab 键补齐）

```
[root@ecs-test ~]# tar -zxvf Porting-advisor-x86_64-linux-1.0.5.tar.gz
Porting-advisor-x86_64-linux-1.0.5/
Porting-advisor-x86_64-linux-1.0.5/install.sh
Porting-advisor-x86_64-linux-1.0.5/Porting-advisor-x86_64-linux-webui-1.0.5.tar.
gz
Porting-advisor-x86_64-linux-1.0.5/Porting-advisor-x86_64-linux-cmd-1.0.5.tar.gz
Porting-advisor-x86_64-linux-1.0.5/Porting-advisor-x86_64-linux-portal-1.0.5.tar
.gz
```

步骤 6　执行如下命令，进入解压后的华为鲲鹏代码迁移工具安装包目录。

`cd Porting-advisor-x86_64-linux-XXX`（使用 tab 键补齐）

```
[root@ecs-test ~]# ls
Porting-advisor-beta-x86_64-linux-1.0.5.tar.gz
Porting-advisor-x86_64-linux-1.0.5
[root@ecs-test ~]# cd Porting-advisor-x86_64-linux-1.0.5/
[root@ecs-test Porting-advisor-x86_64-linux-1.0.5]#
```

步骤 7　执行以下命令，安装华为鲲鹏代码迁移工具的 WEB 模式。

```
sh install.sh web
```

- 按规划输入 Web Server 的 IP 地址，然后按 Enter 键。如果使用默认 IP 地址，不需要输入 IP，直接按 Enter 键即可。默认的 IP 地址为操作系统的 IP 地址。
- 按规划输入 https 端口，然后按 Enter 键。如果使用默认的端口，不需要输入端口号，直接按 Enter 键即可。默认端口为 8084。

回显信息如下：

```
[root@ecs-test ~]# cd Porting-advisor-x86_64-linux-1.0.5
[root@ecs-test Porting-advisor-x86_64-linux-1.0.5]# sh install.sh web
please enter ip address(default is all ip addresses):
defult ip address
Cent OS
please enter https port(default: 8084):
```

注意：如果存在“/opt/portadv”目录，需执行如下命令删除该目录。

```
rm -rf /opt/portadv
```

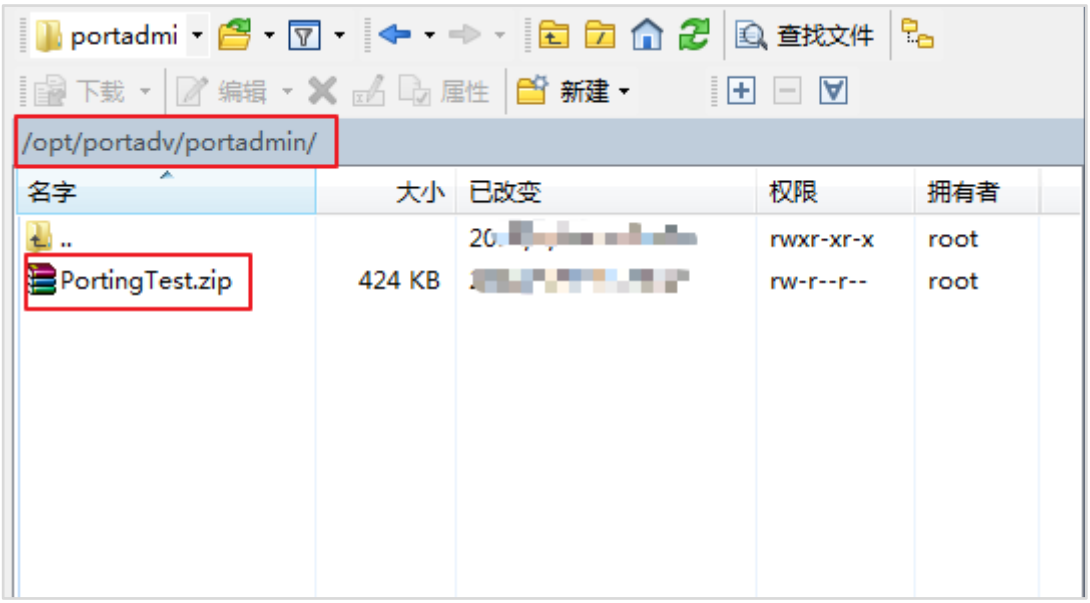
步骤 8 等待 3~5 分钟完成安装，出现以下显示，表示安装成功。

```
  Apply all migrations: admin, auth, contenttypes, operationlog, sessions, taskm
anager, usermanager
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying operationlog.0001_initial... OK
  Applying sessions.0001_initial... OK
  Applying taskmanager.0001_initial... OK
Successfully installed porting advisor in /opt/portadv/tools
[root@ecs-test Porting-advisor-x86_64-linux-1.0.5]#
```

1.4.2 使用代码迁移工具进行代码移植

步骤 1 打开 WinSCP 工具，以 root 用户登录 CentOS 操作系统。

步骤 2 将源代码文件传到 CentOS 操作系统的”/opt/portadv/portadmin”路径下面。配置实验任务。



步骤 3　执行如下命令，进入/opt/portadv/portadmin 路径下面。

```
cd /opt/portadv/portadmin
```

步骤 4　执行如下命令，解压刚上传的源代码文件压缩包。

```
unzip PortingTest.zip
```

```
[root@ecs-test ~]# cd /opt/portadv/portadmin
[root@ecs-test portadmin]# ll
total 424
-rw-r--r-- 1 root root 434134 Sep 29 15:02 PortingTest.zip
[root@ecs-test portadmin]# unzip PortingTest.zip
Archive:  PortingTest.zip
   creating: PortingTest/
   creating: PortingTest/examples/
   creating: PortingTest/examples/cmdline/
  inflating: PortingTest/examples/cmdline/commands.c
  inflating: PortingTest/examples/cmdline/commands.h
  inflating: PortingTest/examples/cmdline/main.c
  inflating: PortingTest/examples/cmdline/parse_obj_list.c
  inflating: PortingTest/examples/cmdline/parse_obj_list.h
   creating: PortingTest/examples/distributor/
```

步骤 5　打开本地 PC 机的浏览器，在地址栏输入 https://所记录的 ECS 的 EIP:端口号，例如：https://121.36.19.170:8084），按“Enter”。

步骤 6　在华为鲲鹏代码迁移工具页面，输入用户名和密码。

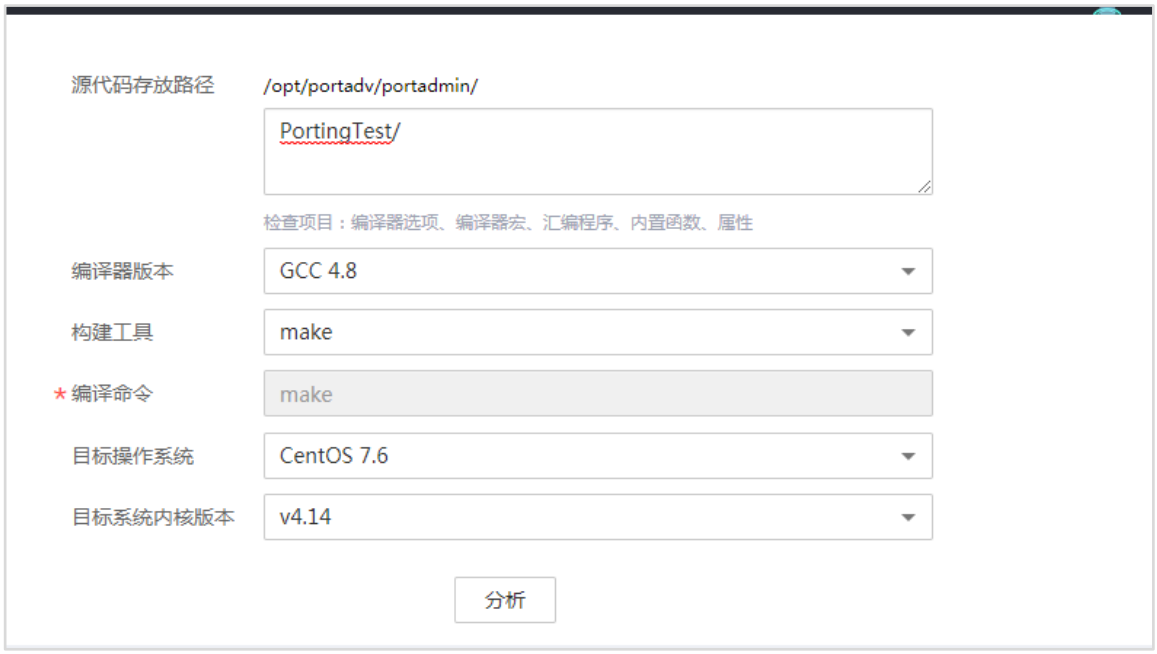
系统的默认用户名为 **portadmin**，默认密码为 **Admin@9000**

步骤 7　单击“登录”。首次登录 Web 的用户，系统提示修改默认密码。将密码改为 XXXXX（按照密码规则自定义密码）。

步骤 8　成功登录后，显示首页界面，界面右上角将显示登录的用户名。

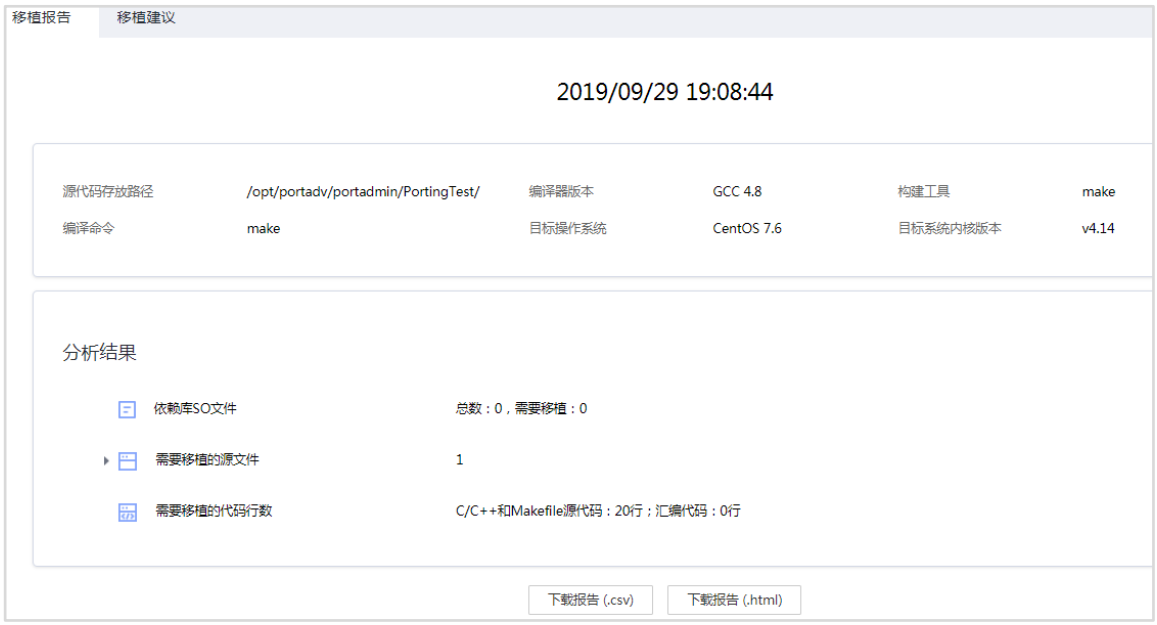
步骤 9　在创建分析任务区对以下参数进行配置。

- 源代码存放路径：PortingTest/
- 编译版本：GCC4.8
- 构建工具：make
- 编译命令：make
- 目标操作系统：CentOS 7.6
- 目标系统内核版本：v4.14



步骤 10　单击“分析”，生成分析报告。

弹窗页面显示任务分析进度，分析完成后，自动跳转至“移植报告”界面。



步骤 11　点击下载报告，可以保存移植报告到本地。

步骤 12　单击“移植建议”页签。进入“移植建议”界面，勾选“显示源代码”。点击“确认”。

步骤 13　选择 C/C++ Source File 查看到“建议源代码”。

步骤 14　返回 putty 登录的 ECS 界面，进入到移植建议的代码 C/C++源文件下，此处为：

```
cd /opt/portadv/portadmin/PortingTest/examples/ip_pipeline
vi main.h
```

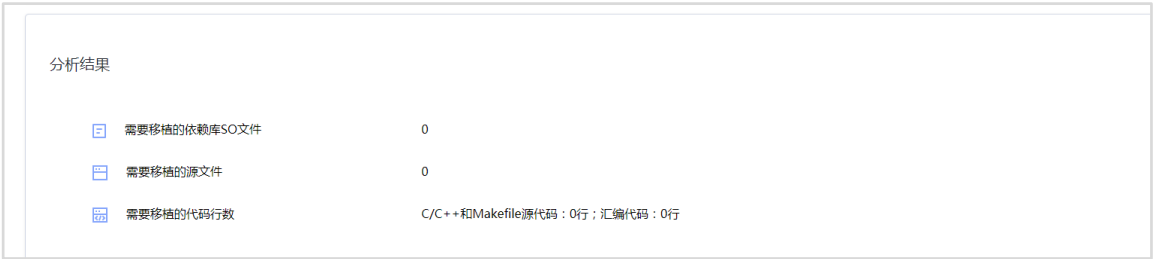
步骤 15　输入 **i**，即可进入 main.h 文件编辑模式，根据步骤 13 中的建议提示逐行修改源代码。

注：此实验仅要求掌握工具使用，不涉及代码修改具体操作，不对代码进行验证，相关实验参考综合实验。

步骤 16　由于此实验仅要求掌握工具使用，不涉及代码修改操作，这里不对代码进行验证，后续综合实验部分会做更进一步练习，因此执行命令：**wq！**，保存并退出代码文件。

1.4.3 结果验证

步骤 1 重复 1.4.2 中步骤 5~步骤 10，对修改后的代码进行移植分析。如果分析结果中，需要移植的依赖库、代码源文件、代码函数均为零的话，则说明没有依赖项和移植修改项，代码修改完成。



1.5 思考题

我们在迁移软件时，包含编译型语言的程序，也包含解释型语言的程序，请思考：

1. 编译型语言和解释型语言分别有哪些？
2. 不同语言代码迁移流程的区别点有哪些？

2 容器迁移实验

2.1 实验介绍

2.1.1 关于本实验

本实验在拥有华为云账号，并且购买了鲲鹏服务器产品的前提下，通过 Docker 安装，构建基础镜像，根据基础镜像安装 Redis，验证 Redis 镜像等 4 个模块的实验练习和实训，让学员掌握容器的迁移指导。

2.1.2 实验目的

- 掌握 Docker 安装。
- 掌握 Docker 构建基础镜像。
- 掌握 Docker 根据基础镜像安装 Redis。
- 掌握验证构建的 Redis 镜像。

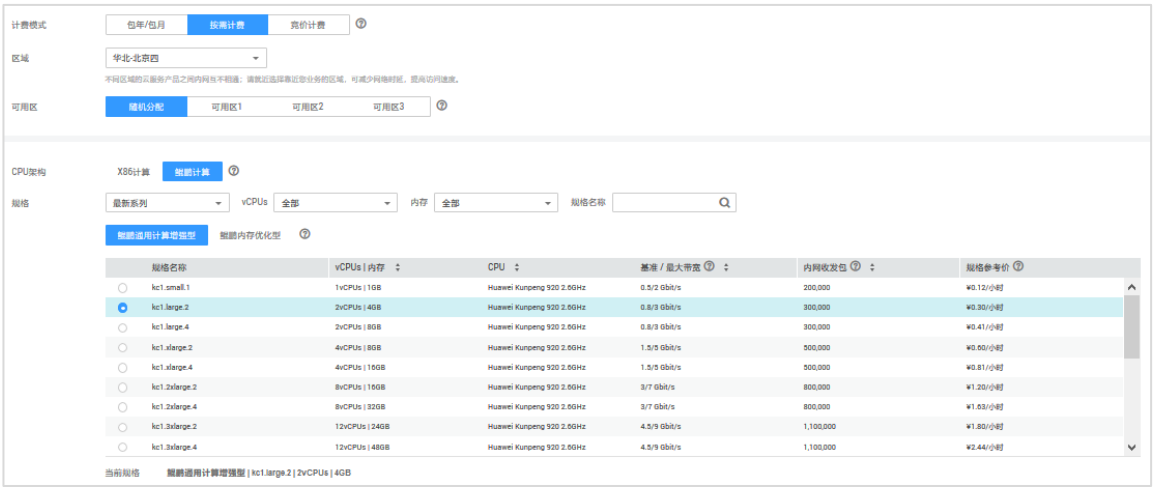
2.2 登录云服务器 ECS

步骤 1 登录华为云账号，在弹性云服务器 ECS 页面下，点击“立即购买”。

https://www.huaweicloud.com/product/ecs.html

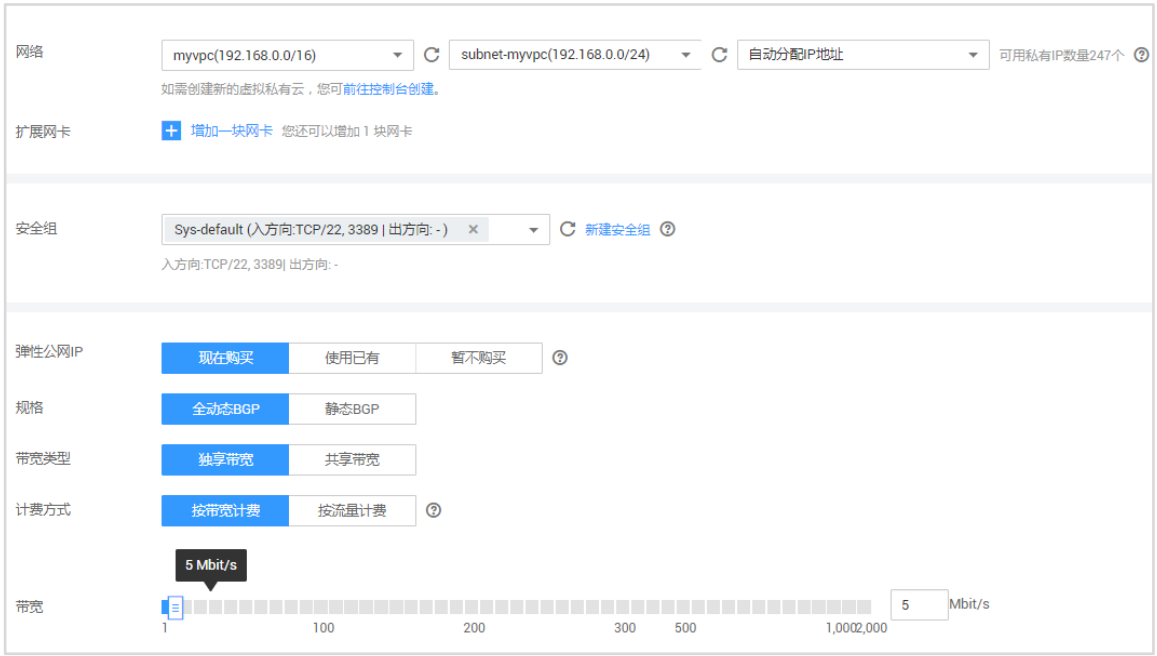
步骤 2 填写如下基础配置信息，然后点击“下一步”。

- 计费模式：按需计费
- 区域：**华北-北京四**
- 可用区：随机分配
- CPU 架构：鲲鹏计算
- 规格：鲲鹏通用计算增强型 kc1.large.2
- 镜像：公共镜像 CentOS7.6 64bit with ARM(40GB)
- 系统盘：高 IO|40G



步骤 3 填写如下网络配置信息（选择实验 1 中已创建好的 VPC 和子网），然后点击“下一步”。

- 网络：选择已创建的网络和子网，如 myvpc 和 subnet-myvpc
- 安全组：Sys-default
- 弹性公网 IP：现在购买
- 规格：全动态 BGP
- 带宽类型：独享带宽
- 计费方式：按带宽计费
- 带宽：5 Mbit/s



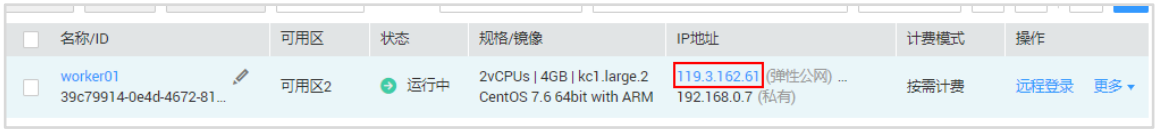
步骤 4 填写如下高级配置信息，然后点击“下一步”。

- 云服务器名称：worker01
- 登录凭证：密码
- 密码/确认密码：XXXXX（按照密码规则自定义密码）

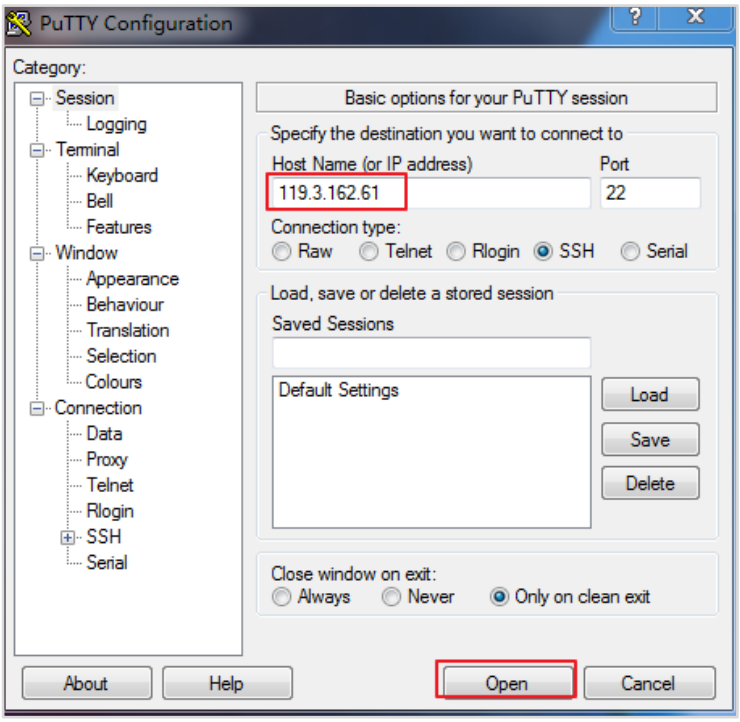
- 云备份：暂不购买

步骤 5 在确认配置界面，勾选“我已经阅读.....”后，点击“立即购买”。

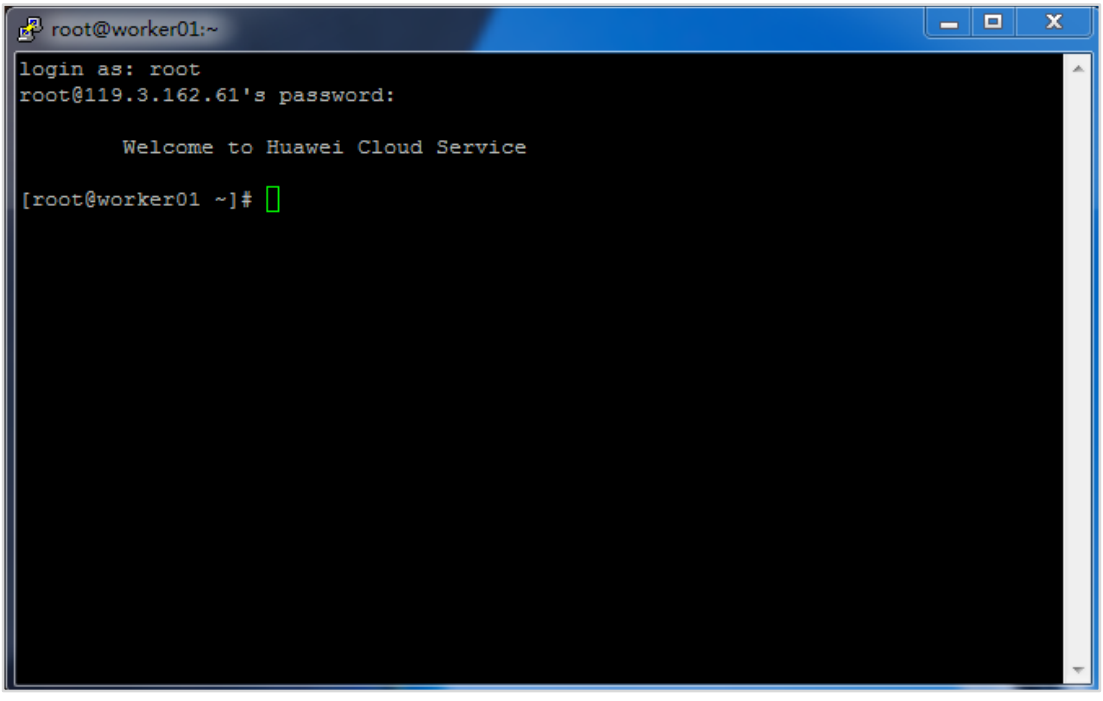
步骤 6 购买完成后，在云服务器控制台可以看到正常运行中的 ECS，记录弹性 IP 地址。



步骤 7 打开 putty，输入弹性 IP 地址后，点击 open。



步骤 8 用 root 用户名，和之前设置的密码登录 ECS。



登录成功后按照以下步骤进行容器迁移实验。

2.3 容器迁移操作

2.3.1 Docker 安装

步骤 1 检查内核版本。

```
uname -r
```

```
[root@worker01 ~]# uname -r
4.14.0-115.5.1.el7a.aarch64
```

说明：Docker 支持的 Centos 版本要求

- CentOS 7，系统为 64 位、系统内核版本为 3.10 以上
- CentOS 6.5 或更高，系统为 64 位、系统内核版本为 2.6.32-431 或者更高版本

查出的 centos7 的系统为 64 位，内核版本为 4.14，符合版本要求。

步骤 2 移除旧的版本。

```
yum remove docker docker-client docker-client-latest docker-common docker-
latest docker-latest-logrotate docker-logrotate docker-selinux docker-engine-
selinux docker-engine
```

```
[root@worker01 ~]# yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-lo
grotate docker-logrotate docker-selinux docker-engine-selinux docker-engine
Loaded plugins: fastestmirror
No Match for argument: docker
No Match for argument: docker-client
No Match for argument: docker-client-latest
No Match for argument: docker-common
No Match for argument: docker-latest
No Match for argument: docker-latest-logrotate
No Match for argument: docker-logrotate
No Match for argument: docker-selinux
No Match for argument: docker-engine-selinux
No Match for argument: docker-engine
No Packages marked for removal
```

步骤 3 安装 Docker 依赖工具。

```
yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
[root@worker01 ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
epel/aarch64/metalink                                                   | 5.2 kB  00:00:00
 * base: mirror.xtom.com.hk
 * epel: hkg.mirror.rackspace.com
 * extras: mirror.xtom.com.hk
 * updates: mirror.xtom.com.hk
base                                                                    | 3.6 kB  00:00:00
docker-ce-stable                                                        | 3.5 kB  00:00:00
epel                                                                    | 5.4 kB  00:00:00
extras                                                                  | 2.9 kB  00:00:00
iRedMail                                                                | 2.9 kB  00:00:00
updates                                                                 | 2.9 kB  00:00:00
(1/6): epel/aarch64/updateinfo                                          | 1.0 MB  00:00:01
epel/aarch64/primary_db        FAILED                                ]  16 kB/s | 1.6 MB  00:11:46 ETA
https://mirror.pregi.net/epel/7/aarch64/repodata/f44e982437a73d1eff8eb69ca0d53674c4b7c8c0867001ccc6f51a67f195c196-pr
imary.sqlite.bz2: [Errno 12] Timeout on https://mirror.pregi.net/epel/7/aarch64/repodata/f44e982437a73d1eff8eb69ca0d
53674c4b7c8c0867001ccc6f51a67f195c196-primary.sqlite.bz2: (28, 'Operation too slow. Less than 1000 bytes/sec transfe
rred the last 30 seconds')
Trying other mirror.
(2/6): extras/7/aarch64/primary_db                                      | 158 kB  00:00:33
(3/6): epel/aarch64/primary_db                                          | 6.6 MB  00:00:06
updates/7/aarch64/primary_db   FAILED                                ] 110 kB/s | 8.5 MB  00:00:41 ETA
http://mirror.0x.sg/centos-altarch/7.7.1908/updates/aarch64/repodata/f81f90e2fbf945e54225c9130ab589cd9bc6ca1bd009d57
0173fd3512a377e34-primary.sqlite.bz2: [Errno 12] Timeout on http://mirror.0x.sg/centos-altarch/7.7.1908/updates/aarc
```

步骤 4 安装 Docker。

```
yum -y install docker
```

```
[root@worker01 ~]# yum -y install docker
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.huaweicloud.com
 * epel: hkg.mirror.rackspace.com
 * extras: mirrors.huaweicloud.com
 * updates: mirrors.huaweicloud.com
Resolving Dependencies
--> Running transaction check
---> Package docker.aarch64 2:1.13.1-103.git7f2769b.el7.centos will be installed
--> Processing Dependency: docker-common = 2:1.13.1-103.git7f2769b.el7.centos for package: 2:docker-1.13.1-103.git7f2769b.el7
.centos.aarch64
--> Processing Dependency: docker-client = 2:1.13.1-103.git7f2769b.el7.centos for package: 2:docker-1.13.1-103.git7f2769b.el7
.centos.aarch64
--> Processing Dependency: subscription-manager-rhsm-certificates for package: 2:docker-1.13.1-103.git7f2769b.el7.centos.aarc
h64
--> Running transaction check
---> Package docker-client.aarch64 2:1.13.1-103.git7f2769b.el7.centos will be installed
---> Package docker-common.aarch64 2:1.13.1-103.git7f2769b.el7.centos will be installed
--> Processing Dependency: skopeo-containers >= 1:0.1.26-2 for package: 2:docker-common-1.13.1-103.git7f2769b.el7.centos.aarc
h64
--> Processing Dependency: oci-umount >= 2:2.3.3-3 for package: 2:docker-common-1.13.1-103.git7f2769b.el7.centos.aarch64
```

步骤 5 启动 Docker 后台服务。

```
systemctl start docker
```

步骤 6 测试运行 hello-world。

```
docker run hello-world
```

```
[root@worker01 ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
3b4173355427: Pull complete
Digest: sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f2a5301c00
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

由于本地没有 hello-world 这个镜像，所以会下载一个 hello-world 的镜像，并在容器内运行。

步骤 7 查看下载的 hello-world 镜像。

```
docker images
```

```
[root@worker01 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
hello-world         latest              de6f0c40d4e5        8 months ago
4.75kB
```

2.3.2 Docker 构建基础镜像

步骤 1 查找 Docker Hub 上的 centos 镜像。

```
docker search centos
```

```
[root@worker01 ~]# docker search centos
NAME                              DESCRIPTION                                     STARS               OFFICIAL
      AUTOMATED
centos                            The official build of CentOS.                   5566                [OK]

ansible/centos7-ansible           Ansible on Centos7                              123
      [OK]
jdeathe/centos-ssh                CentOS-6 6.10 x86_64 / CentOS-7 7.6.1810 x86…   112
      [OK]
consol/centos-xfce-vnc            Centos container with "headless" VNC session…   99
      [OK]
centos/mysql-57-centos7           MySQL 5.7 SQL database server                   62

imagine10255/centos6-lnmp-php56   centos6-lnmp-php56                              57
      [OK]
tutum/centos                      Simple CentOS docker image with SSH access      45

centos/postgresql-96-centos7      PostgreSQL is an advanced Object-Relational …   39
```

步骤 2 拉取官方的镜像,标签为 7。

```
docker pull arm64v8/centos:7
```

```
[root@worker01 ~]# docker pull arm64v8/centos:7
7: Pulling from arm64v8/centos
4856e02b0d50: Pull complete
Digest: sha256:df89b0a0b42916b5b31b334fd52d3e396c226ad97dfe772848bdd6b00fb42bf0
Status: Downloaded newer image for arm64v8/centos:7
docker.io/arm64v8/centos:7
```

步骤 3 本地镜像列表里查到 REPOSITORY 为 arm64v8/centos,标签为 7 的镜像。

```
docker images arm64v8/centos:7
```

```
[root@worker01 ~]# docker images arm64v8/centos:7
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
arm64v8/centos      7                   0cb4fb73950e        4 weeks ago         239MB
```

步骤 4 使用镜像 arm64v8/centos:7 以交互模式启动一个容器，在容器内执行/bin/bash 命令。

```
docker run -it arm64v8/centos:7 /bin/bash
```

```
[root@worker01 ~]# docker run -it arm64v8/centos:7 /bin/bash
[root@f4603f1f918a /]#
```

此时进入到容器内部，图示中“f4603f1f918a”为容器 ID。

步骤 5 容器内安装 Redis 依赖包。

```
yum install wget gcc make libgcc gcc-c++ glibc-devel -y
```

```
[root@f4603f1f918a /]# yum install wget gcc make libgcc gcc-c++ glibc-devel -y
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: mirror.xtom.com.hk
 * extras: mirror.xtom.com.hk
 * updates: mirror.xtom.com.hk
base                                                                    | 3.6 kB  00:00:00
extras                                                                  | 2.9 kB  00:00:00
updates                                                                 | 2.9 kB  00:00:00
(1/4): extras/7/aarch64/primary_db                                      | 158 kB  00:00:02
(2/4): updates/7/aarch64/primary_db                                     | 234 kB  00:00:02
(3/4): base/7/aarch64/group_gz                                          | 165 kB  00:00:03
base/7/aarch64/primary_db      FAILED
http://mirror-hk.koddos.net/centos-altarch/7.7.1908/os/aarch64/repodata/99dbead306f663ac001823b6437924771b436f3b3542
c22a5c2508abb03818c9-primary.sqlite.bz2: [Errno 12] Timeout on http://mirror-hk.koddos.net/centos-altarch/7.7.1908/o
s/aarch64/repodata/99dbead306f663ac001823b6437924771b436f3b3542c22a5c2508abb03818c9-primary.sqlite.bz2: (28, 'Operat
ion too slow. Less than 1000 bytes/sec transferred the last 30 seconds')
Trying other mirror.
(4/4): base/7/aarch64/primary_db                                        | 4.8 MB  00:00:07
Resolving Dependencies
--> Running transaction check
---> Package gcc.aarch64 0:4.8.5-39.el7 will be installed
--> Processing Dependency: libgomp = 4.8.5-39.el7 for package: gcc-4.8.5-39.el7.aarch64
--> Processing Dependency: cpp = 4.8.5-39.el7 for package: gcc-4.8.5-39.el7.aarch64
--> Processing Dependency: libmpfr.so.4()(64bit) for package: gcc-4.8.5-39.el7.aarch64
--> Processing Dependency: libmpc.so.3()(64bit) for package: gcc-4.8.5-39.el7.aarch64
```

步骤 6 输入 exit，退出容器，查看容器 id。

```
docker ps -a
```

```
[root@worker01 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
  NAMES
f4603f1f918a        arm64v8/centos:7    "/bin/bash"         14 hours ago        Exited (0) 2 minutes ago
  busy_khorana
7c69973c7050        hello-world         "/hello"            14 hours ago        Exited (0) 14 hours ago
  mystifying_brown
```

由此可知，容器 ID 为 f4603f1f918a。

步骤 7 根据容器 ID 创建一个新的镜像作为 Redis 的基础镜像。

```
docker commit -a "huawei.com" -m "redis images" f4603f1f918a t_arm64v8/centos:7
```

其中：

- huawei.com：提交的镜像作者
- f4603f1f918a：容器 ID
- redis images：提交时的说明文字
- t_arm64v8/centos:7：新生成的镜像名称

```
[root@worker01 ~]# docker commit -a "huawei.com" -m "redis images" f4603f1f918a t_arm64v8/centos:7
sha256:ac4b911562b0b8ec231cb4a4e216f17b524f90e088742f557c110de7456f0c25
[root@worker01 ~]#
```

步骤 8 查看新构建的基础镜像。

```
docker images
```

```
[root@worker01 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
t_arm64v8/centos    7                   ac4b911562b0        48 seconds ago      385MB
arm64v8/centos      7                   0cb4fb73950e        4 weeks ago         239MB
hello-world         latest              de6f0c40d4e5        8 months ago        4.75kB
```

由此可知，新镜像 t_arm64v8/centos:7 构建成功。

2.3.3 Docker 根据基础镜像安装 Redis

步骤 1 创建 redis 目录。

```
mkdir -p ~/redis ~/redis/data
```

data 目录将映射为 redis 容器配置的/data 目录，作为 redis 数据持久化的存储目录。

步骤 2 进入创建的 redis 目录，创建 Dockerfile。

```
cd redis
vi Dockerfile
```

步骤 3 输入 i，编辑如下内容到 Dockerfile 中。

```
FROM t_arm64v8/centos:7
WORKDIR /home
RUN wget http://download.redis.io/releases/redis-5.0.5.tar.gz && \
tar -xvzf redis-5.0.5.tar.gz && \
mv redis-5.0.5/ redis && \
rm -f redis-5.0.5.tar.gz
WORKDIR /home/redis
RUN make && make install

EXPOSE 6379
CMD ["redis-server"]
```

步骤 4 完成后点击“ecs”，然后输入:wq 保存退出文档。

步骤 5 通过 Dockerfile 创建 redis 镜像。

```
docker build -t t_arm64v8/centos_redis:5.05 .
```

```
[root@worker01 redis]# docker build -t t_arm64v8/centos_redis:5.05 .
Sending build context to Docker daemon  124.6MB
Step 1/7 : FROM t_arm64v8/centos:7
 ---> ac4b911562b0
Step 2/7 : WORKDIR /home
 ---> Running in 27e0d1db3f7f
Removing intermediate container 27e0d1db3f7f
 ---> 6d947ee18245
Step 3/7 : RUN wget http://download.redis.io/releases/redis-5.0.5.tar.gz &&        tar -xvzf redis-5.0.5.tar.gz &&        m
v redis-5.0.5/ redis &&         rm -f redis-5.0.5.tar.gz
 ---> Running in c4380adfd537
--2019-09-21 02:58:18--  http://download.redis.io/releases/redis-5.0.5.tar.gz
Resolving download.redis.io (download.redis.io)... 109.74.203.151
Connecting to download.redis.io (download.redis.io)|109.74.203.151|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1975750 (1.9M) [application/x-gzip]
Saving to: 'redis-5.0.5.tar.gz'
```

```
Removing intermediate container ac44ef6d7d9a
 ---> 1040c9fb40a2
Step 6/7 : EXPOSE 6379
 ---> Running in 4a8ec827e80a
Removing intermediate container 4a8ec827e80a
 ---> 997030afae9d
Step 7/7 : CMD ["redis-server"]
 ---> Running in cac3f91e9f32
Removing intermediate container cac3f91e9f32
 ---> e732d1ff296b
Successfully built e732d1ff296b
Successfully tagged t_arm64v8/centos_redis:5.05
[root@worker01 redis]#
```

步骤 6 查看创建的 redis 镜像。

```
docker images
```

```
[root@worker01 redis]# docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
t_arm64v8/centos_redis  5.05                e732d1ff296b        21 minutes ago      542MB
t_arm64v8/centos        7                   ac4b911562b0        58 minutes ago      385MB
arm64v8/centos          7                   0cb4fb73950e        4 weeks ago         239MB
hello-world             latest              de6f0c40d4e5        8 months ago        4.75kB
```

说明 Redis 镜像创建成功，镜像名称：t_arm64v8/centos_redis，标签 5.05，镜像 ID e732d1ff296b。

2.3.4 验证 Redis 镜像

步骤 1 运行容器，执行 redis-server。

```
docker run -p 6379:6379 -v $PWD/data:/data -d t_arm64v8/centos_redis:5.05
redis-server --appendonly yes
```

命令说明：

- -p 6379:6379 : 将容器的 6379 端口映射到主机的 6379 端口。
- -v $PWD/data:/data : 将主机中当前目录下的 data 挂载到容器的/data。
- redis-server --appendonly yes : 在容器执行 redis-server 启动命令，并打开 redis 持久化配置。

```
[root@worker01 redis]# docker run -p 6379:6379 -v $PWD/data:/data -d t_arm64v8/centos_redis:5.05 redis-server --appendonly ye
s
566f60cc5f6fedc3ac7c92c09ea36d79c1a9880459141dec492eaac3c3637ddf
```

步骤 2 查看容器启动状态，记录容器 ID，下一步中会通过容器 ID 进入容器。

```
docker ps
```

```
[root@worker01 redis]# docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS
                NAMES
566f60cc5f6f        t_arm64v8/centos_redis:5.05    "redis-server --appe…"   About a minute ago  Up About a minute   0.0.0.0:6
379->6379/tcp   gracious_hofstadter
```

步骤 3 执行 redis-cli 命令连接到刚启动的容器。

```
docker exec -it 566f60cc5f6f redis-cli
127.0.0.1:6379> info
```

其中 566f60cc5f6f 为容器 ID。

```
[root@worker01 redis]# docker exec -it 566f60cc5f6f redis-cli
127.0.0.1:6379> info
# Server
redis_version:5.0.5
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:d66412af2962c379
redis_mode:standalone
os:Linux 4.14.0-115.5.1.el7a.aarch64 aarch64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:4.8.5
process_id:1
run_id:db158c55dcb89559465f6f823cc7f054323b9d8a
tcp_port:6379
uptime_in_seconds:252
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:8757623
executable:/home/redis/redis-server
config_file:
```

说明连接 redis-server 成功。

步骤 4 使用 redis 容器。

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set runkey "hello redis"
OK
127.0.0.1:6379> get runkey
"hello redis"
```

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set runkey "hello redis"
OK
127.0.0.1:6379> get runkey
"hello redis"
127.0.0.1:6379>
```

说明：

- ping 返回 PONG 说明检测到 redis 服务已经启动。
- set runkey "hello redis"：设置 runkey 值为"hello redis"，返回 OK，说明设置成功。
- get runkey：获取 runkey 的值，返回"hello redis"说明与设置的相匹配。

3 资源删除

3.1 删除弹性云服务器及相关资源

完成实验后请务必删除华为云上的收费资源，以免造成不必要的收费。找到创建的弹性云服务器 ECS，按照如下步骤进行删除。

步骤 1 打开云服务器控制台，在需要删除的云服务器后面选择“更多>删除”。

步骤 2 在弹出对话框中勾选“释放云服务器绑定的弹性公网 IP 地址”和“删除云服务器挂载的数据盘”，然后点击“是”。

步骤 3　查看到列表中已没有资源时，表示弹性云服务器已删除。

华为认证 Kunpeng 系列教程

HCIA- Kunpeng Application Developer

应用部署与发布

实验指导手册

版本:1.0

华为技术有限公司

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： http://e.huawei.com

华为认证体系介绍

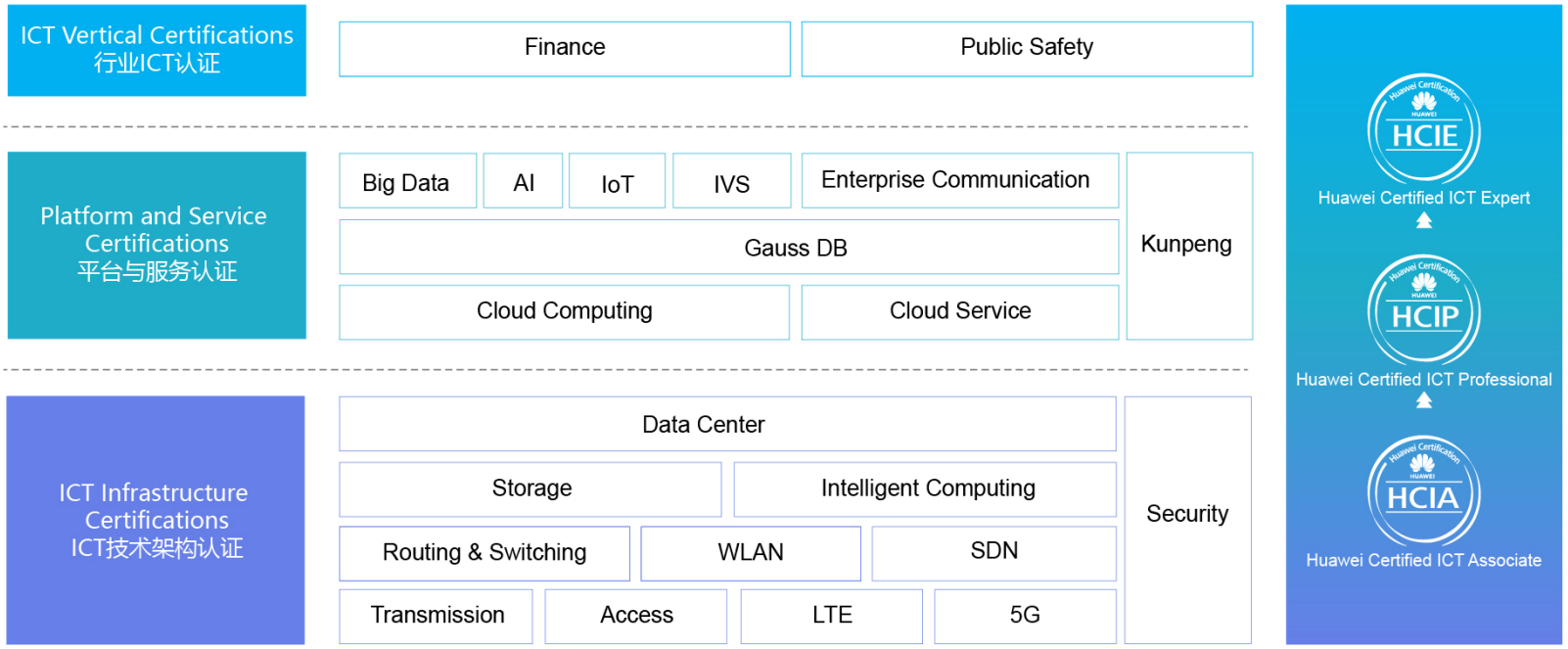
华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的ICT技术架构认证、平台与服务认证、行业ICT认证三类认证，是业界唯一覆盖ICT（Information and Communications Technology 信息通信技术）全技术领域的认证体系。

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。

华为认证HCIA-Kunpeng Application Developer V1.0定位于培养与认证能够应用华为鲲鹏计算平台进行业务部署与迁移，性能测试与调优，常见解决方案设计与规划的工程师。

通过HCIA-Kunpeng Application Developer V1.0认证，您将掌握对鲲鹏计算平台的使用与维护方法，具备对鲲鹏计算平台上的应用进行全生命周期管理的能力，能够胜任鲲鹏平台的应用开发和运维岗位。

Huawei Certification



前 言

简介

本书为 HCIA-Kunpeng Application Developer 认证培训教程，适用于准备参加 HCIA-Kunpeng Application Developer 考试的学员或者希望了解华为鲲鹏架构下的应用部署与发布相关操作的读者。

内容描述

本实验指导书书共包含 3 个实验，从虚拟机创建开始，逐步介绍了在华为鲲鹏架构下的应用部署与发布步骤。

- 实验一为开发环境搭建，通过华为公有云申请 ECS，搭建并验证 x86 编译服务器交叉编译环境。
- 实验二为 rpm 打包实验，本书以 Redis 打包 rpm 为例，介绍通过 Redis 源码进行 rpm 打包。
- 实验三云服务私有镜像的制作，通过前述的步骤，将已经创建的 ECS 制作成私有镜像。

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：

- 具有基本的 Linux 操作命令，同时了解 rpm 概念。
- 了解 Redis 相关概念，并熟悉基本的 SQL 语句。

实验环境说明

组网说明

本实验环境部署在华为公有云上，需要创建 VPC 及子网，实验所涉及到的 ECS 在一个子网内。

环境需求介绍

为了满足实验需要，建议每套实验环境采用以下配置：

名称	版本	规格	用途
鲲鹏ECS	CentOS 7.6	kc1.large.2 | 2 vCPUs | 4GB	用于验证可执行程序
x86 ECS	CentOS 7.6	s6.large.2 | 2 vCPUs | 4GB	用于搭建编译环境
Putty	/	/	ssh登录服务器

实验拓扑

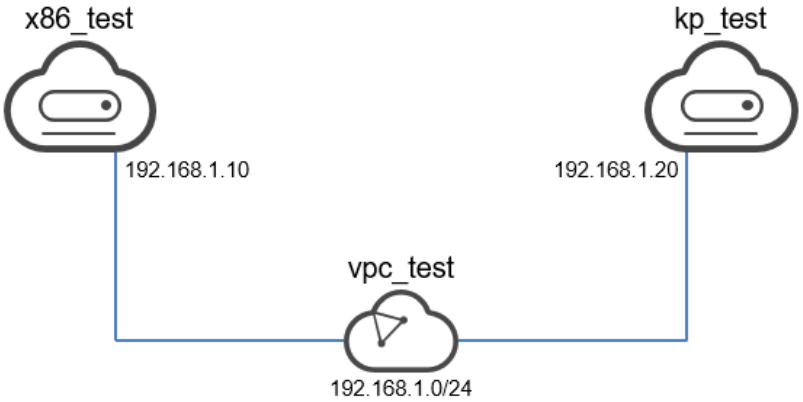


图3-1 实验拓扑

目录

1 开发环境搭建

1.1 实验介绍

1.1.1 关于本实验

本实验通过拥有华为公有云账号的前提下，选择一款适合自己的云服务器进行购买，在购买的 x86 云服务器上搭建交叉编译环境，并且编译生成 ARM 平台上的可执行代码。

1.1.2 实验目的

- 掌握云服务器购买流程。
- 掌握交叉编译的基本原理。
- 掌握 x86 服务器上交叉编译器安装。
- 验证交叉编译是否成功。

1.2 购买云服务器

1.2.1 购买云服务器

步骤 1 打开华为公有云 www.huaweicloud.com 网页，点击右上角“登录”，在登录窗口中输入账号密码登录华为公有云。

步骤 2 选择“产品”→“基础服务”→“虚拟私有云 VPC”。

步骤 3 点击“访问控制台”，进入网络控制台 VPC 页签。在网络控制台 VPC 页签点击右上角“创建虚拟私有云”选项。

步骤 4 按照如下表格配置 VPC 属性，然后点击右下角“立即创建”。

参数	配置
区域	华北-北京四
名称	vpc-test
网段	192.168.1.0/24
子网可用区	可用区 1
子网名称	subnet-test
子网网段	192.168.1.0/24

步骤 5 展开网络控制台左侧列表的访问控制，选择“安全组”，进入安全组页签。点击右上角“创建安全组”。

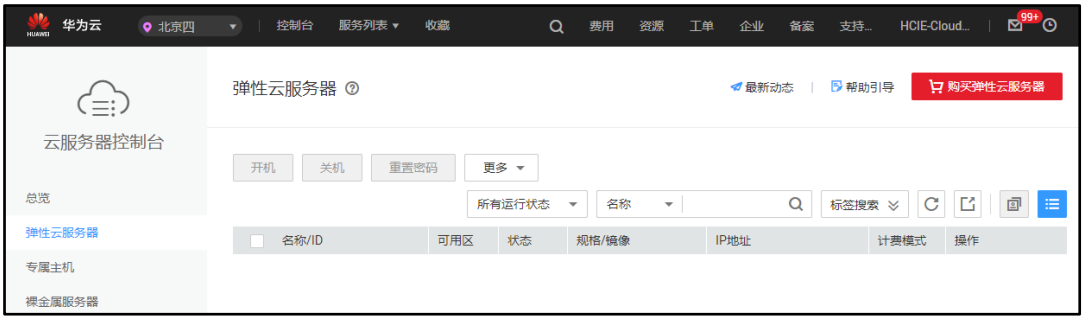
步骤 6 在“创建安全组”窗口中配置模板选择为“开放全部端口”，名称设置为“sg-test”，然后点击“确定”，完成安全组的创建。

步骤 7 展开网络控制台左侧导航栏的“弹性公网 IP 和共享带宽”，选择“共享带宽”，进入共享带宽页签。

步骤 8 点击右上角“购买共享带宽”，按照如下参数配置共享带宽参数，然后点击“立即购买”，在订单详情界面点击“提交”。

参数	配置
计费模式	按需计费
区域	华北-北京四
计费方式	按带宽计费
带宽大小	5 Mbit/s
带宽名称	bandwidth-test

步骤 9 选择“服务列表”→“计算”→“弹性云服务器 ECS”，进入云服务器控制台的弹性云服务器页签。

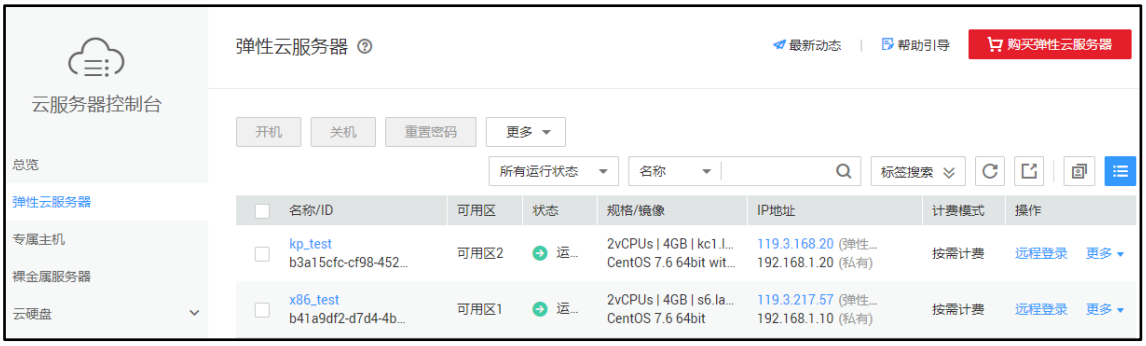


步骤 10 点击“购买弹性云服务器”，按如下参数分别购买 x86_test 和 kp_test 两台弹性云服务器。

参数	kp_test 配置	x86_test 配置
计费模式	按需计费	按需计费
区域	华北-北京四	华北-北京四
CPU 架构	鲲鹏计算	X86 计算
规格	kc1.large.2 | 2vCPUs | 4GB	s6.large.2 | 2vCPUs | 4GB
公共镜像	Centos7.6 64bit with ARM	Centos7.6 64bit
系统盘	高 I/O，40GB	高 I/0，40GB
网络	vpc-test | subnet-test | 手动分配 IP 地址 | 192.168.1.20	vpc-test | subnet-test | 手动分配 IP 地址 | 192.168.1.10
安全组	sg-test	sg-test
弹性公网 IP	现在购买	现在购买
弹性公网 IP 规格	全动态 BGP	全动态 BGP
宽带类型	共享	共享
宽带名称	bandwidth-test	bandwidth-test
云服务器名称	kp_test	x86_test
登录凭证	密码	密码
用户名	root	root
密码/确认密码	XXXXX（按照密码规则自定义密码）	XXXXX（按照密码规则自定义密码）

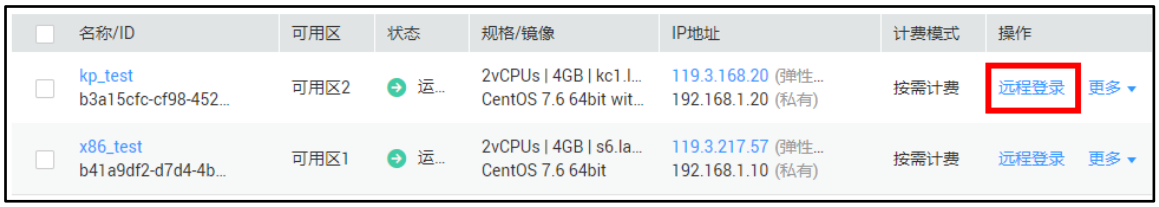
云备份	暂不购买	暂不购买

步骤 11 购买完成后，点击“返回云服务器列表”，查看购买的服务器状态信息。



1.2.2 测试验证

步骤 1 点击 kp_test 弹性云服务器的“远程登录”。



步骤 2 在弹出的 VNC 网页窗口中输入用户名 root 以及密码，弹出如下提示时，表示 ECS 正常。

步骤 3 按照上述两个步骤验证 x86_test 服务器是否正常。

1.3 Linaro 交叉环境搭建

1.3.1 Linaro 交叉环境搭建

步骤 1 参考 2.2.2 步骤 1~步骤 2，登录 x86_test 服务器。

```
CentOS Linux 7 (Core)
Kernel 3.10.0-1062.1.1.el7.x86_64 on an x86_64

x86_test login: root
Password: _

Last login: Sat Oct 12 11:17:40 on tty1

        Welcome to Huawei Cloud Service

[root@x86_test ~]#
```

步骤 2　执行如下命令，安装开发环境：

```
yum -y groupinstall Development Tools
```

步骤 3　在/usr/local/ 目录下创建 ARM-toolchain 目录。

```
mkdir /usr/local/ARM-toolchain
```

步骤 4　进入/usr/local/ARM-toolchain/目录，使用 wget 下载 gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz。

```
cd /usr/local/ARM-toolchain/
wget https://releases.linaro.org/components/toolchain/binaries/latest-5/aarch64-linux-gnu/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz
```

```
[root@x86_test ARM-toolchain]# wget https://releases.linaro.org/components/toolchain/binaries/latest
aro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz
--2019-10-12 11:27:39--  https://releases.linaro.org/components/toolchain/binaries/latest-5/aarch64-
017.10-x86_64_aarch64-linux-gnu.tar.xz
Resolving releases.linaro.org (releases.linaro.org)... 13.228.101.204
Connecting to releases.linaro.org (releases.linaro.org)|13.228.101.204|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://publishing-ap-linaro-org.s3.amazonaws.com/releases/components/toolchain/binaries/l
c-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz?Signature=G9GsZxUxCUZo2hkFiRwIbkdZ4U8%3D&Expi
d=AKIAIELXU2RYNAHFUP7A [following]
--2019-10-12 11:27:42--  https://publishing-ap-linaro-org.s3.amazonaws.com/releases/components/toolc
64-linux-gnu/gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz?Signature=G9GsZxUxCUZo2hkFiRwI
2&AWSAccessKeyId=AKIAIELXU2RYNAHFUP7A
Resolving publishing-ap-linaro-org.s3.amazonaws.com (publishing-ap-linaro-org.s3.amazonaws.com)... 5
Connecting to publishing-ap-linaro-org.s3.amazonaws.com (publishing-ap-linaro-org.s3.amazonaws.com)|
ed.
HTTP request sent, awaiting response... 200 OK
Length: 94218924 (90M) [application/octet-stream]
Saving to: 'gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz'

 3% [==>                                                                    ] 3,419,
```

步骤 5　等待下载 100%完成后，解压压缩包。

```
tar -Jxvf gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu.tar.xz
# 修改目录名称
mv gcc-linaro-5.5.0-2017.10-x86_64_aarch64-linux-gnu linaro
```

步骤 6　修改环境变量

```
vim /etc/profile
```

按 i 键进入插入模式，按上下左右键将光标移动到最后一行的最末尾，按回车添加新的一行，在新的一行输入如下内容，然后按 esc 键退出插入模式，按 冒号+wq 键保存退出。

```
export PATH=$PATH:/usr/local/ARM-toolchain/linaro/bin/
```

步骤 7　更新环境变量

source /etc/profile

步骤 8 输入如下命令查看 aarch64-gcc 版本信息，显示如图所示信息表示配置正确。

```
aarch64-linux-gnu-gcc -v
```

```
[root@x86_test ARM-toolchain]# aarch64-linux-gnu-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/usr/local/ARM-toolchain/linaro/bin/../libexec/gcc/aarch64-linux-gnu/5.5.0/lto-wrapper
Target: aarch64-linux-gnu
Configured with: '/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/snapshots/gcc.git~linaro-5.5-2017.10/configure' SHELL=/bin/bash --with-mpc=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --with-mpfr=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gmp=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu --with-gnu-as --with-gnu-ld --disable-libmudflap --enable-lto --enable-shared --without-included-gettext --enable-nls --disable-sjlj-exceptions --enable-gnu-unique-object --enable-linker-build-id --disable-libstdcxx-pch --enable-c99 --enable-clocale=gnu --enable-libstdcxx-debug --enable-long-long --with-cloog=no --with-ppl=no --with-isl=no --disable-multilib --enable-fix-cortex-a53-835769 --enable-fix-cortex-a53-843419 --with-arch=armv8-a --enable-threads=posix --enable-multiarch --enable-libstdcxx-time=yes --with-build-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/sysroots/aarch64-linux-gnu --with-sysroot=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu/aarch64-linux-gnu/libc --enable-checking=release --disable-bootstrap --enable-languages=c,c++,fortran,lto --build=x86_64-unknown-linux-gnu --host=x86_64-unknown-linux-gnu --target=aarch64-linux-gnu --prefix=/home/tcwg-buildslave/workspace/tcwg-make-release/builder_arch/amd64/label/tcwg-x86_64-build/target/aarch64-linux-gnu/_build/builds/destdir/x86_64-unknown-linux-gnu
Thread model: posix
gcc version 5.5.0 (Linaro GCC 5.5-2017.10)
```

1.3.2 测试验证

步骤 1 在/tmp/目录下创建目录用于保存代码。

```
mkdir /tmp/test/
```

步骤 2 进入/tmp/test/目录，创建 hello.c 文件，并按如下所示编写该文件。

```
cd /tmp/test/
vim hello.c
# 按 i 键进入插入模式，按如下所示输入信息，然后按 esc 键退出插入模式，按 冒号+wq 键保存退出。
```

```
#include <stdio.h>
int main(void)
{
printf("hello linux\n");
return 0;
}
```

步骤 3 执行如下命令，交叉编译。

```
aarch64-linux-gnu-gcc -o kp-hello hello.c
```

步骤 4 执行编译后的脚本。

```
./kp-hello
# 此时会有报如下错误
```

```
[root@x86_test test]# sh kp-hello
kp-hello: kp-hello: cannot execute binary file
```

步骤 5 执行如下命令，将脚本拷贝至 kp_test 服务器。

```
scp kp-hello 192.168.1.20:/tmp
```

依次输入“yes”以及 kp_test 服务器 root 密码，完成拷贝。

步骤 6 参考 1.2.2 步骤 1~步骤 2，登录 kp_test 服务器。

步骤 7 执行如下命令，验证脚本执行。

```
/tmp/kp-hello
```

当出现如图所示信息时，表示执行成功。

2 Redis rpm 打包

2.1 实验介绍

2.1.1 关于本实验

本实验主要介绍利用 Redis 源码进行 rpm 包的打包操作方法。

2.1.2 实验目的

- 能将 Redis 的源码包打包成 rpm 包。

2.2 Redis rpm 打包

步骤 1 参考 2.2.2 步骤 1~步骤 2，登录 kp_test 服务器。

步骤 2 安装 rpmbuild 工具。

```
yum -y install rpm-build
```

步骤 3 安装 wget 工具。

```
yum -y install wget
```

步骤 4 使用 wget 下载 Redis 源代码。

```
wget http://download.redis.io/releases/redis-4.0.9.tar.gz
```

步骤 5 等待进度达到 100%下载完成后将下载的压缩包拷贝到~/rpmbuild/SOURCES/目录下。

```
# 创建一个 rpmbuild 目录。此步骤会报错，属于正常现象。
rpmbuild -ba test
# 移动源码包到 SOURCES 目录。
mv redis-4.0.9.tar.gz ~/rpmbuild/SOURCES/
```

步骤 6　进入~/rpmbuild/SPECS/目录，新建并按要求编辑 redis.spec 文件。

```
cd ~/rpmbuild/SPECS/
vim redis.spec
# 按 i 键进入插入模式，删除原有内容，再输入如下内容后，按 esc 键退出插入模式，再按 冒号+wq 保存退出。
```

```
Name: redis
Version: 4.0.9
Release: 1%{?dist}
Summary: This is a RedisDB
License: GPL
URL: https://redis.io
Source0: redis-4.0.9.tar.gz
BuildRequires: gcc
#Requires: zlib-devel,readline-devel

%description
RedisDB
%prep
%setup -q
%build
make %{?_smp_mflags}
%install
make install PREFIX=%{buildroot}%{_prefix}
install -p -D -m 644 %{name}.conf %{buildroot}%{_sysconfdir}/%{name}.conf
chmod 755 %{buildroot}%{_bindir}/%{name}-*
mkdir -p %{buildroot}%{_sbindir}
mv %{buildroot}%{_bindir}/%{name}-server %{buildroot}%{_sbindir}/%{name}-server
%clean
rm -rf %{buildroot}
%files
%defattr(-,root,root,-)
%{_bindir}/%{name}-*
%{_sbindir}/%{name}-*
%config(noreplace)%{_sysconfdir}/%{name}.conf
%changelog
```

步骤 7　输入如下命令生成二进制版本的 rpm 包。

```
rpmbuild -bb redis.spec
```

步骤 8　输入 ls ~/rpmbuild/RPMS/aarch64 查看刚生成的 rpm 包，若显示如图所示，表示正常。

```
ls ~/rpmbuild/RPMS/aarch64
# 回显
```

```
[root@kp-test SPECS]# ls ~/rpmbuild/RPMS/aarch64/
redis-4.0.9-1.el7.aarch64.rpm  redis-debuginfo-4.0.9-1.el7.aarch64.rpm
```

步骤 9 安装 Redis。

```
rpm -ivh ~/rpmbuild/RPMS/aarch64/redis-4.0.9-1.el7.aarch64.rpm
```

步骤 10 查看已安装的 Redis 客户端版本

```
redis-cli -v
# 回显
```

```
[root@kp-test aarch64]# redis-cli -v
redis-cli 4.0.9
```

3 Linux 私有镜像制作

3.1 实验介绍

3.1.1 关于本实验

本指导书以“通过云服务器创建 Linux 系统盘镜像”为例，指导用户如何制作 Linux 私有镜像。

3.1.2 实验目的

- 熟悉 Redis 数据库简单操作
- 熟悉私有镜像制作流程

3.2 环境数据准备

步骤 1　参考 2.2.2 步骤 1~步骤 2，登录 kp_test 服务器。

步骤 2　执行如下命令，查看 Redis 服务端版本。

```
redis-server -v
# 回显
```

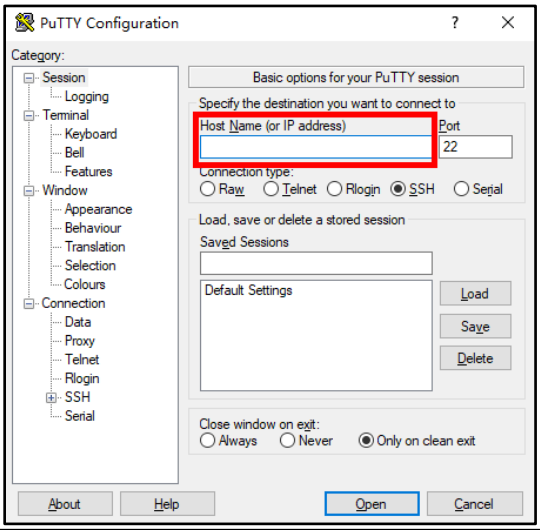
```
[root@kp-test aarch64]# redis-server -v
Redis server v=4.0.9 sha=00000000:0 malloc=jemalloc-4.0.3 bits=64 build=6f57f8d6fada4ec4
```

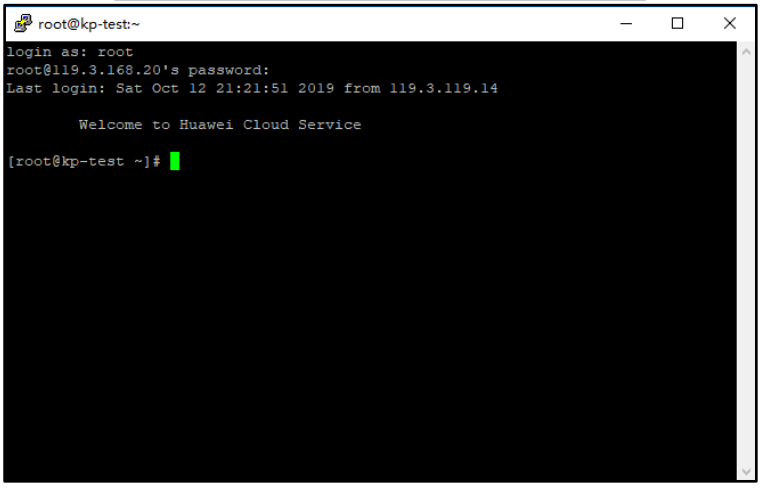
步骤 3　执行如下命令，启动 Redis。

```
redis-server
# 此界面不要关闭
```

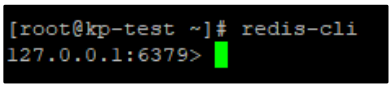
步骤 4 在云服务器控制台的弹性云服务器页签查看 kp_test 服务器弹性 IP 地址。

步骤 5 打开自己电脑上的 putty 工具，将获取到的弹性 IP 地址写入“Host Name (or IP address)处”，然后点击“Open”若有提示框，选择“yes”，然后输入用户名 root 以及密码，登录 kp_test 服务器。



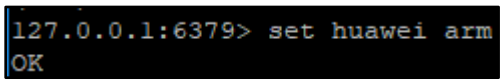


步骤 6 输入 redis-cli 连接数据库服务器。



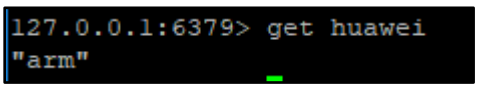
步骤 7　输入 set huawei arm，写入数据库。

```
127.0.0.1:6379> set huawei arm
ok
```



步骤 8　输入 get huawei，获取 K=huawei 的值。

```
127.0.0.1:6379> get huawei
“arm”
```



步骤 9　按 Ctrl+D 退出 Redis 连接，同时，在 kp_test 的 VNC 窗口中按 Ctrl+C，关闭 Redis 数据库。关闭 putty 窗口，关闭 kp_test 服务器 VNC 窗口。

3.3 镜像制作

步骤 1　在云服务器控制台界面的左侧导航栏，点击“镜像服务器”，进入镜像服务页签。

步骤 2　点击“创建私有镜像”，按照下表信息配置私有镜像参数，配置好后点击“立即创建”。在信息汇总页面确认信息无误后继续完成镜像创建，然后返回镜像列表。

参数	配置
区域	华北-北京四
创建方式	系统盘镜像
镜像源	云服务器 | kp_test
镜像名称	image_kp_redis

步骤 3　等待镜像创建完成后，可以在镜像服务页签的私有镜像处看到已经创建的镜像。

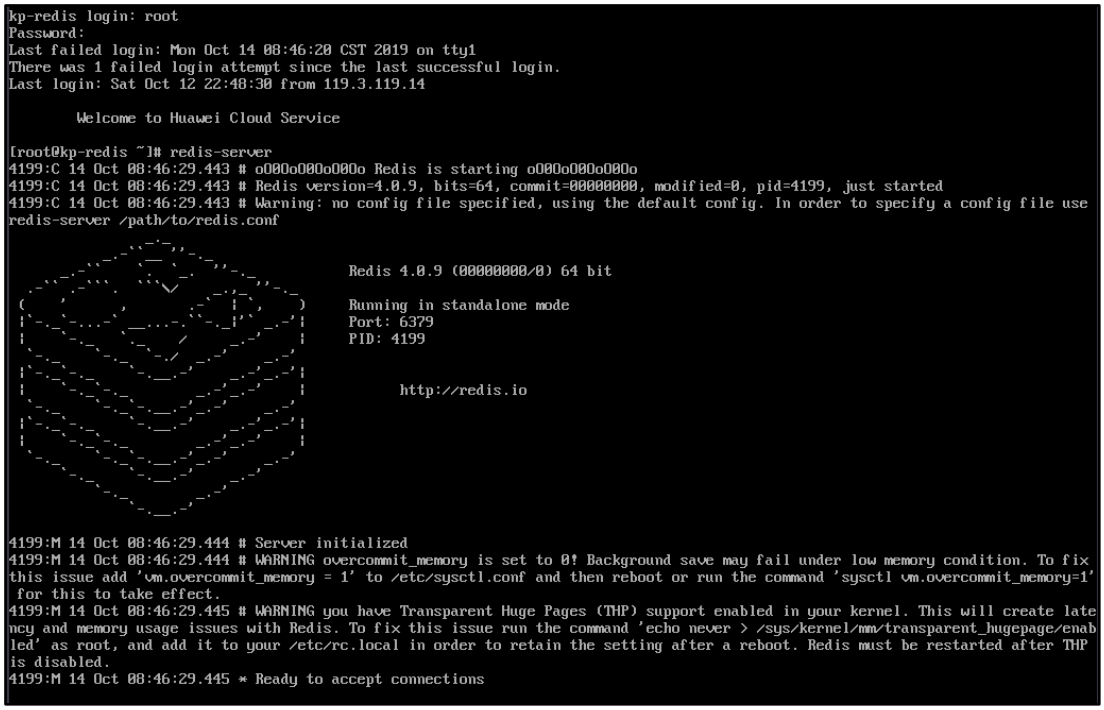
步骤 4　进入弹性云服务器页签，点击“购买弹性云服务器”。

步骤 5　按照下表配置弹性云服务器参数，完成弹性云服务器的创建。

参数	kp_redis 配置
计费模式	按需计费
区域	华北-北京四
可用区	随机分配
CPU 架构	鲲鹏计算
规格	kc1.large.2
镜像	私有镜像 | image_kp_redis
网络	vpc-test | subnet-test | 自动分配 IP 地址
安全组	sg-test
弹性公网 IP	现在购买

弹性公网 IP 规格	全动态 GGP
带宽类型	共享带宽
带宽名称	bandwidth-test
云服务器名称	kp_redis
登录凭证	密码
密码/确认密码	XXXXX（按照密码规则自定义密码）
云备份	暂不购买
高级选择	不勾选

步骤 6 返回弹性云服务器列表，等待 kp_redis 创建完成后，点击“远程登录”，输入用户名 root 及密码登录 kp_redis 云服务器。输入 redis-server 启动 Redis。



步骤 7 参考 4.2 步骤 5，使用自己电脑上的 putty 远程登录到 kp_redis 云服务器。

步骤 8 输入如下命令，连接 Redis。

```
redis-cli
```

步骤 9 输入 set huawei arm，然后再输入 get huawei，查询 key=huawei 的值，验证 Redis 服务是否正常。

```
127.0.0.1:6379> set huawei arm
ok
127.0.0.1:6379> get huawei
 “arm”
```

```
127.0.0.1:6379> set huawei arm
OK
127.0.0.1:6379> get huawei
"arm"
```

华为认证 Kunpeng 系列教程

HCIA-Kunpeng Application Developer

华为鲲鹏平台应用软件移植调优综合实验指导手册

版本:1.0

华为技术有限公司

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： http://e.huawei.com

华为认证体系介绍

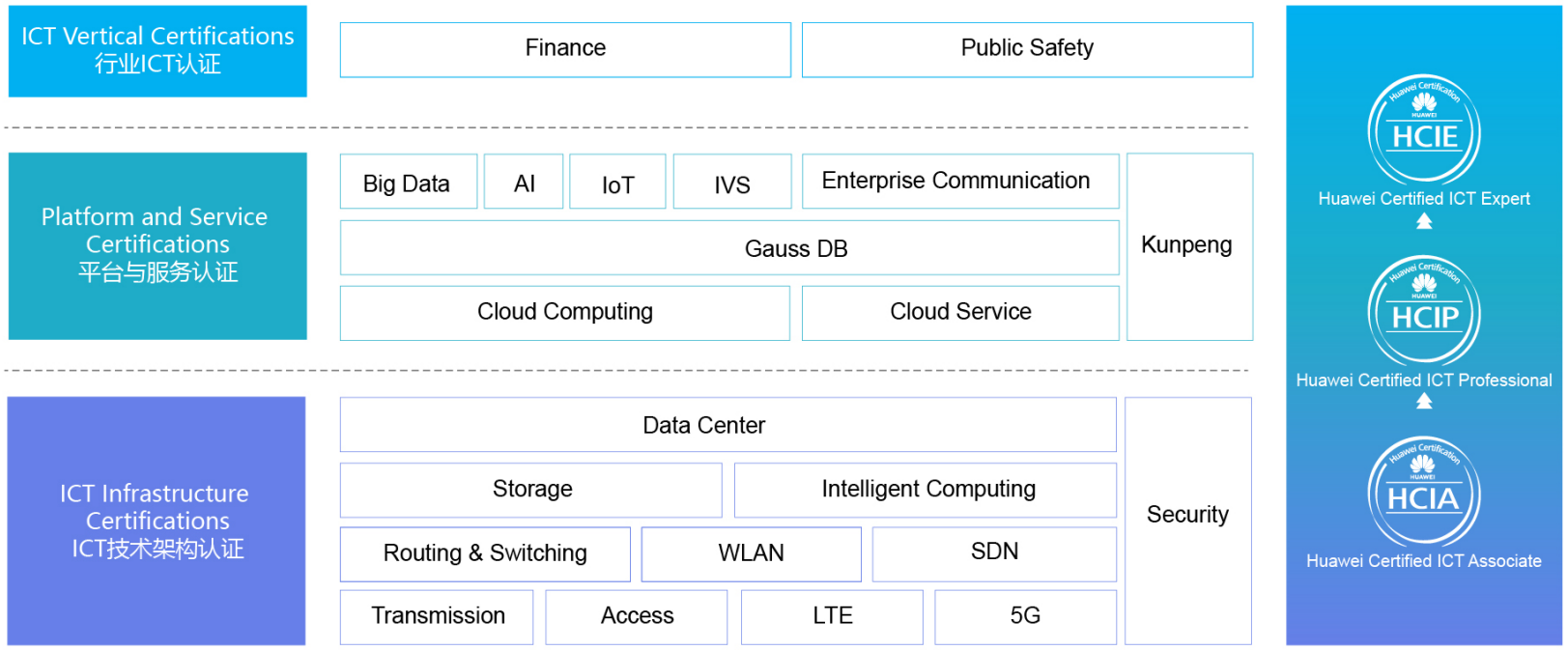
华为认证是华为公司基于“平台+生态”战略，围绕“云-管-端”协同的新ICT技术架构，打造的ICT技术架构认证、平台与服务认证、行业ICT认证三类认证，是业界唯一覆盖ICT（Information and Communications Technology 信息通信技术）全技术领域的认证体系。

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。

华为认证HCIA-Kunpeng Application Developer V1.0定位于培养与认证能够应用华为鲲鹏计算平台进行业务部署与迁移，性能测试与调优，常见解决方案设计与规划的工程师。

通过HCIA-Kunpeng Application Developer V1.0认证，您将掌握对鲲鹏计算平台的使用与维护方法，具备对鲲鹏计算平台上的应用进行全生命周期管理的能力，能够胜任鲲鹏平台的应用开发和运维岗位。

Huawei Certification



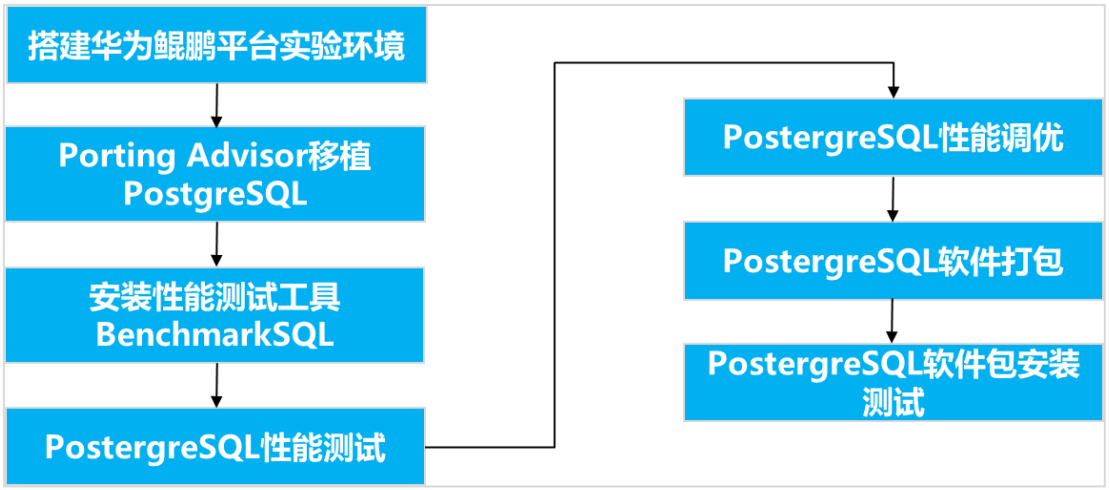
前 言

简介

本书为 HCIA-Kunpeng Application Developer 认证培训教程，适用于准备参加 HCIA-Kunpeng Application Developer 考试的学员或者希望了解华为鲲鹏平台 PostgreSQL 相关性能测试、性能调优以及软件打包和安装等相关技术的读者。

内容描述

本实验以数据库 PostgreSQL 为例，介绍在华为鲲鹏平台上对应用软件进行软件移植、性能测试、性能调优等全流程的综合实验。



本实验指导书共包含 8 个实验，从搭建华为鲲鹏平台实验环境开始，逐一介绍了 PostgreSQL 通过源码移植到华为鲲鹏平台的配置与实现。

- 实验一为通过购买 3 台华为云 ECS 云服务器搭建华为鲲鹏平台实验环境。
- 实验二为通过 Porting Advisor 工具对 PostgreSQL 源码包进行移植分析，并按修改建议对源码进行修改实现代码移植。
- 实验三为在华为鲲鹏平台上通过源码安装部署 BenchmarkSQL。
- 实验四为在华为鲲鹏平台上安装性能测试工具 BenchmarkSQL。
- 实验五为通过性能测试工具 BenchmarkSQL 对 PostgreSQL 进行性能测试。
- 实验六为通过分析性能测试结果对 PostgreSQL 进行性能调优。
- 实验七为通过 rpmbuild 工具在华为鲲鹏平台上制作 PostgreSQL 的 RPM 安装包。

- 实验八为对制作完成的 PostgreSQL 的 RPM 安装包进行安装测试，完成 PostgreSQL 的快速安装部署。

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：

- 具有基本的云服务基础知识以及 Linux 操作系统基础知识。

实验环境说明

组网说明

本实验环境面向准备 HCIA-Kunpeng 考试的鲲鹏应用开发工程师。每套实验环境包括 3 台华为云 ECS 云服务器。每套实验环境适用于 1 名学员同时上机操作。

设备介绍

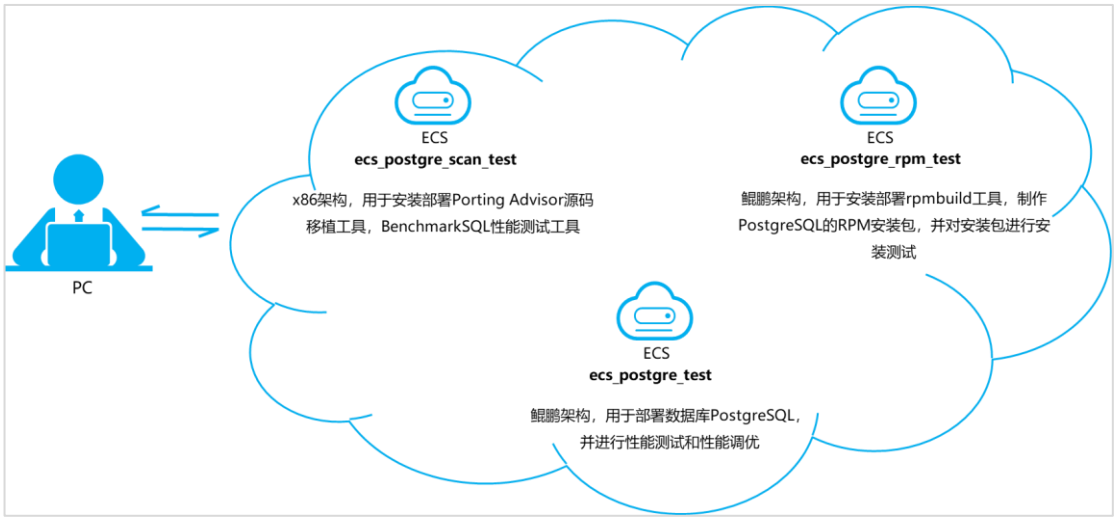
为了满足 HCIA-Kunpeng 实验需要，建议每套实验环境采用以下配置：

3 台华为云 ECS 云服务器及其配置、OS 版本的对应关系如下：

表3-1 设备参数表

设备名称	云主机名称	配置	OS版本	备注
x86云服务器	ecs_postgre_scan_test	c6.large.2 2vCPUs | 4GB	CentOS 7.6 64bit	x86架构，用于安装部署Porting Advisor源码移植工具，BenchmarkSQL性能测试工具。
鲲鹏云服务器	ecs_postgre_test	kc1.2xlarge.2 8vCPUs | 16GB	CentOS 7.6 64bit	鲲鹏架构，用于部署数据库PostgreSQL，并进行性能测试和性能调优。
鲲鹏云服务器	ecs_postgre_rpm_test	kc1.large.2 2vCPUs | 4GB	CentOS 7.6 64bit	鲲鹏架构，用于安装部署rpmbuild工具，制作PostgreSQL的RPM安装包，并对安装包进行安装测试。

实验拓扑



实验拓扑说明：

在华为云上购买的 3 台 ECS 云服务器之间网络互通，且与 PC 之间也网络互通。

准备实验环境

在实验开始之前，请每组学员确认是否已获取华为云账号和密码。

目录

1 搭建华为鲲鹏平台实验环境

1.1 实验介绍

本实验通过购买华为云 ECS 云服务器搭建华为鲲鹏平台实验环境，主要包含三台华为云 ECS 云服务器：

- 一台是基于 x86 架构，用于安装部署 Porting Advisor 源码移植工具，BenchmarkSQL 性能测试工具。
- 一台是基于鲲鹏架构，用于部署数据库 PostgreSQL，并进行性能测试和性能调优。
- 一台是基于鲲鹏架构，用于安装部署 rpmbuild 工具，制作 PostgreSQL 的 RPM 安装包，并对安装包进行安装测试。

1.2 前提条件

- 已获取登录华为云的账号、用户名和密码。

1.3 登录华为云

1.3.1 操作步骤

步骤 1 使用 PC 上的浏览器访问华为云官网：https://www.huaweicloud.com/?locale=zh-cn，单击页面右上角的“登录”，进入华为云账号登录页面。

步骤 2　单击右下角的“IAM 用户登录（中国站）”，进入华为云 IAM 用户登录页面。

步骤 3　输入账号名，用户名和密码，单击下方的“登录”，登录华为云官网。

--结束

1.4 购买华为云 ECS 云服务器

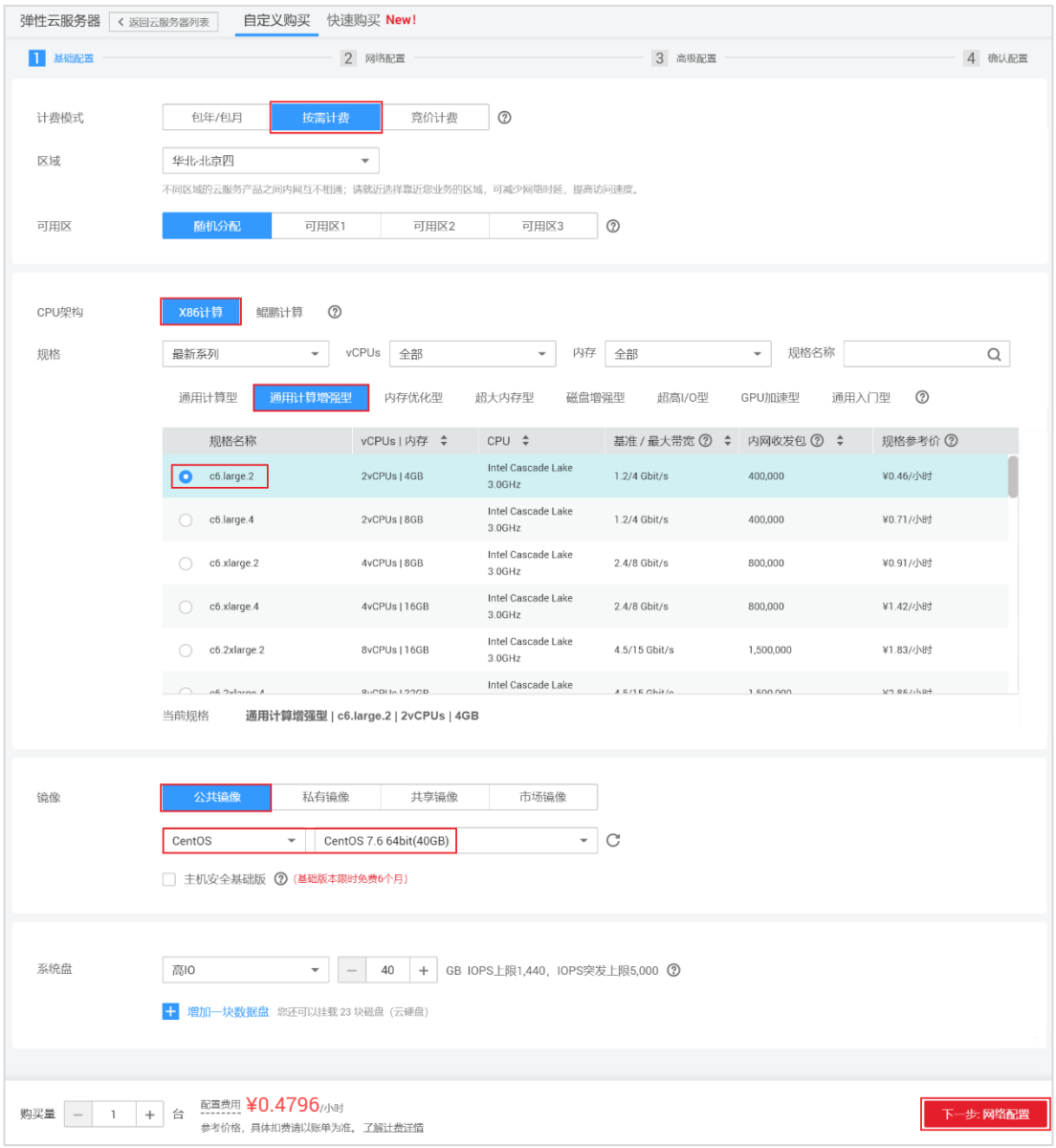
1.4.1 操作步骤

步骤 1　在华为云首页，单击右上角的“控制台”，进入控制台操作页面。

步骤 2　在页面左上角，选择区域“北京四”，单击“服务列表”，选择“计算 > 弹性云服务器 ECS”，进入弹性云服务器列表页面。

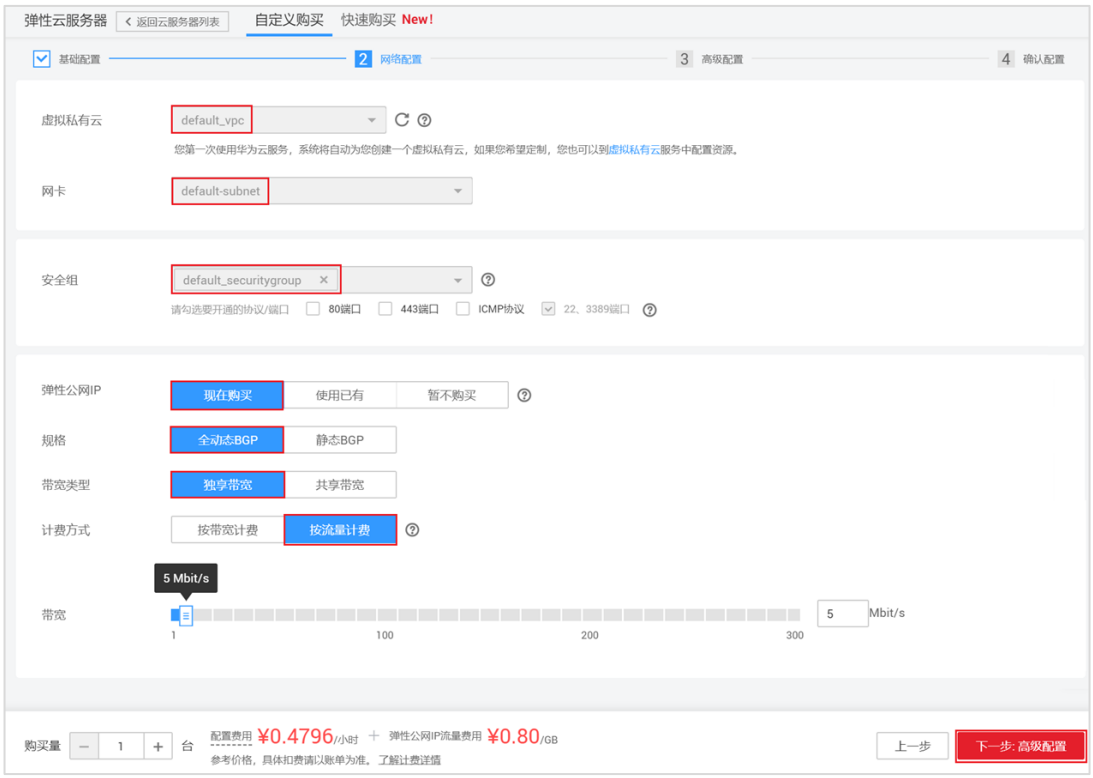
步骤 3　单击页面右上角的“购买弹性云服务器”。

步骤 4　进入弹性云服务器的基础配置页面，基础配置（X86 计算，通用计算增强型，c6.large.2 2vCPUs | 4GB，CentOS 7.6 64bit）如下图所示：



步骤 5　单击“下一步：网络配置”。

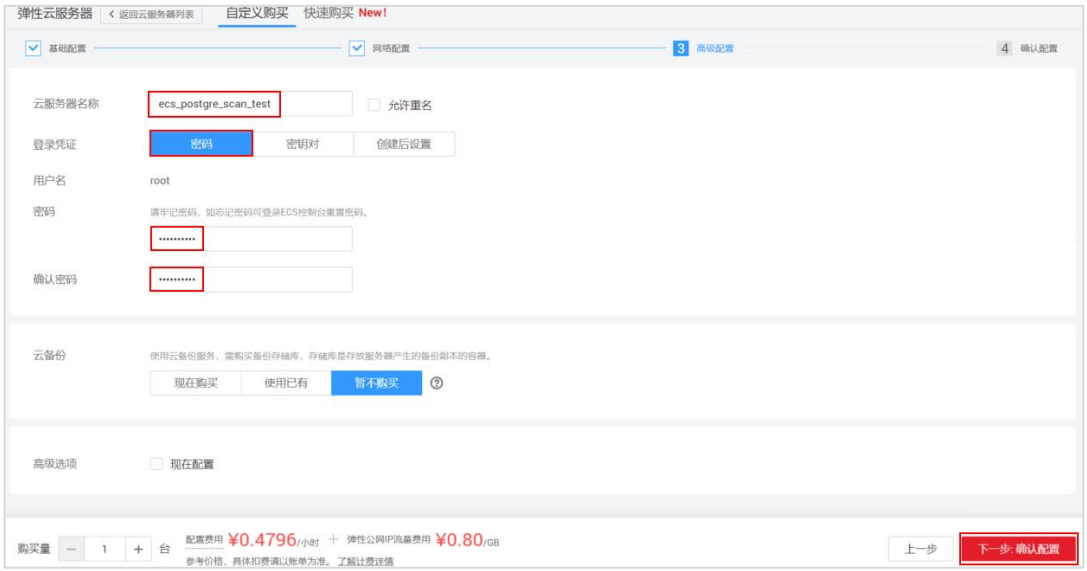
步骤 6　进入弹性云服务器的网络配置页面，网络配置如下图所示：



步骤 7　单击“下一步：高级配置”。

步骤 8　进入弹性云服务器的高级配置页面，高级配置如下图所示：

- 云服务器名称：ecs_postgre_scan_test
- 密码：XXXXX（按照密码规则自定义密码），并再次确认密码



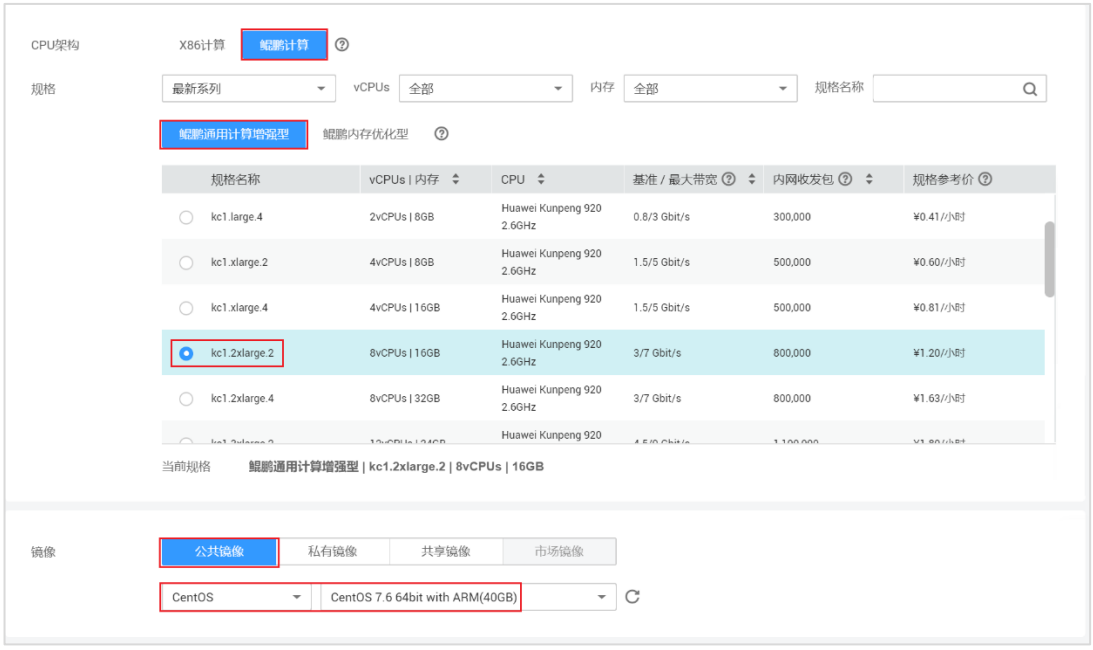
步骤 9　单击“下一步：确认配置”。

步骤 10 进入弹性云服务器的确认配置页面，核对信息无误后，勾选下方的“我已经阅读并同意《华为镜像免责声明》”，单击“立即购买”，完成用于部署 Porting Advisor 源码移植工具的 x86 云服务器的购买。

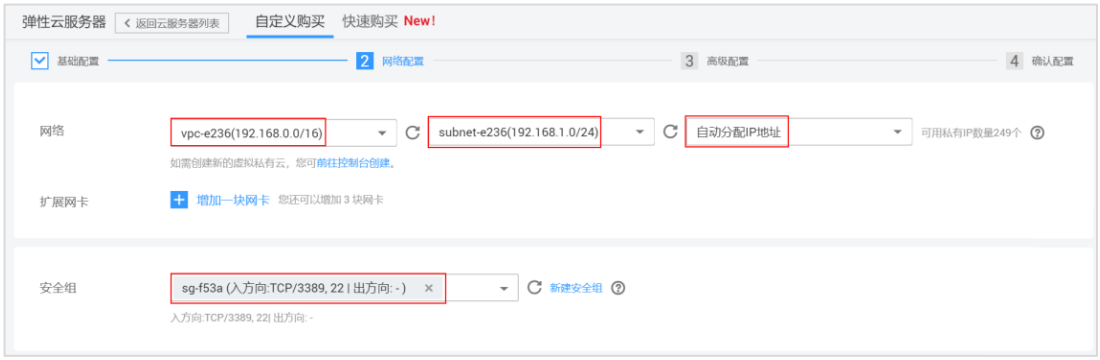
步骤 11 返回弹性云服务器列表页面，查看刚刚购买的弹性云服务器 ecs_postgre_scan_test，等待其状态变为“运行中”。

步骤 12 参考步骤 3~11，购买用于部署 PostgreSQL 的鲲鹏云服务器。

基础配置（鲲鹏计算，鲲鹏通用计算增强型，kc1.2xlarge.2 8vCPUs | 16GB，CentOS 7.6 64bit）如下所示（其他配置同云服务器 ecs_postgre_scan_test）：



网络配置如下所示（其他配置同云服务器 ecs_postgre_scan_test）：

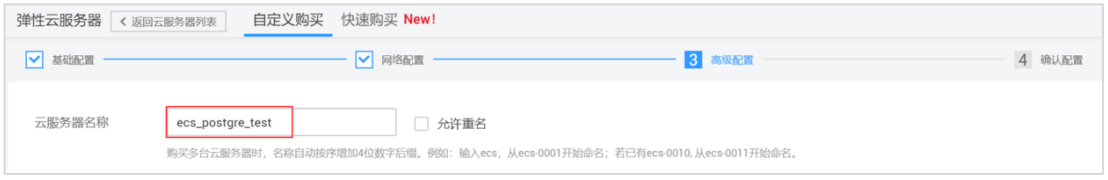


说明：

上图中的 vpc-e236(192.168.0.0/16)、subnet-e236(192.168.1.0/24)和 sg-f53a (入方向:TCP/3389, 22 | 出方向: -)，为购买云服务器 ecs_postgre_scan_test 时创建，此处与云服务器 ecs_postgre_scan_test 使用相同的网络和安全组。

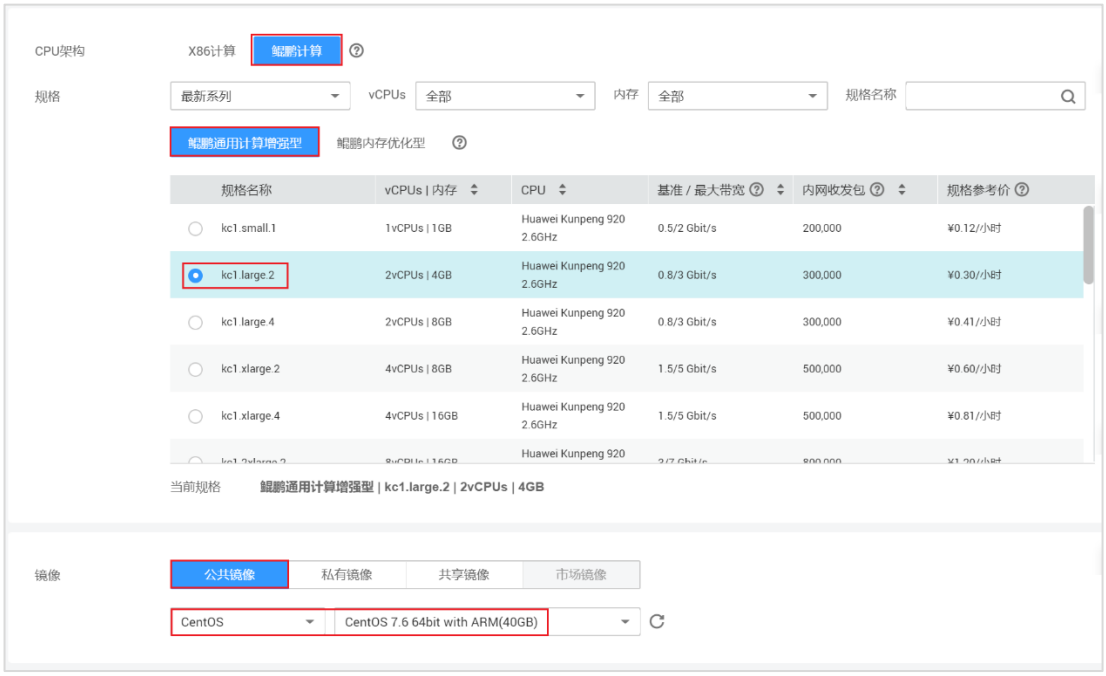
高级配置如下所示（其他配置同云服务器 ecs_postgre_scan_test）：

- 云服务器名称：ecs_postgre_test

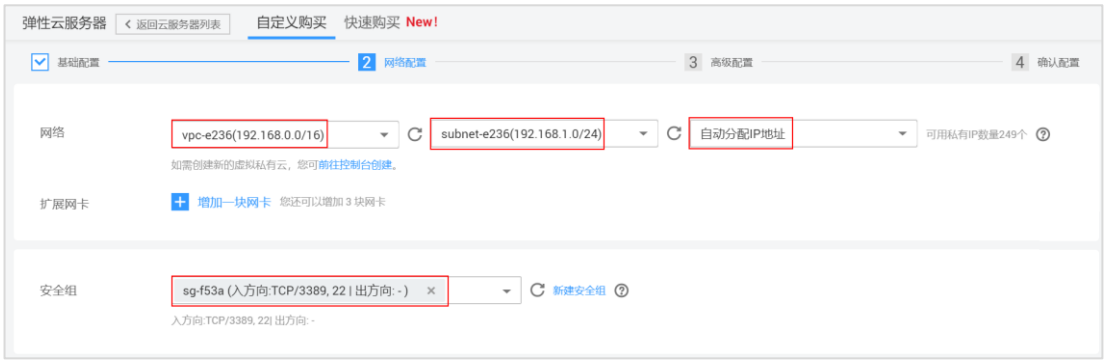


步骤 13 参考步骤 3~11，购买用于制作 PostgreSQL 的 RPM 安装包，并对安装包进行安装测试的鲲鹏云服务器。

基础配置（鲲鹏计算，鲲鹏通用计算增强型，kc1.large.2 2vCPUs | 4GB，CentOS 7.6 64bit）如下所示（其他配置同云服务器 ecs_postgre_scan_test）：



网络配置如下所示（其他配置同云服务器 ecs_postgre_scan_test）：

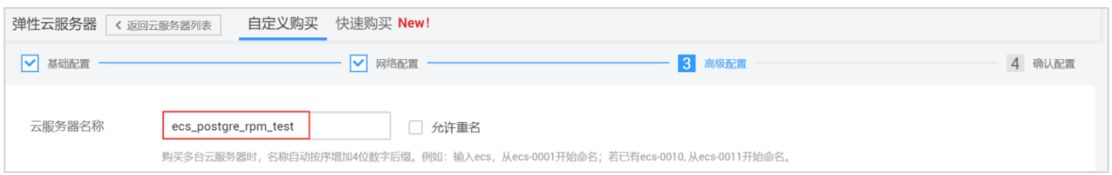


说明：

上图中的 vpc-e236(192.168.0.0/16)、subnet-e236(192.168.1.0/24)和 sg-f53a (入方向:TCP/3389, 22 | 出方向: -)，为购买云服务器 ecs_postgre_scan_test 时创建，此处与云服务器 ecs_postgre_scan_test 使用相同的网络和安全组。

高级配置如下所示（其他配置同云服务器 ecs_postgre_scan_test）：

- 云服务器名称：ecs_postgre_rpm_test



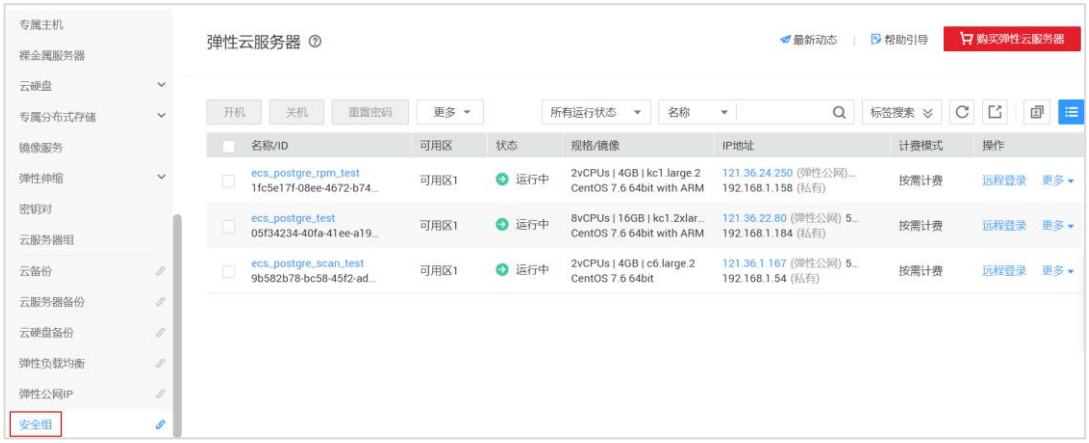
步骤 14 返回弹性云服务器列表页面，查看购买的 3 台弹性云服务器，等待其状态均变为“运行中”。

--结束

1.5 修改安全组规则

1.5.1 操作步骤

步骤 1 在弹性云服务器列表页面的左侧导航栏，选择“安全组”，进入安全组列表页面。



步骤 2 在弹性云服务器所关联的安全组所在行，单击“配置规则”，进入安全组规则列表页面。

步骤 3 在如下安全组规则所在行，单击“修改”。

步骤 4　将源地址修改为“IP 地址：0.0.0.0/0”，并单击“确认”。

步骤 5　若弹出如下信息提示框，勾选“我已知晓安全风险，暂不开启”，并单击“确定”。

步骤 6　返回安全组规则列表，查看源地址是否修改成功。

--结束

2 安装 Porting Advisor 代码移植工具

2.1 实验介绍

本实验通过在 x86 云服务器上安装 Porting Advisor 代码移植工具，为后续移植数据库 PostgreSQL 做准备。

2.2 前提条件

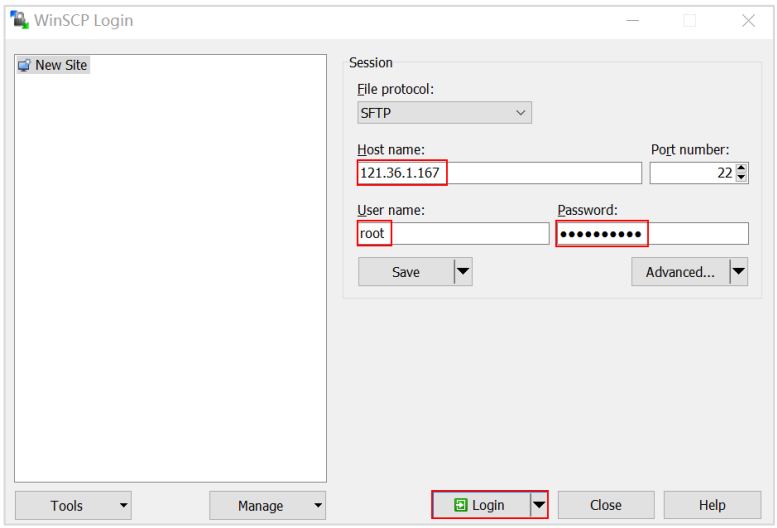
- x86 云服务器已上电。
- x86 云服务器与 PC 之间网络互通，且能访问外网。
- x86 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP，下载地址分别为：https://hcia-kunpeng-application-developer.obs.cn-north-1.myhwclouds.com/putty.exe ，https://hcia-kunpeng-application-developer.obs.cn-north-1.myhwclouds.com/WinSCP.zip。
- 已获取华为鲲鹏代码迁移工具安装包 Porting-advisor-x86_64-linux-1.0.5.tar.gz ,下载地址：https://www.huaweicloud.com/kunpeng/software/portingadvisor.html。

2.3 安装 Porting Advisor

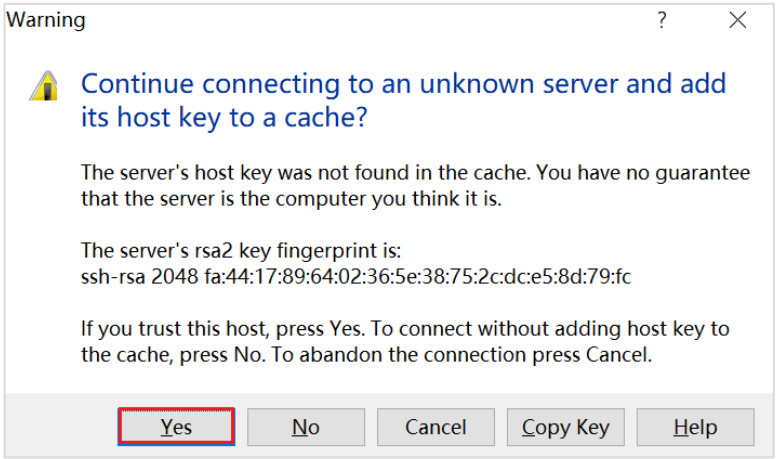
2.3.1 操作步骤

步骤 1 在弹性云服务器列表中，记录 x86 云服务器 ecs_postgre_scan_test 的弹性公网 IP 地址，如下图中的“121.36.1.167”。

步骤 2　打开 PC 上的 WinSCP，输入 x86 云服务器 ecs_postgre_scan_test 的弹性公网 IP 地址，用户名“root”和密码，单击下方的“Login”。

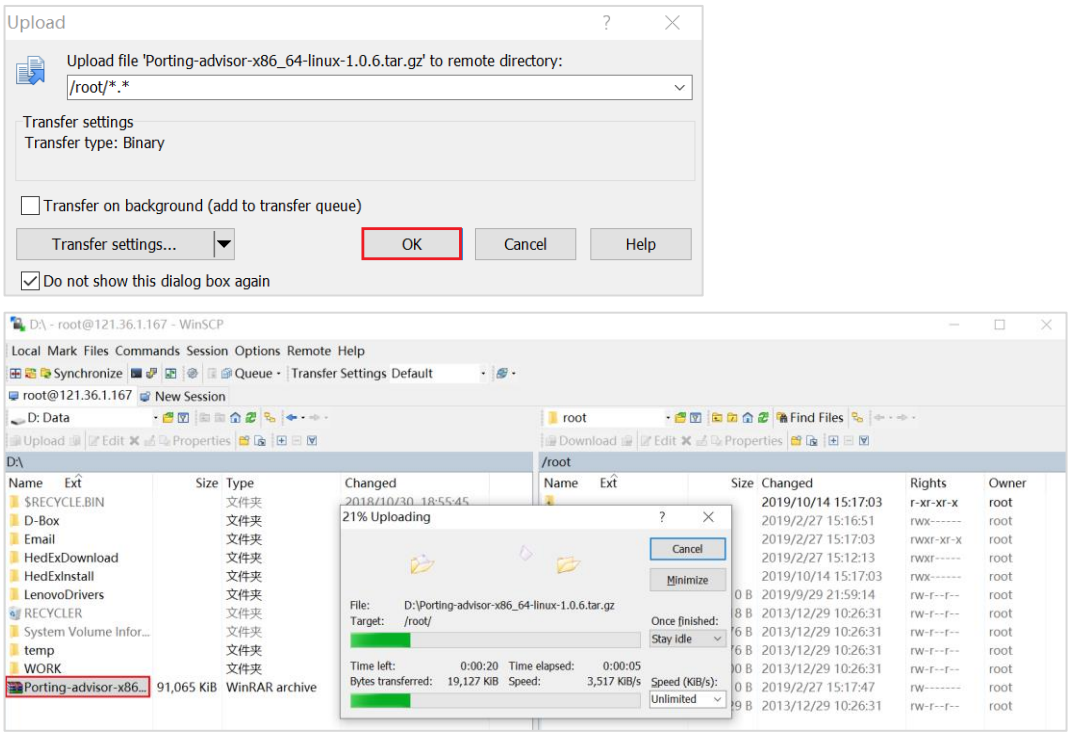


步骤 3　首次访问会弹出如下信息提示框，单击“Yes”。

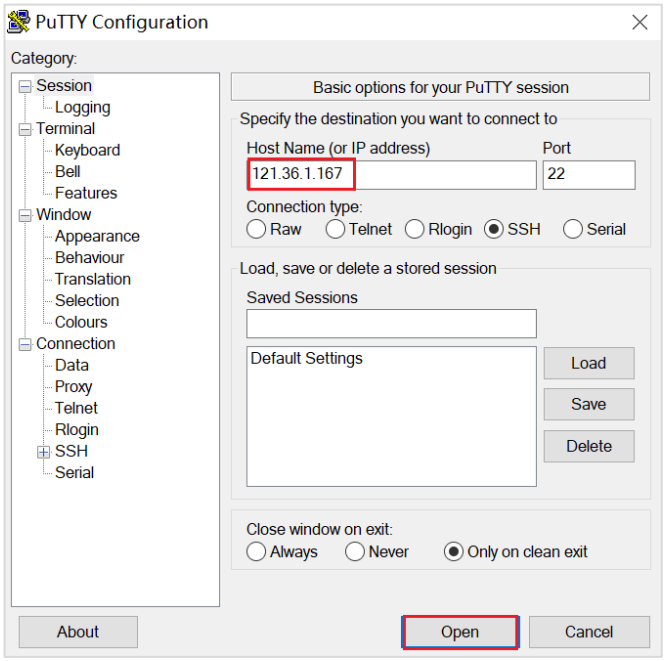


步骤 4　进入如下页面，在左侧选择华为鲲鹏代码迁移工具安装包 Porting-advisor-x86_64-linux-1.0.5.tar.gz 所在路径，并将安装包拖至右侧 x86 云服务器 ecs_postgre_scan_test 的 /root 目录下。

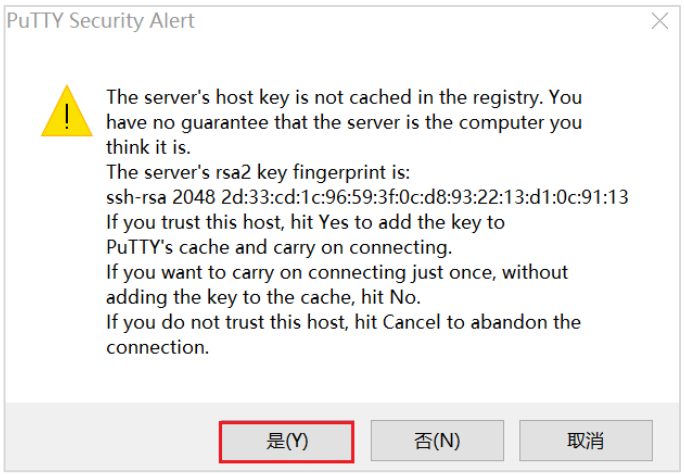
若弹出如下信息提示框，单击“OK”，等待上传完成。



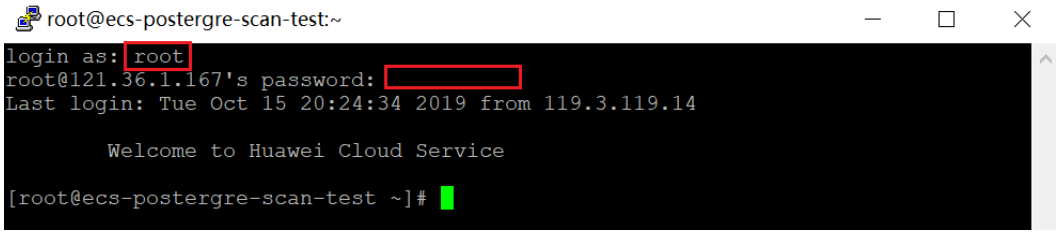
步骤 5　打开 PC 上的 putty，输入 x86 云服务器 ecs_postgre_scan_test 的弹性公网 IP 地址，单击下方的“Open”。



步骤 6　首次登录会弹出如下信息提示框，单击“是(Y)”。

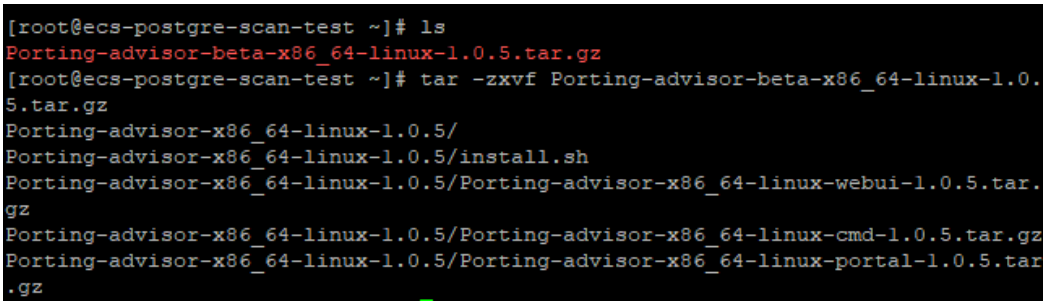


步骤 7 输入 x86 云服务器 ecs_postgre_scan_test 的用户名“root”和密码，进入如下命令行界面，表示登录成功。



步骤 8 执行以下命令，查看是否存在华为鲲鹏代码迁移工具安装包 Porting-advisor-x86_64-linux-1.0.5.tar.gz，若存在则直接解压缩，否则重复步骤 2~4 重新上传。

```
ls
tar -zxvf Porting-advisor-beta-x86_64-linux-1.0.5.tar.gz
```



步骤 9 执行以下命令，进入解压后的华为鲲鹏代码迁移工具安装包目录，安装其 Web 模式。

```
ls
cd Porting-advisor-bete-x86_64-linux-1.0.5
sh install.sh web
```

- 配置 Web Server 的 IP 地址：按 Enter 键，默认为操作系统上所有 IP 地址。
- 配置 https 端口：按 Enter 键，默认为 8084。

```
[root@ecs-postgre-scan-test Porting-advisor-x86_64-linux-1.0.5]# sh install.sh w
eb
please enter ip address(default is all ip addresses):
defult ip address
Cent OS
please enter https port(default: 8084):
```

说明：

若出现如下报错：

```
[root@ecs-postgre-scan-test Porting-advisor-x86_64-linux-1.0.5]# sh install.sh w
eb
 /opt/portadv exists
 Please delete directory!
```

执行以下命令，删除“/opt/portadv”目录。

```
rm -rf /opt/portadv
```

```
[root@ecs-postergre-scan-test ~]# rm -rf /opt/portadv/
```

步骤 10 等待 3~5 分钟完成安装，直到出现如下回显信息，表示安装成功。

```
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying operationlog.0001_initial... OK
  Applying sessions.0001_initial... OK
  Applying taskmanager.0001_initial... OK
Successfully installed porting advisor in /opt/portadv/tools
```

--结束

3 Porting Advisor 移植 PostgreSQL

3.1 实验介绍

本实验以 PostgreSQL 的源码为例，使用 Porting Advisor 代码移植工具对 PostgreSQL 进行移植分析，并根据分析报告和修改建议，修改源码，实现代码的移植。

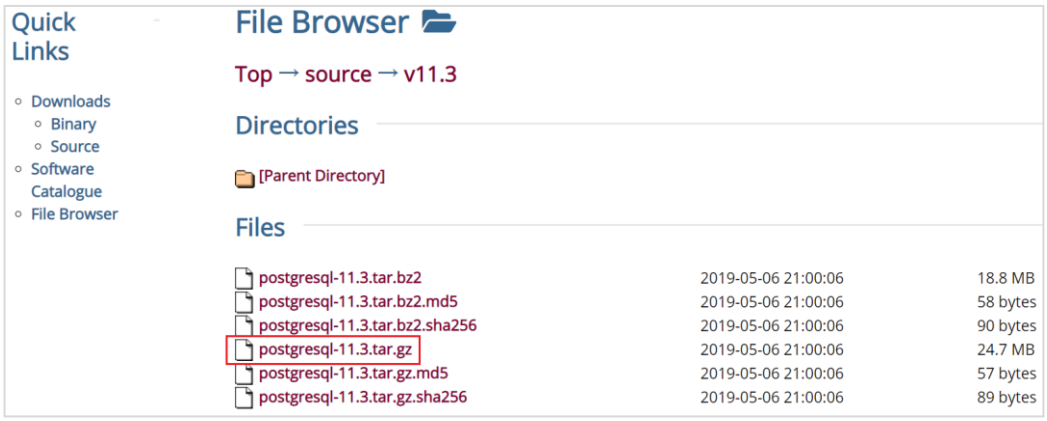
3.2 前提条件

- x86 云服务器和鲲鹏云服务器已上电。
- x86 云服务器、鲲鹏云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- x86 云服务器上已安装 Porting Advisor 工具。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。

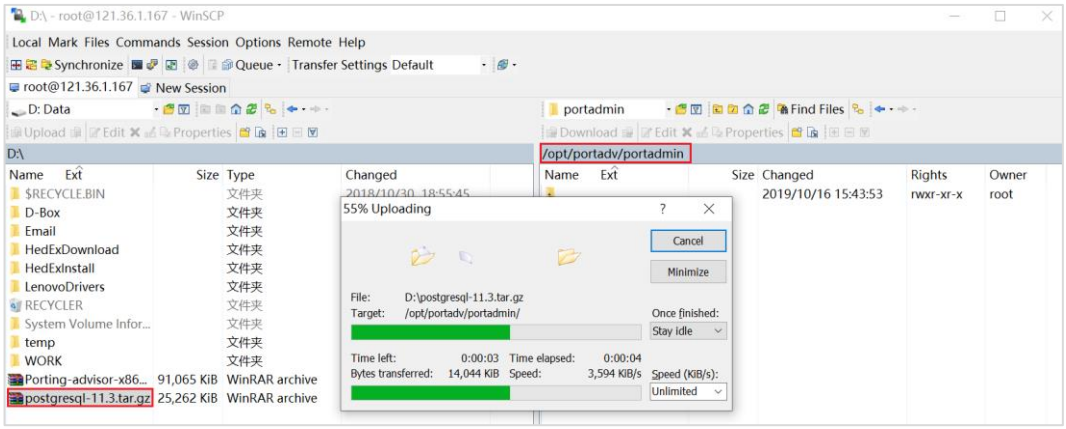
3.3 分析 PostgreSQL 源码

3.3.1 操作步骤

步骤 1 使用 PC 上的浏览器访问：https://www.postgresql.org/ftp/source/v11.3/，下载 PostgreSQL 11.3 版本源代码包 postgresql-11.3.tar.gz。



步骤 2　参考 2.3.1 中的步骤 2~4，打开 WinSCP，将下载的 PostgreSQL 源代码包 postgresql-11.3.tar.gz 上传至 x86 云服务器 ecs_postgre_scan_test 的/opt/portadv/portadmin/目录下。

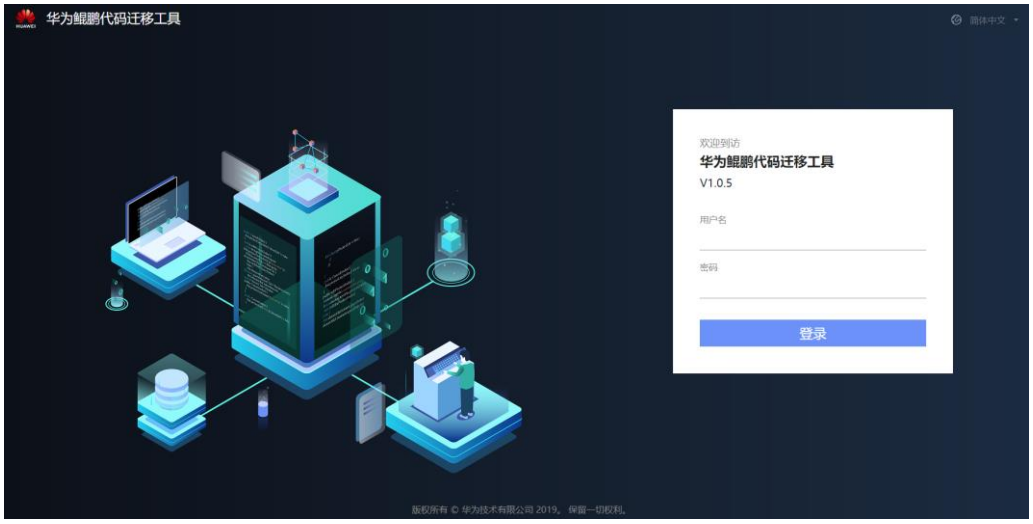


步骤 3　返回 putty，登录 x86 云服务器 ecs_postgre_scan_test，执行以下命令，进入 PostgreSQL 源代码包 postgresql-11.3.tar.gz 所在路径，查看是否存在源代码包，若存在则直接解压缩，否则需要重新上传。

```
cd /opt/portadv/portadmin
ls
tar -zxvf postgresql-11.3.tar.gz
```

```
[root@ecs-postergre-scan-test ~]# cd /opt/portadv/portadmin/
[root@ecs-postergre-scan-test portadmin]# ls
postgresql-11.3.tar.gz
[root@ecs-postergre-scan-test portadmin]# tar -zxvf postgresql-11.3.tar.gz
```

步骤 4　使用 PC 上的浏览器访问：**https://x86 云服务器的弹性公网 IP 地址：8084**，如 https://121.36.1.167:8084，进入华为鲲鹏代码移植工具的 Web 页面。



说明：

如果弹出“此网站的安全证书有问题”的告警提示，请选择“继续浏览此网站（不推荐）”，进入登录界面。

步骤 5 输入用户名和密码，单击“登录”，进入登录界面。

说明：

系统的默认用户名为 portadmin，默认密码为 Admin@9000。

首次登录 Web 的用户，系统提示修改默认密码。请按提示修改密码，密码需要满足如下复杂度要求：

- 密码长度为 6～32 个字符。
- 必须包含大写字母、小写字母、数字、特殊字符（`~!@#$%^&*()-_=+\|[{}];:'",<.>/?）中的至少两种及以上的字符。

按要求修改初始密码后单击“确认”，或者直接单击“取消”，后续再自行修改密码。

步骤 6　配置源代码存放路径，其他参数保持默认值。

说明：

这里的源代码存放路径参数填写“postgresql-11.3/”，其中“/opt/portadv/portadmin/”为固定路径，源码代码全路径为“/opt/portadv/portadmin/postgresql-11.3/”。

界面上的编译器版本 GCC 4.8 与后台 CentOS 上安装的编译器 GCC 版本相互独立，两者没有关联关系。

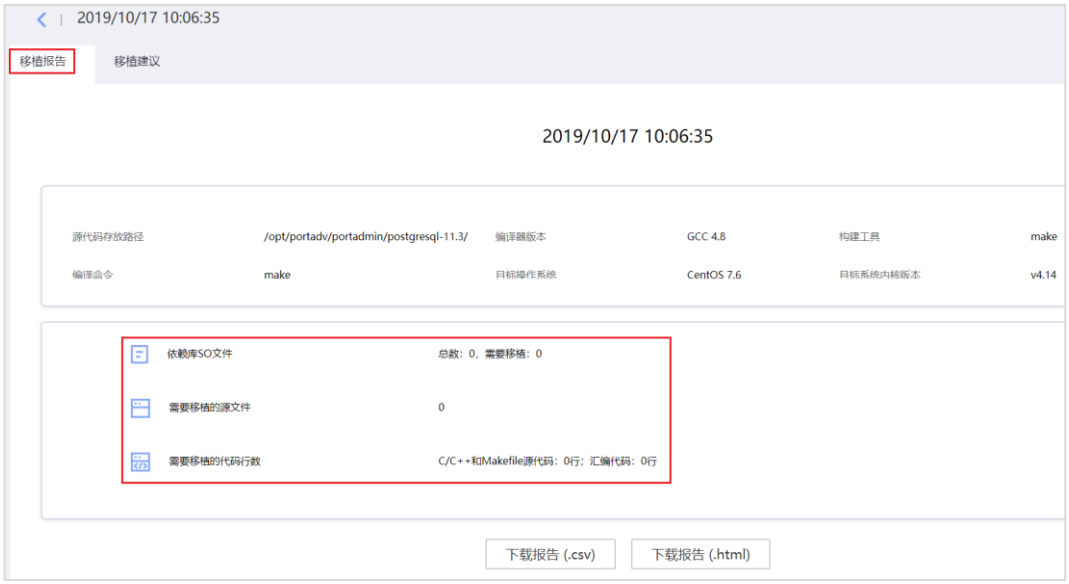
步骤 7　单击“分析”，生成分析报告。

弹窗页面显示任务分析进度，分析完成后，自动跳转至“移植报告”界面。

说明：

在分析进行中，请勿关闭当前页面，任务分析完成后可直接跳转至“移植报告”界面，也可以在历史报告的列表中，单击指定分析任务的报告日期。

步骤 8 待分析完成后，选择左上角的“移植报告”和“移植建议”页签，查看移植报告和移植建议。



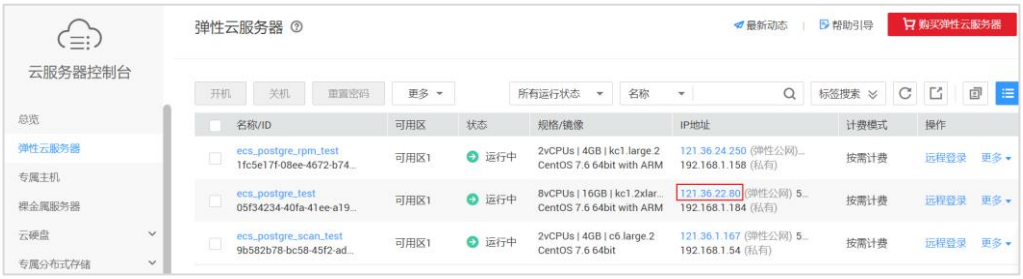
从移植报告中可以看出，需要移植的依赖库 SO 文件、源文件、代码行数均为 0，说明 postgresql-11.3 源码不需要修改，可以直接进行源码编译安装。

--结束

3.4 升级 GCC 编译器版本

3.4.1 操作步骤

步骤 1　在弹性云服务器列表中，记录鲲鹏云服务器 ecs_postgre_test 的弹性公网 IP 地址，如下图中的“121.36.22.80”。



步骤 2　参考 2.3.1 中的步骤 5~7，打开 putty，登录鲲鹏云服务器 ecs_postgre_test。

步骤 3 执行以下命令，配置 yum 源，并安装 PostgreSQL 依赖包。

```
yum -y install gcc gcc-c++ automake zlib zlib-devel bzip2 bzip2-devel bzip2-
libs readline readline-devel bison ncurses ncurses-devel libaio-devel openssl
openssl-devel gmp gmp-devel mpfr mpfr-devel libmpc libmpc-devel
```

```
[root@ecs-postgre-test ~]# yum -y install gcc gcc-c++ automake zlib zlib-devel b
zip2 bzip2-devel bzip2-libs readline readline-devel bison ncurses ncurses-devel
libaio-devel openssl openssl-devel gmp gmp-devel mpfr mpfr-devel libmpc libmpc-d
evel
Loaded plugins: fastestmirror
Determining fastest mirrors
epel/aarch64/metalink                                    | 7.2 kB     00:00
 * base: mirrors.huaweicloud.com
 * epel: mirrors.yun-idc.com
 * extras: mirrors.huaweicloud.com
 * updates: mirror-hk.koddos.net
```

继续等待，直到出现如下回显信息，表示安装完成。

```
Updated:
  gcc.aarch64 0:4.8.5-39.el7                 gcc-c++.aarch64 0:4.8.5-39.el7
  openssl.aarch64 1:1.0.2k-19.el7            readline.aarch64 0:6.2-11.el7

Dependency Updated:
  cpp.aarch64 0:4.8.5-39.el7                 e2fsprogs.aarch64 0:1.42.9-16.el7
  e2fsprogs-libs.aarch64 0:1.42.9-16.el7     krb5-libs.aarch64 0:1.15.1-37.el7_7.2
  libcom_err.aarch64 0:1.42.9-16.el7         libgcc.aarch64 0:4.8.5-39.el7
  libgomp.aarch64 0:4.8.5-39.el7             libss.aarch64 0:1.42.9-16.el7
  libstdc++.aarch64 0:4.8.5-39.el7           libstdc++-devel.aarch64 0:4.8.5-39.el7
  openssl-libs.aarch64 1:1.0.2k-19.el7

Complete!
```

步骤 4 执行以下命令，检查当前环境中的 GCC 编译器是否符合版本要求。

```
gcc -v
```

```
[root@ecs-postergre-test ~]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/aarch64-redhat-linux/4.8.5/lto-wrapper
Target: aarch64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/s
hare/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstrap --ena
ble-shared --enable-threads=posix --enable-checking=release --with-system-zlib --ena
ble-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-
linker-build-id --with-linker-hash-style=gnu --enable-languages=c,c++,objc,obj-c++,j
ava,fortran,ada,lto --enable-plugin --enable-initfini-array --disable-libgcj --with-
isl=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-aarch64-redhat-linux/isl-install --
with-cloog=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-aarch64-redhat-linux/cloog-i
nstall --enable-gnu-indirect-function --build=aarch64-redhat-linux
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-39) (GCC)
```

- 若版本为 5.3 及以上，则表示符合版本要求，结束。
- 否则，请执行步骤 5，升级 GCC 版本（本实验以升级到 7.3.0 版本为例进行介绍）。

步骤 5 执行以下命令，安装 wget 工具。

```
yum -y install wget
```

```
[root@ecs-postgre-test ~]# yum -y install wget
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.huaweicloud.com
 * epel: mirrors.yun-idc.com
 * extras: mirrors.huaweicloud.com
 * updates: mirror-hk.koddos.net
```

步骤 6 执行以下命令，下载 7.3.0 版本的 GCC 软件包。

```
wget https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz
```

```
[root@ecs-postergre-test ~]# wget https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7
.3.0/gcc-7.3.0.tar.gz
--2019-10-16 16:31:02--  https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-
7.3.0.tar.gz
Resolving mirrors.tuna.tsinghua.edu.cn (mirrors.tuna.tsinghua.edu.cn)... 101.6.8.193
, 2402:f000:1:408:8100::1
Connecting to mirrors.tuna.tsinghua.edu.cn (mirrors.tuna.tsinghua.edu.cn)|101.6.8.19
3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 112201917 (107M) [application/x-gzip]
Saving to: 'gcc-7.3.0.tar.gz'

100%[======================================>] 112,201,917 8.60MB/s   in 12s

2019-10-16 16:31:14 (8.56 MB/s) - 'gcc-7.3.0.tar.gz' saved [112201917/112201917]
```

步骤 7 执行以下命令，对软件包进行解压缩。

```
ls
tar -xvf gcc-7.3.0.tar.gz
```

```
[root@ecs-postergre-test ~]# ls
gcc-7.3.0.tar.gz
[root@ecs-postergre-test ~]# tar -xvf gcc-7.3.0.tar.gz
```

步骤 8 执行以下命令，编译安装 GCC。

```
ls
cd gcc-7.3.0
./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --
enable-bootstrap --enable-shared --enable-threads=posix --enable-
checking=release --with-system-zlib --enable-__cxa_atexit --disable-libunwind-
exceptions --enable-gnu-unique-object --enable-linker-build-id --with-linker-
hash-style=gnu --enable-languages=c,c++,objc,obj-c++,fortran,lto --enable-
plugin --enable-initfini-array --disable-libgcj
make -j 96
make install
```

此过程需要等待较长时间，直到出现如下回显信息，表示安装完成。

```
See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
make[4]: Nothing to be done for `install-data-am'.
make[4]: Leaving directory `/root/gcc-7.3.0/aarch64-unknown-linux-gnu/libatomic'
make[3]: Leaving directory `/root/gcc-7.3.0/aarch64-unknown-linux-gnu/libatomic'
make[2]: Leaving directory `/root/gcc-7.3.0/aarch64-unknown-linux-gnu/libatomic'
make[1]: Leaving directory `/root/gcc-7.3.0'
```

步骤 9 执行以下命令，再次查看 GCC 的版本，验证是否升级成功。

```
gcc -v
```

```
[root@ecs-postergre-test gcc-7.3.0]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/aarch64-unknown-linux-gnu/7.3.0/lto-wrapper
Target: aarch64-unknown-linux-gnu
Configured with: ./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/sh
are/info --enable-bootstrap --enable-shared --enable-threads=posix --enable-checking
=release --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --e
nable-gnu-unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enab
le-languages=c,c++,objc,obj-c++,fortran,lto --enable-plugin --enable-initfini-array
--disable-libgcj
Thread model: posix
gcc version 7.3.0 (GCC)
```

--结束

3.5 移植 PostgreSQL

3.5.1 操作步骤

步骤 1 返回 putty，登录鲲鹏云服务器 ecs_postgre_test，执行以下命令，将 x86 云服务器 ecs_postgre_scan_test 上的 PostgreSQL 源码包拷贝至“/home”目录下。

```
scp root@121.36.1.167:/opt/portadv/portadmin/postgresql-11.3.tar.gz /home/
```

此处 IP 地址请参照自己的实验环境 ecs_postgre_scan_test 的弹性公网 IP。

首次连接会出现如下提示，输入“yes”，按 Enter。

按提示输入 x86 云服务器 ecs_postgre_scan_test 的 root 用户的密码。

等待拷贝完成。

```
[root@ecs-postergre-test ~]# scp root@121.36.1.167:/opt/portadv/portadmin/postgresql-
11.3.tar.gz /home/
The authenticity of host '121.36.1.167 (121.36.1.167)' can't be established.
ECDSA key fingerprint is SHA256:x/T12QhE2eDq+J161I42QaE/ltqvFzJ8WlOY8DmoN1k.
ECDSA key fingerprint is MD5:74:45:77:7f:8f:a5:0f:a1:bb:2d:2e:b8:2b:4a:a7:9b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '121.36.1.167' (ECDSA) to the list of known hosts.
root@121.36.1.167's password:
postgresql-11.3.tar.gz                        100%   25MB   6.5MB/s   00:03
```

步骤 2 执行以下命令，进入/home 目录，查看是否存在 PostgreSQL 源码包，若存在则直接解压缩，否则重复步骤 1 重新拷贝。

```
cd /home/
ls
tar -zxvf postgresql-11.3.tar.gz
```

```
[root@ecs-postergre-test ~]# cd /home/
[root@ecs-postergre-test home]# ls
postgresql-11.3.tar.gz
[root@ecs-postergre-test home]# tar -zxvf postgresql-11.3.tar.gz
```

步骤 3 执行以下命令，进入解压后的源码目录，创建安装目录/home/pgsql。

```
cd postgresql-11.3
```

```
mkdir /home/pgsql
```

```
[root@ecs-postergre-test home]# cd postgresql-11.3
[root@ecs-postergre-test postgresql-11.3]# mkdir /home/pgsql
```

步骤 4　执行以下命令，生成 Makefile 文件。

```
./configure -prefix=/home/pgsql
```

直到出现如下回显信息，表示生成成功。

```
config.status: creating GNUmakefile
config.status: creating src/Makefile.global
config.status: creating src/include/pg_config.h
config.status: creating src/include/pg_config_ext.h
config.status: creating src/interfaces/ecpg/include/ecpg_config.h
config.status: linking src/backend/port/tas/dummy.s to src/backend/port/tas.s
config.status: linking src/backend/port/dynloader/linux.c to src/backend/port/dy
nloader.c
config.status: linking src/backend/port/posix_sema.c to src/backend/port/pg_sema
.c
config.status: linking src/backend/port/sysv_shmem.c to src/backend/port/pg_shme
m.c
config.status: linking src/backend/port/dynloader/linux.h to src/include/dynload
er.h
config.status: linking src/include/port/linux.h to src/include/pg_config_os.h
config.status: linking src/makefiles/Makefile.linux to src/Makefile.port
```

步骤 5　执行以下命令，编译并安装 PostgreSQL。

```
make -j 64
make install
```

直到出现如下回显信息，表示安装完成。

```
make[2]: Entering directory `/home/postgresql-11.3/src/test/perl'
make[2]: Nothing to be done for `install'.
make[2]: Leaving directory `/home/postgresql-11.3/src/test/perl'
/usr/bin/mkdir -p '/home/pgsql/lib/pgxs/src'
/usr/bin/install -c -m 644 Makefile.global '/home/pgsql/lib/pgxs/src/Makefile.gl
obal'
/usr/bin/install -c -m 644 Makefile.port '/home/pgsql/lib/pgxs/src/Makefile.port
'
/usr/bin/install -c -m 644 ./Makefile.shlib '/home/pgsql/lib/pgxs/src/Makefile.s
hlib'
/usr/bin/install -c -m 644 ./nls-global.mk '/home/pgsql/lib/pgxs/src/nls-global.
mk'
make[1]: Leaving directory `/home/postgresql-11.3/src'
make -C config install
make[1]: Entering directory `/home/postgresql-11.3/config'
/usr/bin/mkdir -p '/home/pgsql/lib/pgxs/config'
/usr/bin/install -c -m 755 ./install-sh '/home/pgsql/lib/pgxs/config/install-sh'
/usr/bin/install -c -m 755 ./missing '/home/pgsql/lib/pgxs/config/missing'
make[1]: Leaving directory `/home/postgresql-11.3/config'
PostgreSQL installation complete.
```

步骤 6　执行以下命令，创建 postgres 用户和用户组。

```
/usr/sbin/groupadd -g 1001 postgres
/usr/sbin/useradd -u 1012 -m -g postgres postgres
```

```
[root@ecs-postergre-test postgresql-11.3]# /usr/sbin/groupadd -g 1001 postgres
[root@ecs-postergre-test postgresql-11.3]# /usr/sbin/useradd -u 1012 -m -g postg
res postgres
```

步骤 7　执行以下命令，设置 postgres 用户密码。

```
passwd postgres
```

按提示输入 postgres 用户的密码，并再次确认。

```
[root@ecs-postergre-test postgresql-11.3]# passwd postgres
Changing password for user postgres.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

步骤 8　执行以下命令，切换到 postgres 用户，并初始化数据库。

```
su - postgres
/home/pgsql/bin/initdb -D pgsql/
```

```
[root@ecs-postergre-test postgresql-11.3]# su - postgres
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/initdb -D pgsql/
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory pgsql ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    /home/pgsql/bin/pg_ctl -D pgsql/ -l logfile start
```

步骤 9　执行以下命令，启动 PostgreSQL 数据库。

```
/home/pgsql/bin/pg_ctl -D pgsql/ -l logfile start
```

```
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/pg_ctl -D pgsql/ -l logfile start
waiting for server to start.... done
server started
```

步骤 10　执行以下命令，确认 PostgreSQL 数据库进程是否正常启动。

```
ps -ef | grep postgres
```

```
[postgres@ecs-postgre-test ~]$ ps -ef | grep postgres
root     26852 10617  0 20:14 pts/0    00:00:00 su - postgres
postgres 26853 26852  0 20:14 pts/0    00:00:00 -bash
postgres 26890     1  0 20:15 pts/0    00:00:00 /home/pgsql/bin/postgres -D pgsql
postgres 26892 26890  0 20:15 ?        00:00:00 postgres: checkpointer
postgres 26893 26890  0 20:15 ?        00:00:00 postgres: background writer
postgres 26894 26890  0 20:15 ?        00:00:00 postgres: walwriter
postgres 26895 26890  0 20:15 ?        00:00:00 postgres: autovacuum launcher
postgres 26896 26890  0 20:15 ?        00:00:00 postgres: stats collector
postgres 26897 26890  0 20:15 ?        00:00:00 postgres: logical replication launcher
postgres 26898 26853  0 20:15 pts/0    00:00:00 ps -ef
postgres 26899 26853  0 20:15 pts/0    00:00:00 grep --color=auto postgres
```

如上图所示，PostgreSQL 相关进程已经正常启动了。

步骤 11 执行以下命令，验证是否可以正常登录 PostgreSQL 数据库。

说明：

初次登录数据库，无需密码。

```
/home/pgsql/bin/psql -U postgres
```

```
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/psql -U postgres
psql (11.3)
Type "help" for help.

postgres=#
```

如上图所示，可以正常登录 PostgreSQL 数据库。

输入“\q”，退出数据库。

```
postgres=# \q
[postgres@ecs-postergre-test ~]$
```

步骤 12 执行以下命令，编辑配置文件 pg_hba.conf。

```
cd /home/postgres/pgsql/
vi pg_hba.conf
```

```
[postgres@ecs-postergre-test ~]$ cd /home/postgres/pgsql/
[postgres@ecs-postergre-test pgsql]$ vi pg_hba.conf
```

按“i”进入编辑模式，输入以下内容，按“Esc”退出编辑模式，输入“:wq!”，保存退出。

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             ■.■.■.■/■               trust
```

说明：

该行内容用以允许非本地访问 PostgreSQL 数据库，IP 地址段表示允许访问 PostgreSQL 数据库的地址段，具体请按实际情况配置。例如，此处配置为 172.168.1.1，表示只允许 172.168.1.1 这一台主机访问。

步骤 13 执行以下命令，编辑配置文件 postgresql.conf。

```
vi postgresql.conf
```

```
[postgres@ecs-postergre-test pgsql]$ vi postgresql.conf
```

修改配置连接信息，按“i”进入编辑模式，将“listen_addresses”后的“localhost”修改为“*”，并删除前面的“#”，按“Esc”退出编辑模式，输入“:wq!” ，保存退出。

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
#port = 5432                    # (change requires restart)
max_connections = 100           # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/tmp'       # comma-separated list of directories
                                # (change requires restart)
#unix_socket_group = ''         # (change requires restart)
#unix_socket_permissions = 0777         # begin with 0 to use octal notation
                                # (change requires restart)
#bonjour = off                  # advertise server via Bonjour
                                # (change requires restart)
```

说明：

“listen_addresses = ‘*’ ”，表示监听所有 IP 地址。

步骤 14 配置完成后，执行以下命令，返回上级目录，重启数据库服务，使配置文件生效。

```
cd ..
```

/home/pgsql/bin/pg_ctl -D pgsql/ stop

/home/pgsql/bin/pg_ctl -D pgsql/ start

```
[postgres@ecs-postergre-test pgsql]$ cd ..
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/pg_ctl -D pgsql/ stop
waiting for server to shut down.... done
server stopped
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/pg_ctl -D pgsql/ start
waiting for server to start....2019-10-18 15:15:54.619 CST [7156] LOG:  listening on
IPv4 address "0.0.0.0", port 5432
2019-10-18 15:15:54.619 CST [7156] LOG:  listening on IPv6 address "::", port 5432
2019-10-18 15:15:54.626 CST [7156] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432
"
2019-10-18 15:15:54.639 CST [7157] LOG:  database system was shut down at 2019-10-18
15:15:49 CST
2019-10-18 15:15:54.643 CST [7156] LOG:  database system is ready to accept connectio
ns
 done
server started
```

--结束

4 安装 BenchmarkSQL 性能测试工具

实验介绍

本实验介绍在 x86 云服务器 ecs_postgre_scan_test 上安装性能测试工具 BenchmarkSQL，为后续对数据库 PostgreSQL 进行性能测试做准备。

前提条件

- x86 云服务器已上电。
- x86 云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。
- 已获取性能测试工具源码文件 BenchmarkSQL.zip，下载地址：https://hcia-kunpeng-application-developer.obs.cn-north-1.myhwclouds.com/BenchmarkSQL.zip。

4.1 安装 jdk

4.1.1 操作步骤

步骤 1 打开 putty，登录 x86 云服务器 ecs_postgre_scan_test，执行以下命令，查看是否有安装 jdk。

```
java -version
```

```
[root@ecs-postergre-scan-test ~]# java -version
-bash: java: command not found
```

如上图所示，表示未安装 jdk。

- 若已安装，则表示符合环境要求，结束。
- 否则，请执行步骤 2，开始安装 jdk（本实验以安装 java-1.8.0-openjdk 为例进行介绍）。

步骤 2　执行以下命令，安装 java-1.8.0-openjdk。

```
yum -y install java-1.8.0-openjdk
```

```
[root@ecs-postgre-scan-test ~]# yum -y install java-1.8.0-openjdk
```

步骤 3　安装完成后，执行以下命令，设置环境变量。

```
vi /etc/profile
```

```
[root@ecs-postergre-scan-test ~]# vi /etc/profile
```

说明：

安装完之后，默认的安装目录在: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.222.b10-1.el7_7.x86_64。

按“i”进入编辑模式，输入以下内容，按“Esc”退出编辑模式，输入“:wq”，保存退出。

```
# ---Set Java Environment---
JAVA_HOME=/usr/local/java/jdk1.8.0_181
JRE_HOME=/usr/local/java/jdk1.8.0_181/jre
CLASS_PATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export JAVA_HOME JRE_HOME CLASS_PATH PATH
# ---Set Java Environment---
```

```
# ---Set Java Environment---
JAVA_HOME=/usr/local/java/jdk1.8.0_181
JRE_HOME=/usr/local/java/jdk1.8.0_181/jre
CLASS_PATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export JAVA_HOME JRE_HOME CLASS_PATH PATH
# ---Set Java Environment---
```

步骤 4　执行以下命令，使修改的配置生效。

```
source /etc/profile
```

```
[root@ecs-postergre-scan-test ~]# source /etc/profile
```

步骤 5　执行以下命令，再次查看 jdk 的版本。

```
java -version
```

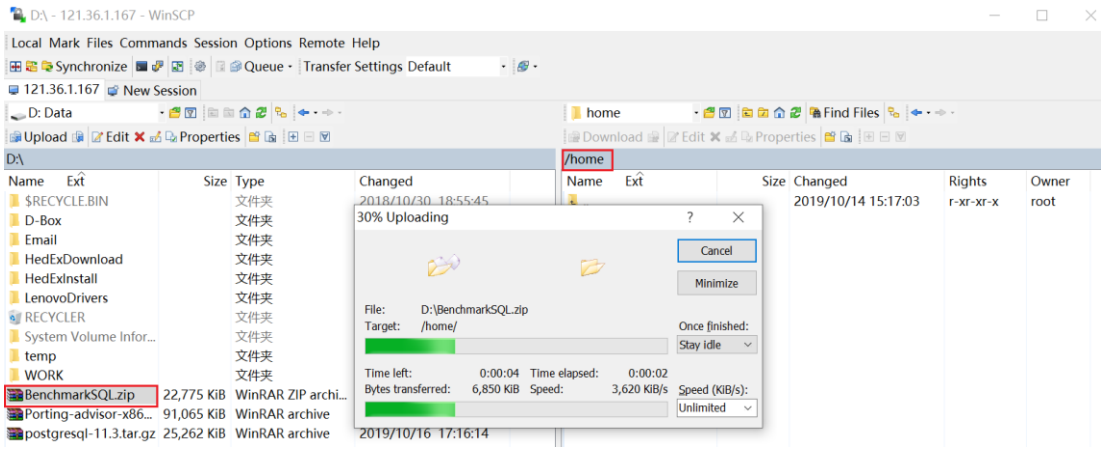
```
[root@ecs-postergre-scan-test ~]# java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

--结束

4.2 安装 BenchmarkSQL

4.2.1 操作步骤

步骤 1 参考 2.3.1 中的步骤 2~4，打开 WinSCP，将性能测试工具源码文件 BenchmarkSQL.zip 上传至 x86 云服务器 ecs_postgre_scan_test 的/home 目录下。



步骤 2 执行以下命令，查看下载的 BenchmarkSQL 源码文件，并解压。

```
cd /home/
ls
unzip BenchmarkSQL.zip
```

```
[root@ecs-postergre-scan-test ~]# cd /home/
[root@ecs-postergre-scan-test home]# ls
BenchmarkSQL.zip
[root@ecs-postergre-scan-test home]# unzip BenchmarkSQL.zip
```

说明：

BenchmarkSQL 不需要再编译安装，可以在 run 目录下，直接通过修改配置文件，连接 PostgreSQL 数据库。

--结束

5 PostgreSQL 性能测试

5.1 实验介绍

本实验介绍使用 BenchmarkSQL 性能测试工具对 PostgreSQL 数据库进行性能测试的方法。

5.2 前提条件

- x86 云服务器已上电。
- x86 云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- x86 云服务器上已安装 BenchmarkSQL 性能测试工具。
- 鲲鹏云服务器上已安装 PostgreSQL 数据库，并已创建数据库用户 postgres。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。

5.3 创建数据库 tpcc

5.3.1 操作步骤

步骤 1 返回 putty，登录鲲鹏云服务器 ecs_postgre_test，执行以下命令，切换到 postgres 用户，登录数据库。

```
su - postgres
/home/pgsql/bin/psql -U postgres
```

```
[root@ecs-postergre-test ~]# su - postgres
Last login: Fri Oct 18 16:46:48 CST 2019 on pts/1
[postgres@ecs-postergre-test ~]$ /home/pgsql/bin/psql -U postgres
psql (11.3)
Type "help" for help.

postgres=#
```

步骤 2　执行以下命令，查看已创建的数据库列表。

```
\l
```

```
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
(3 rows)
```

步骤 3　执行以下命令，创建数据库 tpcc，并查看是否创建成功（先不要退出数据库）。

```
create database tpcc;
\l
```

```
postgres=# create database tpcc;
CREATE DATABASE
postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 tpcc      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(4 rows)
```

--结束

5.4 PostgreSQL 性能测试

5.4.1 操作步骤

步骤 1　返回 putty，登录 x86 云服务器 ecs_postgre_scan_test，执行以下命令，添加 BenchmarkSQL 目录的执行权限。

```
chmod -R u+x BenchmarkSQL/
```

```
[root@ecs-postergre-scan-test home]# chmod -R u+x BenchmarkSQL/
```

步骤 2　执行以下命令，进入 BenchmarkSQL 的 run 目录，按实际情况修改配置文件。

```
cd /home/BenchmarkSQL/run/
cp props.pg arm_postgres.properties
vi arm_postgres.properties
```

```
[root@ecs-postergre-scan-test home]# cd /home/BenchmarkSQL/run/
[root@ecs-postergre-scan-test run]# cp props.pg arm_postgres.properties
[root@ecs-postergre-scan-test run]# vi arm_postgres.properties
```

修改以下内容，其余内容保持默认：

表5-1 数据库参数说明

参数	说明
conn	“121.36.22.80”表示数据库服务器地址，按实际填写。 “5432”表示PostgreSQL数据库端口，按实际填写。 “tpcc”是数据库名称，按实际填写。
user	修改为创建数据库用户帐号。
password	修改为创建数据库用户密码。
warehouses	参数是初始化加载数据时，需要创建多少仓库的数据。例如输入10，则创建10个仓库数据，每一个数据仓库的数据量大概是 76823.04KB，可能有少量的上下浮动，因为测试过程中将会插入或删除现有记录。
loadworkers	参数表示加载数据时，每次提交的进程数。
terminals	终端数量，指同时有多少终端并发执行，表示并发程度。
runTxnsPerTerminal	每分钟每个终端执行的事务数。
runMins	执行多少分钟。
limitTnxsPermin	每分钟执行的事务总数。
terminalWarehouseFixed	用于指定终端和仓库的绑定模式，设置为true时可以运行4.x兼容模式，意思为每个终端都有一个固定的仓库。设置为false时可以均匀的使用数据库整体配置。 **注意：** runMins和runTxnsPerTerminal这两个参数指定了两种运行方式，前者是按照指定运行时间执行，以时间为标准；后者以指定每个终端的事务数为标准执行。两者不能同时生效，必须有一个设定为0。

按“i”进入编辑模式，如下图所示进行修改，按“Esc”退出编辑模式，输入“:wq”，保存退出。

```
db=postgres
driver=org.postgresql.Driver
conn=jdbc:postgresql://121.36.22.80:5432/tpcc
user=postgres
password=H[redacted]

warehouses=10
loadWorkers=100

terminals=1
//To run specified transactions per terminal- runMins must equal zero
runTxnsPerTerminal=100
//To run for specified minutes- runTxnsPerTerminal must equal zero
runMins=0
//Number of total transactions per minute
limitTxnsPerMin=300

//Set to true to run in 4.x compatible mode. Set to false to use the
//entire configured database evenly.
terminalWarehouseFixed=true
```

步骤 3　执行以下命令，初始化数据。

1. 在/run/目录下，执行 runDatabaseBuild.sh 脚本，加载数据。

./runDatabaseBuild.sh arm_postgres.properties

```
[root@ecs-postergre-scan-test run]# ./runDatabaseBuild.sh arm_postgres.properties
```

如下图所示的过程，需要等待较长时间。

```
Starting BenchmarkSQL LoadData

driver=org.postgresql.Driver
conn=jdbc:postgresql://121.36.22.80:5432/tpcc
user=postgres
password=***********
warehouses=10
loadWorkers=100
fileLocation (not defined)
csvNullValue (not defined - using default 'NULL')

Worker 000: Loading ITEM
Worker 001: Loading Warehouse      1
Worker 002: Loading Warehouse      2
Worker 003: Loading Warehouse      3
Worker 004: Loading Warehouse      4
Worker 005: Loading Warehouse      5
Worker 006: Loading Warehouse      6
Worker 007: Loading Warehouse      7
Worker 008: Loading Warehouse      8
Worker 010: Loading Warehouse     10
Worker 009: Loading Warehouse      9
Worker 000: Loading ITEM done
Worker 008: Loading Warehouse      8 done
Worker 007: Loading Warehouse      7 done
Worker 001: Loading Warehouse      1 done
Worker 002: Loading Warehouse      2 done
Worker 004: Loading Warehouse      4 done
Worker 009: Loading Warehouse      9 done
Worker 006: Loading Warehouse      6 done
Worker 005: Loading Warehouse      5 done
Worker 010: Loading Warehouse     10 done
Worker 003: Loading Warehouse      3 done
```

直到出现如下回显信息，表示数据加载成功。

```
# ------------------------------------------------------------
# Loading SQL file ./sql.postgres/buildFinish.sql
# ------------------------------------------------------------
-- ----
-- Extra commands to run after the tables are created, loaded,
-- indexes built and extra's created.
-- PostgreSQL version.
-- ----
vacuum analyze;
```

2. 等待加载完成后，返回 putty，登录鲲鹏云服务器 ecs_postgre_test，执行以下命令，连接 tpcc 数据库，并查看数据库表。

```
\c tpcc
\dt
```

```
postgres=# \c tpcc
You are now connected to database "tpcc" as user "postgres".
tpcc=# \dt
              List of relations
 Schema |       Name       | Type  |  Owner
--------+------------------+-------+----------
 public | bmsql_config     | table | postgres
 public | bmsql_customer   | table | postgres
 public | bmsql_district   | table | postgres
 public | bmsql_history    | table | postgres
 public | bmsql_item       | table | postgres
 public | bmsql_new_order  | table | postgres
 public | bmsql_oorder     | table | postgres
 public | bmsql_order_line | table | postgres
 public | bmsql_stock      | table | postgres
 public | bmsql_warehouse  | table | postgres
(10 rows)

tpcc=#
```

如上图所示，创建并初始化的表包括：9 张表（warehouse，stock，item，order-line，new-order，history，distirct，customer，oorder）和 1 张配置表。

3. 执行以下命令，退出数据库 tpcc。

```
\q
```

```
tpcc=# \q
[postgres@ecs-postgre-test ~]$
```

步骤 4 返回 putty，登录 x86 云服务器 ecs_postgre_scan_test，执行以下命令，运行 BenchmarkSQL 程序，对 PostgreSQL 数据库进行压力测试。

```
./runBenchmark.sh arm_postgres.properties
```

```
[root@ecs-postgre-scan-test run]# ./runBenchmark.sh arm_postgres.properties
22:11:30,207 [main] INFO   jTPCC : Term-00, C value for C_LAST during load: 17
22:11:30,207 [main] INFO   jTPCC : Term-00, C value for C_LAST this run:    127
22:11:30,207 [main] INFO   jTPCC : Term-00,                Term-00, Running Average tpmTOTAL: 305.84    Current tpmTOTAL: 696    Memory Usage: 15MB / 57MB
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00,
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00,
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00, Measured tpmC (NewOrders) = 128.83
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00, Measured tpmTOTAL = 302.62
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00, Session Start     = 2019-10-19 22:11:30
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00, Session End       = 2019-10-19 22:11:50
22:11:50,242 [Thread-1] INFO   jTPCC : Term-00, Transaction Count = 100
```

记录 Running Average tpmTOTAL 值“305.84”和 Current tpmTOTAL 值“696”。

步骤 5 执行以下命令，删除数据库和数据。

```
./runDatabaseDestroy.sh arm_postgres.properties
```

```
[root@ecs-postgre-scan-test run]# ./runDatabaseDestroy.sh arm_postgres.properties
# ------------------------------------------------------------
# Loading SQL file ./sql.common/tableDrops.sql
# ------------------------------------------------------------
drop table bmsql_config;
drop table bmsql_new_order;
drop table bmsql_order_line;
drop table bmsql_oorder;
drop table bmsql_history;
drop table bmsql_customer;
drop table bmsql_stock;
drop table bmsql_item;
drop table bmsql_district;
drop table bmsql_warehouse;
drop sequence bmsql_hist_id_seq;
```

--结束

6 PostgreSQL 性能调优

6.1 实验介绍

本实验介绍对 PostgreSQL 数据库进行性能调优的方法。

6.2 前提条件

- 鲲鹏云服务器已上电。
- 鲲鹏云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- x86 云服务器上已安装 BenchmarkSQL 性能测试工具。
- 鲲鹏云服务器上已安装 PostgreSQL 数据库 ,并已创建数据库用户 postgres 和数据库 tpcc。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。

6.3 PostgreSQL 性能调优

6.3.1 操作步骤

步骤 1 返回 putty，登录鲲鹏云服务器 ecs_postgre_test，编译参数调优。

重新移植 PostgreSQL 数据，使用更高效率的 ARMv8 原子指令。

1. 执行以下命令，退出 postgres 用户，重新切换到 root 用户，进入/home 目录，删除以下目录。

```
exit
cd /home/
ls
rm -rf /home/postgresql-11.3
rm -rf /home/pgsql/
rm -rf /home/postgres/logfile
```

```
rm -rf /home/postgres/pgsql/
```

```
[root@ecs-postgre-test ~]# exit
logout
[root@ecs-postgre-test postgresql-11.3]# cd /home/
[root@ecs-postgre-test home]# ls
pgsql  postgres  postgresql-11.3  postgresql-11.3.tar.gz
[root@ecs-postgre-test home]# rm -rf /home/postgresql-11.3
[root@ecs-postgre-test home]# rm -rf /home/pgsql/
[root@ecs-postgre-test home]# rm -rf /home/postgres/logfile
[root@ecs-postgre-test home]# rm -rf /home/postgres/pgsql/
```

2. 参考 3.5.1 中的步骤 2~4，重新解压 PostgreSQL 源码包，生成 Makefile 文件。

3. 执行以下命令，修改“src/Makefile.global”文件。

```
vi src/Makefile.global
```

```
[root@ecs-postgre-test postgresql-11.3]# vi src/Makefile.global
```

输入“:set nu”显示行号，在 261 行 CFLAGS 中，按“i”进入编辑模式，增加"-march=armv8-a+crc+lse"（性能优化选项 enable lse），按“Esc”退出编辑模式，输入“:wq!”，保存退出。

```
261 CFLAGS = -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-s
    tatement -Wendif-labels -Wmissing-format-attribute -Wformat-security -fno
    -strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncati
    on -O2 -march=armv8-a+crc+lse
```

4. 修改完成后，执行以下命令，安装 PostgreSQL 数据库。

```
make && make install
```

```
[root@ecs-postgre-test postgresql-11.3]# make && make install
```

步骤 2 参考 3.5.1 中的步骤 8~13，对 PostgreSQL 数据库进行初始化并启动，修改配置文件 pg_hba.conf。

步骤 3 数据库参数调优。

对于不同业务场景，通过在调整数据库的参数配置，可以有效提升服务器性能。

1. 执行以下命令，打开配置文件 postgresql.conf。

```
vi postgresql.conf
```

```
[postgres@ecs-postgre-test pgsql]$ vi postgresql.conf
```

2. 根据实际情况，按“i”进入编辑模式，如下图所示修改配置文件，并删除前面的“#”，按“Esc”退出编辑模式，输入“:wq!”，保存退出。

```
listen_addresses = '*'
bgwriter_delay = 10ms
bgwriter_lru_maxpages = 800
max_wal_size = 20GB
min_wal_size = 1GB
checkpoint_completion_target = 0.9
max_connections = 1000
checkpoint_timeout = 60min
full_page_writes = off
```

```
max_files_per_process = 100000
max_prepared_transactions = 2048
shared_buffers = 9GB
wal_buffers = 1GB
work_mem = 1GB
log_min_messages = FATAL
synchronous_commit = on
fsync = on
maintenance_work_mem = 2GB
vacuum_cost_limit = 10000
autovacuum = on
autovacuum_max_workers = 5
autovacuum_naptime = 20s
autovacuum_vacuum_scale_factor = 0.002
autovacuum_analyze_scale_factor = 0.001
```

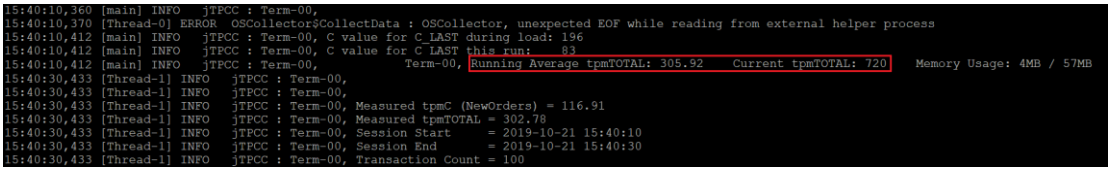
表6-1 数据库参数说明

参数名称	参数含义	优化建议
max_connections	允许客户端的最大并发连接数目。	可根据实际情况设置。
bgwriter_delay	后台写数据库进程的睡眠时间。	设置为最小10ms。
bgwriter_lru_maxpages	后台写数据库进程每次写脏数据块时，写到外部文件中的脏数据块的最大个数。	
max_wal_size	最大WAL x_log文件大小。	设置稍大以免频繁checkpoint。
checkpoint_completion_target	表示checkpoint的完成时间要在两个checkpoint间隔时间的N%内完成。	设置长一点，避免对性能测试的影响。
checkpoint_timeout	检查点周期。	
full_page_writes	在checkpoint之后在对页面的第一次写时将整个页面写到wal里面。	
shared_buffers	可以被PostgreSQL用于缓存数据的内存大小。	建议不超过物理内存60%，可根据实际情况调整。
wal_buffers	日志缓存区的大小。	
work_mem	数据库的排序操作和哈希表使用的内存缓冲区的大小。	

log_min_messages	控制写到数据库日志文件中的消息的级别。	
fsync	数据同步更新到磁盘。	默认打开，关闭会影响数据可靠性。
synchronous_commit	同步提交日志。	默认打开，关闭会影响数据可靠性。
maintenance_work_mem	数据库的维护操作使用的内存空间的大小，如VACUUM、CREATE INDEX和ALTER TABLE ADD FOREIGN KEY等。	
autovacuum_max_workers	设置系统自动清理工作进程的最大数量。	
autovacuum_naptime	设置两次系统自动清理操作之间的间隔时间。	

步骤 4　参考 3.5.1 中的步骤 14，重启数据库服务，使配置文件生效。

步骤 5　参考 5.3.1 中的步骤 1~3 和 5.4.1 中的步骤 3~4，返回 putty，登录 x86 云服务器 ecs_postgre_scan_test ，对 PostgreSQL 数据库重新进行性能测试。



从调优后的性能测试数据可以看到，Running Average tpmTOTAL 值“305.92”和 Current tpmTOTAL 值“720”与调优前相比均有提升，已达到调优的效果。若要达到最优性能，需要反复进行性能调优和测试，找到性能最优点。

--结束

7 PostgreSQL 软件打包

7.1 实验介绍

通过将 PostgreSQL 源码制作成 RPM 包或 DEB 包的方式，可以实现 PostgreSQL 数据库在新鲲鹏云服务器上的快速安装部署。本实验介绍了制作 PostgreSQL 源码 RPM 包的方法。

📖 说明：

制作 PostergreSQL 的 RPM 包是很复杂的过程，以下是精简的示例，不是完整的 RPM 包示例。

7.2 前提条件

- 鲲鹏云服务器已上电。
- 鲲鹏云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。

7.3 制作 RPM 包

7.3.1 操作步骤

步骤 1 参考 3.5.1 中的步骤 1，返回 putty，登录鲲鹏云服务器 ecs_postgre_rpm_test，执行以下命令，将鲲鹏云服务器 ecs_postgre_test 上的 PostgreSQL 源代码拷贝到鲲鹏云服务器 ecs_postgre_rpm_test。

```
cd /usr/local/src/
scp root@121.36.22.80:/home/postgresql-11.3.tar.gz ./
# 此处 IP 地址请参照自己的实验环境 ecs_postgre_test 的弹性公网 IP。
```

```
[root@ecs-postgre-rpm-test ~]# cd /usr/local/src/
[root@ecs-postgre-rpm-test src]# scp root@121.36.22.80:/home/postgresql-11.3.tar.gz ./
The authenticity of host '121.36.22.80 (121.36.22.80)' can't be established.
ECDSA key fingerprint is SHA256:7zPsrd9OzwUjfpgKh5DSoPGJOqZnl9pSG5oj7ikxk60.
ECDSA key fingerprint is MD5:23:9e:16:59:0a:f2:b5:d4:e3:af:38:5a:b6:c3:97:18.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '121.36.22.80' (ECDSA) to the list of known hosts.
root@121.36.22.80's password:
postgresql-11.3.tar.gz                        100%   25MB  13.7MB/s   00:01
```

步骤 2　执行以下命令，安装 RPM 包制作工具 rpmbuild。

```
yum -y install rpm-build
```

```
[root@ecs-postgre-rpm-test src]# yum -y install rpm-build
```

步骤 3　执行以下命令，生成制作 RPM 包的相关目录。

```
rpmbuild -ba pgsql.spec
```

```
[root@ecs-postgre-rpm-test src]# rpmbuild -ba pgsql.spec
error: failed to stat /usr/local/src/pgsql.spec: No such file or directory
```

说明：

若报错提示找不到 pgsql.spec，可忽略。执行以下命令，查看 root 目录，可以看到生成的 rpmbuild 目录以及相关的子目录。

```
ll /root/rpmbuild/
```

```
[root@ecs-postgre-rpm-test src]# ll /root/rpmbuild/
total 24
drwxr-xr-x 2 root root 4096 Oct 20 22:34 BUILD
drwxr-xr-x 2 root root 4096 Oct 20 22:34 BUILDROOT
drwxr-xr-x 2 root root 4096 Oct 20 22:34 RPMS
drwxr-xr-x 2 root root 4096 Oct 20 22:34 SOURCES
drwxr-xr-x 2 root root 4096 Oct 20 22:34 SPECS
drwxr-xr-x 2 root root 4096 Oct 20 22:34 SRPMS
```

步骤 4　执行以下命令，将 PostgreSQL 源码包拷贝至 rpmbuild 的 SOURCE 目录下。

```
cp /usr/local/src/postgresql-11.3.tar.gz /root/rpmbuild/SOURCES/
```

```
[root@ecs-postgre-rpm-test src]# cp /usr/local/src/postgresql-11.3.tar.gz /root/
rpmbuild/SOURCES/
```

步骤 5　执行以下命令，安装依赖包。

```
yum -y install readline-devel zlib-devel
```

```
[root@ecs-postgre-rpm-test src]# yum -y install readline-devel zlib-devel
```

步骤 6　执行以下命令，进入 rpmbuild 的 SPECS 目录，创建并打开编写制作 postgreSQL RPM 包的 spec 文件。

```
cd /root/rpmbuild/SPECS/
vi pgsql.spec
```

```
[root@ecs-postgre-rpm-test src]# cd /root/rpmbuild/SPECS/
[root@ecs-postgre-rpm-test SPECS]# vi pgsql.spec
```

按“i”进入编辑模式，输入以下内容，按“Esc”退出编辑模式，输入“:wq!”，保存退出。

```
Name:    postgresql
```

```
Version:  11.3
Release:  1%{?dist}
Summary:  This is a DB
License:  PostgreSQL
URL:      https://ftp.postgresql.org/
Source0:  postgresql-11.3.tar.gz
BuildRequires:   gcc
Requires: zlib-devel, readline-devel
%description
postgresql 11.3.
%prep
%setup -q
%build
%configure
make %{?_smp_mflags}
%install
make install DESTDIR=%{buildroot}
%files
%defattr(-,root,root,-)
%{_prefix}
%changelog
```

```
Name:     postgresql
Version:  11.3
Release:  1%{?dist}
Summary:  This is a DB
License:  PostgreSQL
URL:      https://ftp.postgresql.org/
Source0:  postgresql-11.3.tar.gz
BuildRequires:   gcc
Requires: zlib-devel, readline-devel
%description
postgresql 11.3.
%prep
%setup -q
%build
%configure
make %{?_smp_mflags}
%install
make install DESTDIR=%{buildroot}
%files
%defattr(-,root,root,-)
%{_prefix}
%changelog
```

步骤 7　执行以下命令，制作 RPM 包。

```
rpmbuild -ba pgsql.spec
```

```
[root@ecs-postgre-rpm-test SPECS]# rpmbuild -ba pgsql.spec
```

步骤 8　执行以下命令，查看生成的 RPM 包。

```
ll /root/rpmbuild/RPMS/aarch64/
```

```
[root@ecs-postgre-rpm-test SPECS]# ll /root/rpmbuild/RPMS/aarch64/
total 39488
-rw-r--r-- 1 root root 23036804 Oct 21 09:04 postgresql-11.3-1.el7.aarch64.rpm
-rw-r--r-- 1 root root 17392204 Oct 21 09:04 postgresql-debuginfo-11.3-1.el7.aarch64.rpm
```

--结束

8 PostgreSQL 软件包安装测试

8.1 实验介绍

本实验介绍了在新鲲鹏云服务器上通过制作的 PostgreSQL 源码 RPM 包，实现 PostgreSQL 数据库的快速安装部署。

8.2 前提条件

- 鲲鹏云服务器已上电。
- 鲲鹏云服务器与 PC 之间网络互通，且能访问外网。
- 云服务器上已安装 CentOS 7.6 操作系统，且已获取 root 用户帐号和密码。
- 鲲鹏云服务器上已完成 PostgreSQL 源码 RPM 包的制作。
- PC 上已安装 SSH 远程登录工具 putty 和 WinSCP。

8.2.1 RPM 包安装测试

8.2.2 操作步骤

步骤 1 返回 putty，登录鲲鹏云服务器 ecs_postgre_rpm_test，执行以下命令，将制作的 RPM 包拷贝到/home 目录下。

```
cp /root/rpmbuild/RPMS/aarch64/* /home
cd /home/
ls
```

```
[root@ecs-postgre-rpm-test SPECS]# cp /root/rpmbuild/RPMS/aarch64/* /home
[root@ecs-postgre-rpm-test SPECS]# cd /home/
[root@ecs-postgre-rpm-test home]# ls
postgresql-11.3-1.el7.aarch64.rpm  postgresql-debuginfo-11.3-1.el7.aarch64.rpm
```

步骤 2 执行以下命令，使用 RPM 包部署 PostgreSQL 数据库。

```
rpm -ivh postgresql-11.3-1.el7.aarch64.rpm --nodeps --force
```

```
[root@ecs-postgre-rpm-test home]# rpm -ivh postgresql-11.3-1.el7.aarch64.rpm --node
ps --force
Preparing...                          ################################# [100%]
Updating / installing...
   1:postgresql-11.3-1.el7            ################################# [100%]
```

步骤 3　执行以下命令，测试 postgreSQL 数据库。

创建 pgsql 运行用户和目录，并修改目录所属组。

```
useradd pgsql
cd /root/
mkdir -p /data/pgsql/data
chown -R pgsql:pgsql /data
```

```
[root@ecs-postgre-rpm-test home]# useradd pgsql
[root@ecs-postgre-rpm-test home]# cd /root/
[root@ecs-postgre-rpm-test ~]# mkdir -p /data/pgsql/data
[root@ecs-postgre-rpm-test ~]# chown -R pgsql:pgsql /data
```

切换至 pgsql 用户。

```
su - pgsql
```

```
[root@ecs-postgre-rpm-test ~]# su - pgsql
```

初始化数据库。

```
initdb -D /data/pgsql/data
```

```
[pgsql@ecs-postgre-rpm-test ~]$ initdb -D /data/pgsql/data
The files belonging to this database system will be owned by user "pgsql".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /data/pgsql/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /data/pgsql/data -l logfile start
```

启动数据库。

```
pg_ctl -D /data/pgsql/data -l logfile start
```

```
[pgsql@ecs-postgre-rpm-test ~]$ pg_ctl -D /data/pgsql/data -l logfile start
waiting for server to start.... done
server started
```

创建数据库。

```
createdb test
```

```
[pgsql@ecs-postgre-rpm-test ~]$ createdb test
```

进入数据库。

```
psql test
```

```
[pgsql@ecs-postgre-rpm-test ~]$ psql test
psql (11.3)
Type "help" for help.

test=#
```

创建表。

```
CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL, NAME TEXT  NOT NULL);
```

```
test=# CREATE TABLE COMPANY(ID INT PRIMARY KEY NOT NULL, NAME TEXT  NOT NULL);
CREATE TABLE
test=#
```

插入记录

```
INSERT INTO COMPANY VALUES (1, 'Hello');
```

```
test=# INSERT INTO COMPANY VALUES (1, 'Hello');
INSERT 0 1
test=#
```

查询数据记录。

```
select * from COMPANY;
```

```
test=# select * from COMPANY;
 id | name
----+-------
  1 | Hello
(1 row)

test=#
```

删除数据记录。

```
DELETE FROM COMPANY WHERE ID=1;
```

```
test=# DELETE FROM COMPANY WHERE ID=1;
DELETE 1
test=#
```

测试完成。

--结束