KYLINSOFT 麒麟软件

银河麒麟高级服务器操作系统软件适配常见问题指导手册

麒麟软件有限公司

生态发展中心

2025 年 12 月 08 日

版本说明

版本号	版本说明	作者	日期	变更内容
V1.0	首次发布	康佳楠、孙志刚、赵堃	2022-07-18	首次创建
V1.1	模板变更	刘佳鑫	2022-11-30	模板变更
V1.2	内容添加	唐金义	2023-06-28	内容添加
V1.3	内容添加	唐金义	2023-11-20	内容添加
V1.4	内容添加	冯培培	2025-10-23	添加 5 个常见问题、增加联系方式
V1.5	模板变更	冯培培	2025-12-08	模板变更

目 录

1 目的 4
2 范围 4
3 服务器软件适配常见问题 4
3.1 信息查询 4
3.1.1 如何下载麒麟系统镜像 4
3.1.2 如何获取系统的版本信息 4
3.1.3 如何获取系统的 cpu 信息 5
3.1.4 如何获取系统的架构信息 6
3.1.5 V10 常用的源地址 6
3.2 系统升级与安装问题 7
3.2.1 如何安装系统 7
3.2.2 如何将 V10 SP1 升级到 V10 SP2 24
3.3 YUM 源配置问题 34
3.3.1 如何搭建源服务器 34
3.3.2 本地 iso 镜像挂载 yum 源安装软件包 38
3.3.3 通过 ks.cfg 实现系统无人值守安装 41
3.4 服务器系统设置问题 45
3.4.1 locale 显示语言环境为 en_US.UTF-8 45
3.4.2 如何设置 core 文件 46
3.4.3 系统无法识别 exfat 格式的硬盘 46
3.4.4 系统中 netstat 命令不可用 46
3.4.5 如何修改网卡名称 46
3.4.6 如何安装图形化界面 48
3.4.7 如何利用 yum 降级安装软件包 50
3.4.8 如何将 txt 文件转换为 pdf 51
3.4.9 如何进入单用户模式 52
3.4.10 如何编译内核模块 55
3.4.11 如何开启/关闭开机自启动服务 59
3.4.12 手动挂载磁盘 61
3.4.13 根目录扩容 65
3.5 软件安装问题 69
3.5.1 如何安装 Mysql 数据库 69
3.5.2 如何安装其它版本的 docker 74
3.5.3 如何安装旧版本 cryptography 77
3.5.4 如何安装 Xpdf 79
3.5.5 如何安装 fish 80
3.5.6 如何安装 RabbitMQ 81
3.5.7 如何安装 PostgreSQL 84
3.5.8 如何安装 ODBC Driver for PostgreSQL 86
3.5.9 如何安装 sentencepiece 88
3.5.10 如何安装 zabbix 91

3.5.11 如何安装 geckodriver 92
3.5.12 如何安装.NET CORE 94
3.5.13 如何安装 apache-dophinescheduler 97
3.5.14 如何安装 mongodb-4.2.6 101
3.5.15 如何安装 mesos-1.8.1 105
3.5.16 如何安装 canu-1.8 109
3.5.17 如何安装 glibc-2.29 111
3.5.18 如何安装 codis-3.2.2 114
3.5.19 如何安装 kong-1.3.0 115
3.5.20 如何安装 sysak-1.3.0 122
3.5.21 如何安装 vernemq-1.11.0 124
3.5.22 如何安装 emqx-4.4.0 126
3.5.23 如何安装 fastjson-1.2.79 129
3.5.24 如何安装 FATE-1.9.0 132
3.5.25 如何安装 tars-2.4.12 141
3.5.26 如何安装 greenplum-6.20.0 143
3.6 虚拟机问题 151
3.6.1 如何扩展 swap 交换空间 151
3.6.2 KVM 虚拟机如何进行磁盘扩容 152
3.6.3 如何使用 virt-install 安装虚拟机 157
3.7 远程连接服务器问题 160
3.7.1 VNC 界面如何启动中文输入法 160
3.7.2 如何使用密钥连接服务器 161
3.7.3 如何部署虚拟多用户访问 ftp 163
3.7.4 xrdp 如何使用 xorg 登录 166
4 联系我们 170

1 目的

为了提高在银河麒麟高级服务器操作系统上的软件适配效率、增强麒麟软件在适配方面的服务质量，现对适配过程中遇到的有代表性的问题及解决方案进行梳理汇总，修订此指导手册，以供适配相关人员参考。

2 范围

本手册适用于第三方软件适配相关人员在与银河麒麟高级服务器操作系统适配过程中遇到问题时查阅参考。

3 服务器软件适配常见问题

3.1 信息查询

3.1.1 如何下载麒麟系统镜像

解决方法：登录麒麟生态网站 https://eco.kylinos.cn 点击合作伙伴-镜像下载。

备注：请先进行注册，填写信息提交申请，注册通过后即可下载系统镜像文件试用版。

3.1.2 如何获取系统的版本信息

解决方法：使用以下命令中的一个能查到即可：

```
# cat /etc/.productinfo
```

```
[root@localhost ~]# cat /etc/.productinfo
Kylin Linux Advanced Server
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
```

```
# nkvers
```

```
[root@localhost ~]# nkvers
############## Kylin Linux Version #################
Release:
Kylin Linux Advanced Server release V10 (Sword)

Kernel:
4.19.90-24.4.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
####################################################
```

```
# cat /etc/os-release
```

```
[root@localhost ~]# cat /etc/os-release
NAME="Kylin Linux Advanced Server"
VERSION="V10 (Sword)"
ID="kylin"
VERSION_ID="V10"
PRETTY_NAME="Kylin Linux Advanced Server V10 (Sword)"
ANSI_COLOR="0;31"
```

3.1.3 如何获取系统的 cpu 信息

解决方法：使用以下命令中的一个能查到即可:

```
# cat /proc/cpuinfo
```

```
[root@localhost ~]# cat /proc/cpuinfo
processor       : 0
vendor_id       : CentaurHauls
cpu family      : 6
model           : 94
model name      : Intel Core Processor (Skylake, IBRS)
stepping        : 3
cpu MHz         : 2694.620
cache size      : 16384 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid tsc_known_freq pni
lmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ibrs ibpb fsgsbase tsc_adjust bmi1 smep bmi2 invpcid rd
ed adx sha_ni xsaveopt arat umip arch_capabilities
bugs            : spectre_v1 spectre_v2 spec_store_bypass mds swapgs itlb_multihit
bogomips        : 5389.24
clflush size    : 64
cache_alignment : 128
address sizes   : 40 bits physical, 48 bits virtual
power management:
```

```
# lscpu
```

```
[root@localhost ~]# lscpu
架构：                           x86_64
CPU 运行模式：                   32-bit, 64-bit
字节序：                         Little Endian
Address sizes:                   40 bits physical, 48 bits virtual
CPU:                             16
在线 CPU 列表：                  0-15
每个核的线程数：                 1
每个座的核数：                   1
座：                             16
NUMA 节点：                      1
厂商 ID:                         CentaurHauls
CPU 系列：                       6
型号：                           94
型号名称：                       Intel Core Processor (Skylake, IBRS)
步进：                           3
CPU MHz:                         2694.620
BogoMIPS:                        5389.24
超管理器厂商：                   KVM
虚拟化类型：                     完全
L1d 缓存：                       512 KiB
L1i 缓存：                       512 KiB
L2 缓存：                        64 MiB
L3 缓存：                        256 MiB
NUMA 节点0 CPU:                  0-15
Vulnerability Itlb multihit:     KVM: Vulnerable
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Vulnerable: Clear CPU buffers attempted, no mic
                                 rocode; SMT Host state unknown
Vulnerability Meltdown:          Not affected
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user
                                  pointer sanitization
Vulnerability Spectre v2:        Mitigation; Full generic retpoline, IBPB condit
                                 ional, IBRS_FW, STIBP disabled, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
标记：                           fpu vme de pse tsc msr pae mce cx8 apic sep mtr
                                 r pge mca cmov pat pse36 clflush mmx fxsr sse s
                                 se2 ss syscall nx pdpe1gb rdtscp lm constant_ts
                                 c rep_good nopl xtopology cpuid tsc_known_freq
                                 pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2a
                                 pic movbe popcnt tsc_deadline_timer aes xsave a
                                 vx f16c rdrand hypervisor lahf_lm abm 3dnowpref
                                 etch invpcid_single ibrs ibpb fsgsbase tsc_adju
                                 st bmi1 smep bmi2 invpcid rdseed adx sha_ni xsa
                                 veopt arat umip arch_capabilities
```

3.1.4 如何获取系统的架构信息

解决方法：使用以下命令中的一个能查到即可:

```
# uname -a
```

```
[root@localhost ~]# uname -a
Linux localhost.localdomain 4.19.90-24.4.v2101.ky10.x86_64 #1 SMP Mon May 24 12:
14:55 CST 2021 x86 64 x86 64 x86 64 GNU/Linux
```

```
# arch
```

```
[root@localhost ~]# arch
x86_64
```

3.1.5 V10 常用的源地址

解决方法：http://update.cs2c.com.cn:8080/NS/V10/

3.2 系统升级与安装问题

3.2.1 如何安装系统

3.2.1.1 系统版本

适用系统：V10(SP1)、V10(SP2)

适用架构：全架构

其他版本可作参考。

3.2.1.2 问题描述

服务器如何安装银河麒麟操作系统?

3.2.1.3 问题分析

利用光盘或者 U 盘启动盘进行安装。

3.2.1.4 解决方案

一、安装前准备

利用 DVD 光盘或者 U 盘启动盘安装。

1) U 盘启动盘制作方法：

在 windows 系统上使用 UltraISO 或其他启动盘制作工具一般制作的都是针对 windows 系统的启动盘，对 linux 系统不是很适用，需要使用其他软件或命令来制作，一般有如下两个方法：

a. 在 windows 系统安装 FedoraMediaWriter,使用自定义的方式制作镜像；（容易导致 U 盘制作成启动盘后在 windows 系统无法识别）

b. 在 linux 下使用 dd 命令来制作 U 盘启动盘,dd 命令制作方法具体如下(推荐此方法）：

首先我们需要找一台 linux 系统。

①下载镜像，检验 MD5 码

在终端，使用 md5sum ***.iso 命令来查看镜像文件的 md5 值（***.iso

指的是下载的 iso 镜像），查看是否和下载时提供的 MD5 值是否一致，如果一致则镜像完整，如果不一致说明镜像下载不完整，需要重新下载。

```
# md5sum ***.iso
```

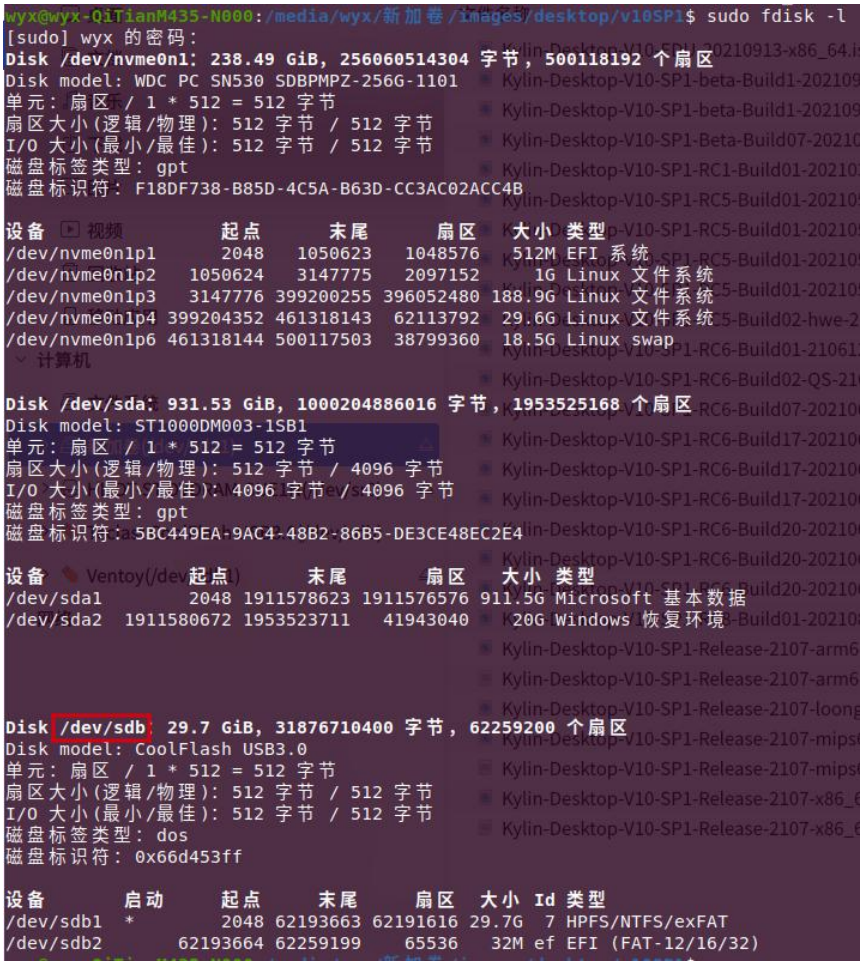
②U 盘连接电脑

将 U 盘插入电脑。

③查看挂载的 U 盘设备路径：

使用命令 sudo fdisk -l 查看 U 盘挂载的设备路径,此处可以看到是 /dev/sdb，一般都是这个，有些可能是 sda 或 sdc。

```
# sudo fdisk -l
```



④写入镜像

```
# sudo dd if=/***.iso of=/dev/xxx
```

#if 后接镜像文件路径，of 后接写入 U 盘的路径

如上图所示，这种代表已经写入完成了，现在就可以拿着 U 盘去做系统了。

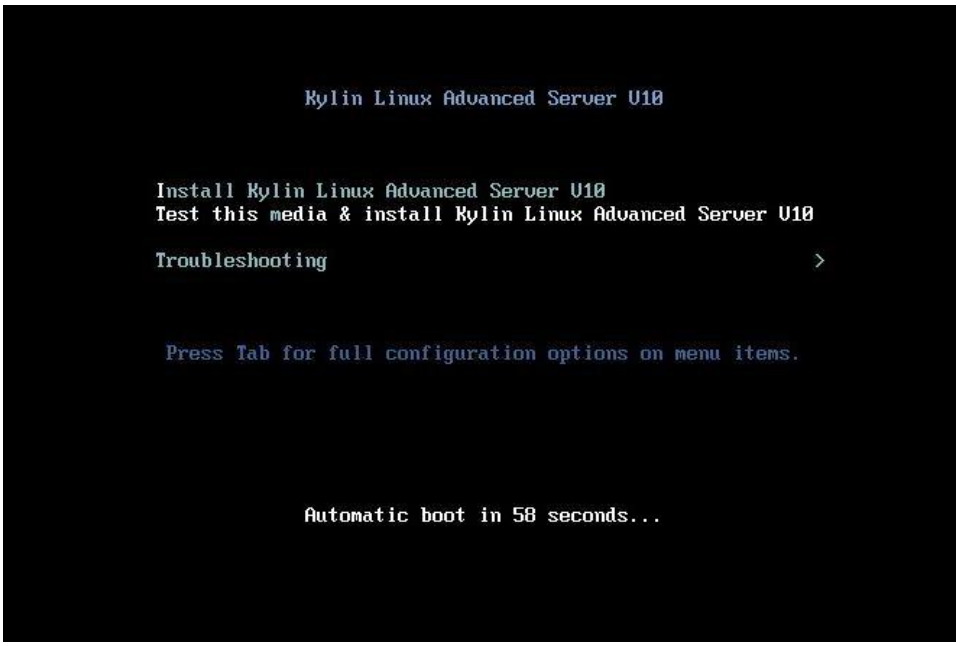
2) 利用 DVD 光盘安装

请将安装光盘放入光驱中，并设置计算机 BIOS 为光驱引导，重新启动计算机后，安装光盘会自动运行，这样就可以开始安装银河麒麟服务器操作系统了。

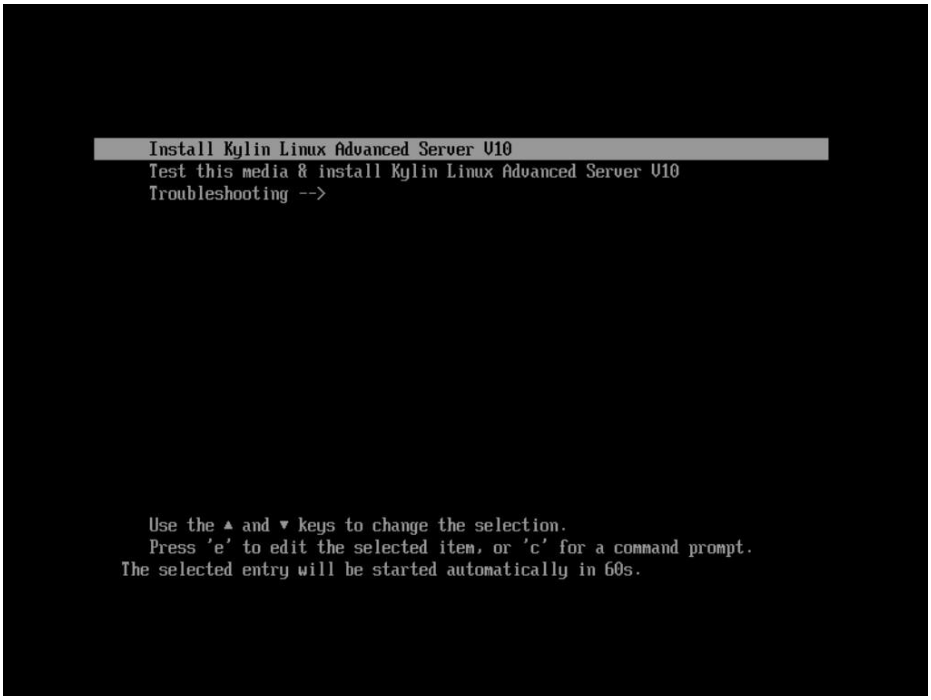
安装银河麒麟服务器操作系统建议最少准备 8G 内存、30GB 磁盘空间，并与其他操作系统（如 Windows 或其他版本的 Linux）使用的硬盘空间分开。

二、引导安装

从光盘引导安装时首先进入的是安装引导页面，如下图：

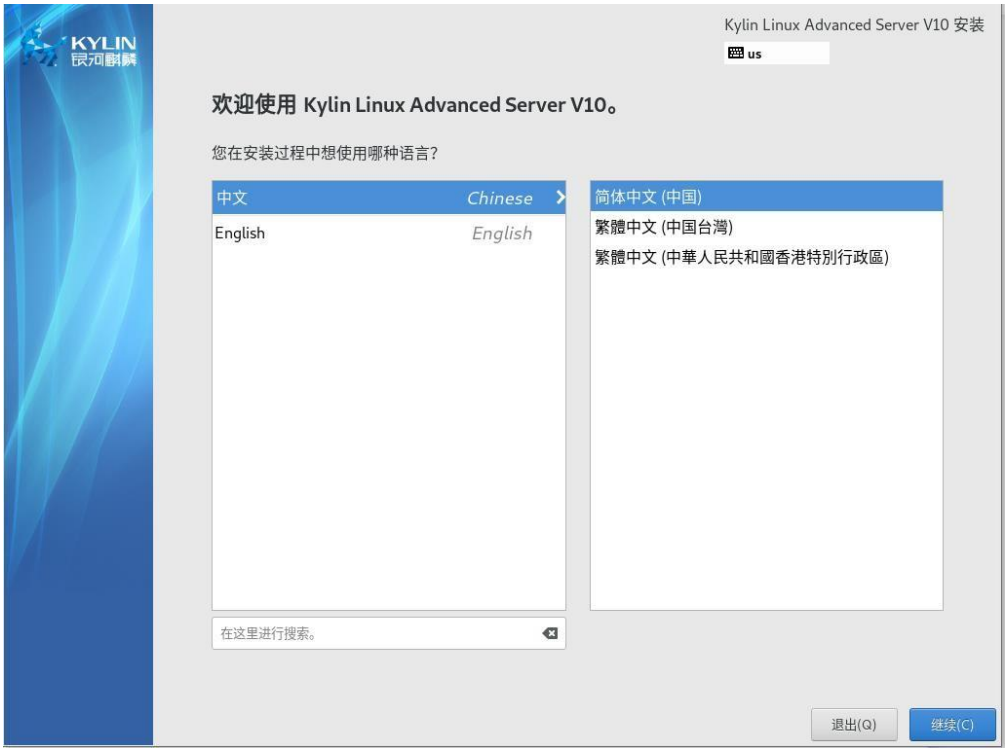


使用向上方向键<↑>选择“Install Kylin Linux Advanced Server V10”，按<Enter>进入安装过程。



三、欢迎页面

安装过程首先进入欢迎页面【欢迎使用 Kylin Linux Advanced Server V10】，如下图：



选择安装过程中所使用的语言，默认采用【简体中文(中国)】。确定安装语言后，点击【继续(C)】进入【安装信息摘要】页面。

四、安装信息摘要

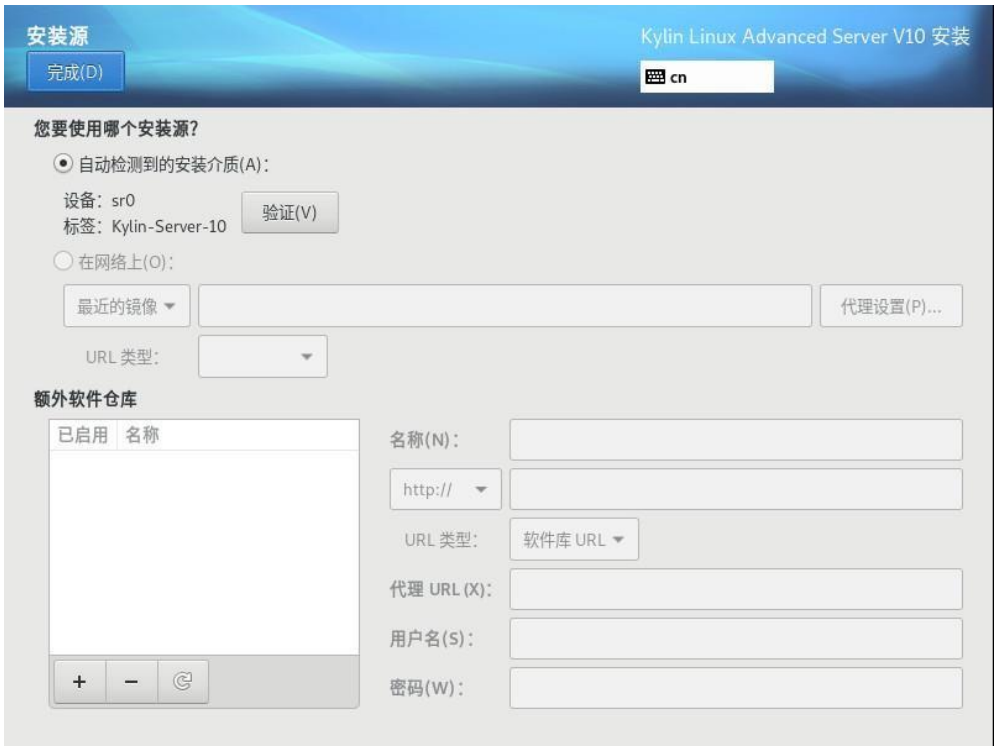
在【安装信息摘要】页面配置所有与安装相关的信息，如下：

点击相应配置项的图标进入配置页面，需要注意的是【软件】和【系统】两类配置。安装程序会自动检测各个配置项；如果检测存在无法确认或不正确配置项，相应配置项的图标上会显示感叹号。在该配置界面只有正确进行了全部配置，才能进行下一步操作。

在【软件】中，有【安装源】、【软件选择】的配置。

1) 安装源

安装源用于指定银河麒麟服务器操作系统安装介质的位置。使用 DVD 进行光盘安装时会自动识别安装介质，通常您不需要改动。除了光盘安装，您还可以选择 ISO 文件、网络或 USB 盘安装，并且可以配置额外软件仓库。如下图所示：



2) 软件选择

根据不同业务系统运行的服务器操作系统环境需求,安装程序默认提供了几种【基本环境】安装选择，安装程序默认选项是【带 UKUI GUI 的服务器】。

【基本环境】安装选择包括(在每个选项的右侧窗口，可选择需要安装的组件):

a. 最小安装【基本功能】

b. 基础设施服务器【集成的易于管理的服务器】

c. 文件及打印服务器【用于企业的文件、打印及存储服务器】

d. 基本网页服务器【提供静态及动态互联网内容的服务器】

e. 虚拟化主机【最小虚拟化主机】

f. 带 UKUI GUI 的服务器【带有用于操作网络基础设施服务 UKUI GUI 的服务器】

具体如下图所示：

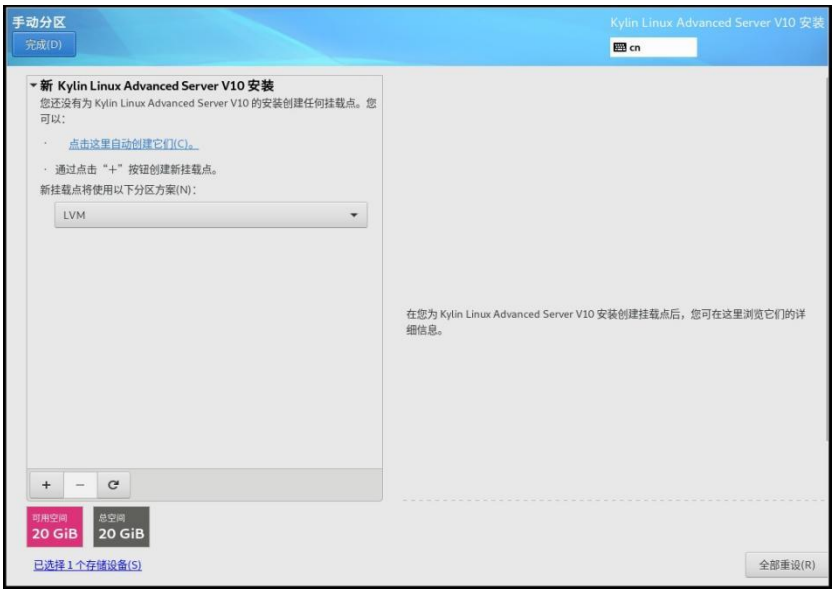
从【基本环境】中选择分组后，该分组默认必须安装的软件不可配置，但是可以从【已选环境的附加选项】定制安装其他软件。

在【系统】中，为用户提供了【安装位置】、【KDUMP】、【网络和主机名】的配置。

3) 安装位置

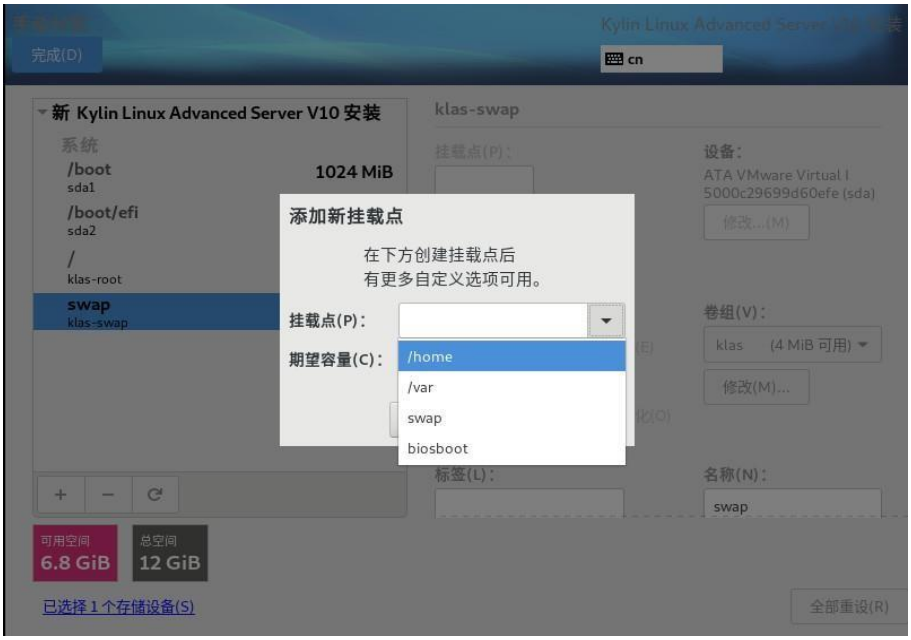
本页面用于配置安装磁盘及分区,如果使用的是全新的磁盘并且希望使用全部磁盘空间，可以使用默认的自动分区配置，直接点击【完成】返回【安装信息摘要】页面即可，如下图：

如果将要安装的机器还有其他操作系统或数据分区需要保留，千万注意提前做好数据备份。这时需要选择手动分区，请选择【自定义】，并点击【完成】，进入【手动分区】页面，如下图所示：



在【手动分区】页面中，可以使用【点这里自动创建他们(C)。】进行自动分区创建，也可以手动选择合适的分区方案并手动创建挂载点，一般使用【标准分区】方案。点击【＋】，会弹出【添加新挂载点】窗口，该窗口可完成【挂载点(P)】和【期望容量(C)】的设置，需要注意的是在 AARCH 平台下无 biosboot

挂在点，如下图：



配置挂载点时一般会单独设置【/boot】分区、【swap】分区和【/】分区，成功完成这步后，可以显示已配置挂载点并可以进行一些修改，如下图所示：

系统默认采用 XFS 文件系统，龙芯平台默认采用 ext4 文件系统。如果有其他需要可以更改文件系统类型。当然，也可以在【手动分区】中，通过点击

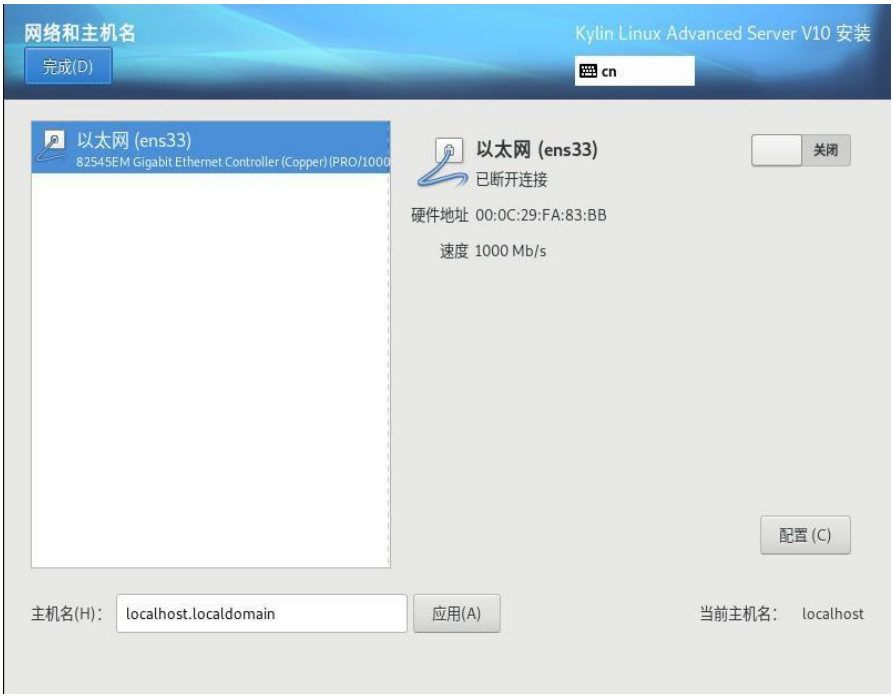
【-】来删除划错或不需要的分区。

4) KDUMP

Kdump 是一个内核崩溃转储机制。在系统崩溃时，kdump 将捕获系统信息，有助于诊断系统崩溃原因。注意：kdump 需要预留一部分内存，并且不可以被其它用户使用。运行 free 命令也是不会显示这部分保留内存的。默认在安装中是启用 kdump 的。用户可以根据自己的需求进行取舍。

5) 网络和主机名

安装程序自动探测可本地访问的接口。探测到的接口列在左侧方框中。在右侧点击列表中的接口显示详情。要激活或者取消激活网络接口，请将页面右侧的开关转到【开启】或者【关闭】。



在连接列表下方，在【主机名(H)】输入框中输入这台计算机的主机名。主机名可以是完全限定域名（FQDN），其格式为 hostname.domainname，也可以是简要主机名，其格式为 hostname。很多网络有动态主机配置协议（DHCP）服务，它可自动提供带域名的连接的系统。要允许 DHCP 服务为这台机器分配域名，只指定简要主机名即可。

五、开始安装

完成全部配置后，在【安装信息摘要】页面点击【开始安装】按钮，就进入自动安装进程。

在安装过程中，必须设置【ROOT 密码】，可以选择是否【创建用户】，系统安装完毕并完成配置后，会提示【重启(R)】系统。

六、安装完成

安装完成后第一次启动银河麒麟高级服务器操作系统系统,需要进行相关的初始设置。初始设置程序会提示是否接受许可协议和最终用户隐私声明；接受许可协议才可完成配置。如果在安装过程中没有创建登录系统的普通用户账户，还可以在这里创建普通用户账号。

现在可以正式使用银河麒麟高级服务器操作系统了。

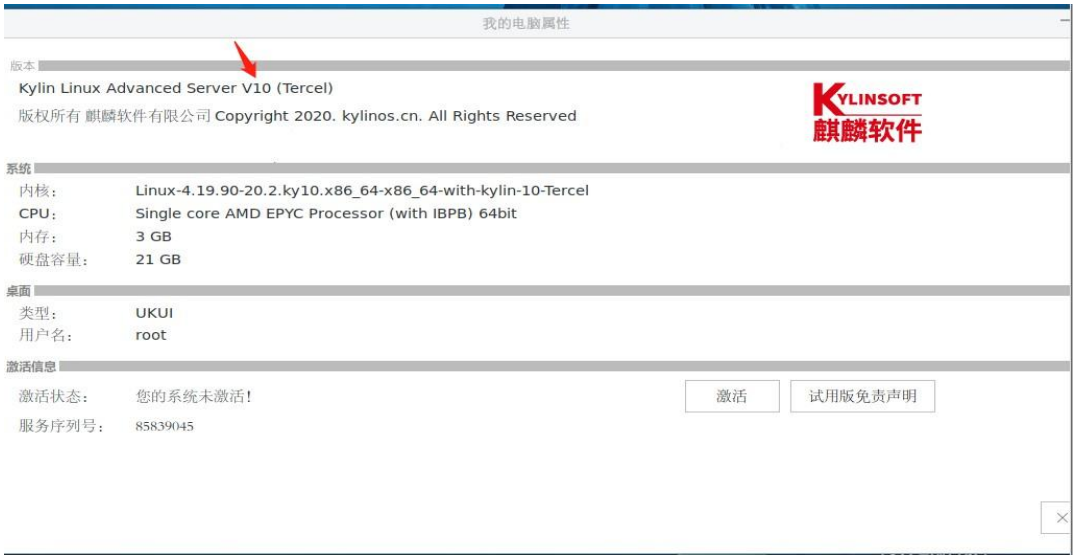
七、授权激活

1) 扫码激活

系统未激活时“关于银河麒麟”的“服务序列号”显示为 8 位的服务序列号、“客户单位”显示客户单位名称，如下图：

鼠标右键点击我的电脑->属性，会显示操作系统版本信息、系统信息、桌

面环境信息以及激活状态，系统未激活时激活状态显示为“您的系统还未激活，请激活”，“服务序列号”显示为 8 位的服务序列号。如下图：



点击“激活”按钮，进入如下激活窗口：

点击“序列号”的编辑按钮可以填写 8 位及以上的服务序列号、点击“客户

单位”的编辑按钮可以填写客户单位信息，如下图：

填写好序列号和客户单位信息就可通过微信扫描二维码进行系统激活，如下图进入企业微信平台。

点击“获取激活码”按钮跳转，输入机器型号、装机地点以及验证码来获取

激活码，如下图：

激活码获取成功，系统已激活，如下图：

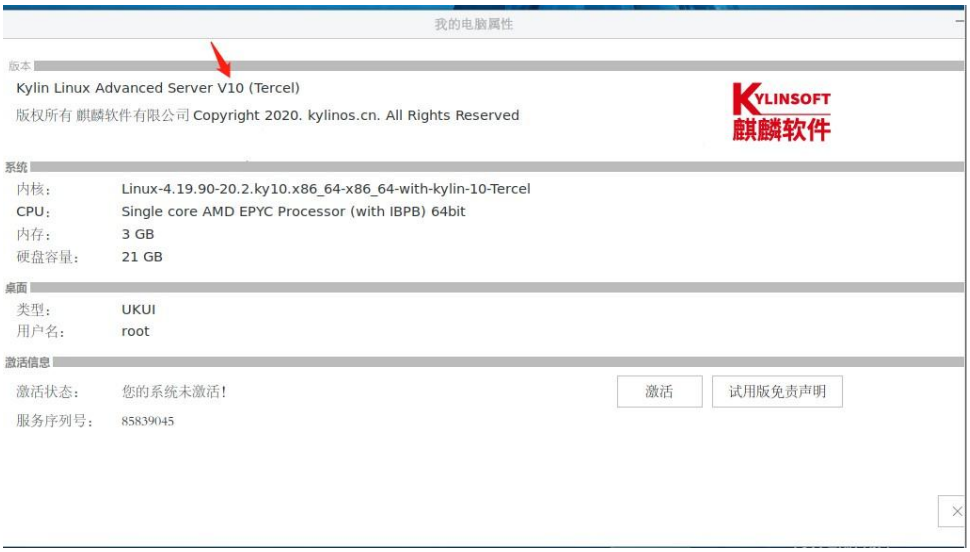
输入“激活码”，点击“激活/延长服务”，会提示“激活/延长服务，请重启系统！”，激活成功如下图：

2） Ukey 激活

ukey 中包含批次号和服务序列号，ukey 与系统信息检验逻辑如下。

		Ukey	
		含批次号、含服务序列号	**不含批次号、含服务序列号**
系统	批次号	批次号一致才可以激活系统	可以激活系统（默认不提供）
	服务序列号	服务序列号一致，才可以激活系统	服务序列号一致，才可以激活系统

激活成功后“我的电脑属性”的“激活状态”显示为“您的系统已激活，”“服务序列号”显示为 8 位及以上的服务序列号。



系统激活后“关于银河麒麟”的“服务序列号”显示为 8 位及以上的服务序列号、“客户单位”显示激活时填写的单位名称。

3.2.2 如何将 V10 SP1 升级到 V10 SP2

3.2.2.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH

其他版本和架构可作参考。

3.2.2.2 问题描述

如何将 V10 SP1 系统直接升级到 V10 SP2 系统?

3.2.2.3 问题分析

该解决方法对银河麒麟高级服务器操作系统软件通过外部仓库源分别从 V10SP1 Build04（20200711）、Build10（20201202）、Build19（20210319）、Build20（20210518）更新到 V10SP2 Build09（20210524）的基本步骤进行说明。

如果把 V10 SP1 低版本的升级到高版本的也可以使用此方法。

此方法升级后会把内核及软件版本升级到最新（updates 中）的，如果不想升级到最新的,只升级到正常 iso 安装的状态的话,可以把外部源中 updates 的 enabled 开关关闭（改成 0）。

以 x86 平台为例进行说明，对于 aarch64 平台，请将 x86_64 替换为 aarch64。

3.2.2.4 解决方案

V10SP2 仓库源更新

前提：由于 audit、setroubleshoot 等安全相关的软件包会受 selinux 的 Enforcing 状态影响，建议先将 selinux 的状态设置为 Disabled 或者 Permissive 状态，然后再执行更新。

```
[root@localhost ~]# setenforce 0
```

1) 从 SP1 Build04 更新到 SP2 Build09

a. 更新前检查

执行 nkvers，检查 OS 的版本信息为 V10SP1 Build04

```
[root@localhost ~]# nkvers
############### Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Tercel)

Kernel:
4.19.90-17.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP1) /(Tercel)-x86_64-Build04/20200711
#######################################################
#
```

b. 编辑仓库配置文件

```
# vi /etc/yum.repos.d/kylin_x86_64.repo
```

①修改 base 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-os]
name = Kylin Linux Advanced Server 10 - Os
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/base/$basearch/
gpgcheck = 0
enabled = 1
```

②修改 update 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/updates/$basea
rch/
gpgcheck = 0
enabled = 1
```

c. 执行更新

客户可以根据具体需要更新某个软件包，也可以选择全量更新。需要注意的是：如果选择全量更新，则 OS 的版本信息也会被修改。

①更新指定的软件包

此处以更新 kylin-menu 为例进行说明

```
# yum update kylin-menu
```

②全量更新

更新 yum 工具

```
# yum update yum
```

执行全量更新

```
# yum update
```

执行成功后使用新版本的仓库配置

```
# cd /etc/yum.repos.d
# cp kylin_x86_64.repo kylin_x86_64.repo.0711
# mv kylin_x86_64.repo.rpmnew kylin_x86_64.repo
```

③重启

当进行了全量更新或核心包（如 kernel 等）的更新时，请执行 reboot 重启 OS。

④说明

更新或使用中遇到的问题，请参考 FAQ 手册【FAQ 的获取请咨询产品部】

d. 更新后检查

当更新 kylin-release 包或执行全量更新后，系统版本会发生变化。可以执行 nkvers 进行检查，OS 的版本信息被更新为 V10SP2 Build09。

```
[root@localhost ~]# nkvers
############### Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Sword)

Kernel:
4.19.90-24.4.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
#######################################################
#
```

2) 从 SP1 Build10 更新到 SP2 Build09

a. 更新前检查

执行 nkvers，检查 OS 的版本信息为 V10SP1 Build10

```
[root@localhost ~]# nkvers
############### Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Tercel)

Kernel:
4.19.90-21.2.ky10.x86_64

Build:
```

```
Kylin Linux Advanced Server
release V10 (SP1) /(Tercel)-x86_64-Build10/20201202
##########################################################
#
```

b. 编辑仓库配置文件

```
# vi /etc/yum.repos.d/kylin_x86_64.repo
```

①修改 base 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-os]
name = Kylin Linux Advanced Server 10 - Os
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/base/$basearch/
gpgcheck = 0
enabled = 1
```

②修改 update 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/updates/$basea
rch/
gpgcheck = 0
enabled = 1
```

c. 执行更新

客户可以根据具体需要更新某个软件包，也可以选择全量更新。需要注意的是：如果选择全量更新，则 OS 的版本信息也会被修改。

①更新指定的软件包

此处以更新 kylin-menu 为例进行说明

```
# yum update kylin-menu
```

②全量更新

更新 yum 工具

```
# yum update yum
```

执行全量更新

```
# yum update
```

执行成功后使用新版本的仓库配置

```
# cd /etc/yum.repos.d
# cp kylin_x86_64.repo kylin_x86_64.repo.1202
# mv kylin_x86_64.repo.rpmnew kylin_x86_64.repo
```

③重启

当进行了全量更新或核心包（如 kernel 等）的更新时，请执行 reboot 重启 OS。

④说明

更新或使用中遇到的问题，请参考 FAQ 手册【FAQ 的获取请咨询产品部】

d. 更新后检查

当更新 kylin-release 包或执行全量更新后，系统版本会发生变化。可以执行 nkvers 进行检查，OS 的版本信息被更新为 V10SP2 Build09。

```
[root@localhost ~]# nkvers
############## Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Sword)

Kernel:
4.19.90-24.4.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
#####################################################
#
```

3) 从 SP1 Build19 更新到 SP2 Build09

a. 更新前检查

执行 nkvers，检查 OS 的版本信息为 V10SP1 Build19

```
# nkvers
```

```
############## Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Tercel)

Kernel:
4.19.90-23.6.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP1) /(Tercel)-x86_64-Build19/20210319
#############################################################
#
```

b. 编辑仓库配置文件

```
# vi /etc/yum.repos.d/kylin_x86_64.repo
```

①修改 base 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-os]
name = Kylin Linux Advanced Server 10 - Os
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/base/$basearch/
gpgcheck = 1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
enabled = 1
```

②修改 update 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/updates/$basea
rch/
gpgcheck = 1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
enabled = 1
```

c. 执行更新

客户可以根据具体需要更新某个软件包，也可以选择全量更新。需要注意的是：如果选择全量更新，则 OS 的版本信息也会被修改。

①更新指定的软件包

此处以更新 kylin-menu 为例进行说明

```
# yum update kylin-menu
```

②全量更新

更新 yum 工具

```
# yum update yum
```

执行全量更新

```
# yum update
```

执行成功后使用新版本的仓库配置

```
# cd /etc/yum.repos.d
# cp kylin_x86_64.repo kylin_x86_64.repo.0319
# mv kylin_x86_64.repo.rpmnew kylin_x86_64.repo
```

③重启

当进行了全量更新或核心包（如 kernel 等）的更新时，请执行 reboot 重启 OS。

④说明

更新或使用中遇到的问题，请参考 FAQ 手册【FAQ 的获取请咨询产品部】

d. 更新后检查

当更新 kylin-release 包或执行全量更新后，系统版本会发生变化。可以执行 nkvers 进行检查，OS 的版本信息被更新为 V10SP2 Build09。

```
[root@localhost ~]# nkvers
############### Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Sword)

Kernel:
4.19.90-24.4.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
```

```
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
##################################################
#
```

4) SP1 Build20 更新到 SP2 Build09

a. 更新前检查

执行 nkvers，检查 OS 的版本信息为 V10SP1 Build20

```
[root@localhost ~]# nkvers
############## Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Tercel)

Kernel:
4.19.90-23.8.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP1) /(Tercel)-x86_64-Build20/20210518
##################################################
#
```

b. 编辑仓库配置文件

```
# vi /etc/yum.repos.d/kylin_x86_64.repo
```

①修改 base 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-os]
name = Kylin Linux Advanced Server 10 - Os
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/base/$basearch/
gpgcheck = 1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
enabled = 1
```

②修改 update 仓库源

修改 baseurl 为红色字体部分

```
[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl
=  http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/updates/$basea
```

```
rch/
gpgcheck = 1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
enabled = 1
```

c. 执行更新

客户可以根据具体需要更新某个软件包，也可以选择全量更新。需要注意的是：如果选择全量更新，则 OS 的版本信息也会被修改。

①更新指定的软件包

此处以更新 kylin-menu 为例进行说明

```
# yum update kylin-menu
```

②全量更新

更新 yum 工具

```
# yum update yum
```

执行全量更新

```
# yum update
```

执行成功后使用新版本的仓库配置

```
# cd /etc/yum.repos.d
# cp kylin_x86_64.repo kylin_x86_64.repo.0518
# mv kylin_x86_64.repo.rpmnew kylin_x86_64.repo
```

③重启

当进行了全量更新或核心包（如 kernel 等）的更新时，请执行 reboot 重启 OS。

④说明

更新或使用中遇到的问题，请参考 FAQ 手册【FAQ 的获取请咨询产品部】

d. 更新后检查

当更新 kylin-release 包或执行全量更新后，系统版本会发生变化。可以执行 nkvers 进行检查，OS 的版本信息被更新为 V10SP2 Build09。

```
# nkvers
############## Kylin Linux Version ##################
Release:
Kylin Linux Advanced Server release V10 (Sword)

Kernel:
4.19.90-24.4.v2101.ky10.x86_64

Build:
Kylin Linux Advanced Server
release V10 (SP2) /(Sword)-x86_64-Build09/20210524
#######################################################
#
```

3.3 yum 源配置问题

3.3.1 如何搭建源服务器

3.3.1.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：全架构

3.3.1.2 问题描述

如何搭建源服务器?

3.3.1.3 问题分析

主要修改/etc/yum.repos.d 文件里的源地址。

3.3.1.4 解决方案

以 V10(SP1)为例。

1) 创建文件夹

a. 创建存放 base 目录下文件的文件夹

```
# mkdir -p /yum/var/www/html/V10/V10SP1/os/adv/lic/base
```

b. 创建存放 addons 目录下文件的文件夹

```
# mkdir -p /yum/var/www/html/V10/V10SP1/os/adv/lic/addons
```

c. 创建存放 updates 目录下文件的文件夹

```
# mkdir -p /yum/var/www/html/V10/V10SP1/os/adv/lic/updates
```

2) 备份现有 repo 文件

```
# cp /etc/yum.repos.d/kylin_aarch64.repo /etc/yum.repos.d/kylin_aarch64.repo_bak
```

3) 创建新的 repo 文件

```
# mkdir /etc/yum.repos.d/yumsync.repo
```

内容如下：

```
[x86_64]
name = x86_64
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/base/x86_64/
gpgcheck = 0
enabled = 1

[aarch64]
name = aarch64
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/base/aarch64/
gpgcheck = 0
enabled = 1

[mips64el]
name = mips64el
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/base/mips64el/
gpgcheck = 0
enabled = 1
```

以上内容为同步 base 目录下文件，如果需要同步 addons/updates 目录下文件内容如下：

```
addons
[x86_64]
name = x86_64
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/addons/x86_64/
gpgcheck = 0
enabled = 1

[aarch64]
name = aarch64
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/addons/aarch64/
```

```
gpgcheck = 0
enabled = 1

[mips64el]
name = mips64el
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/addons/mips64e
l/
gpgcheck = 0
enabled = 1
updates

[aarch64]
name = aarch64
baseurl
=  http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/updates/aarch6
4/
gpgcheck = 0
enabled = 1

[mips64el]
name = mips64el
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/updates/mips64
el/
gpgcheck = 0
enabled = 1

[x86_64]
name = x86_64
baseurl
= http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/updates/x86_64/
gpgcheck = 0
enabled = 1
```

4) 更新索引

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

5) 同步数据

以同步 base 目录下文件为例

```
# reposync -p /yum/var/www/html/V10/V10SP1/os/adv/lic/base
```

6) 生成索引

```
# createrepo  /yum/var/www/html/V10/V10SP1/os/adv/lic/base/aarch64
# createrepo  /yum/var/www/html/V10/V10SP1/os/adv/lic/base/x86_64
# createrepo  /yum/var/www/html/V10/V10SP1/os/adv/lic/base/mips64el
```

7) 更新索引

```
# createrepo --update /yum/var/www/html/V10/V10SP1/os/adv/lic/base/aarch64
# createrepo --update /yum/var/www/html/V10/V10SP1/os/adv/lic/base/x86_64
# createrepo --update /yum/var/www/html/V10/V10SP1/os/adv/lic/base/mips64el
```

8) 修改权限

```
# chmod 755 -R /yum/var/www/html/
```

如果同步 addons/updates 目录下文件请按 2）-8）步执行一遍，需要修改 repo 文件地址。

9) 搭建 apache

a. 安装 httpd

```
# yum install httpd
```

b. 启动 httpd

```
# systemctl start httpd.server
# systemctl enable httpd.server
```

c. 查看是否启动成功

```
# netstat -luntp | grep 80                #80 端口为 apache 默认端口
```

显示如下表示启动成功

```
tcp6  0  0:::80  :::*  LISTEN  88645/httpd
```

10)发布

a. 建立软链接

```
# ln -s /yum/var/www/html/ /var/www/html/
```

b. 打开 80 端口访问权限

```
# iptables -I INPUT -p TCP --dport 80 -j ACCEPT
```

此时应该可以通过 http 访问/yum/var/www/html 目录。

11)使用

打开/etc/yum.repos.d/目录下的 repo 文件

修改里面的链接地址前缀如下：

修改完成后保存，然后更新索引

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

接下来就可以正常使用本地源。

关于源定时同步可以编写脚本，然后配置 crontab 定期执行更新。

路径：/etc/crontab

3.3.2 本地 iso 镜像挂载 yum 源安装软件包

3.3.2.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：全架构

3.3.2.2 问题描述

yum 如何挂载本地镜像源?

3.3.2.3 问题分析

主要修改/etc/yum.repos.d 文件里的源地址。

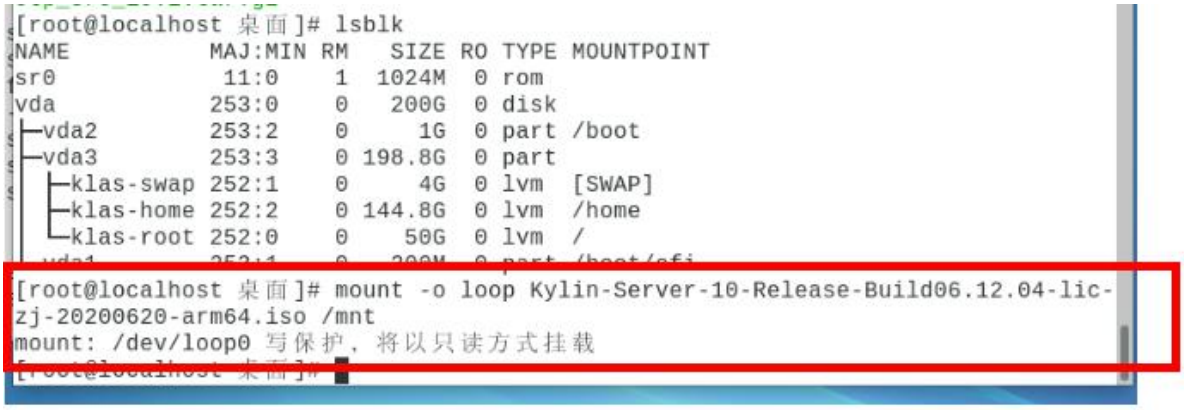
3.3.2.4 解决方案

1) 拷贝镜像到本地

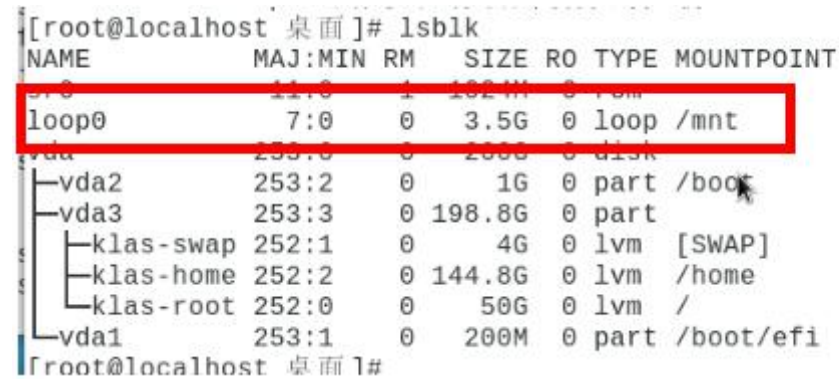
2) 执行以下命令：

```
# mount –o loop 镜像路径及镜像名字 /mnt（或 media）
```

挂载前



挂载后

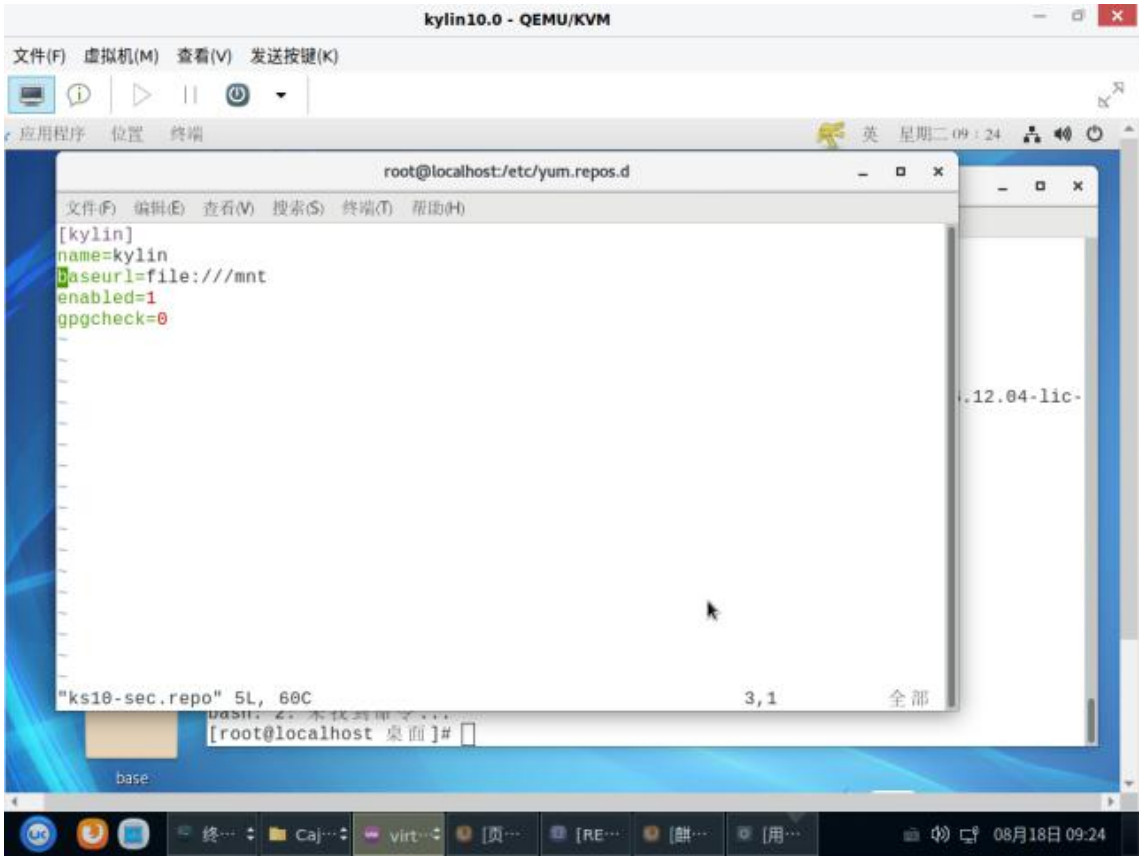


3) 进入/etc/yum.repos.d（yum.repos.d 是一个目录，该目录是分析 RPM 软件后所产生的软件属性依赖数据放置处）

```
[root@localhost etc]# cd yum
yum/          yum.repos.d/
[root@localhost etc]# cd yum.repos.d/
[root@localhost yum.repos.d]# ls
ks10-adv.repo.bak  ks10-sec.repo
[root@localhost yum.repos.d]# pwd
/etc/yum.repos.d
[root@localhost yum.repos.d]# vim ks10-sec.repo
[root@localhost yum.repos.d]# ls
ks10-adv.repo.bak  ks10-sec.repo
[root@localhost yum.repos.d]#
```

4) 备份/etc/yum.repos.d 下所有文件(本地源和外部源不能同时存在，修改外部源 ks10-adv.repo 的名字)，在 ks10-sec.repo（本地源）中加入以下内容：

```
[kylin]
name=kylin
baseurl=file:///mnt
enabled=1
gpgcheck=0
```



修改完之后退出编辑模式，执行：wq 保存退出。

5)

a. 清除本地内存

```
# yum clean all
```

b. 把服务器的包信息下载到本地电脑缓存起来

```
# yum makecache
```

c. 升级所有包同时也升级软件和系统内核

```
# yum update
```

3.3.3 通过 ks.cfg 实现系统无人值守安装

3.3.3.1 系统版本

适用系统：V10（SP2）、V10（SP3）

适用架构：X86、AARCH

3.3.3.2 问题描述

如何通过 ks.cfg 实现系统无人值守安装？

3.3.3.3 解决方案

本例镜像文件：Kylin-Server-V10-SP3-General-Release-2303-X86_64.iso

1）挂载镜像并拷贝一份出来

```
# mount -o loop Kylin-Server-V10-SP3-General-Release-2303-X86_64.iso /mnt/
# mkdir /root/iso
# rsync -avz /mnt/ /root/iso
```

注意：拷贝以后'll -a'确认下，一定要把文件拷贝全

```
[root@node1 ~]# cd /root/iso/
[root@node1 iso]# ll -a
总用量 248
dr-xr-xr-x   9 root root    254  3月 24  2023 .
dr-xr-x---+ 29 root root   4096  2月  5 09:53 ..
-r--r--r--   1 root root     45  3月 15  2023 .discinfo
dr-xr-xr-x   3 root root     35  3月 15  2023 EFI
dr-xr-xr-x   3 root root     76  3月 15  2023 images
dr-xr-xr-x   2 root root    256  3月 15  2023 isolinux
-r--r--r--   1 root root    260  3月 31  2023 .kyinfo
-r--r--r--   1 root root     97  3月 18  2023 .kylin-post-actions-nochroot
dr-xr-xr-x   5 root root     79  3月 24  2023 kylin-sm-package
-r--r--r--   1 root root    441  3月 31  2023 LICENSE
dr-xr-xr-x   2 root root    214  3月 15  2023 manual
dr-xr-xr-x   2 root root 167936  3月 24  2023 Packages
-r--r--r--   1 root root     79  3月 24  2023 .productinfo
dr-xr-xr-x   2 root root   4096  3月 24  2023 repodata
-r--r--r--   1 root root   2883  3月 31  2023 TRANS.TBL
-r--r--r--   1 root root    437  3月 15  2023 .treeinfo
[root@node1 iso]#
```

2）使用通用系统按照自己需求安装一个环境，将安装好的环境里的/root/anaconda-ks.cfg 应答文件拷贝到/root/iso 下，重命名为 ks.cfg

```
[root@node1 iso]# mv anaconda-ks.cfg ks.cfg
[root@node1 iso]# ll -a
总用量 252
dr-xr-xr-x   9 root root    268  2月   5 10:05 .
dr-xr-x---+ 29 root root   4096  2月   5 09:53 ..
-r--r--r--   1 root root     45  3月 15  2023 .discinfo
dr-xr-xr-x   3 root root     35  3月 15  2023 EFI
dr-xr-xr-x   3 root root     76  3月 15  2023 images
dr-xr-xr-x   2 root root    256  3月 15  2023 isolinux
-rw-r--r--   1 root root   3292  2月   5 10:05 ks.cfg
-r--r--r--   1 root root    260  3月 31  2023 .kyinfo
-r--r--r--   1 root root     97  3月 18  2023 .kylin-post-actions-nochroot
dr-xr-xr-x   5 root root     79  3月 24  2023 kylin-sm-package
-r--r--r--   1 root root    441  3月 31  2023 LICENSE
dr-xr-xr-x   2 root root    214  3月 15  2023 manual
dr-xr-xr-x   2 root root 167936  3月 24  2023 Packages
-r--r--r--   1 root root     79  3月 24  2023 .productinfo
dr-xr-xr-x   2 root root   4096  3月 24  2023 repodata
-r--r--r--   1 root root   2883  3月 31  2023 TRANS.TBL
-r--r--r--   1 root root    437  3月 15  2023 .treeinfo
[root@node1 iso]#
```

3）定制内容-定制加入包

① 在/root/iso/下创建 alpha 目录，把所需要的 rpm 包或者其他安装文件拷贝进去 alpha 目录（以放入 mysql 的 rpm 安装包为例）

```
[root@node1 alpha]# ll
总用量 460824
-rw-r--r-- 1 root root  19109628  2月   5 10:29 mysql-community-client-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root    292424  2月   5 10:29 mysql-community-common-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root   3056496  2月   5 10:29 mysql-community-devel-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root  46461544  2月   5 10:29 mysql-community-embedded-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root 140569284  2月   5 10:29 mysql-community-embedded-devel-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root   1600224  2月   5 10:29 mysql-community-libs-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root 141031844  2月   5 10:30 mysql-community-server-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root 119654236  2月   5 10:30 mysql-community-test-5.7.29-1.p01.ky10.ky10.x86_64.rpm
-rw-r--r-- 1 root root     51400  2月   5 10:30 perl-JSON-2.97.001-6.ky10.noarch.rpm
[root@node1 alpha]# pwd
/root/iso/alpha
[root@node1 alpha]#
```

②编辑 ks.cfg

在%post 下面加入安装命令：

```
# yum remove mariadb
# rpm -ivh /alpha/*.rpm
```

（此处还可以加入其他定制命令）

```
%post

systemctl disable systemd-networkd-wait-online.service
systemctl disable multipathd.service

### do kylin post action
if [ -e /bin/.kylin-post-actions ];then
  /bin/bash -x /bin/.kylin-post-actions &> /var/log/.kylin-post-actions.log
fi

rpm -ivh /alpha/*.rpm

%end
```

③修改 cfg 或者 isolinux.cfg（BIOS 启动用 isolinux.cfg，UEFI 启动用 grub.cfg，选其一即可）

注意：1、不要更改 hd:LABEL=Kylin-Server-10

2、要是刻盘安装物理机，不要使用 windows 系统刻录 U 盘启动盘，会出现启动盘名称截断，后续无法引导安装（可以使用 dd 命令刻盘，或者 windows 系统刻录光盘启动盘）

grub.cfg：

```
### BEGIN /etc/grub.d/10_linux ###
menuentry ' AUTO Install Kylin Linux Advanced Server V10' --class fedora --class gnu-linux --class gnu --class os {
        linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=Kylin-Server-10 inst.ks=hd:LABEL=Kylin-Server-10:/ks.cfg quiet
        initrdefi /images/pxeboot/initrd.img
}
submenu 'Troubleshooting -->' {
        menuentry 'Install Kylin Linux Advanced Server V10 in basic graphics mode' --class fedora --class gnu-linux --class gnu --class os {
                linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=Kylin-Server-10 nomodeset quiet
                initrdefi /images/pxeboot/initrd.img
        }
        menuentry 'Rescue a Kylin Linux Advanced Server system' --class fedora --class gnu-linux --class gnu --class os {
                linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=Kylin-Server-10 rescue console=tty1 quiet
                initrdefi /images/pxeboot/initrd.img
        }
}
```

isolinux.cfg：

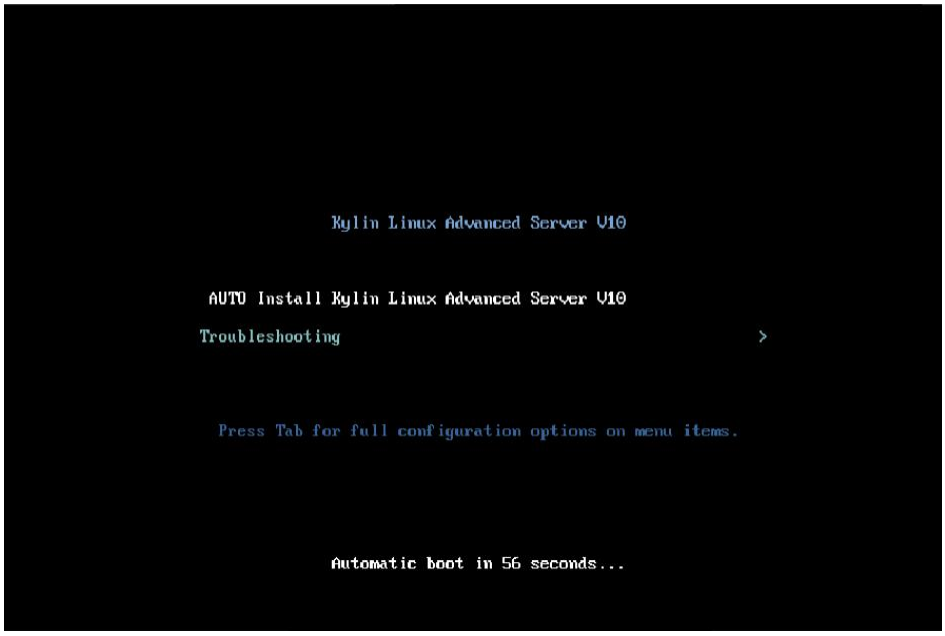
```
# Press [Tab] message
menu color tabmsg 0 #ff3a6496 #00000000 none

# Unselected menu item
menu color unsel 0 #84b8ffff #00000000 none

# Selected hotkey
menu color hotsel 0 #84b8ffff #00000000 none

# Unselected hotkey
menu color hotkey 0 #ffffffff #00000000 none

# Help text
menu color help 0 #ffffffff #00000000 none

# A scrollbar of some type? Not sure.
menu color scrollbar 0 #ffffffff #ff355594 none

# Timeout msg
menu color timeout 0 #ffffffff #00000000 none
menu color timeout_msg 0 #ffffffff #00000000 none

# Command prompt text
menu color cmdmark 0 #84b8ffff #00000000 none
menu color cmdline 0 #ffffffff #00000000 none

# Do not display the actual menu unless the user presses a key. All that is displayed is a timeout message.

menu tabmsg Press Tab for full configuration options on menu items.

menu separator # insert an empty line
menu separator # insert an empty line

label linux
  menu default
  menu label ^ AUTO Install Kylin Linux Advanced Server V10
  kernel vmlinuz
  append initrd=initrd.img inst.stage2=hd:LABEL=Kylin-Server-10 inst.ks=hd:LABEL=Kylin-Server-10:/ks.cfg quiet

menu separator # insert an empty line

# utilities submenu
menu begin ^Troubleshooting
  menu title Troubleshooting
```

4）定制化完成之后打包镜像

```
# mkisofs -J -T -r -V "Kylin-Server-10" -b isolinux/isolinux.bin -no-emul-boot -boot-load-size 4 -boot-info-table -eltorito-alt-boot -e images/efiboot.img -no-emul-boot -o /data/Kylin-Server-V10-SP3-General-Release-2303-X86_64-new.iso ./
```

注意：新镜像命名尽量在原名基础上加自定义的标识

5）安装验证



3.4 服务器系统设置问题

3.4.1 locale 显示语言环境为 en_US.UTF-8

解决方法：安装系统的时候没有选择中文语言安装，导致环境语言不是中文。

1) 查看已经安装的语言环境

```
# locale -a
```

2) 然后，更改/etc/locale.conf 文件，将其中语言环境改为中文：

```
# vim /etc/locale.conf
LANG="zh_CN.UTF-8"
```

3) 最后重启电脑。

3.4.2 如何设置 core 文件

解决方法：修改/etc/systemd/coredump.conf 文件：

```
# vim /etc/systemd/coredump.conf

ProcessSizeMax=8G
ExternalSizeMax=8G
JournalSizeMax=8G
```

然后，重新加载配置

```
# systemctl daemon-reload
```

3.4.3 系统无法识别 exfat 格式的硬盘

解决方法：从系统源里下载相应的驱动：

```
# yum install exfat-utils fuse-exfat
```

3.4.4 系统中 netstat 命令不可用

解决方法：安装 net-tools 软件包

```
# yum install net-tools
```

3.4.5 如何修改网卡名称

3.4.5.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可做参考。

3.4.5.2 问题描述

如何修改原网卡为指定名称。

3.4.5.3 问题分析

这是一个修改系统配置的问题，与 centos 系统修改网卡名称不一样，银河麒麟高级服务器系统不仅需要修改名称，还需要修改规则，即需要修改其对

应网卡的 MAC 地址，然后重启系统才能生效。

3.4.5.4 解决方案

1) 修改网卡名称

a. 查看需要修改的网卡名称（以 enp1s0 为例）：

```
# ifconfig
```

b. 在/etc/sysconfig/network-scripts/下找到 ifcfg-enp1s0 文件

```
# cd /etc/sysconfig/network-scripts/
# ls
```

c. 修改 ifcfg-enp1s0 文件名为 ifcfg-eth0

```
# mv ifcfg-enp1s0 ifcfg-eth0
```

d. 打开修改后的文件 ifcfg-eth0 将内容中的 NAME 和 DEVICE 后面的值为改为 eth0

```
# vim ifcfg-eth0
```

如下：

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0
UUID=ec12c1ac-ffc9-3f80-84b6-428962544def
ONBOOT=yes
AUTOCONNECT_PRIORITY=-999
DEVICE=eth0
IPADDR=[redacted]
PREFIX=24
GATEWAY=[redacted]
DNS1=[redacted]
DNS2=[redacted]
IPV6_PRIVACY=no
```

2) 修改规则

a. 在/etc/udev/rules.d 下找到规则文件 75-network.rules（如果没有该文件就新建一个）

```
# cd /etc/udev/rules.d
# ls
```

b. 打开规则文件 75-network.rules：

```
# vim 75-network.rules
```

将其内容修改如下：

```
SUBSYSTEM==”net”,ACTION==”add”,ATTR(address)==”52:54:00:0e:58:7c”,
NAME=”eth0”,
```

其中 ATTR 后面的值为 enp1s0 网卡对应的 MAC 地址，NAME 后面的值为要修改的网卡名称。

3) 生效规则

a. 重启系统使规则生效。

```
# reboot
```

b. 查看修改后的网卡名称

```
# ifconfig
```

```
[root@localhost rules.d]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet [redacted]  netmask [redacted]  broadcast [redacted]
        inet6 fe80::8ea5:ee8d:5d5e:10c9  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:0e:58:7c  txqueuelen 1000  (Ethernet)
        RX packets 348528  bytes 55780759 (53.1 MiB)
        RX errors 0  dropped 33958  overruns 0  frame 0
        TX packets 1368  bytes 116132 (113.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.4.6 如何安装图形化界面

3.4.6.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可做参考。

3.4.6.2 问题描述

利用最小化安装系统后，无图形化界面，只有命令行，而后续操作需要用到图形化，则需要再安装图形化界面。

3.4.6.3 问题分析

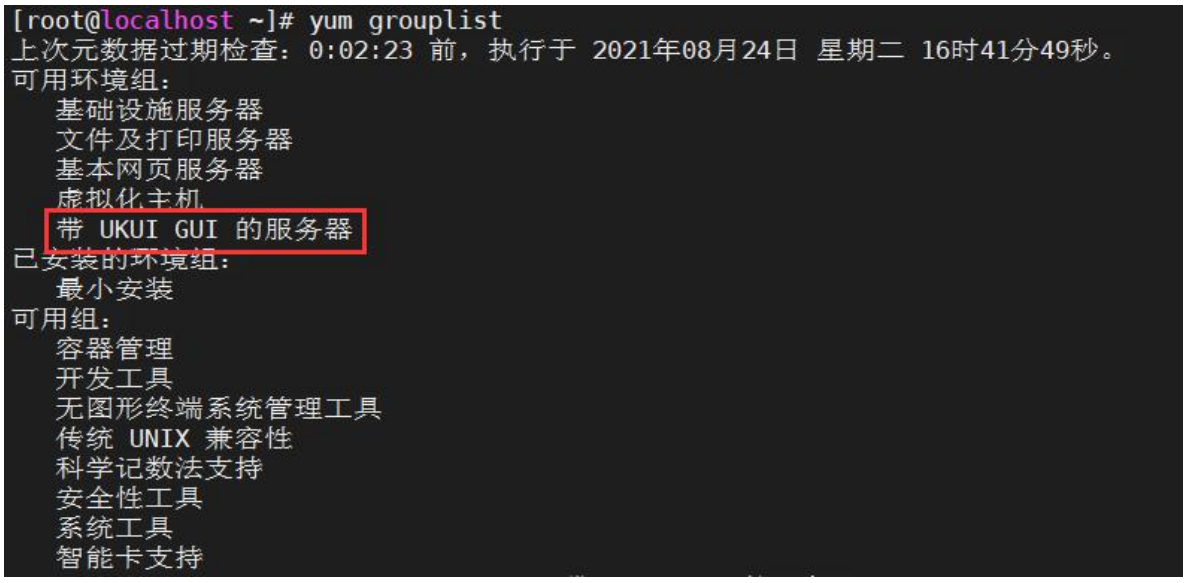
安装系统的时候，为了方便，有时候只需要最小化安装即可，但是后续如果要用到图形化界面，只需要安装图形化界面即可，不需要重新安装系统。

3.4.6.4 解决方案

1) 配置网络，确保 yum 源可用，有外网环境可直接使用 yum 源，无外网环境则可挂载镜像作为本地源。

2) 查询安装软件

```
# yum grouplist
```



3) 安装图形化界面

```
# yum groupinstall "带 UKUI GUI 的服务器"
```

注：这里注意双引号和空格

4) 不重启系统的情况下启动图形化界面

```
# startx
```

5) 如果想要重启系统后仍然可以启动图形化界面，则可查看开机启动项，并设置图形化作为默认启动项

```
# systemctl get-default
# systemctl set-default graphical.target
# reboot
```

```
[root@localhost ~]# systemctl get-default
multi-user.target
[root@localhost ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/graphical.target.
[root@localhost ~]# systemctl get-default
graphical.target
[root@localhost ~]#
```

3.4.7 如何利用 yum 降级安装软件包

3.4.7.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可做参考。

3.4.7.2 问题描述

有时候需要安装低版本的软件包，但是如果系统中已经安装了较新版本的该软件包，然后再使用 yum install package-name 的话，则不会再次安装低版本的，所以就得将该软件包降级。

3.4.7.3 问题分析

先查看源中都有哪些版本的软件包，然后直接利用“yum install 具体版本的软件包”进行安装。

3.4.7.4 解决方案

以 yum 软件包为例。

前提：yum 源中有低版本的软件包。

1) 使用 yum 直接安装（因为已经安装的 yum 版本比较新，所以不会更新）

```
# yum install yum
```

```
[root@host-192-168-9-3 ~]# yum install yum
上次元数据过期检查：0:00:31 前，执行于 2021年11月19日 星期五 16时04分19秒。
软件包 yum-4.2.23-3.p02.ky10.noarch 已安装。
依赖关系解决。
无需任何处理。
完毕！
```

2) 查看源中 yum 包含的版本

```
# yum list --showduplicates yum
```

```
[root@host-192-168-9-3 ~]# yum list --showduplicates yum
上次元数据过期检查：0:02:07 前，执行于 2021年11月19日 星期五 16时04分19秒。
已安装的软件包
yum.noarch                                                    4.2.23-3.p02.ky10
可安装的软件包
yum.noarch                                                    4.2.15-7.ky10
```

3) 选择需要的 yum 包的版本并使用 yum 安装

```
# yum install yum-4.2.15-7.ky10
```

```
[root@host-192-168-9-3 ~]# yum install yum-4.2.15-7.ky10
上次元数据过期检查：0:02:33 前，执行于 2021年11月19日 星期五 16时04分19秒。
依赖关系解决。
================================================================================
 Package                 Architecture        Version                 Repository                 Size
================================================================================
降级：
 dnf                     noarch              4.2.15-7.ky10           ks10-adv-os                29 k
 dnf-automatic           noarch              4.2.15-7.ky10           ks10-adv-os                25 k
 libdnf                  x86_64              0.37.2-2.ky10           ks10-adv-os               570 k
 python3-dnf             noarch              4.2.15-7.ky10           ks10-adv-os               400 k
 python3-hawkey          x86_64              0.37.2-2.ky10           ks10-adv-os                74 k
 python3-libdnf          x86_64              0.37.2-2.ky10           ks10-adv-os               606 k
 yum                     noarch              4.2.15-7.ky10           ks10-adv-os               8.2 k

事务概要
================================================================================
降级  7 软件包

总下载：1.7 M
确定吗？[y/N]：  y
下载软件包：
(1/7): dnf-automatic-4.2.15-7.ky10.noarch.rpm                  228 kB/s |  25 kB     00:00
(2/7): dnf-4.2.15-7.ky10.noarch.rpm                            235 kB/s |  29 kB     00:00
(3/7): python3-hawkey-0.37.2-2.ky10.x86_64.rpm                 230 kB/s |  74 kB     00:00
(4/7): libdnf-0.37.2-2.ky10.x86_64.rpm                         271 kB/s | 570 kB     00:02
(5/7): yum-4.2.15-7.ky10.noarch.rpm                            223 kB/s | 8.2 kB     00:00
(6/7): python3-dnf-4.2.15-7.ky10.noarch.rpm                    161 kB/s | 400 kB     00:02
(7/7): python3-libdnf-0.37.2-2.ky10.x86_64.rpm                 203 kB/s | 606 kB     00:02
--------------------------------------------------------------------------------
总计                                                           496 kB/s
运行事务检查
事务检查成功。
运行事务测试
事务测试成功。
运行事务
  准备中  :                                                                1/1
  降级    : libdnf-0.37.2-2.ky10.x86_64                                   1/14
```

4) 检查是否降级成功

```
# yum list installed | grep yum
```

```
[root@host-192-168-9-3 ~]# yum list installed | grep yum
yum.noarch                                4.2.15-7.ky10                @ks10-adv-os
yum-langpacks-help.noarch                 0.4.5-10.oe1                 @anaconda
yum-metadata-parser-help.x86_64           1.1.4-24.ky10                @anaconda
```

3.4.8 如何将 txt 文件转换为 pdf

3.4.8.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.4.8.2 问题描述

银河麒麟高级服务器操作系统下只能创建.txt 文件格式，而有些文件需

要.pdf 格式，所以就需要将.txt 格式转换为.pdf 格式。

3.4.8.3 问题分析

先将.txt 格式的文件转换为.ps 格式，再将.ps 文件转换为.pdf 格式。其中，.ps 是 PostScript 的缩写，PostScript 是一种页面描述语言，而 pdf 文件就是 PostScript 发展而来的，所以.ps 可以转换为.pdf。enscript 命令行工具可以将.txt 格式的文本文件转换成.ps 格式，ghostscript 是阅读.ps 文件的语言解释器。

3.4.8.4 解决方案

1） 安装软件

```
# yum install enscript ghostscript
```

2） 准备 txt 文件

3） 将 txt 文件转换为 ps 格式

```
# enscript -p filename.ps filename.txt
```

4） 将 ps 文件转换为 pdf 格式

```
# ps2pdf filename.ps filename.pdf
```

```
[root@localhost pdf]# enscript -p output.ps test.txt
[ 1 page * 1 copy ] left in output.ps
3 lines were wrapped
30 non-printable characters
[root@localhost pdf]# vim output.ps
[root@localhost pdf]# ps2pdf output.ps output.pdf
[root@localhost pdf]# ls
output.pdf  output.ps  test.txt
[root@localhost pdf]#
```

3.4.9 如何进入单用户模式

3.4.9.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.4.9.2 问题描述

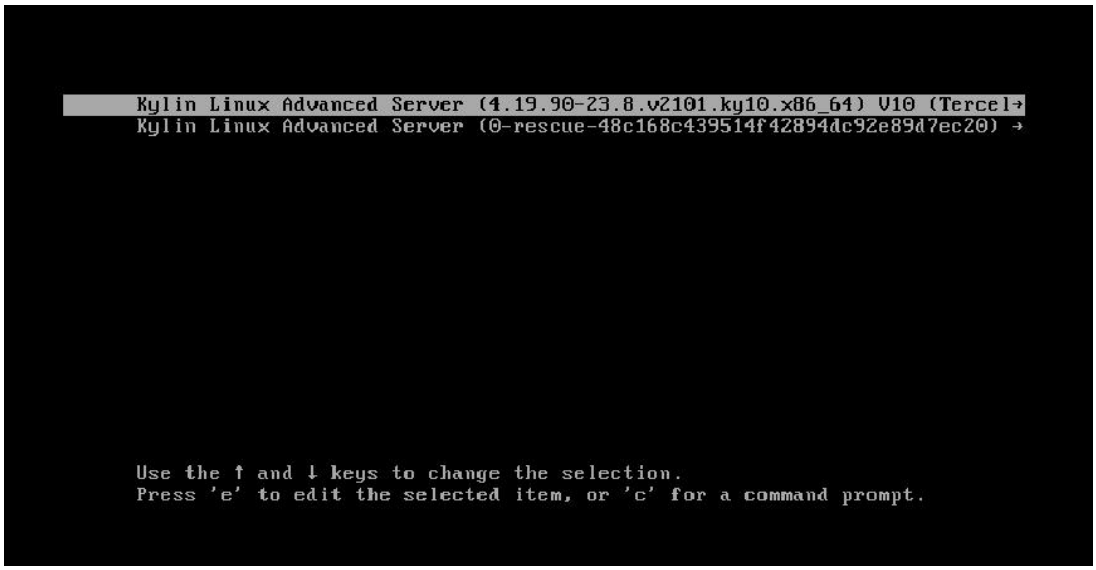
有时候需要对系统进行修改或是遗忘了 root 密码，则可以进入单用户模式进行修改。

3.4.9.3 问题分析

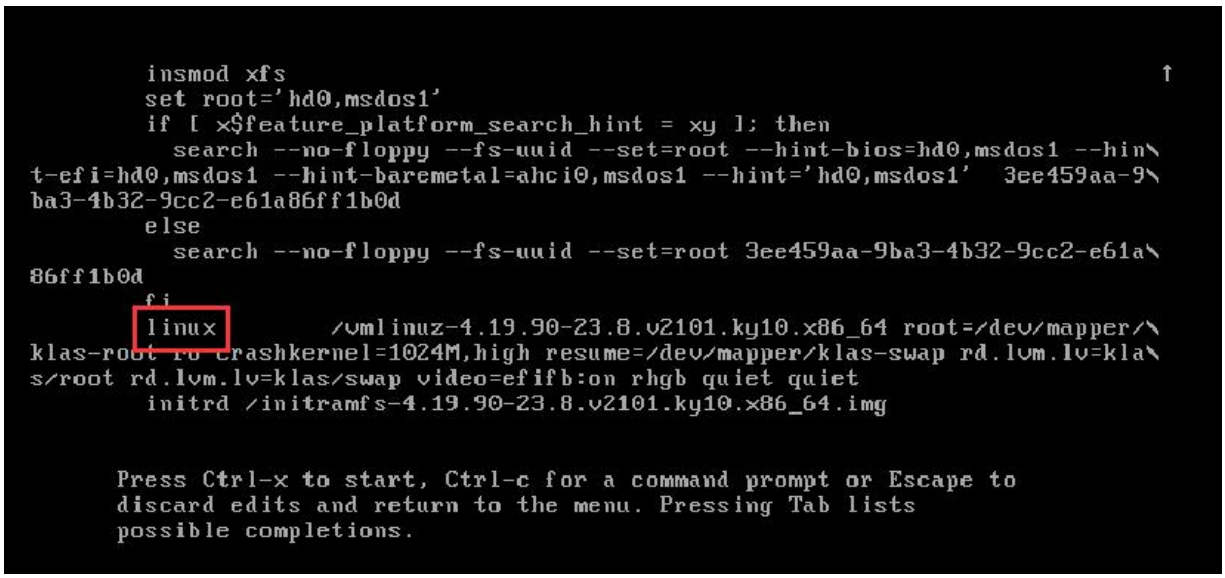
单用户即系统并没有完全运行起来，只是部分程序运行，这时也不能进行远程登录到 Linux 系统，单用户是以 root 身份进入，这时的 root 用户对系统有完全操作权限，可以修复系统的同时，也能随时的对系统进行破坏。

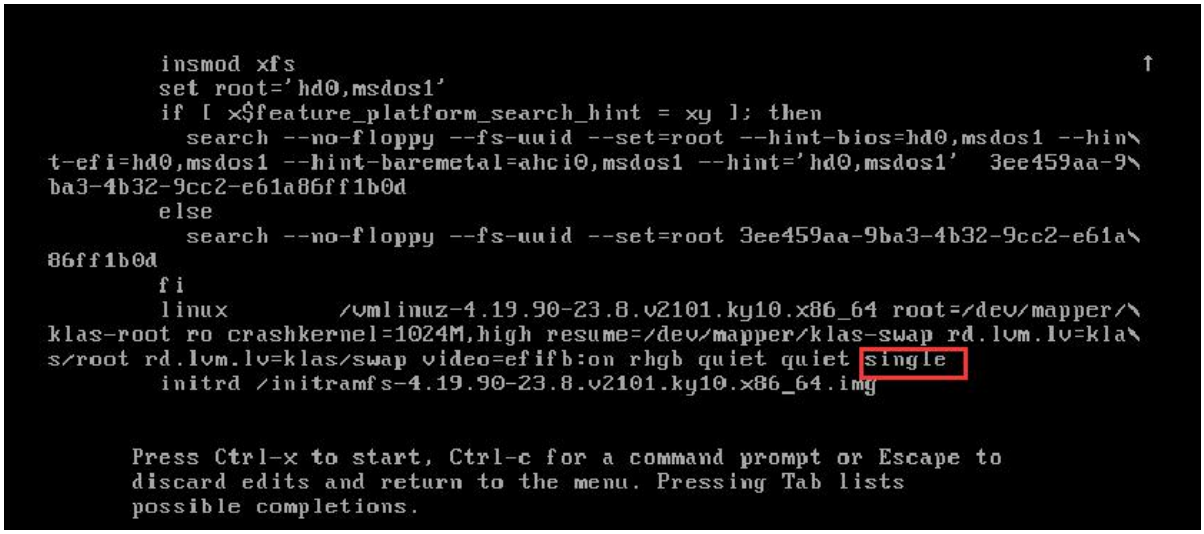
3.4.9.4 解决方案

1) 在此界面按“e”进入编辑模式

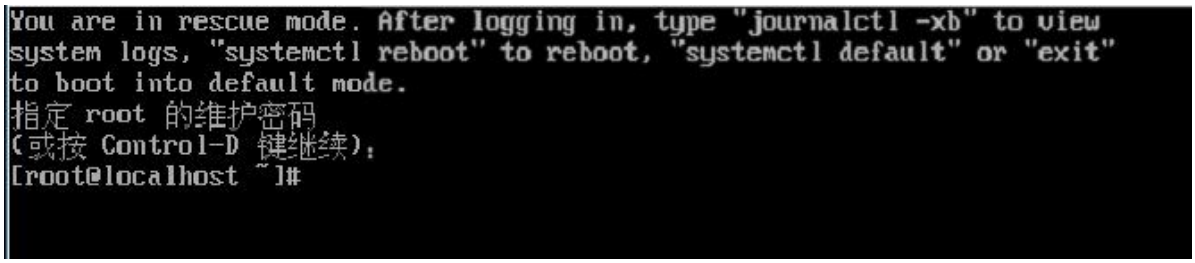


2) 找到“linux”行，在末尾输入”single“，按 Ctrl+x 进入单用户模式





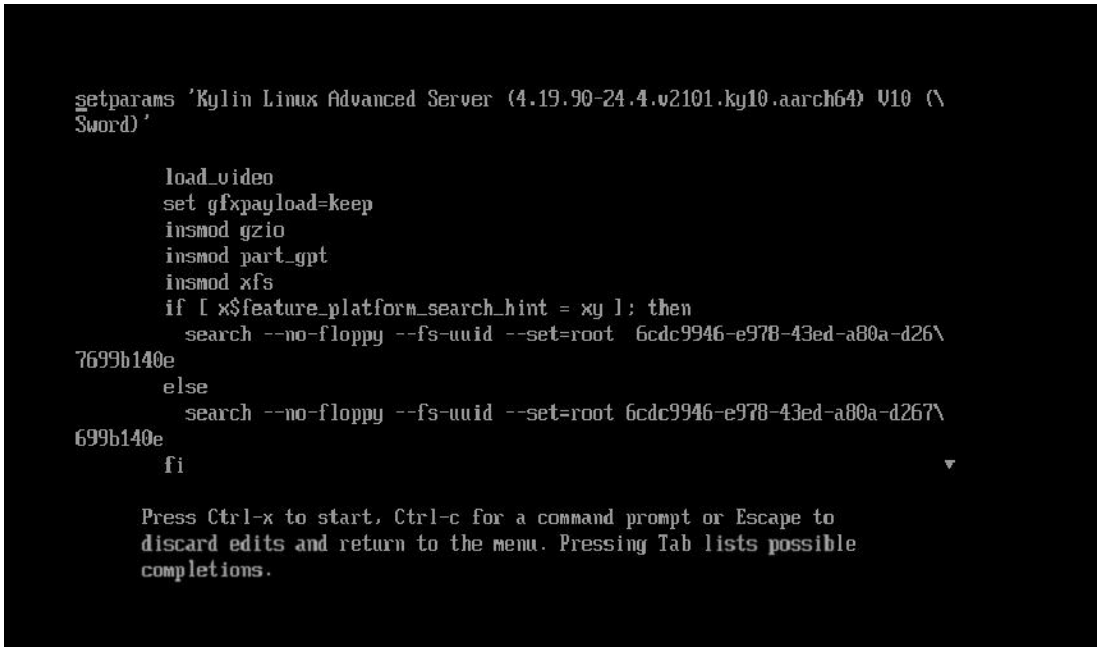
3) 输入机器密码进入



4) 有的系统版本在第一步按”e“后，需要输入指定的用户名和密码才可以进入编辑模式，其他步骤与 2），3）步一样

用户名：root

密码：Kylin123123



5) 修改完成后，输入 reboot 重启系统，正常进入系统即可。

3.4.10 如何编译内核模块

3.4.10.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.4.10.2 问题描述

内核驱动模块.ko 文件大多为内核自带的,但是由于有时候需要根据具体的需求自己编写。

3.4.10.3 问题分析

系统缺少需要的内核模块，所以自己需要编写和编译等。

3.4.10.4 解决方案

1) 内核驱动模块编译环境搭建

a. 根据系统内核版本“uname –r”安装对应的开发包

```
[root@localhost ~]# uname -r
4.19.90-24.4.v2101.ky10.x86_64
```

或者使用以下命令进行内核安装或升级（此处不进行升级）：

```
# yum install kernel-devel
```

```
[root@localhost ~]# yum install kernel-devel
上次元数据过期检查：1:05:40 前，执行于 2022年03月14日 星期一 09时59分34秒。
软件包 kernel-devel-4.19.90-24.4.v2101.ky10.x86_64 已安装。
依赖关系解决。
================================================================================
 Package        Arch     Version                        Repository          Size
================================================================================
安装：
 kernel-devel   x86_64   4.19.90-25.11.v2101.ky10       ks10-adv-updates    14 M
移除依赖的软件包：
 kernel-devel   x86_64   4.19.90-24.4.v2101.ky10        @anaconda           46 M

事务概要
================================================================================
安装  1 软件包
移除  1 软件包

总下载：14 M
确定吗？[y/N]:  n
操作中止。
```

安装完成后会在/usr/src/kernels 下产生对应内核名称的目录

```
[root@localhost ~]# cd /usr/src/kernels/
[root@localhost kernels]# ls
4.19.90-24.4.v2101.ky10.x86_64
```

b. 然后会在/usr/lib/modules 下对应的内核版本中产生一个 build 的软连接，指向上面的地址。

```
[root@localhost kernels]# cd /usr/lib/modules
[root@localhost modules]# ls
4.19.90-24.4.v2101.ky10.x86_64
[root@localhost modules]# cd 4.19.90-24.4.v2101.ky10.x86_64/
[root@localhost 4.19.90-24.4.v2101.ky10.x86_64]# ls
bls.conf           modules.builtin           modules.modesetting  symvers.gz
build              modules.builtin.alias.bin modules.networking   System.map
config             modules.builtin.bin       modules.order        updates
kernel             modules.dep               modules.softdep      vdso
modules.alias      modules.dep.bin           modules.symbols      vmlinuz
modules.alias.bin  modules.devname           modules.symbols.bin  weak-updates
modules.block      modules.drm               source
```

2) 编译好.c 文件

例如在/opt 目录下编写一个 hello.c 的文件

```
/*加载头文件*/
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("test");
MODULE_DESCRIPTION("hello");
MODULE_VERSION("1.00");
/*编写初始化函数*/
static int hello_init(void)
{
printk(KERN_ALERT "hello_init\n");
}
/*编写退出函数*/
static void hello_exit(void)
{
printk(KERN_ALERT "hello_exit\n");
}
/*调用函数*/
module_init(hello_init);
module_exit(hello_exit);
```

```c
/*加载头文件*/
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("test");
MODULE_DESCRIPTION("hello");
MODULE_VERSION("1.00");
/*编写初始化函数*/
static int hello_init(void)
{
        printk(KERN_ALERT "hello_init\n");
}
/*编写退出函数*/
static void hello_exit(void)
{
        printk(KERN_ALERT "hello_exit\n");
}
/*调用函数*/
module_init(hello_init);
module_exit(hello_exit);
```

其中，函数没有任何意义，只是显示出来加载过程。

3) 编写 Makefile

在/opt 目录下编写一个 Makefile 文件

```
obj-m := hello.o
default:
        make -C /lib/modules/`uname -r`/build SUBDIRS=`pwd` modules
```

4) 编译

a. 在/opt 目录下执行以下命令

```
# make
```

b. make 完成后会在/opt 目录下生成对应的.ko 和.o 文件等

```
[root@localhost opt]# ls
firefox  hello.ko     hello.mod.o  kernel    modules.order
hello.c  hello.mod.c  hello.o      Makefile  Module.symvers
```

5) 加载模块并查看

```
# insmod /opt/hello.ko
# lsmod | grep hello
```

```
[root@localhost opt]# insmod /opt/hello.ko
[root@localhost opt]# lsmod | grep hello
hello                  16384  0
```

```
# dmesg
```

```
[ 5147.328146] hello init
[ 5147.328152] do_init_module: 'hello'->init suspiciously returned 10, it should follow 0/-E convention
               do_init_module: loading module anyway...
[ 5147.328155] CPU: 1 PID: 7137 Comm: insmod Kdump: loaded Tainted: G           OE     4.19.90-25.8.v2101.ky10.x86_64
 #1
[ 5147.328156] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS rel-1.12.1-0-ga5cab58e9a3f-prebuilt.qemu.o
rg 04/01/2014
```

6) 卸载模块

```
# rmmod hello.ko
# lsmod | grep hello
```

```
[root@localhost opt]# lsmod | grep hello
[root@localhost opt]#
```

```
# dmesg
```

```
[ 5147.328236] R10: 0000000000000003 R11: 0000000000000246 R12: 0000000000000000
[ 5147.328237] R13: 0000558d7130c7d0 R14: 0000000000000000 R15: 0000000000000000
[ 5341.185019] hello exit
```

卸载模块完成。

3.4.11 如何开启/关闭开机自启动服务

3.4.11.1 系统版本

适用系统：银河麒麟高级服务器 V10、V10(SP1)、V10(SP2)

适用架构：X86、ARM、MIPS、Loongarch

其他版本和架构可作参考。

3.4.11.2 解决方案

如果是原生 systemd 服务，则可以使用

```
# systemctl list-unit-files
```

查看

其中 enable 表示已开启自启动，disabled 表示未开启自启动，static 表示未进行设置，generated 表示衍生

```
UNIT FILE                                  STATE
proc-sys-fs-binfmt_misc.automount          static
-.mount                                    generated
backup.mount                               generated
boot-efi.mount                             generated
boot.mount                                 generated
dev-hugepages.mount                        static
dev-mqueue.mount                           static
proc-fs-nfsd.mount                         static
proc-sys-fs-binfmt_misc.mount              static
sys-fs-fuse-connections.mount              static
sys-kernel-config.mount                    static
sys-kernel-debug.mount                     static
tmp.mount                                  disabled
var-lib-nfs-rpc_pipefs.mount               static
cups.path                                  enabled
ostree-finalize-staged.path                disabled
systemd-ask-password-console.path          static
systemd-ask-password-plymouth.path         static
systemd-ask-password-wall.path             static
session-2.scope                            transient
session-4.scope                            transient
accounts-daemon.service                    enabled
anaconda-direct.service                    static
anaconda-nm-config.service                 static
anaconda-noshell.service                   static
anaconda-pre.service                       static
anaconda-shell@.service                    static
anaconda-sshd.service                      static
anaconda-tmux@.service                     static
anaconda.service                           static
arp-ethers.service                         disabled
atd.service                                enabled
auditd.service                             enabled
auth-rpcgss-module.service                 static
autovt@.service                            enabled
blivet.service                             static
blk-availability.service                   disabled
bluetooth-mesh.service                     disabled
bluetooth.service                          enabled
bmc-snmp-proxy.service                     disabled
btattach-bcm@.service                      static
canberra-system-bootup.service             disabled
canberra-system-shutdown-reboot.service    disabled
canberra-system-shutdown.service           disabled
cgconfig.service                           disabled
cgdcbxd.service                            disabled
chrony-dnssrv@.service                     static
```

开启开机自启动服务

执行

```
# systemctl enable NAME
```

```
[root@2303-arm ~]# systemctl enable firewalld.service
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service → /usr/lib/systemd/system/firewalld.service.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service → /usr/lib/systemd/system/firewalld.service.
[root@2303-arm ~]#
```

执行

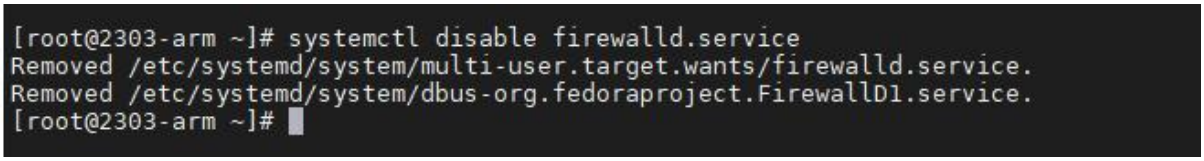
```
# systemctl list-unit-files
```

查看是否生效

关闭开机自启动服务

执行

```
# systemctl disable NAME
```



执行

```
# systemctl list-unit-files
```

查看是否生效

3.4.12 手动挂载磁盘

3.4.12.1 系统版本

适用系统：银河麒麟高级服务器 V10、V10(SP1)、V10(SP2)

适用架构：X86、ARM、MIPS

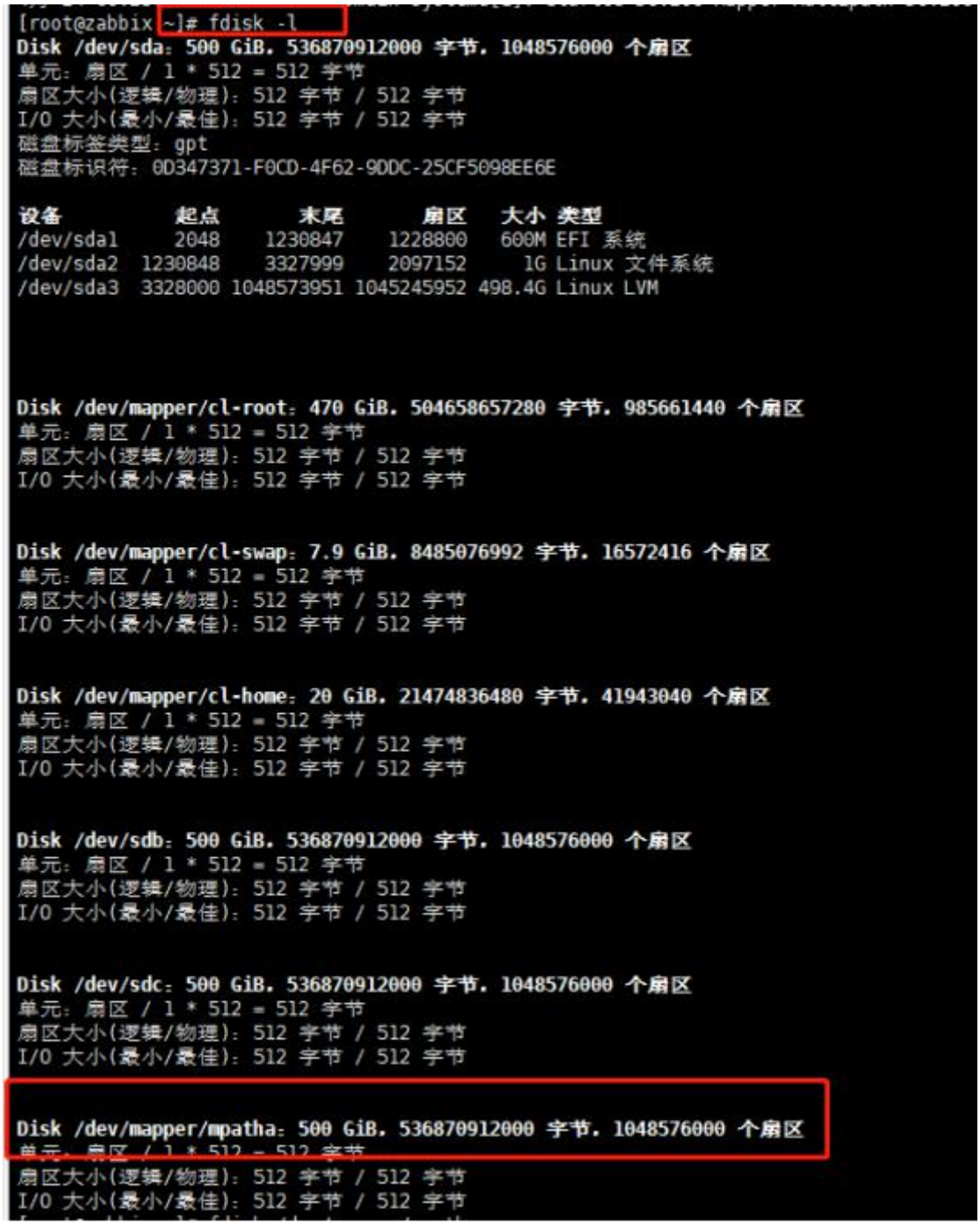
其他版本和架构可作参考。

3.4.12.2 解决方案

查看磁盘是否已存在

执行

```
# fdisk -l
```



初始化磁盘

执行 fdisk 查询到的磁盘（如上图的/dev/mapper/mpatha）

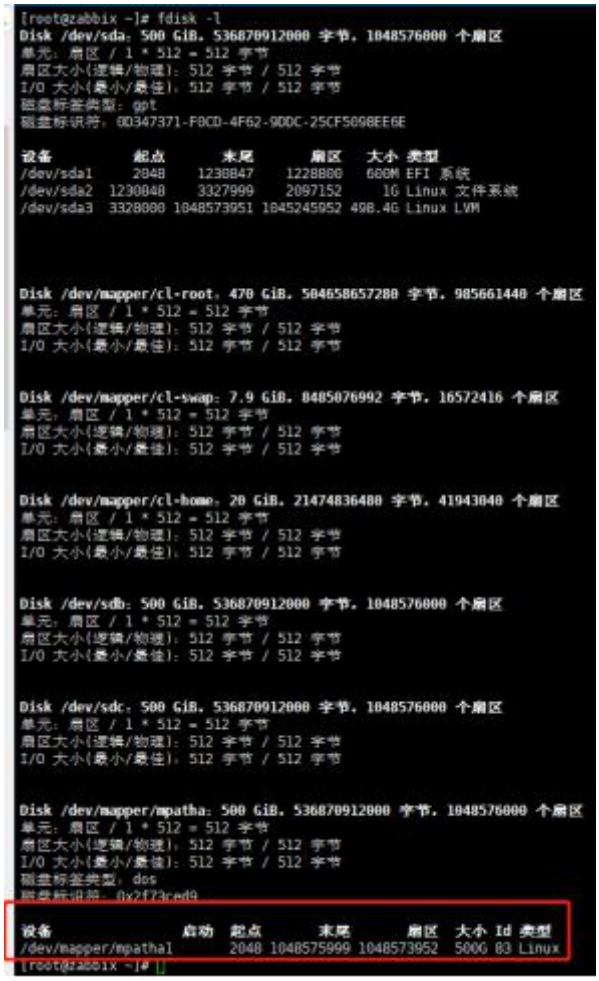
```
# fdisk /dev/mapper/mpatha
```

然后分别输入：n

p

1

W



然后将磁盘格式化成 ext4 格式，执行

```
# mkfs.ext4 /dev/mapper/mpatha
```

创建挂载点

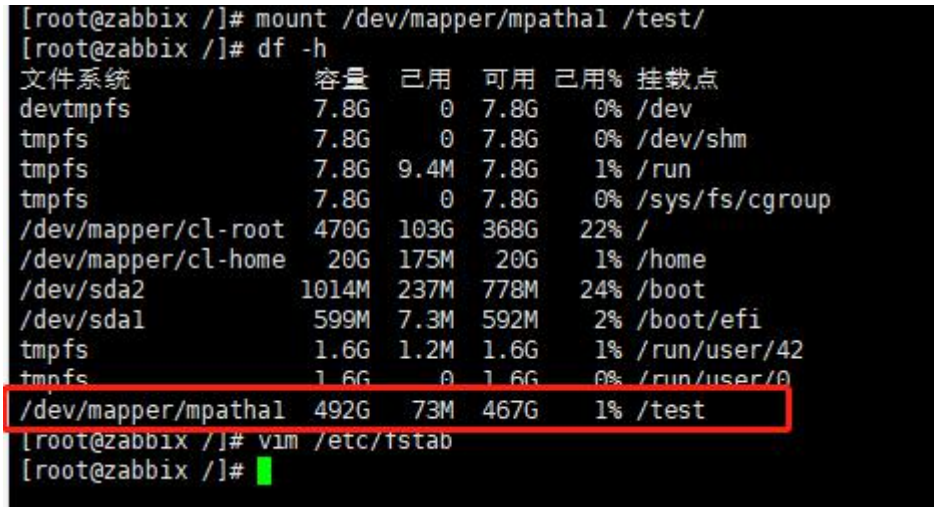
创建要挂载到的目录，比如/test，执行 mkdir /test

然后挂载磁盘目录，执行

```
# mount /dev/mapper/mpatha1 /test/
```

然后查看挂载是否成功，执行

```
# df -h
```



手动挂载命令（重启后会失效）：

```
#mount /dev/mapper/maptha1 /test
```

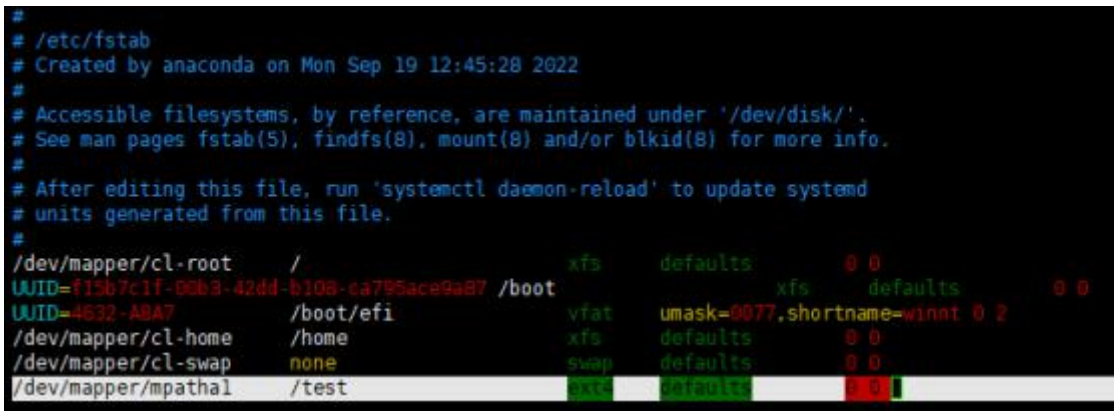
若永久生效可写入 fstab 自启动

执行

```
# vim /etc/fstab
```

然后在该文本中添加

```
/dev/mapper/mpatha1  /test        ext4    defaults     0 0
```



3.4.13 根目录扩容

3.4.13.1 系统版本

适用系统：银河麒麟高级服务器 V10、V10(SP1)、V10(SP2)、V10（SP3）

适用架构：全架构

其他版本和架构可作参考。

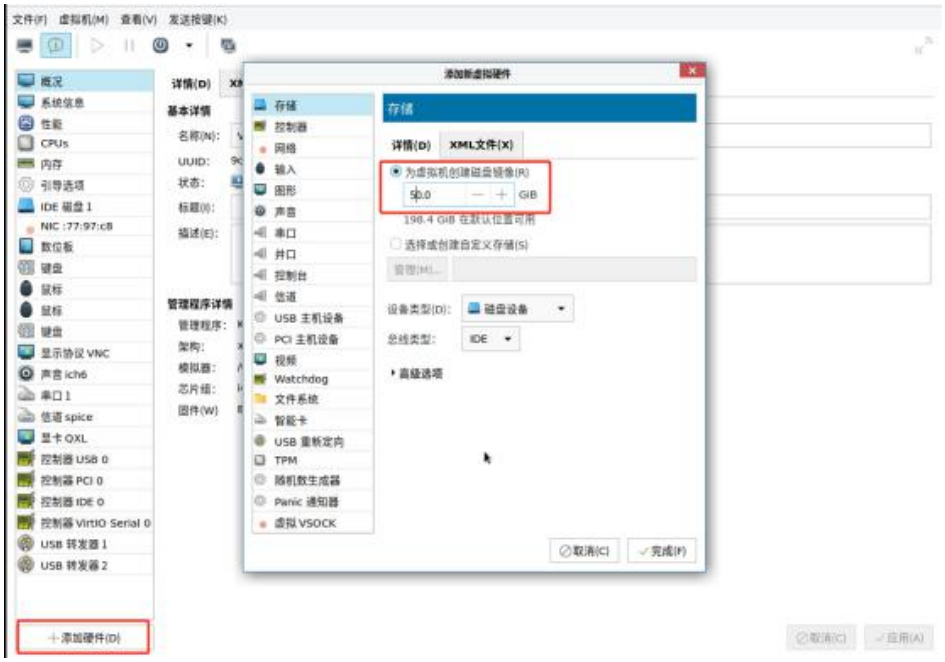
3.4.13.2 解决方案

原始根目录大小，如下图：

```
[root@client ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    2M  0 part
├─sda2            8:2    0    1G  0 part /boot
└─sda3            8:3    0   99G  0 part
  └─klas-root 253:0    0   99G  0 lvm  /
[root@client ~]#
```

若将根目录扩大 50GB，步骤如下：

1）以安装在 virt-manager 里的虚拟机为例，添加一块 50GB 大小的磁盘



2）重启进入系统，可以看到已有这块盘，如下图

```
[root@client ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    2M  0 part
├─sda2            8:2    0    1G  0 part /boot
└─sda3            8:3    0   99G  0 part
  └─klas-root 253:0    0   99G  0 lvm  /
sdb               8:16   0   50G  0 disk
[root@client ~]#
```

3）对 sdb 进行分区

```
# fdisk /dev/sdb
```

```
[root@client ~]# fdisk /dev/sdb

欢迎使用 fdisk (util-linux 2.35.2)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。

设备不包含可识别的分区表。
创建了一个磁盘标识符为 0x9e73a652 的新 DOS 磁盘标签。

命令(输入 m 获取帮助)：p
Disk /dev/sdb：50 GiB，53687091200 字节，104857600 个扇区
磁盘型号：QEMU HARDDISK
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x9e73a652

命令(输入 m 获取帮助)：n   arm架构下不会显示下面的分区类型，直接回车就行
分区类型
   p   主分区 (0 primary, 0 extended, 4 free)
   e   扩展分区 (逻辑分区容器)
选择 (默认 p)：p
分区号 (1-4，默认  1)：回车
第一个扇区 (2048-104857599，默认 2048)：回车
最后一个扇区，+/-sectors 或 +size{K,M,G,T,P} (2048-104857599，默认 104857599)：回车

创建了一个新分区 1，类型为“Linux”，大小为 50 GiB。

命令(输入 m 获取帮助)：t 更改分区类型
已选择分区 1
Hex code or alias (type L to list all)：8e 需要改为LVM，选择8e
已将分区“Linux”的类型更改为“Linux LVM”。

命令(输入 m 获取帮助)：p 打印分区信息
Disk /dev/sdb：50 GiB，53687091200 字节，104857600 个扇区
磁盘型号：QEMU HARDDISK
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x9e73a652

设备       启动  起点      末尾      扇区 大小 Id 类型
/dev/sdb1        2048 104857599 104855552  50G 8e Linux LVM

命令(输入 m 获取帮助)：w 保存上述配置
分区表已调整。
将调用 ioctl() 来重新读分区表。
正在同步磁盘。
```

此时再次查看 lsblk，已经分好了:

```
[root@client ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    2M  0 part
├─sda2            8:2    0    1G  0 part /boot
└─sda3            8:3    0   99G  0 part
  └─klas-root 253:0    0   99G  0 lvm  /
sdb               8:16   0   50G  0 disk
└─sdb1            8:17   0   50G  0 part
[root@client ~]#
```

4）创建卷信息

```
# pvcreate /dev/sdb1
```

```
[root@client ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
[root@client ~]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda3
  VG Name               klas
  PV Size               <99.00 GiB / not usable 0
  Allocatable           yes (but full)
  PE Size               4.00 MiB
  Total PE              25343
  Free PE               0
  Allocated PE          25343
  PV UUID               1bAtpf-2MHL-I67q-i3HP-XXd4-90kW-K4lgTE

  "/dev/sdb1" is a new physical volume of "<50.00 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               <50.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               Xy2avX-Znde-4yjD-9s2u-Z1ga-Ixy7-FzhzqA
```

5）使用 vgdisplay 可以看到 vg 中的 VG Name 是 klas

```
[root@client ~]# vgdisplay
  --- Volume group ---
  VG Name               klas
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
```

6）将上面创建的 pv 加入到 vg 中

```
# vgextend klas /dev/sdb1
```

```
[root@client ~]# vgextend klas /dev/sdb1
  Volume group "klas" successfully extended
[root@client ~]#
```

再次使用 vgdisplay 查看 vg 信息，可以看到 Cur PV 和 Act PV 的值变成了 2，添加已完成：

```
[root@client ~]# vgdisplay
  --- Volume group ---
  VG Name               klas
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  3
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               148.99 GiB
  PE Size               4.00 MiB
  Total PE              38142
  Alloc PE / Size       25343 / <99.00 GiB
  Free  PE / Size       12799 / <50.00 GiB
  VG UUID               piuywx-aCXy-ZTpx-bxLm-oyeq-c6ty-M0yEFa

[root@client ~]#
```

7）使用 df -Th 查看一下根目录的信息

```
[root@client ~]# df -Th
文件系统              类型      容量  已用  可用 已用% 挂载点
devtmpfs              devtmpfs  4.3G     0  4.3G    0% /dev
tmpfs                 tmpfs     4.3G   16K  4.3G    1% /dev/shm
tmpfs                 tmpfs     4.3G  9.3M  4.3G    1% /run
tmpfs                 tmpfs     4.3G     0  4.3G    0% /sys/fs/cgroup
/dev/mapper/klas-root xfs        99G   11G   89G   11% /
tmpfs                 tmpfs     4.3G  4.0K  4.3G    1% /tmp
/dev/sda2             xfs      1014M  185M  830M   19% /boot
tmpfs                 tmpfs     871M   36K  871M    1% /run/user/0
[root@client ~]#
```

可以确定根目录的挂载路径为/dev/mapper/klas-root

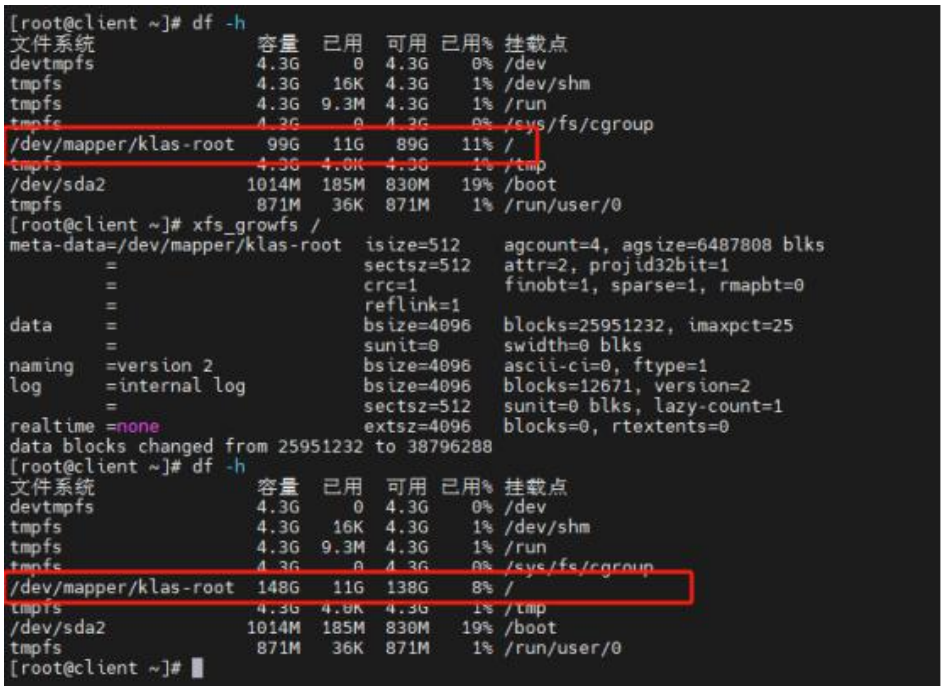
使用 lvextend 命令对根目录的挂载目录进行扩容

```
# lvextend -L +49G /dev/mapper/klas-root
```

```
[root@client ~]# lvextend -L +49G /dev/mapper/klas-root
  Size of logical volume klas/root changed from <99.00 GiB (25343 extents) to <148.00 GiB (37887 extents).
  Logical volume klas/root successfully resized.
[root@client ~]#
```

8）更新根目录空间，扩容完成

```
# xfs_growfs /
```



3.5 软件安装问题

3.5.1 如何安装 Mysql 数据库

3.5.1.1 系统版本

适用系统：V10(SP1)

适用架构：X86、AARCH、LOONGARCH64

其他版本和架构可作参考。

3.5.1.2 问题描述

如何在银河麒麟高级服务器操作系统下安装 Mysql 数据库?

3.5.1.3 问题分析

X86 和 AARCH 架构的源中自带 Mysql 安装包,所以可以下载对应的 rpm 包，然后再进行安装，而 Loongarch64 架构下没有自带 Mysql 安装包，需要用源码进行安装，当然在安装 Mysql 前需要卸载系统自带的 mariadb。

AARCH 架构下载地址：

http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/icbc-appstore/aarch64/Packages/

X86 架构下载地址：

http://update.cs2c.com.cn:8080/NS/V10/V10SP1/os/adv/lic/icbc-appstore/x86_64/Packages/

（目前 AARCH、X86 已有版本均为 5.7.29、8.0.17）

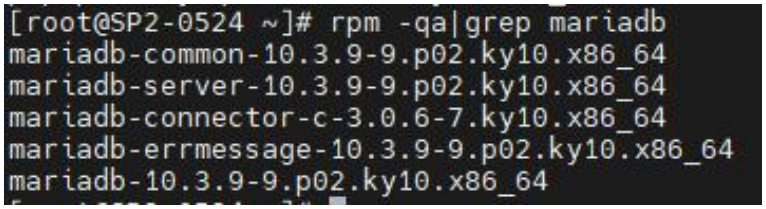
Loongarch64 架构下没有自带 Mysql 安装包，需要用源码进行安装，而在安装 Mysql 前需要卸载系统自带的 mariadb。

3.5.1.4 解决方案

一、对于 X86 和 AARCH 架构

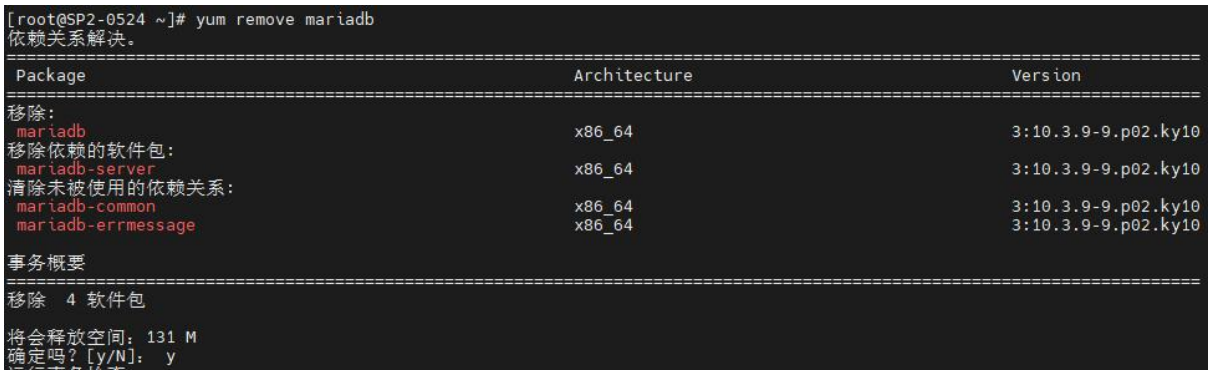
1） 查看是否已安装 mariadb，若是已安装，需要卸载

```
# rpm -qa|grep mariadb
```



要是有，卸载 mariadb

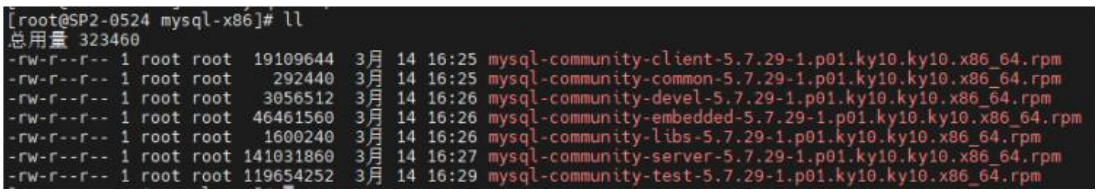
```
# yum remove mariadb
```



2） 将下载好的安装包文件夹上传到服务器，进到该文件夹中

安装

```
# yum localinstall *.rpm
```



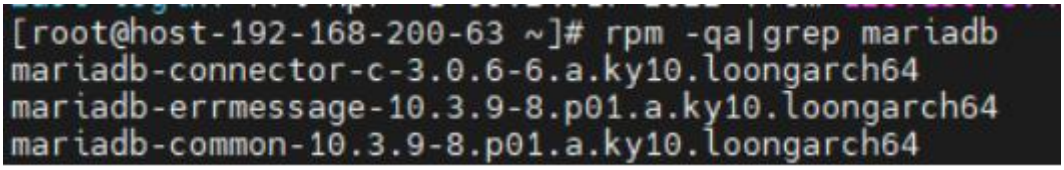
3) 安装成功后启动服务

```
# systemctl start mysqld
```

二、对于 Loongarch64 架构

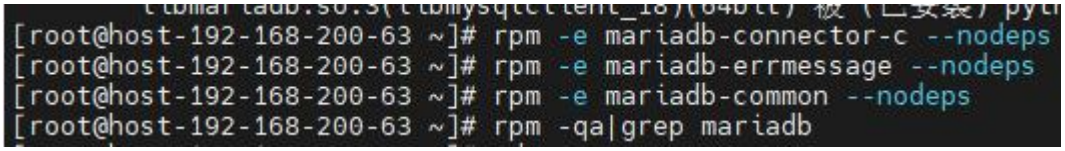
1) 查看是否已安装 mariadb，若是已安装，需要卸载

```
# rpm -qa|grep mariadb
```



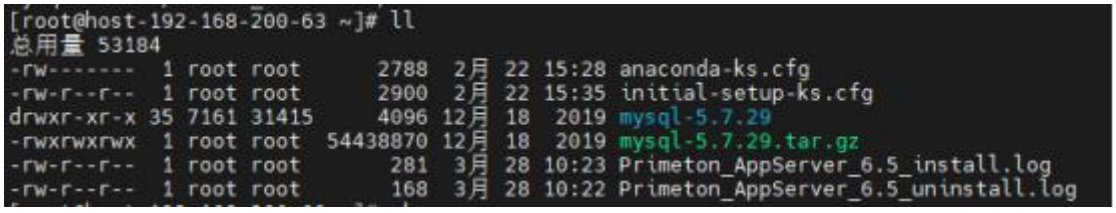
要是有，卸载 mariadb

```
# yum remove mariadb
```



2) 下载源码包并解压

```
# wget https://cdn.mysql.com/archives/mysql-5.7/mysql-5.7.29.tar.gz
# tar -zxvf mysql-5.7.29.tar.gz
```



3) yum 安装编译所需要的工具和库

```
# yum install gcc gcc-c++ cmake ncurses-devel bison openssl-devel rpcgen
```

4) 创建 mysql 的安装目录及数据库存放目录

a. 安装 Mysql

```
# mkdir -p /mysqlapp/mysql
```

b. 存放数据库

```
# mkdir -p /mysqlapp/mysql/data
```

c. 创建 mysql 组

```
# groupadd mysql
```

d. 创建 Mysql 用户，同时属于 mysql 组

```
# useradd -g mysql mysql
```

e. 设置目录权限

```
# chown -R root:mysql   /mysqlapp/mysql
# chown -R mysql:mysql /mysqlapp/mysql/data
```

```
[root@host-192-168-200-63 ~]# mkdir -p /mysqlapp/mysql
[root@host-192-168-200-63 ~]# mkdir -p /mysqlapp/mysql/data
[root@host-192-168-200-63 ~]# groupadd mysql
groupadd：“mysql”组已存在
[root@host-192-168-200-63 ~]# useradd -g mysql mysql
useradd：用户“mysql”已存在
```

5) 进到 mysql-5.7.29，编译安装

```
# cmake . -DCMAKE_INSTALL_PREFIX=/mysqlapp/mysql
-DMYSQL_DATADIR=/mysqlapp/mysql/data
-DSYSCONFDIR=/etc   -DDOWNLOAD_BOOST=1
-DWITH_BOOST=/root/mysql-5.7.29/include/boost_1_59_0
-DWITHOUT_PARTITION_STORAGE_ENGINE=0
# make
# make install
```

6) 配置/etc/my.cnf 文件

注意：5.7 版本没有模板文件 /application/mysql/support-files/my-default.cnf，可根据需要自行添加

```
[mysqld]
port = 3306
user = mysql
basedir = /mysqlapp/mysql
datadir = /mysqlapp/mysql/data
pid-file = /mysqlapp/mysql/data/mysql.pid
sql_mode='ONLY_FULL_GROUP_BY'
log_error = /mysqlapp/mysql/mysql-error.log
[client]
port = 3306
```

```
[mysqld]
port = 3306
user = mysql
basedir = /mysqlapp/mysql
datadir = /mysqlapp/mysql/data
pid-file = /mysqlapp/mysql/data/mysql.pid
sql_mode='ONLY_FULL_GROUP_BY'
log_error = /mysqlapp/mysql/mysql-error.log
#!includedir /etc/my.cnf.d

[client]
port = 3306
~
```

7) 初始化数据库

```
# /mysqlapp/mysql/bin/mysqld --initialize-insecure --user=mysql
--basedir=/mysqlapp/mysql
--datadir=/mysqlapp/mysql/data
```

8) 设置环境变量

```
# echo 'export PATH=/mysqlapp/mysql/bin:$PATH' >> /etc/profile
# source /etc/profile
# tail -1 /etc/profile
```

```
[root@host-192-168-200-63 federated]# /mysqlapp/mysql/bin/mysqld --user=mysql --basedir=/mysqlapp/mysql --datadir=/mysqlapp/mysql/data
[root@host-192-168-200-63 federated]# echo 'export PATH=/mysqlapp/mysql/bin:$PATH' >> /etc/profile
[root@host-192-168-200-63 federated]# source /etc/profile
[root@host-192-168-200-63 federated]# tail -1 /etc/profile
export PATH=/mysqlapp/mysql/bin:$PATH
[root@host-192-168-200-63 federated]#
```

9) 拷贝启动脚本、启动服务、登录数据库（没有密码）并查看版本

```
# cp /mysqlapp/mysql/support-files/mysql.server /etc/init.d/mysqld
# /etc/init.d/mysqld start
```

```
[root@host-192-168-200-63 federated]# cd /mysqlapp/mysql/support-files/
[root@host-192-168-200-63 support-files]# ll
总用量 24
-rw-r--r-- 1 root root   773 12月 18  2019 magic
-rwxr-xr-x 1 root root  1061  4月  1 10:29 mysqld_multi.server
-rwxr-xr-x 1 root root   889  4月  1 10:29 mysql-log-rotate
-rwxr-xr-x 1 root root 10570  4月  1 10:29 mysql.server
[root@host-192-168-200-63 support-files]# cp mysql.server /etc/init.d/mysqld
```

```
[root@host-192-168-200-63 mysql]# /etc/init.d/mysqld start
Starting MySQL. SUCCESS!
```

```
[root@host-192-168-200-63 mysql]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.29 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.29    |
+-----------+
1 row in set (0.00 sec)
```

3.5.2 如何安装其它版本的 docker

3.5.2.1 系统版本

适用系统：V10(SP1)

适用架构：X86、AARCH

其他版本和架构可作参考。

3.5.2.2 问题描述

目前系统自带的 docker 版本为 18.09，有些用户可能需要高版本的 docker，所以需要手动安装其它版本 docker。

3.5.2.3 问题分析

先下载需要的 docker 版本，然后将其解压到相应位置，再配置好 docker 服务，安装好之后进行验证即可。

3.5.2.4 解决方案

1) docker 下载地址（社区版二进制文件）：

https://download.docker.com/linux/static/stable/

2) 配置安装（以 docker 19.03.5 arm 版为例）

a. 下载软件

```
# wget https://download.docker.com/linux/static/stable/aarch64/docker-19.03.5.tgz
```

b. 解压并放置到/usr/bin 下面

```
# tar -zxvf docker-19.03.5.tgz
# cd docker
# cp * /usr/bin
```

c. 配置 docker 服务

```
# vim /usr/lib/systemd/system/docker.service
```

```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target

[Service]
Type=notify
ExecStart=/usr/bin/dockerd
ExecReload=/bin/kill -s HUP $MAINPID
LimitNOFILE=infinity
LimitNPROC=infinity
TimeoutStartSec=0
Delegate=yes
KillMode=process
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

[Install]
WantedBy=multi-user.target
~
~
```

d. 编辑 daemon.json 文件

```
# mkdir /etc/docker
# vim /etc/docker/daemon.json
```

3) 启动 docker 服务

a. 设置开机自动启动 docker

```
# systemctl enable docker
```

b. 重新加载服务

```
# systemctl daemon-reload
```

c. 启动 docker

```
# systemctl start docker
```

4) 版本验证

```
# docker -v
```

```
[root@server1 桌面]# docker -v
Docker version 19.03.5, build 633a0ea
```

```
# docker info
```

```
[root@server1 桌面]# docker info
Client:
 Debug Mode: false

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 19.03.5
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: b34a5c8af56e510852c35414db4c1f4fa6172339
 runc version: 3e425f80a8c931f88e6d94a8c831b9d5aa481657
 init version: fec3683
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 4.19.90-23.8.v2101.ky10.aarch64
```

```
 Kernel Version: 4.19.90-23.8.v2101.ky10.aarch64
 Operating System: Kylin Linux Advanced Server V10 (Tercel)
 OSType: linux
 Architecture: aarch64
 CPUs: 4
 Total Memory: 6.647GiB
 Name: server1
 ID: X2WU:DTYT:YQH5:GGM4:5GUM:EA3Q:3J4D:RF6Y:QCAK:S3KL:UQ5A:ZG4A
 Docker Root Dir: /var/lib/docker
 Debug Mode: false
 Registry: https://index.docker.io/v1/
 Labels:
 Experimental: false
 Insecure Registries:
  127.0.0.0/8
 Registry Mirrors:
  https://registry.docker-cn.com/
 Live Restore Enabled: true
 Product License: Community Engine
```

5) 使用验证

a. 下载 hello-world 镜像

```
# docker pull hello-world
```

```
[root@server1 桌面]# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
109db8fad215: Pull complete
Digest: sha256:0fe98d7debd9049c50b597ef1f85b7c1e8cc81f59c8d623fcb2250e8bec85b38
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
[root@server1 桌面]#
```

b. 运行 hello-world 镜像

```
# docker run hello-world
```

```
[root@server1 桌面]# docker run hello-world

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/

[root@server1 桌面]#
```

c. 查看本地 docker 镜像

```
# docker images
```

```
[root@server1 桌面]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              bc11b176a293        5 weeks ago         9.14kB
```

新版本 docker 安装完成。

3.5.3 如何安装旧版本 cryptography

3.5.3.1 系统版本

适用系统：V10(SP1)

适用架构：X86、AARCH

其他版本和架构可作参考。

3.5.3.2 问题描述

有时候安装软件需要用到 cryptography==1.5.3 版本，此为 python 模块，但在安装过程中由于系统自带的 openssl 1.1.1f 版本的库文件和 cryptography 使用的库文件版本不一致导致部分函数不能使用。

3.5.3.3 问题分析

通过编译安装实现 1.5.3 版本的 cryptography 的安装。

3.5.3.4 解决方案

系统版本以 Kylin-Server-10-SP1-Release-Build20-20210518-aarch64 为例。

1） 安装开发环境 openssl-del

```
# yum install openssl-devel
```

2） 安装低版本的 openssl

a. 下载低版本的 openssl

```
# cd /usr/local/src
```

下载地址：https://www.openssl.org/source/openssl-1.0.2k.tar.gz

b. 然后解压：

```
# tar -xf openssl-1.0.2k.tar.gz
# cd openssl-1.0.2k
```

c. 编译安装低版本的 openssl

```
# ./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared zlib
# make
# make install
```

3） 安装 cryptography==1.5.3

```
# yum install python2-pip
# LDFLAGS="-L/usr/local/ssl/lib -Wl,-rpath,/usr/local/ssl/lib"CFLAGS="-I/usr/local/ssl/include" pip install cryptography==1.5.3
-i http://pypi.douban.com/simple --trusted-host pypi.douban.com
```

```
Collecting cffi>=1.4.1 (from cryptography==1.5.3)
  Downloading http://pypi.doubanio.com/packages/2e/92/87bb61538d7e60da8a7ec247dc048f7671afe17016cd0008b3b710012804/cffi-1.14.6.tar.gz (475kB)
    100% |████████████████████████████████| 481kB 886kB/s
Collecting pycparser (from cffi>=1.4.1->cryptography==1.5.3)
  Downloading http://pypi.doubanio.com/packages/ae/e7/d9c3a176ca4b02024debf82342dab36efadfc5776f9c8db077e8f6e71821/pycparser-2.20-py2.py3-none-any.whl (112kB)
    100% |████████████████████████████████| 112kB 41.3MB/s
Installing collected packages: idna, pyasn1, enum34, ipaddress, pycparser, cffi, cryptography
  Running setup.py install for cffi ... done
  Running setup.py install for cryptography ... done
Successfully installed cffi-1.14.6 cryptography-1.5.3 enum34-1.1.10 idna-2.10 ipaddress-1.0.23 pyasn1-0.4.8 pycparser-2.20
[root@host-192-168-9-24 openssl-1.0.2k]#
```

```
# pip show cryptography
```

```
[root@host-192-168-9-24 openssl-1.0.2k]# pip show cryptography
Name: cryptography
Version: 1.5.3
Summary: cryptography is a package which provides cryptographic recipes and primitives to Python developers.
Home-page: https://github.com/pyca/cryptography
Author: The cryptography developers
Author-email: cryptography-dev@python.org
License: BSD or Apache License, Version 2.0
Location: /usr/lib64/python2.7/site-packages
Requires: idna, pyasn1, six, setuptools, enum34, ipaddress, cffi
Required-by:
[root@host-192-168-9-24 openssl-1.0.2k]#
```

4） 验证

```
# python
```

```
>>>from cryptography.fernet import Fernet
```

```
[root@host-192-168-9-24 openssl-1.0.2k]# python
Python 2.7.16 (default, Jul  9 2020, 06:35:45)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from cryptography.ferent import Fernet
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named ferent
>>> from cryptography.fernet import Fernet
>>> key=Fernet.generate_key()
>>> f=Fernet(key)
>>> token=f.encrypt(b"A really secret message.Not for prying eyes.")
>>> token
'gAAAAABhPrSSF3-WXBEMCsR7YzqDYxwerPZDX0f4ohhFfCyYI8b3kcP4fhf6XtKv-Ka5JF4__P5Asmd69aRtuCmXfwOqljm9fcFBFeD9O7oECWtoWEcwUoc_4Bs1z
VjdzfXfy8EAtguc'
>>> f.decrypt(token)
'A really secret message.Not for prying eyes.'
>>>
```

3.5.4 如何安装 Xpdf

3.5.4.1 系统版本

适用系统：V10(SP2)

适用架构：AARCH

其他版本和架构可作参考。

3.5.4.2 解决方案

1) 安装依赖

```
# yum install cmake freetype zlib libpng gcc-c++ qt5* cups*
```

2) 下载源码，源码地址为：

https://dl.xpdfreader.com/xpdf-4.03.tar.gz

3) 进入源码目录

```
# cd xpdf-4.03
# cmake -DCMAKE_BUILD_TYPE=4.03
```

```
[root@host-192-168-9-20 xpdf-4.03]# cmake -DCMAKE_BUILD_TYPE=4.03
CMake Warning:
  No source or binary directory provided.  Both will be assumed to be the
  same as the current working directory, but note that this warning will
  become a fatal error in future CMake releases.


-- Found FreeType (old-style includes): /usr/lib64/libfreetype.so
-- Qt5 found
-- Found Cups: /usr/lib64/libcups.so (found version "2.2.13")
-- Found fontconfig
-- Configuring done
-- Generating done
-- Build files have been written to: /root/xpdf-4.03
```

4) 编译安装

```
# make && make install
```

5） 验证

```
# pdftotext -v
```

```
[root@host-192-168-9-20 xpdf-4.03]# pdftotext -v
pdftotext version 4.03 [www.xpdfreader.com]
Copyright 1996-2021 Glyph & Cog, LLC
```

3.5.5 如何安装 fish

3.5.5.1 系统版本

适用系统：V10(SP2)

适用架构：AARCH

其他版本和架构可作参考。

3.5.5.2 解决方案

1） 下载源码

fish-3.1.2.tar.gz

下载地址：

链接：https://fishshell.com

2） 上传至机器并解压

```
# tar -zxvf fish-3.1.2.tar.gz
# cd fish-3.1.2
```

3） 安装依赖

```
# yum install gcc-c++
# yum install  ncurses-devel
```

4） 执行编译安装

```
# cmake .
# make
# make install
```

5） 验证

```
# fish
```

```
[root@host-192-168-9-3 fish-3.1.2]# fish
Welcome to fish, the friendly interactive shell
Type `help` for instructions on how to use fish
root@host-192-168-9-3 ~/fish-3.1.2# fish --version
fish, version 3.1.2
root@host-192-168-9-3 ~/fish-3.1.2#
```

3.5.6 如何安装 RabbitMQ

3.5.6.1 系统版本

适用系统：V10(SP1)

适用架构：AARCH

其他版本和架构可作参考。

3.5.6.2 解决方案

以系统为 V10 (SP1)-aarch64-20200711 为例：

方法一：利用 rpm 包安装

1) 安装 erlang

```
# yum localinstall erlang-23.1.5-1.ky10.aarch64.rpm
```

2) 安装 rabbitmq

```
# yum localinstall rabbitmq-server-3.7.28-1.el7.noarch.rpm
```

3) 启动服务

```
# systemctl start rabbitmq-server.service
```

4) 查看服务状态

```
# systemctl status rabbitmq-server.service
```

```
[root@host-192-168-9-24 rabbitmq]# systemctl status rabbitmq-server.service
● rabbitmq-server.service - RabbitMQ broker
   Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2021-09-29 16:33:04 CST; 9s ago
 Main PID: 31748 (beam.smp)
   Status: "Initialized"
    Tasks: 91
   Memory: 89.0M
   CGroup: /system.slice/rabbitmq-server.service
           ├─31748 /usr/lib64/erlang/erts-11.1.3/bin/beam.smp -W w -A 64 -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcs 30 -P 1048576 -t 5000000 -st
           ├─32121 /usr/lib64/erlang/erts-11.1.3/bin/epmd -daemon
           ├─33061 erl_child_setup 32768
           ├─40034 inet_gethost 4
           └─40039 inet_gethost 4

9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:   ##  ##
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:   ##  ##      RabbitMQ 3.7.28. Copyright (c) 2007-2020 Pivotal Software, Inc.
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:   ##########  Licensed under the MPL.  See https://www.rabbitmq.com/
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:   ######  ##
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:   ##########  Logs: /var/log/rabbitmq/rabbit@host-192-168-9-24.log
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:                 /var/log/rabbitmq/rabbit@host-192-168-9-24_upgrade.log
9月 29 16:33:02 host-192-168-9-24 rabbitmq-server[31748]:               Starting broker...
9月 29 16:33:04 host-192-168-9-24 rabbitmq-server[31748]: systemd unit for activation check: "rabbitmq-server.service"
9月 29 16:33:04 host-192-168-9-24 systemd[1]: Started RabbitMQ broker.
9月 29 16:33:04 host-192-168-9-24 rabbitmq-server[31748]:  completed with 0 plugins.
[root@host-192-168-9-24 rabbitmq]#
```

方法二：利用二进制安装

1) 安装依赖

```
# yum install mesa-libGL-devel mesa-libGLU-devel make gcc gcc-c++ openssl
openssl-devel unixODBC unixODBC-devel kernel-devel m4 ncurses-devel
```

2) 解压并编译安装 wxWidgets-3.0.5，该插件在编译 erlang 时会用到

```
# tar -xvf wxWidgets-3.0.5.tar.bz2
# cd wxWidgets-3.0.5
# ./configure --with-opengl --enable-debug --enable-unicode
# make && make install
```

3) 解压并编译安装 erlang

```
# tar -zxvf otp_src_22.0.tar.gz
# cd otp_src_22.0
# mkdir –p /usr/local/erlang
# ./configure --prefix=/usr/local/erlang --without-javac
# make && make install
```

4) 设置环境变量

```
# vim /etc/profile  添加以下内容

export ERLANG_HOME=/usr/local/erlang
export
RABBITMQ_HOME=/usr/local/rabbitmq_software/rabbitmq_server-3.7.28
export PATH=${ERLANG_HOME}/bin:${RABBITMQ_HOME}/sbin:${PATH}
```

```
# source /etc/profile
```

5)

a. 解压 rabbitmq 至/usr/local/rabbitmq_software 并运行

```
# tar -xvf rabbitmq-server-generic-unix-3.7.28.tar.xz
```

b. 加载网页插件，以便访问

```
# rabbitmq-plugins enable rabbitmq_management
```

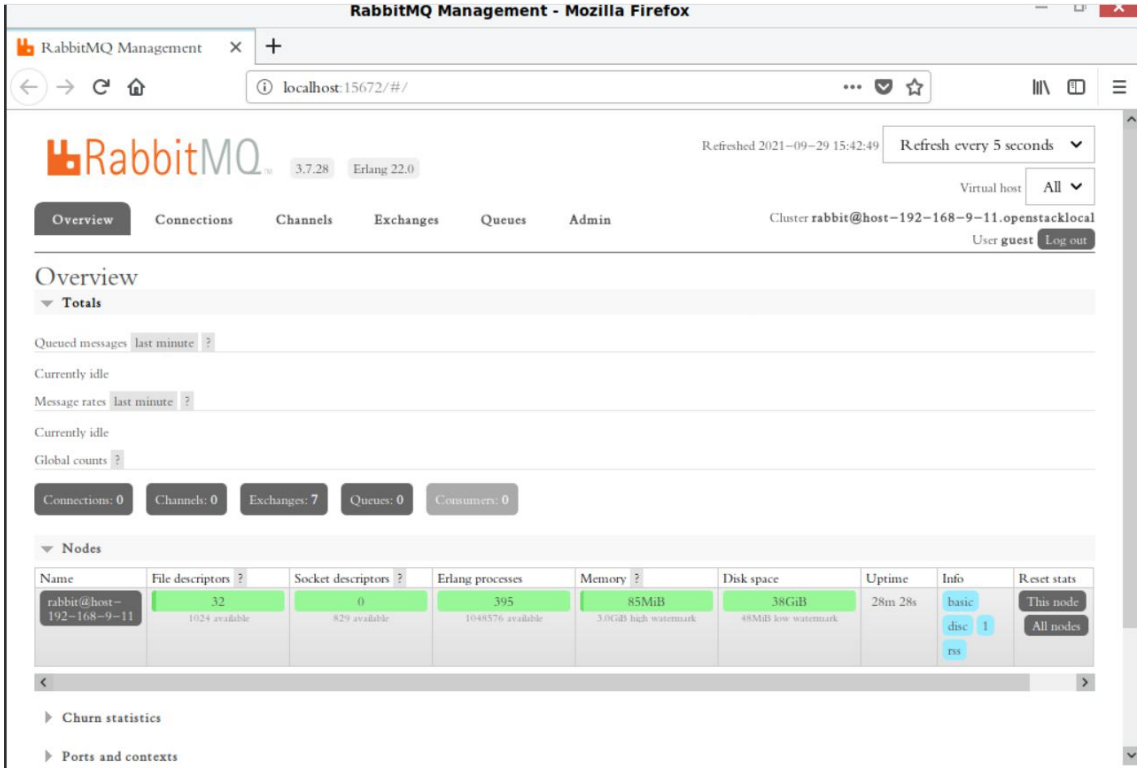
c. 启动后默认监听端口 15672

```
# rabbitmq-server
```

```
[root@host-192-168-9-11 rabbitmq_server-3.7.28]# rabbitmq-server

  ##  ##
  ##  ##      RabbitMQ 3.7.28. Copyright (c) 2007-2020 Pivotal Software, Inc.
  ##########  Licensed under the MPL.  See https://www.rabbitmq.com/
  ######  ##
  ##########  Logs: /usr/local/rabbitmq_software/rabbitmq_server-3.7.28/var/log/rabbitmq/rabbit@host-192-168-9-11.log
                    /usr/local/rabbitmq_software/rabbitmq_server-3.7.28/var/log/rabbitmq/rabbit@host-192-168-9-11_upgrade.log

              Starting broker...
 completed with 3 plugins.
```

```
[root@host-192-168-9-11 ~]# lsof -i:15672
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
beam.smp 156712 root   79u  IPv4 185300      0t0  TCP *:15672 (LISTEN)
```

默认用户名和密码都是 guest，且只能通过本地访问



rabbitmq 中文社区：http://rabbitmq.mr-ping.com/

erlang 与 rabbltmq 版本对应关系：

https://www.rabbitmq.com/which-erlang.html

rabbitmq 官网：https://www.rabbitmq.com/

3.5.7 如何安装 PostgreSQL

3.5.7.1 系统版本

适用系统：V10(SP1)、V10(SP2)

适用架构：X86、AARCH

其他版本和架构可作参考。

3.5.7.2 解决方案

以服务器版本为 release V10 (SP2) /(Sword)-x86_64-Build09/20210524 为例。

1) 卸载系统自带的 mariadb

```
# yum remove mariadb
```

2) 安装 postgresql，安装后会自动创建一个 postgres 用户。

```
# yum install postgresql postgresql-server
```

```
[root@localhost ~]# tail /etc/passwd
tomcat:x:91:91:Apache Tomcat:/usr/share/tomcat:/sbin/nologin
dbus:x:978:995:System Message Bus:/:/usr/sbin/nologin
dnsmasq:x:977:977:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/usr/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/usr/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/usr/sbin/nologin
systemd-timesync:x:976:996:systemd Time Synchronization:/:/usr/sbin/nologin
systemd-coredump:x:975:997:systemd Core Dumper:/:/usr/sbin/nologin
tss:x:59:59:tss user for tpm2:/:/usr/sbin/nologin
tcpdump:x:72:72::/:/sbin/nologin
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
[root@localhost ~]#
```

3) 设置环境变量

```
# echo "export PGDATA=/var/lib/pgsql/data/" >> /etc/profile
# source /etc/profile
```

4) 修改目录权限，初始化 postgresql

```
# chown -R postgres:postgres /var/lib/pgsql/data/
# chmod -R 755 /var/lib/pgsql/data/
```

切换至 postgres 用户

```
# su postgres
# initdb -D /var/lib/pgsql/data/
```

```
bash-5.0$ initdb -D /var/lib/pgsql/data/
could not change directory to "/root": 权限不够
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 "zh_CN.UTF-8".
The default database encoding has accordingly been set to "UTF8".
initdb: could not find suitable text search configuration for locale "zh_CN.UTF-8"
The default text search configuration will be set to "simple".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/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 /var/lib/pgsql/data/ -l logfile start

bash-5.0$
```

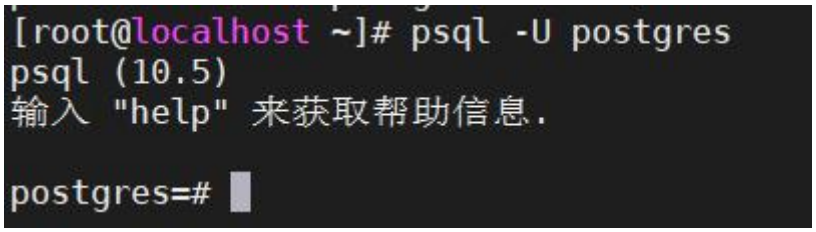
5) 启动数据库服务，查看端口

```
# systemctl start postgresql.service
# systemctl status postgresql.service
# lsof -i:5432
```

```
[root@localhost ~]# systemctl start postgresql.service
[root@localhost ~]# systemctl status postgresql.service
● postgresql.service - PostgreSQL database server
   Loaded: loaded (/usr/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2021-08-02 20:29:53 CST; 4s ago
  Process: 29753 ExecStartPre=/usr/libexec/postgresql-check-db-dir postgresql (code=exited, status=0/SUCCESS)
 Main PID: 29756 (postmaster)
    Tasks: 7
   Memory: 13.0M
   CGroup: /system.slice/postgresql.service
           ├─29756 /usr/bin/postmaster -D /var/lib/pgsql/data
           ├─29758 postgres: checkpointer process
           ├─29759 postgres: writer process
           ├─29760 postgres: wal writer process
           ├─29761 postgres: autovacuum launcher process
           ├─29762 postgres: stats collector process
           └─29763 postgres: bgworker: logical replication launcher

8月 02 20:29:53 localhost.localdomain systemd[1]: Starting PostgreSQL database server...
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.271 CST [29756] 日志:  listening on IPv6 address "::1", port 5432
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.271 CST [29756] 日志:  listening on IPv4 address "127.0.0.1", port 5432
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.284 CST [29756] 日志:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.288 CST [29756] 日志:  listening on Unix socket "/tmp/.s.PGSQL.5432"
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.327 CST [29757] 日志:  数据库上次关闭时间为 2021-08-02 20:25:37 CST
8月 02 20:29:53 localhost.localdomain postmaster[29756]: 2021-08-02 20:29:53.384 CST [29756] 日志:  数据库系统准备接受连接
8月 02 20:29:53 localhost.localdomain systemd[1]: Started PostgreSQL database server.
[root@localhost ~]# lsof -i:5432
COMMAND     PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
postmaste 29756 postgres    3u  IPv6  88235      0t0  TCP localhost:postgres (LISTEN)
postmaste 29756 postgres    4u  IPv4  88236      0t0  TCP localhost:postgres (LISTEN)
[root@localhost ~]#
```

6) 测试数据库连接

```
# psql -U postgres
```



3.5.8 如何安装 ODBC Driver for PostgreSQL

3.5.8.1 系统版本

适用系统：V10(SP2)

适用架构：AARCH

其他版本和架构可作参考。

3.5.8.2 解决方案

以 aarch 架构下 V10(SP2)系统 0524 版本为例。

1) 安装软件包

系统源自带 postgresql 软件包

```
# yum install unixODBC-devel postgresql postgresql-devel postgresql-libs
```

2) 获取 psqlodbc 源码

```
#
wget https://ftp.postgresql.org/pub/odbc/versions/src/psqlodbc-13.00.0000.tar
.gz
# tar -zxf psqlodbc-13.00.0000.tar.gz
# cd psqlodbc-13.00.0000
```

3) 编译 psqlodbc

```
# ./configure
# make
# make install
```

```
[root@localhost psqlodbc-13.00.0000]# make install
make[1]: 进入目录"/opt/psqlodbc-13.00.0000"
 /usr/bin/mkdir -p '/usr/local/lib'
 /bin/sh ./libtool   --mode=install /usr/bin/install -c   psqlodbcw.la psqlodbca.la '/usr/local/lib'
libtool: install: /usr/bin/install -c .libs/psqlodbcw.so /usr/local/lib/psqlodbcw.so
libtool: install: /usr/bin/install -c .libs/psqlodbcw.lai /usr/local/lib/psqlodbcw.la
libtool: install: /usr/bin/install -c .libs/psqlodbca.so /usr/local/lib/psqlodbca.so
libtool: install: /usr/bin/install -c .libs/psqlodbca.lai /usr/local/lib/psqlodbca.la
libtool: finish: PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/root/bin:/sbin" ldconfig -n /usr/local/lib
----------------------------------------------------------------------
Libraries have been installed in:
   /usr/local/lib

If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the `-LLIBDIR'
flag during linking and do at least one of the following:
   - add LIBDIR to the `LD_LIBRARY_PATH' environment variable
     during execution
   - add LIBDIR to the `LD_RUN_PATH' environment variable
     during linking
   - use the `-Wl,-rpath -Wl,LIBDIR' linker flag
   - have your system administrator add LIBDIR to `/etc/ld.so.conf'

See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
make[1]: 对"install-data-am"无需做任何事。
make[1]: 离开目录"/opt/psqlodbc-13.00.0000"
```

编译过程中报错：

make 时出现报错

报错 1

```
odbcapi30.c:120:1: error: conflicting types for 'SQLColAttribute'
 SQLColAttribute(SQLHSTMT StatementHandle,
 ^~~~~~~~~~~~~~~
In file included from psqlodbc.h:112:0,
                 from odbcapi30.c:21:
/usr/include/sql.h:613:24: note: previous declaration of 'SQLColAttribute' was here
     SQLRETURN  SQL_API SQLColAttribute(SQLHSTMT StatementHandle,
                        ^~~~~~~~~~~~~~~
make[1]: *** [Makefile:811: psqlodbcw_la-odbcapi30.lo] 错误 1
make[1]: 离开目录"/opt/psqlodbc-13.00.0000"
make: *** [Makefile:479: all] 错误 2
```

解决方法：

修改 vim /usr/include/sql.h

屏蔽以下内容（使用#if 0 和#endif）

```
613 #if 0
614     SQLRETURN  SQL_API SQLColAttribute(SQLHSTMT StatementHandle,
615                                        SQLUSMALLINT ColumnNumber, SQLUSMALLINT FieldIdentifier,
616                                        SQLPOINTER CharacterAttribute, SQLSMALLINT BufferLength,
617                                        SQLSMALLINT *StringLength, SQLLEN *NumericAttribute );
618 #endif
```

报错 2

```
odbcapi30w.c:266:1: error: conflicting types for 'SQLColAttributeW'
 SQLColAttributeW(SQLHSTMT hstmt,
 ^~~~~~~~~~~~~~~~
In file included from /usr/include/sqlext.h:2201:0,
                 from psqlodbc.h:113,
                 from odbcapi30w.c:15:
/usr/include/sqlucode.h:29:19: note: previous declaration of 'SQLColAttributeW' was here
 SQLRETURN SQL_API SQLColAttributeW(
                   ^~~~~~~~~~~~~~~~
make[1]: *** [Makefile:832: psqlodbcw_la-odbcapi30w.lo] 错误 1
make[1]: 离开目录"/opt/psqlodbc-13.00.0000"
make: *** [Makefile:479: all] 错误 2
```

解决方法：

修改 vim /usr/include/sqlucode.h

屏蔽以下内容（使用#if 0 和#endif）

```
29 #if 0
30 SQLRETURN SQL_API SQLColAttributeW(
31         SQLHSTMT                hstmt,
32         SQLUSMALLINT    iCol,
33         SQLUSMALLINT    iField,
34         SQLPOINTER              pCharAttr,
35         SQLSMALLINT             cbCharAttrMax,
36         SQLSMALLINT     *pcbCharAttr,
37         SQLLEN          *pNumAttr);
38 #endif
39
40 SQLRETURN SQL_API SQLColAttributesW(
41     SQLHSTMT            hstmt,
42     SQLUSMALLINT        icol,
43     SQLUSMALLINT        fDescType,
44     SQLPOINTER          rgbDesc,
```

4) 验证

```
# odbcinst -q -d
```

```
[root@localhost psqlodbc-13.00.0000]# odbcinst -q -d
[PostgreSQL]
```

3.5.9 如何安装 sentencepiece

3.5.9.1 系统版本

适用系统：V10（SP1）

适用架构：X86、LOONGARCH

其他版本和架构可作参考。

3.5.9.2 解决方案

1) 安装依赖

```
# sudo yum install cmake gperftools python3-pip gcc gcc-c++
```

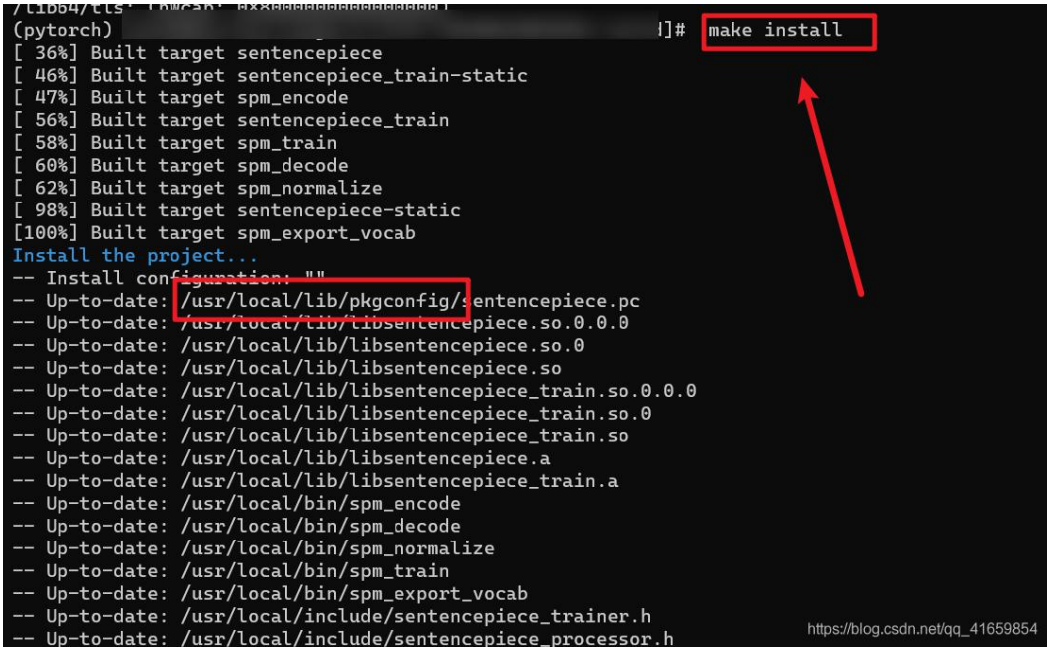
2)

```
# git clone https://github.com/google/sentencepiece
# cd /path/to/sentencepiece
# mkdir build
# cd build
# cmake ..
```

```
# make -j $(nproc)
# make install
```

3) 指定 PKG_CONFIG_PATH 路径

```
# export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig/（该路径在 make install 中可查看）
```



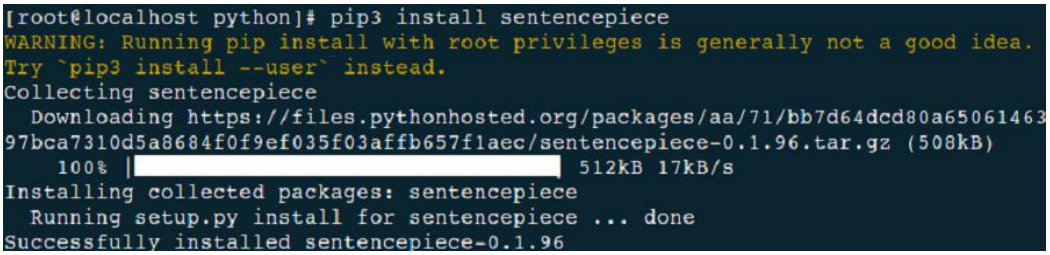
查看 echo $PKG_CONFIG_PATH

4) Python 中安装 sentencepiece，进入到 sentencepiece/python，执行以下两条命令

```
# python3 setup.py build
# python3 setup.py install
```

或者执行

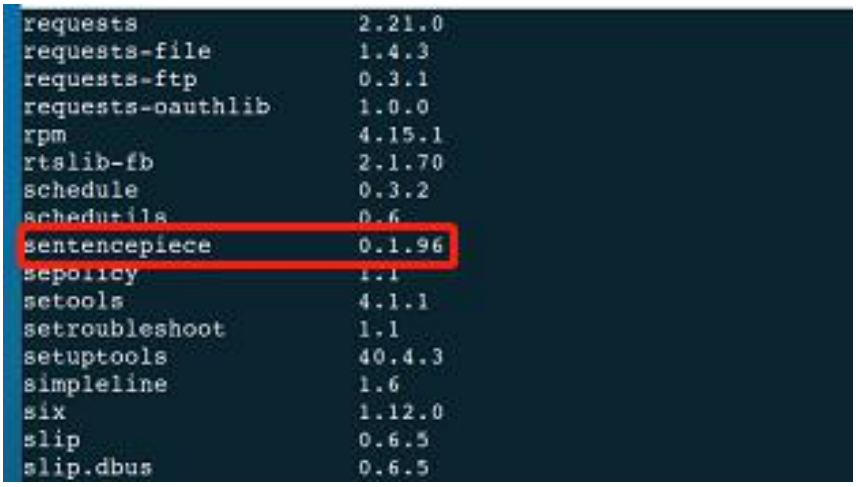
```
# pip3 install sentencepiece
```



（上述方法选其中一个即可）

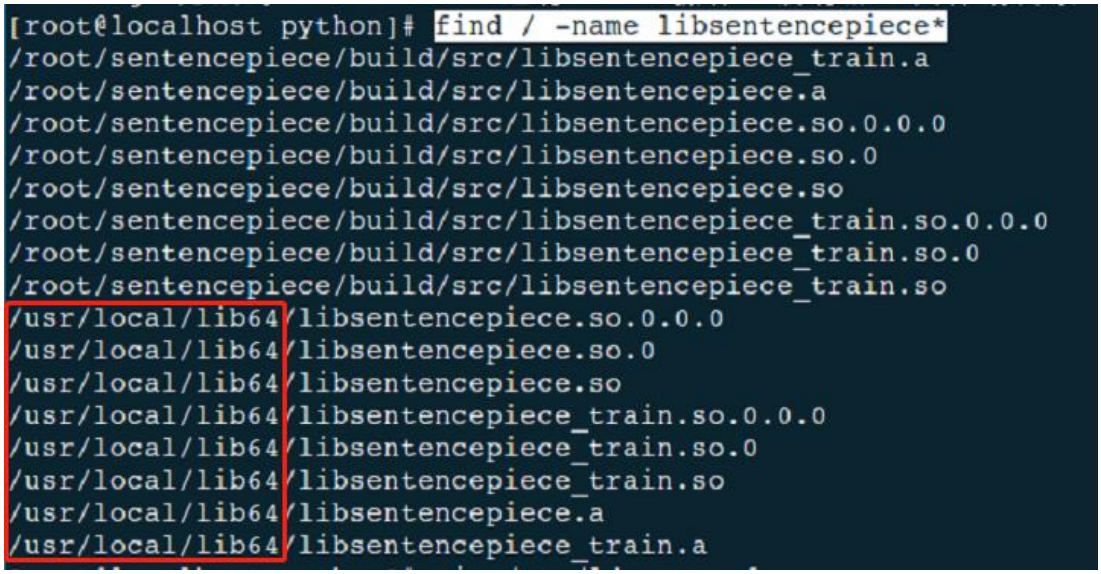
查看是否已安装：

```
# pip3 list
```



5)

```
# find / -name libsentencepiece*
```



6) 编辑 /etc/ld.so.conf 文件，添加该路径

```
# vim /etc/ld.so.conf,

Include ld.so.conf.d/*.conf
/usr/local/lib64
```

7) 进入 python3，然后导入 sentencepiece 进行验证

```
# cd python3
>>> import sentencepiece
```

```
[root@localhost python]# python3
Python 3.7.9 (default, Mar  2 2021, 02:43:11)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sentencepiece
>>> exit()
```

3.5.10 如何安装 zabbix

3.5.10.1 系统版本

适用系统：V10（SP1）

适用架构：X86

其他版本和架构可作参考。

3.5.10.2 解决方案

1) 下载源码并上传至服务器安装依赖

https://www.zabbix.com/download_sources#60LTS

2) 解压并进入源码目录

```
# tar -zxvf zabbix-5.0.22.tar.gz
# cd zabbix-5.0.22
```

3) 安装依赖

```
# yum install mariadb-devel net-snmp-devel OpenIPMI-devel libevent-devel curl-devel
```

4) 创建目录，编译安装

```
# mkdir -p /opt/zabbix
# ./configure --prefix=/opt/zabbix --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2 --with-openipmi
# make && make install
```

5) 创建 zabbix 用户和组，赋予 zabbix 目录权限

```
# groupadd zabbix
# useradd zabbix
# passwd zabbix
# chown -R zabbix:zabbix /opt/zabbix
```

6) 验证安装

```
# su - zabbix
# cd /opt/zabbix
```

```
[zabbix@host-192-168-9-11 ~]$ cd /opt/zabbix/
[zabbix@host-192-168-9-11 zabbix]$ ls
bin  etc  lib  sbin  share
[zabbix@host-192-168-9-11 zabbix]$ tree
.
├── bin
│   ├── zabbix_get
│   ├── zabbix_js
│   └── zabbix_sender
├── etc
│   ├── zabbix_agentd.conf
│   ├── zabbix_agentd.conf.d
│   ├── zabbix_server.conf
│   └── zabbix_server.conf.d
├── lib
│   └── modules
├── sbin
│   ├── zabbix_agentd
│   └── zabbix_server
└── share
    ├── man
    │   ├── man1
    │   │   ├── zabbix_get.1
    │   │   └── zabbix_sender.1
    │   └── man8
    │       ├── zabbix_agentd.8
    │       └── zabbix_server.8
    └── zabbix
        ├── alertscripts
        └── externalscripts

14 directories, 11 files
[zabbix@host-192-168-9-11 zabbix]$
```

```
# sbin/zabbix_server -V
```

```
[zabbix@host-192-168-9-11 zabbix]$ sbin/zabbix_server -V
zabbix_server (Zabbix) 5.0.22
Revision 90ee9e33d3 4 April 2022, compilation time: Apr 15 2022 10:30:17

Copyright (C) 2022 Zabbix SIA
License GPLv2+: GNU GPL version 2 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it according to
the license. There is NO WARRANTY, to the extent permitted by law.
[zabbix@host-192-168-9-11 zabbix]$
```

3.5.11 如何安装 geckodriver

3.5.11.1 系统版本

适用系统：V10（SP1）

适用架构：AARCH

其他版本和架构可作参考。

3.5.11.2 解决方案

1) 安装 rust

```
# curl https://sh.rustup.rs -sSf | sh
```

中间会出现一个交互界面，选 1 后回车

```
Current installation options:


   default host triple: aarch64-unknown-linux-gnu
     default toolchain: stable (default)
               profile: default
  modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1
```

安装完成后初始化环境变量

```
  stable-aarch64-unknown-linux-gnu installed - rustc 1.59.0 (9d1b2106e 2022-02-23)


Rust is installed now. Great!

To get started you may need to restart your current shell.
This would reload your PATH environment variable to include
Cargo's bin directory ($HOME/.cargo/bin).

To configure your current shell, run:
source $HOME/.cargo/env
```

```
# source $HOME/.cargo/env
```

2) 安装 rustc 目标工具链

```
# rustup target install armv7-unknown-linux-gnueabihf
```

3) 上传源码至服务器并解压

源码地址：https://github.com/mozilla/geckodriver

```
# tar -zxvf geckodriver-0.28.0.tar.gz
```

4) 修改 geckodriver/.cargo/config 文件

```
# vim geckodriver-0.28.0/.cargo/config

[target.armv7-unknown-linux-gnueabihf]
linker = "arm-linux-gnueabihf-gcc"
```

5) 编译安装

```
# cd geckodriver-0.28.0
# cargo build --release --target aarch64-unknown-linux-gnu
```

生成的二进制文件位于

geckodriver-0.28.0/target/aarch64-unknown-linux-gnu/release 目录。

```
[root@host-192-168-9-11 release]# ls
build  deps  examples  geckodriver  geckodriver.d  incremental
[root@host-192-168-9-11 release]# pwd
/root/geckodriver-0.28.0/target/aarch64-unknown-linux-gnu/release
```

6) 测试验证

```
# cd geckodriver-0.28.0/src
# cargo test
```

```
test result: ok. 50 passed; 0 failed; 4 ignored; 0 measured; 0 filtered out; finished in 0.02s
```

```
# geckodriver-0.28.0/target/aarch64-unknown-linux-gnu/release/
```

```
[root@host-192-168-9-11 src]# ~/geckodriver-0.28.0/target/aarch64-unknown-linux-gnu/release/geckodriver
1647398159440   geckodriver     INFO    Listening on 127.0.0.1:4444
```

```
[root@host-192-168-9-11 ~]# lsof -i:4444
COMMAND     PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
geckodriv 25519 root    3u  IPv4  59357      0t0  TCP localhost:krb524 (LISTEN)
[root@host-192-168-9-11 ~]#
```

3.5.12 如何安装.NET CORE

3.5.12.1 系统版本

适用系统：V10（SP1）

适用架构：AARCH

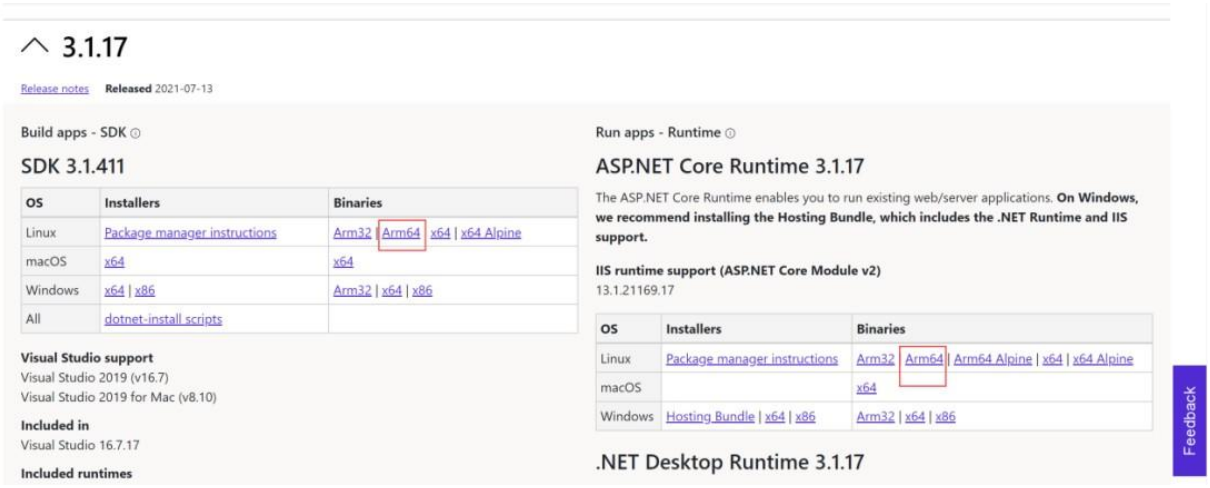
其他版本和架构可作参考。

3.5.12.2 解决方案

1) 手动下载软件包

安装地址如下（下载 sdk 和 asp.net core）：

https://dotnet.microsoft.com/download/dotnet/3.1



下载 arm64 版后上传至服务器

2) 安装 ASP.NET CORE

a. 切换到 root 用户：

```
# sudo su
```

b. 输入以下命令：

```
# mkdir -p $HOME/dotnet && tar zxf aspnetcore-runtime-3.1.17-linux-arm64.tar.gz -C $HOME/dotnet
```

c. 添加环境变量：

```
# vi /etc/profile
```

添加两行内容

```
export DOTNET_ROOT=$HOME/dotnet
export PATH=$PATH:$DOTNET_ROOT
```

d. 保存退出，输入以下命令使环境变量生效

```
# source /etc/profile
```

e. 查看安装版本

```
# dotnet --info
```

```
[root@localhost netcore]# dotnet --info
  It was not possible to find any installed .NET Core SDKs
  Did you mean to run .NET Core SDK commands? Install a .NET Core SDK from:
      https://aka.ms/dotnet-download

Host (useful for support):
  Version: 3.1.17
  Commit:  3a75b805fa

.NET Core SDKs installed:
  No SDKs were found.

.NET Core runtimes installed:
  Microsoft.AspNetCore.App 3.1.17 [/root/dotnet/shared/Microsoft.AspNetCore.App]
  Microsoft.NETCore.App 3.1.17 [/root/dotnet/shared/Microsoft.NETCore.App]

To install additional .NET Core runtimes or SDKs:
  https://aka.ms/dotnet-download
```

```
# dotnet --version
```

```
[root@localhost netcore]# dotnet --version
3.1.411
```

3) 安装 SDK

a. 添加环境变量

如果没有添加环境变量，重复步骤（2）的 cd 即可。

b. 输入以下命令：

```
# mkdir -p $HOME/dotnet && tar zxf dotnet-sdk-3.1.411-linux-arm64.tar.gz -C $HOME/dotnet
```

4) 检验能否运行

a. 创建测试脚本

```
# vi test.sh
```

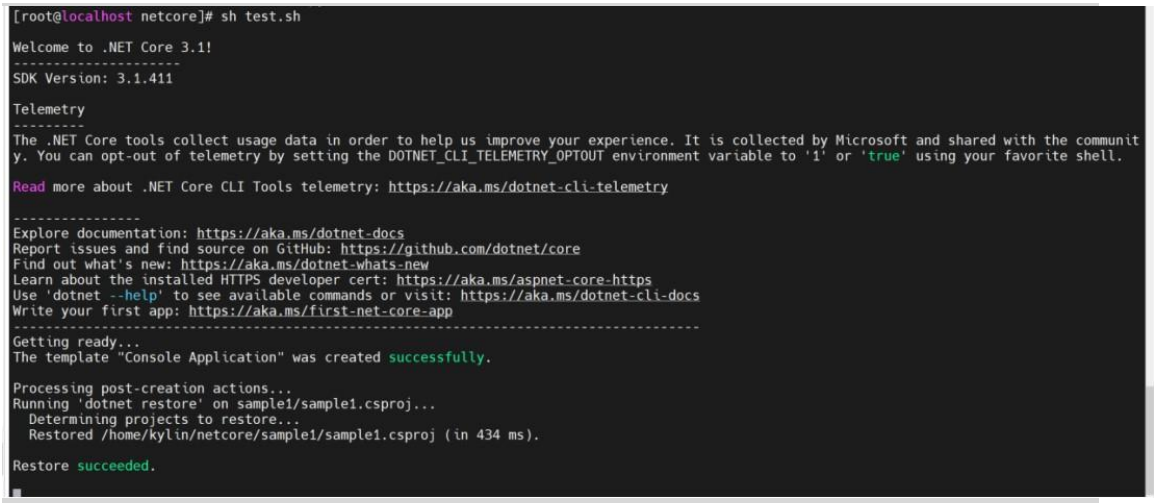
b. 添加以下内容

```
#!/bin/bash
dotnet new console --output sample1
dotnet run --project sample1
```

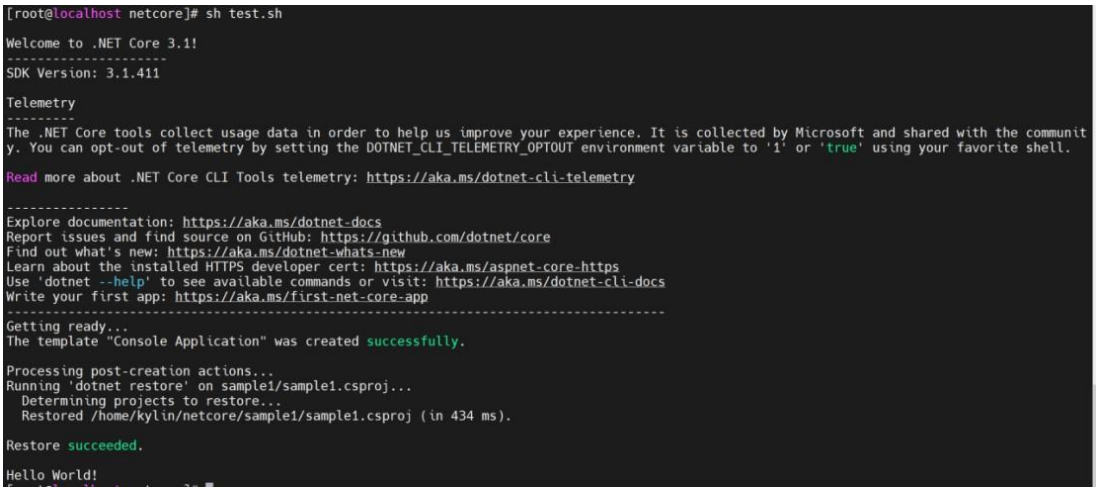
c. 保存退出

d. 执行测试脚本

```
#                    sh                    test.sh
```



e. 查看结果



3.5.13 如何安装 apache-dophinescheduler

3.5.13.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：X86

其他版本或架构可做参考。

3.5.13.2 解决方案

获取源码

```
#
wget https://archive.apache.org/dist/dolphinscheduler/1.3.6/apache-dolphinsc
heduler-1.3.6-bin.tar.gz
```

下载 mysql 和 mysql-connector-j

```
#
wget https://cdn.mysql.com/archives/mysql-5.7/mysql-5.7.29-linux-glibc2.12-x86_64.tar.gz
#
wget https://cdn.mysql.com/archives/mysql-connector-java-8.0/mysql-connector-j-8.0.31.tar.gz
#yum install java-1.8.0-openjdk-devel
```

安装部署 mysql，参考 mysql-8.0.20 在 V10-SP1-aarch64 上编译安装

解压 dolphinscheduler

```
# mkdir -p /opt/dolphinscheduler
# tar -zxvf apache-dolphinscheduler-1.3.6-bin.tar.gz -C /opt/dolphinscheduler
# mv apache-dolphinscheduler-1.3.6-bin  dolphinscheduler-bin
```

创建部署用户

```
# useradd dolphinscheduler
# passwd dolphinscheduler
# sed -i '$adolphinscheduler ALL=(ALL) NOPASSWD: NOPASSWD: ALL' /etc/sudoers
# sed -i 's/Defaults requirett/#Defaults requirett/g' /etc/sudoers
# chown -R dolphinscheduler:dolphinscheduler dolphinscheduler-bin
```

复制第一步中的 mysql 驱动至 dolphinscheduler 的 lib 目录下

```
# tar -zxvf mysql-connector-j-8.0.31.tar.gz
# cd mysql-connector-j-8.0.31/
# cp mysql-connector-j-8.0.31.jar /opt/dolphinscheduler/dolphinscheduler-bin/lib/
```

启动 mysql，创建数据库

```
# su dolphinscheduler
$ mysql -uroot -p
mysql> CREATE DATABASE dolphinscheduler DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;
mysql> GRANT ALL PRIVILEGES ON dolphinscheduler.* TO 'test'@'%' IDENTIFIED BY 'test';
mysql> GRANT ALL PRIVILEGES ON dolphinscheduler.* TO 'test'@'localhost' IDENTIFIED BY 'test';
mysql> flush privileges;
mysql> exit
```

配置 dolphinscheduler

```
# vim /opt/dolphinscheduler/dolphinscheduler-bin/conf/datasource.properties
```

注释掉 postgresql 的相关配置，修改 mysql 的配置

```
# postgresql
#spring.datasource.driver-class-name=org.postgresql.Driver
#spring.datasource.url=jdbc:postgresql://127.0.0.1:5432/dolphinscheduler
#spring.datasource.username=test
#spring.datasource.password=test

# mysql
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/dolphinscheduler?useUnicode=true&characterEncoding=UTF-8
spring.datasource.username=test
spring.datasource.password=test

# connection configuration
#spring.datasource.initialSize=5
# min connection number
#spring.datasource.minIdle=5
# max connection number
#spring.datasource.maxActive=50

# max wait time for get a connection in milliseconds. if configuring maxWait, fair locks are enabled by default and concu
# If necessary, unfair locks can be used by configuring the useUnfairLock attribute to true
```

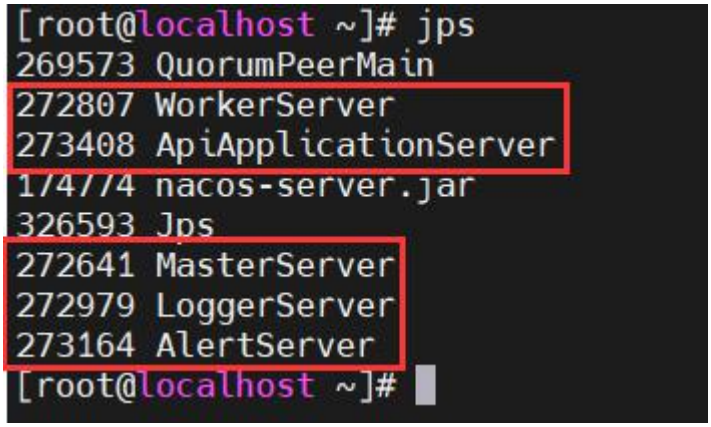
```
#
/opt/dolphinscheduler/dolphinscheduler-bin/scripts/create-dolphinscheduler.s
h
#                                                                    vim
/opt/dolphinscheduler/dolphinscheduler-bin/conf/config/install_config.conf
# Note: if kerberos is enabled, please config hdfsRootUser=
    hdfsRootUser="hdfs"
# kerberos config
    # whether kerberos starts, if kerberos starts, following four items need to
config, otherwise please ignore
    kerberosStartUp="false"
    # kdc krb5 config file path
    krb5ConfPath="$installPath/conf/krb5.conf"
    # keytab username
    keytabUserName="hdfs-mycluster@ESZ.COM"
    # username keytab path
    keytabPath="$installPath/conf/hdfs.headless.keytab"

    # api server port
    apiServerPort="12345"

    # install hosts
    # Note: install the scheduled hostname list. If it is pseudo-distributed, just
write a pseudo-distributed hostname
    ips="localhost"
# ssh port, default 22
    # Note: if ssh port is not default, modify here
    sshPort="22"
# run master machine
    # Note: list of hosts hostname for deploying master
    masters="localhost"
```

```
# run worker machine
    # note: need to write the worker group name of each worker, the default value is "default"
    workers="localhost:default"
# run alert machine
    # note: list of machine hostnames for deploying alert server
    alertServer="localhost"
# run api machine
    # note: list of machine hostnames for deploying api server
    apiServers="localhost"
```

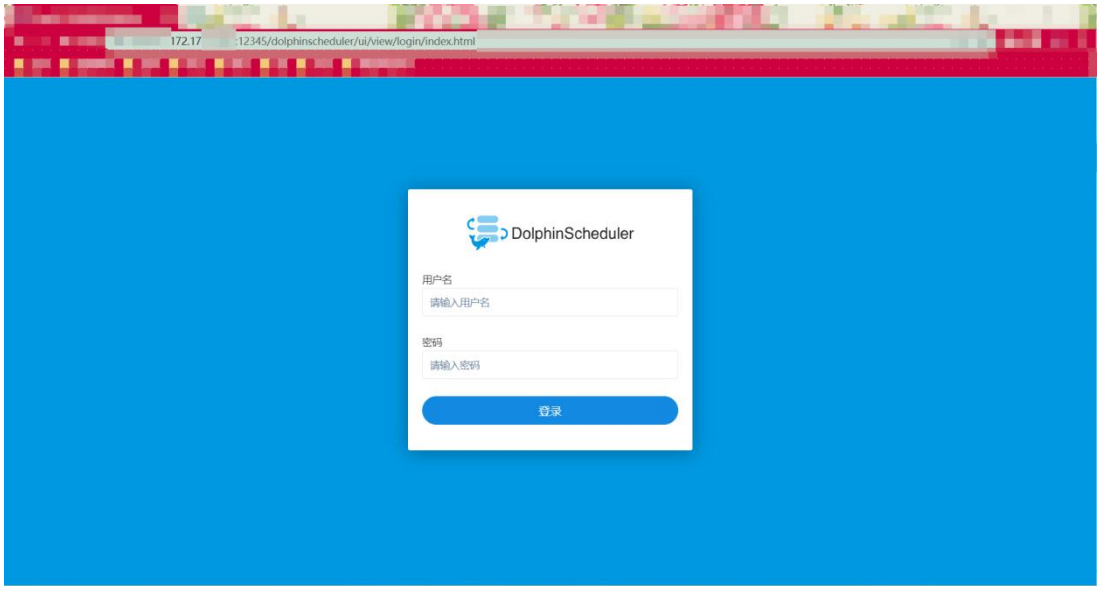
启动 dophinscheduler

```
# opt/dolphinscheduler/dolphinscheduler-bin/install.sh
# jps
```

查看进程会有 5 个服务



访问 ip：12345/dophinscheduler



3.5.14 如何安装 mongodb-4.2.6

3.5.14.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：ARM64 添加架构 X86

其他版本或架构可做参考。

3.5.14.2 解决方案

源码获取

下载

```
https://github.com/mongodb/mongo/archive/refs/tags/r4.2.6.tar.gz
```

编译步骤

gcc 版本要求 8 以上

```
To build MongoDB, you will need:

* A modern C++ compiler capable of compiling C++17. One of the following is required:
    * GCC 8.0 or newer
    * Clang 7.0 (or Apple XCode 10.0 Clang) or newer
    * Visual Studio 2017 version 15.9 or newer (See Windows section below for details)
* On Linux and macOS, the libcurl library and header is required. MacOS includes libcurl.
    * Fedora/RHEL - `dnf install libcurl-devel`
    * Ubuntu/Debian - `apt-get install libcurl-dev`
* Python 3.7.x and Pip modules:
  * See the section "Python Prerequisites" below.
```

此处编译 9.3

下载 gcc-9.3 相关包

```
http://mirrors.aliyun.com/gnu/gcc/gcc-9.3.0/gcc-9.3.0.tar.gz
http://mirrors.aliyun.com/gnu/gmp/gmp-6.1.2.tar.bz2
http://mirrors.aliyun.com/gnu/mpc/mpc-1.1.0.tar.gz
http://mirrors.aliyun.com/gnu/mpfr/mpfr-4.0.2.tar.gz
```

分别把以上四个软件包解压

```
# tar -zxf gcc-9.3.0.tar.gz
# tar -zxf mpc-1.1.0.tar.gz
# tar -zxf mpfr-4.0.2.tar.gz
```

```
# tar -xf gmp-6.1.2.tar.bz2
```

把 mpc mpfr gmp 分别 copy 到 gcc-9.3.0 文件夹下面

```
# mv gmp-6.1.2 gcc-9.3.0/gmp
# mv mpc-1.1.0 gcc-9.3.0/mpc
# mv mpfr-4.0.2 gcc-9.3.0/mpfr
```

创建 gcc-9.3.0 的安装文件夹

```
# mkdir /opt/gcc9
```

进入到 gcc-9.3.0 文件夹

```
# cd gcc-9.3.0
```

安装依赖包

```
# yum install texinfo
```

编译安装 gcc

```
# mkdir build
# cd build
#                    ../configure                    --prefix=/opt/gcc9
--enable-languages=c,c++    --disable-multilib
# make -j`nproc`
# make install
```

添加软连接

```
# cd /opt/gcc9/bin/
# ln -s gcc cc
# ln -s g++ g++-9
# ln -s gcc gcc-9
# ln -s /opt/gcc9/bin/* /usr/local/bin/
```

添加环境变量

```
# vim /etc/profile
```

添加如下内容

```
export GCC9_HOME=/opt/gcc9
export PATH=$GCC9_HOME/bin:$PATH
export CC=gcc-9
export CXX=g++-9
```

生效环境变量

```
# source /etc/profile
```

验证版本

```
[root@localhost ~]# gcc --version
gcc (GCC) 9.3.0
Copyright © 2019 Free Software Foundation, Inc.
本程序是自由软件；请参看源代码的版权声明。本软件没有任何担保；
包括没有适销性和某一专用目的下的适用性担保。
[root@localhost ~]#
```

安装依赖包

```
# yum install python3-pip python3-psutil python3-pymongo python3-PyYAML
python3-packaging curl-devel openssl-devel python3-devel

# pip3 install Cheetah3 regex==2021.11.10 requirements_parser==0.3.1
-i https://pypi.mirrors.ustc.edu.cn/simple/
```

解压

```
# tar -zxf mongo-r4.2.6.tar.gz
# cd /opt/mongo-r4.2.6
# pip3 install -r etc/pip/compile-requirements.txt
-i https://pypi.mirrors.ustc.edu.cn/simple/
```

编译安装

```
# python3 buildscripts/scons.py MONGO_VERSION=4.2.6 all
CFLAGS="-march=armv8-a+crc -mtune=generic" -j 32
--disable-warnings-as-errors
# python3 buildscripts/scons.py MONGO_VERSION=4.2.6
--prefix=/opt/mongodb-4.2.6-bin/ --disable-warnings-as-errors
CFLAGS="-march=armv8-a+crc" install -j 32
```

编译完成的二进制文件在/opt/mongodb-4.2.6-bin 目录下

进入该目录，删除调试信息

```
# strip mongo mongod mongos
```

```
[root@localhost bin]# strip mongo mongod mongos
[root@localhost bin]# ll
总用量 124468
-rwxr-xr-x 1 root root     7694 10月  13 13:53 install_compass
-rwxr-xr-x 1 root root 38526152 10月  13 13:55 mongo
-rwxr-xr-x 1 root root 57475672 10月  13 13:55 mongod
-rwxr-xr-x 1 root root 31439096 10月  13 13:55 mongos
```

编译 mongodb-tools

```
# cd /opt/mongo-r4.2.6/src/mongo/gotools/src/github.com/mongodb/mongo-tools
```

安装依赖

```
# yum install libpcap-devel go
```

配置 go 的环境变量

```
# vim /etc/profile/
export GOROOT=/usr/lib/golang/
export GOPATH=/usr/lib/golang/src
export PATH=$PATH:$GOROOT/bin:$GOPATH:bin
# source /etc/profile
# ./build.sh
```

编译完成的二进制文件在当前目录下的 bin 文件下

使用 strip 命令删除调试信息

```
[root@localhost mongo-tools]# vim build.sh
[root@localhost mongo-tools]# cd bin/
[root@localhost bin]# ls
bsondump  mongodump  mongoexport  mongofiles  mongoimport  mongoreplay  mongorestore  mongostat  mongotop
[root@localhost bin]# ll
total 168000
-rwxr-xr-x 1 root root 14891208 Oct 13 14:11 bsondump
-rwxr-xr-x 1 root root 20327896 Oct 13 14:11 mongodump
-rwxr-xr-x 1 root root 20071688 Oct 13 14:11 mongoexport
-rwxr-xr-x 1 root root 19994440 Oct 13 14:11 mongofiles
-rwxr-xr-x 1 root root 20247840 Oct 13 14:11 mongoimport
-rwxr-xr-x 1 root root 16373056 Oct 13 14:11 mongoreplay
-rwxr-xr-x 1 root root 20730416 Oct 13 14:11 mongorestore
-rwxr-xr-x 1 root root 19901992 Oct 13 14:11 mongostat
-rwxr-xr-x 1 root root 19472432 Oct 13 14:11 mongotop
[root@localhost bin]# strip *
[root@localhost bin]# ll
total 123076
-rwxr-xr-x 1 root root 10531128 Oct 13 14:12 bsondump
-rwxr-xr-x 1 root root 14783624 Oct 13 14:12 mongodump
-rwxr-xr-x 1 root root 14586504 Oct 13 14:12 mongoexport
-rwxr-xr-x 1 root root 14520872 Oct 13 14:12 mongofiles
-rwxr-xr-x 1 root root 14719496 Oct 13 14:12 mongoimport
-rwxr-xr-x 1 root root 13151816 Oct 13 14:12 mongoreplay
-rwxr-xr-x 1 root root 15111656 Oct 13 14:12 mongorestore
-rwxr-xr-x 1 root root 14471656 Oct 13 14:12 mongostat
-rwxr-xr-x 1 root root 14127112 Oct 13 14:12 mongotop
```

安装验证

由于编译的时候使用 gcc-9.3，所有如果编译安装和使用不在一个机器的话，需要把 gcc 的库文件（libstdc++.so.6.0.28）更换成编译 gcc-9.3 时生成的，路径为/usr/lib64 下，更换 libstdc++.so.6 软连接的指向到 libstdc++.so.6.0.28

启动 mongodb

```
# ./mongod
```

```
[root@localhost bin]# ./mongod
2022-10-18T17:56:59.478+0800 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProt
ls 'none'
2022-10-18T17:56:59.480+0800 W  ASIO     [main] No TransportLayer configured during NetworkInterface startup
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] MongoDB starting : pid=3023 port=27017 dbpath=/data/db 64-bit host=localh
.localdomain
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] db version v4.2.6
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] git version: nogitversion
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.1.1f  31 Mar 2020
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] allocator: tcmalloc
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] modules: none
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] build environment:
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten]     distarch: aarch64
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten]     target_arch: aarch64
2022-10-18T17:56:59.481+0800 I  CONTROL  [initandlisten] options: {}
2022-10-18T17:56:59.482+0800 I  STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=3914M,cache_overflow=(file_max=
,session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=
rnal,compressor=snappy),file_manager=(close_idle_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait
,verbose=[recovery_progress,checkpoint_progress],
2022-10-18T17:56:59.615+0800 I  STORAGE  [initandlisten] WiredTiger message [1666087019:615781][3023:0xfffefb050040], txn-recover:
t global recovery timestamp: (0, 0)
2022-10-18T17:56:59.714+0800 I  RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2022-10-18T17:56:59.797+0800 I  STORAGE  [initandlisten] Timestamp monitor starting
2022-10-18T17:56:59.832+0800 I  CONTROL  [initandlisten]
```

```
2022-10-18T17:57:00.274+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: inserted 0 keys from external sorter into index in
 0 seconds
2022-10-18T17:57:00.290+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: done building index lsidTTLIndex on ns config.syst
em.sessions
2022-10-18T17:57:00.340+0800 I  COMMAND  [LogicalSessionCacheRefresh] command config.system.sessions command: createIndexes { createI
ndexes: "system.sessions", indexes: [ { key: { lastUse: 1 }, name: "lsidTTLIndex", expireAfterSeconds: 1800 } ], $db: "config" } numY
ields:0 reslen:114 locks:{ ParallelBatchWriterMode: { acquireCount: { r: 2 } }, ReplicationStateTransition: { acquireCount: { w: 3 }
}, Global: { acquireCount: { r: 1, w: 2 } }, Database: { acquireCount: { r: 1, w: 2, W: 1 } }, Collection: { acquireCount: { r: 4, w:
 1, R: 1, W: 2 } }, Mutex: { acquireCount: { r: 3 } } } flowControl:{ acquireCount: 1, timeAcquiringMicros: 1 } storage:{} protocol:o
p_msg 280ms
2022-10-18T17:57:01.000+0800 I  SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```

3.5.15 如何安装 mesos-1.8.1

3.5.15.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1/SP2）

适用架构：ARM64、X86

其他版本或架构可做参考。

3.5.15.2 解决方案

源码获取

```
https://github.com/apache/mesos/archive/refs/tags/1.8.1.tar.gz
```

编译步骤

此处以 V10-SP1-0518-X86_64 为例：

配置 maven

SP2：使用系统自带的 maven 软件包

SP1：获取 maven 软件包

```
#
wget https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz --no-check-certificate
```

把 maven 放在/opt 目录下，并解压

```
# tar -zxf apache-maven-3.6.3-bin.tar.gz
```

配置 maven 环境变量

```
# vim /etc/profile
```

```
MAVEN_HOME=/opt/apache-maven-3.6.3
export PATH=$MAVEN_HOME/bin:$PATH
```

生效环境变量

```
# source /etc/profile
```

把 maven 配置国内源

```
# vim /opt/apache-maven-3.6.3/conf/settings.xml
```

```
146   <mirrors>
147     <!-- mirror
148      | Specifies a repository mirror site to use instead of a given repository. The reposito
    ry that
149      | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs ar
    e used
150      | for inheritance and direct lookup purposes, and must be unique across the set of mirr
    ors.
151      |
152      -->
153     <mirror>
154       <id>nexus-aliyun</id>
155       <mirrorOf>central</mirrorOf>
156       <name>Nexus aliyun</name>
157       <url>http://maven.aliyun.com/nexus/content/groups/public</url>
158     </mirror>
159   </mirrors>
160
161   <!-- profiles
162    | This is a list of profiles which can be activated in a variety of ways, and which can m
    odify
```

安装依赖

```
# yum install python3-devel python3-six  java-1.8.0-openjdk-devel zlib-devel openssl-devel  cyrus-sasl-md5  apr-util-devel  wget  apr-devel  libidn-devel
```

```
apr-util libcurl-devel cyrus-sasl cyrus-sasl-devel subversion-devel subversion
python3-pytz python3-google-apputils
```

解压 mesos 软件包

```
# tar -zxf mesos-1.8.1.tar.gz
```

创建安装目录

```
# mkdir /opt/mesos-bin
```

编译安装

```
# cd mesos-1.8.1
# ./bootstrap
# ./configure --prefix=/opt/mesos-bin
# make -j `nproc`
# make install
```

安装后的文件在/opt/mesos-bin 目录下

安装验证

上传二进制文件至/opt 目录下并解压

复制 mesos-master-env.sh.template 配置文件并进行修改

```
# cd /opt/mesos-bin/etc/mesos/
# cp mesos-master-env.sh.template mesos-master-env.sh
# vim mesos-master-env.sh
```

修改如下两句：

```
export MESOS_log_dir=/var/log/mesos
export MESOS_work_dir=/opt/mesos-bin/data
```

修改 mesos-slave-env.sh.template 配置文件

```
# vim mesos-slave-env.sh.template
```

修改 MESOS_master

```
export MESOS_master=ip:5050
 (其中 ip 为本机 ip)
```

对主机配置文件/etc/hosts 添加主节点

```
# vim /etc/hosts
```

添加如下内容：

```
IP master
```

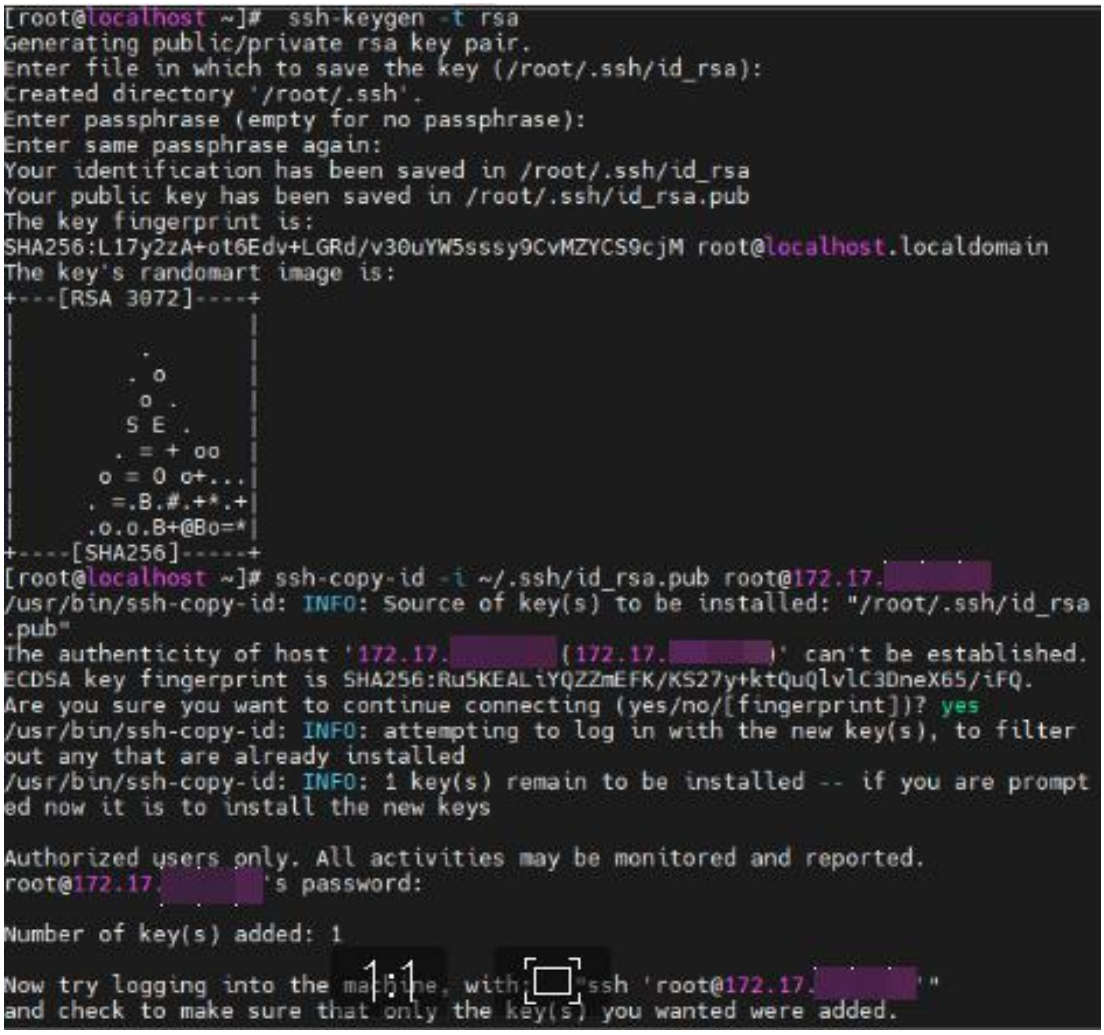
（其中 master 为电脑名）

在/opt/mesos-bin/etc/mesos/目录下编辑主节点配置文件

```
# vim masters
```

对系统进行免密操作

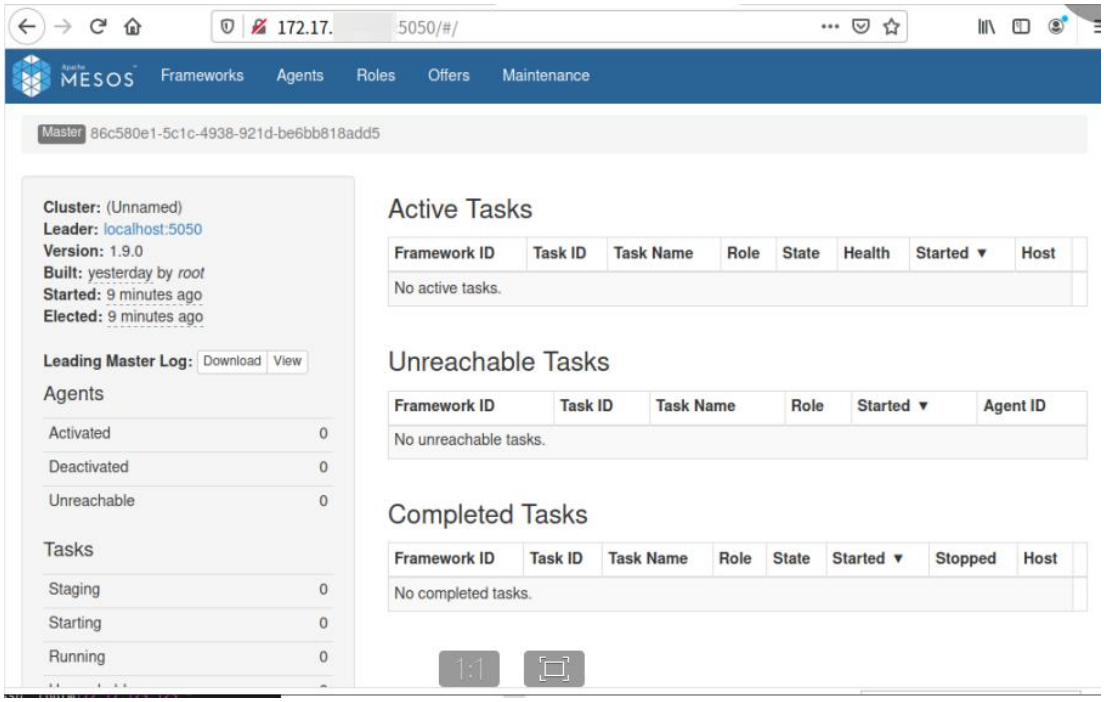
```
# ssh-keygen -t rsa
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@172.17.xxx.xxx
```



启动/opt/mesos-bin/sbin/mesos-start-masters.sh

```
# ./mesos-start-masters.sh
```

打开浏览器访问 ip:5050



3.5.16 如何安装 canu-1.8

3.5.16.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP2）

适用架构：AARCH64

其他版本或架构可做参考。

3.5.16.2 解决方案

源码获取

```
https://github.com/marbl/canu/archive/refs/tags/v1.8.tar.gz
http://downloads.sourceforge.net/project/boost/boost/1.58.0/boost_1_58_0.tar.gz
```

编译步骤

经验证在 V10-SP1-0518-aarch64 架构下，可以使用系统自带的 boost 进行编译安装，

在 V10-SP2-0524-aarch64 下，由于使用的 boost 版本为 1.7.3，导致编译失败，需要更换低版本

此处使用 boost-1.5.8

源码编译安装 boost-1.58 版本

```
# tar -zxf boost_1_58_0.tar.gz
# cd boost_1_58_0
# sed -ri 's/\-m64/\-mabi=lp64/g' `grep -Rl '\-m64'`
# ./bootstrap.sh
# ./b2
# ./b2 install
# mkdir -p /usr/include/boost
# cp -rf /usr/local/include/boost/* /usr/include/boost
# cp -rf /usr/local/lib/* /usr/lib
# ldconfig /usr/local/lib
```

至此 boost-1.58.0 版本安装完成

canu-1.8 编译安装

```
# tar -zxf canu-1.8.tar.gz
# cd canu-1.8/src
# make -j 8
# make install
```

```
#4/obj/bin/fastqSimulate-sort/fastq-utilities/fastqSimulate-sort.o -lcanu
 g++ -o /opt/canu-1.8/Linux-aarch64/bin/utgcns -D_GLIBCXX_PARALLEL -pthread -fope
nmp -lm -L/opt/canu-1.8/Linux-aarch64/lib /opt/canu-1.8/Linux-aarch64/obj/bin/ut
gcns/utgcns/utgcns.o /opt/canu-1.8/Linux-aarch64/obj/bin/utgcns/utgcns/stashCont
ains.o -lcanu

Success!
canu installed in /opt/canu-1.8/Linux-aarch64/bin/canu
```

安装验证

查看帮助

```
[root@localhost bin]# /opt/canu-1.8/Linux-aarch64/bin/canu

usage:   canu [-version] [-citation] \
              [-correct | -trim | -assemble | -trim-assemble] \
              [-s <assembly-specifications-file>] \
               -p <assembly-prefix> \
               -d <assembly-directory> \
               genomeSize=<number>[g|m|k] \
              [other-options] \
              [-pacbio-raw |
               -pacbio-corrected |
               -nanopore-raw |
               -nanopore-corrected] file1 file2 ...

example: canu -d run1 -p godzilla genomeSize=1g -nanopore-raw reads/*.fasta.gz
```

3.5.17 如何安装 glibc-2.29

3.5.17.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP2）

适用架构：AARCH64

其他版本或架构可做参考。

3.5.17.2 解决方案

源码获取

```
https://ftp.gnu.org/gnu/glibc/glibc-2.29.tar.xz
# yum install gcc binutils texinfo gawk bison sed pexpect gdb gettext perl
```

下载源码包到/opt 下

解压

```
# tar -zxf glibc-2.29.tar.gz
# cd glibc-2.29
# mkdir -p /opt/glibc-install      ---编译后的安装目录
# mkdir build
# cd build
```

修改文件

```
# vim /opt/glibc-2.29/nss/makedb.c
```

```
37 #include <sys/param.h>
38 #include <sys/stat.h>
39 #include <sys/uio.h>
40 #include "nss_db/nss_db.h"
41 #include <libc-diag.h>
42
43 /* Get libc version number.  */
44 #include "../version.h"
45
46 /* The hashing function we use.  */
47 #include "../intl/hash-string.h"
48
49 /* SELinux support.  */
```

```
841 }
842
843
844 #ifdef HAVE_SELINUX
845
846 DIAG_PUSH_NEEDS_COMMENT
847 DIAG_IGNORE_NEEDS_COMMENT (10, "-Wdeprecated-declarations");
848
849 static void
850 set_file_creation_context (const char *outname, mode_t mode)
851 {
852   static int enabled;
853   static int enforcing;
854   security_context_t ctx;
855
856   /* Check if SELinux is enabled, and remember. */
857   if (enabled == 0)
858     enabled = is_selinux_enabled () ? 1 : -1;
859   if (enabled < 0)
860     return;
```

```
874
875       freecon (ctx);
876     }
877 }
878
879 DIAG_POP_NEEDS_COMMENT
880
881
882 static void
883 reset_file_creation_context (void)
884 {
```

修改文件

```
# vim /opt/glibc-2.29/nscd/selinux.c
```

```
32 #include <selinux/selinux.h>
33 #ifdef HAVE_LIBAUDIT
34 # include <libaudit.h>
35 #endif
36 #include <libc-diag.h>
37
38 #include "dbg_log.h"
39 #include "selinux.h"
40
41
```

```
319   pthread_mutex_destroy (lock);
320   free (lock);
321 }
322
323 DIAG_PUSH_NEEDS_COMMENT
324 DIAG_IGNORE_NEEDS_COMMENT (10, "-Wdeprecated-declarations");
325
326 /* Initialize the user space access vector cache (AVC) for NSCD along with
327    log/thread/lock callbacks.  */
328 void
329 nscd_avc_init (void)
330 {
331   avc_entry_ref_init (&aeref);
337 #ifdef HAVE_LIBAUDIT
338   audit_init ();
339 #endif
340 }
341
342 DIAG_POP_NEEDS_COMMENT
343
344 DIAG_PUSH_NEEDS_COMMENT
345 DIAG_IGNORE_NEEDS_COMMENT (10, "-Wdeprecated-declarations");
346
347
348 /* Check the permission from the caller (via getpeercon) to nscd.
349    Returns 0 if access is allowed, 1 if denied, and -1 on error.
```

```
431   return rc;
432 }
433 DIAG_POP_NEEDS_COMMENT
434
435
436 /* Wrapper to get AVC statistics.  */
437 void
438 nscd_avc_cache_stats (struct avc_cache_stats *cstats)
439 {
440   avc_cache_stats (cstats);
441 }
```

编译

```
# ../configure --prefix=/opt/glibc-install
# make
```

```
# make install
```

安装完成后在/opt/glibc-install 目录下

安装验证

验证

```
[root@localhost glibc-install]# ls
bin  etc  include  lib  libexec  sbin  share  var
[root@localhost glibc-install]# strings lib/libc.so.6 | grep GLIBC_
GLIBC_2.17
GLIBC_2.18
GLIBC_2.22
GLIBC_2.23
GLIBC_2.24
GLIBC_2.25
GLIBC_2.26
GLIBC_2.27
GLIBC_2.28
GLIBC_2.29
GLIBC_PRIVATE
__create_module_GLIBC_2_0
```

3.5.18 如何安装 codis-3.2.2

3.5.18.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP2）

适用架构：AARCH64/X86

其他版本或架构可做参考。

3.5.18.2 解决方案

源码获取

```
https://github.com/CodisLabs/codis/archive/refs/tags/3.2.2.tar.gz
```

获取 golang 并放在/opt 目录下

```
https://dl.google.com/go/go1.9.2.linux-arm64.tar.gz
https://dl.google.com/go/go1.9.2.linux-amd64.tar.gz
```

解压 golang

```
# tar -zxf go1.9.2.linux-amd64.tar.gz
# tar -zxf go1.9.2.linux-arm64.tar.gz
```

设置 go 的环境变量

```
# export PATH=/opt/go/bin:$PATH
```

安装 java-1.8.0-openjdk-devel

```
# yum install java-1.8.0-openjdk-devel
# cd  /opt/go
```

创建 codis 源码存放路径：

```
# mkdir -p /opt/go/src/github.com/CodisLabs
```

把 codis 源码放在在以上路径

```
# cd /opt/go/src/github.com/CodisLabs
# wget https://github.com/CodisLabs/codis/archive/refs/tags/3.2.2.tar.gz
# tar -zxf codis-3.2.2.tar.gz
# mv codis-3.2.2 codis
# cd codis/
# make
```

编译成功后文件在 bin 目录下

保留图片

```
[root@localhost bin]# ll
总用量 90848
drwxr-xr-x 4 root root      117  8月  14 10:04 assets
-rwxr-xr-x 1 root root 14770016  8月  14 10:04 codis-admin
-rwxr-xr-x 1 root root 15784672  8月  14 10:03 codis-dashboard
-rwxr-xr-x 1 root root 14144440  8月  14 10:04 codis-fe
-rwxr-xr-x 1 root root 13044600  8月  14 10:04 codis-ha
-rwxr-xr-x 1 root root 17969272  8月  14 10:03 codis-proxy
-rwxr-xr-x 1 root root  5866296  8月  14 10:03 codis-server
-rwxr-xr-x 1 root root  2695648  8月  14 10:03 redis-benchmark
-rwxr-xr-x 1 root root  2859504  8月  14 10:03 redis-cli
-rwxr-xr-x 1 root root  5866296  8月  14 10:03 redis-sentinel
-rw-r--r-- 1 root root       96  8月  14 10:03 version
```

3.5.19 如何安装 kong-1.3.0

3.5.19.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP2）

适用架构：AARCH64

其他版本或架构可做参考。

3.5.19.2 解决方案

源码获取

```
https://  github.com/Kong/kong.git
```

安装依赖

```
# yum install zlib-devel libyaml-devel
```

配置 github 代理，另外由于 github 不在使用 git 拉取文件，

```
# vim .gitconfig      添加如下内容
```

```
[url "https: /https://github.com"]
insteadOf = git://github.com
```

获取 kong-build-tools（放在/opt 目录下）

```
# git clone   https://github.com/Kong/kong-build-tools.git
```

切换版本到 2.0.2

```
# cd kong-build-tools
# git checkout tags/2.0.2
```

在当前目录获取 openresty-build-tools 直接使用 master 版本

```
# git clone https://github.com/Kong/openresty-build-tools
# cd openresty-build-tools
# vim kong-ngx-build
```

由于里面的 pcre 的地址已经变更无法获取，可以下载后搭建本地 apache

```
if [ ! -z "$PCRE_VER" ]; then
  PCRE_DOWNLOAD=$DOWNLOAD_CACHE/pcre-$PCRE_VER
  if [ ! -d $PCRE_DOWNLOAD ]; then
    warn "PCRE source not found, downloading..."
    #curl -sSLO https://ftp.pcre.org/pub/pcre/pcre-${PCRE_VER}.tar.gz
    curl -sSLO http://10.41.      /test/pcre-${PCRE_VER}.tar.gz
    if [ ! -z ${PCRE_SHA+x} ]; then
      echo "$PCRE_SHA pcre-${PCRE_VER}.tar.gz" | sha256sum -c -
    fi
    tar -xzvf pcre-${PCRE_VER}.tar.gz
  fi
fi
```

pcre 获取地址：

```
https://sourceforge.net/projects/pcre/files/pcre/
```

```
    # lua-kong-nginx-module
    if [ $KONG_NGINX_MODULE != 0 ]; then
      pushd $DOWNLOAD_CACHE
        if [ ! -d lua-kong-nginx-module ]; then
          warn "lua-kong-nginx-module source not found, cloning..."
          git clone https://github.com/Kong/lua-kong-nginx-module
        fi

        pushd lua-kong-nginx-module
          git fetch
          git reset --hard $KONG_NGINX_MODULE || git reset --hard origin/$KONG_NGINX_MODULE
        popd
      popd
    fi
  popd
```

```
# cd /opt/kong-build-tools
```

创建安装目录

```
# mkdir -p /opt/kong-bin
```

安装依赖

```
# ./openresty-build-tools/kong-ngx-build --prefix /opt/kong-bin --work work --openresty 1.15.8.1 --openssl 1.1.1c --kong-nginx-module 0.0.6 --luarocks 3.1.3 --pcre 8.41 --jobs 16
```

```
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/type"
install -m 644 "src/luarocks/type/rockspec.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/type/rockspec.lua"
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks"
install -m 644 "src/luarocks/fun.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/fun.lua"
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks"
install -m 644 "src/luarocks/rockspecs.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/rockspecs.lua"
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks"
install -m 644 "src/luarocks/signing.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/signing.lua"
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks"
install -m 644 "src/luarocks/loader.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/loader.lua"
mkdir -p "//opt/kong-bin/luarocks/share/lua/5.1/luarocks"
install -m 644 "src/luarocks/path.lua" "//opt/kong-bin/luarocks/share/lua/5.1/luarocks/path.lua"
/opt/kong-build-tools
SUCCESS: LuaRocks 3.1.3 has been built successfully!
SUCCESS: Build finished in 100 seconds. Enjoy!
```

安装完成后会在 kong-bin 目录下产生文件

```
[root@localhost kong-build-tools]# cd ../kong-bin
[root@localhost kong-bin]# ls
luarocks  openresty  openssl
[root@localhost kong-bin]#
```

配置环境变量

```
# export KONG_DIR=/opt/kong-bin
# export OPENSSL_DIR=$KONG_DIR/openssl
# export PATH=$KONG_DIR/openresty/bin:$KONG_DIR/openresty/nginx/sbin:$OPENSSL_DIR/bin:$KONG_DIR/luarocks/bin:$PATH
```

查看版本信息

```
# openssl version -a
```

```
# nginx -V
# resty -v
# openresty -V
# luarocks --version
```

```
[root@localhost kong-bin]# openssl version -a
OpenSSL 1.1.1c  28 May 2019
built on: Sat May  6 04:42:39 2023 UTC
platform: linux-aarch64
options:  bn(64,64) rc4(char) des(int) idea(int) blowfish(ptr)
compiler: gcc -fPIC -Wa,--noexecstack -Wall -O3 -g -DOPENSSL_USE_NODELETE -DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOP
ENSSL_BN_ASM_MONT -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DKECCAK1600_ASM -DVPAES_ASM -DECP_NISTZ256_ASM -DPOLY130
5_ASM -DNDEBUG -DPURIFY
OPENSSLDIR: "/opt/kong-bin/openssl"
ENGINESDIR: "/opt/kong-bin/openssl/lib/engines-1.1"
Seeding source: os-specific
[root@localhost kong-bin]# nginx V
nginx: invalid option: "V"
[root@localhost kong-bin]# nginx -V
nginx version: openresty/1.15.8.1
built by gcc 7.3.0 (GCC)
built with OpenSSL 1.1.1c  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/opt/kong-bin/openresty/nginx --with-cc-opt='-O2 -I/opt/kong-bin/openssl/include'
--add-module=../ngx_devel_kit-0.3.1rc1 --add-module=../echo-nginx-module-0.61 --add-module=../xss-nginx-module-0
.06 --add-module=../ngx_coolkit-0.2 --add-module=../set-misc-nginx-module-0.32 --add-module=../form-input-nginx-
module-0.12 --add-module=../encrypted-session-nginx-module-0.08 --add-module=../srcache-nginx-module-0.31 --add-
module=../ngx_lua-0.10.15 --add-module=../ngx_lua_upstream-0.07 --add-module=../headers-more-nginx-module-0.33 -
-add-module=../array-var-nginx-module-0.05 --add-module=../memc-nginx-module-0.19 --add-module=../redis2-nginx-m
odule-0.15 --add-module=../redis-nginx-module-0.3.7 --add-module=../rds-json-nginx-module-0.15 --add-module=../r
ds-csv-nginx-module-0.09 --add-module=../ngx_stream_lua-0.0.7 --with-ld-opt='-Wl,-rpath,/opt/kong-bin/openresty/
luajit/lib -L/opt/kong-bin/openssl/lib -Wl,-rpath,/opt/kong-bin/openssl/lib' --with-pcre-jit --with-http_ssl_mod
ule --with-http_realip_module --with-http_stub_status_module --with-http_v2_module --with-stream_realip_module -
-with-stream_ssl_preread_module --add-module=/opt/kong-build-tools/work/lua-kong-nginx-module --with-pcre=/opt/k
ong-build-tools/work/pcre-8.41 --with-stream --with-stream_ssl_module --with-stream_ssl_preread_module
[root@localhost kong-bin]# resty -V
resty 0.23
nginx version: openresty/1.15.8.1
built by gcc 7.3.0 (GCC)
built with OpenSSL 1.1.1c  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/opt/kong-bin/openresty/nginx --with-cc-opt='-O2 -I/opt/kong-bin/openssl/include'
--add-module=../ngx_devel_kit-0.3.1rc1 --add-module=../echo-nginx-module-0.61 --add-module=../xss-nginx-module-0
.06 --add-module=../ngx_coolkit-0.2 --add-module=../set-misc-nginx-module-0.32 --add-module=../form-input-nginx-
module-0.12 --add-module=../encrypted-session-nginx-module-0.08 --add-module=../srcache-nginx-module-0.31 --add-
module=../ngx_lua-0.10.15 --add-module=../ngx_lua_upstream-0.07 --add-module=../headers-more-nginx-module-0.33 -
-add-module=../array-var-nginx-module-0.05 --add-module=../memc-nginx-module-0.19 --add-module=../redis2-nginx-m
odule-0.15 --add-module=../redis-nginx-module-0.3.7 --add-module=../rds-json-nginx-module-0.15 --add-module=../r
ds-csv-nginx-module-0.09 --add-module=../ngx_stream_lua-0.0.7 --with-ld-opt='-Wl,-rpath,/opt/kong-bin/openresty/
luajit/lib -L/opt/kong-bin/openssl/lib -Wl,-rpath,/opt/kong-bin/openssl/lib' --with-pcre-jit --with-http_ssl_mod
ule --with-http_realip_module --with-http_stub_status_module --with-http_v2_module --with-stream_realip_module -
-with-stream_ssl_preread_module --add-module=/opt/kong-build-tools/work/lua-kong-nginx-module --with-pcre=/opt/k
ong-build-tools/work/pcre-8.41 --with-stream --with-stream_ssl_module --with-stream_ssl_preread_module
[root@localhost kong-bin]# openresty -V
nginx version: openresty/1.15.8.1
built by gcc 7.3.0 (GCC)
built with OpenSSL 1.1.1c  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/opt/kong-bin/openresty/nginx --with-cc-opt='-O2 -I/opt/kong-bin/openssl/include'
--add-module=../ngx_devel_kit-0.3.1rc1 --add-module=../echo-nginx-module-0.61 --add-module=../xss-nginx-module-0
.06 --add-module=../ngx_coolkit-0.2 --add-module=../set-misc-nginx-module-0.32 --add-module=../form-input-nginx-
module-0.12 --add-module=../encrypted-session-nginx-module-0.08 --add-module=../srcache-nginx-module-0.31 --add-
module=../ngx_lua-0.10.15 --add-module=../ngx_lua_upstream-0.07 --add-module=../headers-more-nginx-module-0.33 -
-add-module=../array-var-nginx-module-0.05 --add-module=../memc-nginx-module-0.19 --add-module=../redis2-nginx-m
odule-0.15 --add-module=../redis-nginx-module-0.3.7 --add-module=../rds-json-nginx-module-0.15 --add-module=../r
ds-csv-nginx-module-0.09 --add-module=../ngx_stream_lua-0.0.7 --with-ld-opt='-Wl,-rpath,/opt/kong-bin/openresty/
luajit/lib -L/opt/kong-bin/openssl/lib -Wl,-rpath,/opt/kong-bin/openssl/lib' --with-pcre-jit --with-http_ssl_mod
ule --with-http_realip_module --with-http_stub_status_module --with-http_v2_module --with-stream_realip_module -
-with-stream_ssl_preread_module --add-module=/opt/kong-build-tools/work/lua-kong-nginx-module --with-pcre=/opt/k
ong-build-tools/work/pcre-8.41 --with-stream --with-stream_ssl_module --with-stream_ssl_preread_module
[root@localhost kong-bin]# luarocks --version
/opt/kong-bin/luarocks/bin/luarocks 3.1.3
LuaRocks main command-line interface
```

获取 kong 源码

```
# cd /opt/kong-bin
# git clone https://ghproxy.com/https://github.com/Kong/kong.git
```

```
# cd kong
# git checkout tags/1.3.0
```

编译安装

```
# make install
```

```
kong 1.3.0-0 depends on kong-plugin-request-transformer ~> 1.2 (not installed)
Installing https://luarocks.org/kong-plugin-request-transformer-1.2.8-0.src.rock

kong-plugin-request-transformer 1.2.8-0 is now installed in /opt/kong-bin/luarocks (license: Apache 2.0)

kong 1.3.0-0 depends on kong-plugin-session ~> 2.1 (not installed)
Installing https://luarocks.org/kong-plugin-session-2.1.2-1.src.rock
Missing dependencies for kong-plugin-session 2.1.2-1:
   lua-resty-session 2.24 (not installed)

kong-plugin-session 2.1.2-1 depends on lua-resty-session 2.24 (not installed)
Installing https://luarocks.org/lua-resty-session-2.24-1.src.rock

lua-resty-session 2.24-1 is now installed in /opt/kong-bin/luarocks (license: BSD)

kong-plugin-session 2.1.2-1 is now installed in /opt/kong-bin/luarocks (license: Apache 2.0)

kong 1.3.0-0 is now installed in /opt/kong-bin/luarocks (license: Apache 2.0)
```

安装验证

设置环境变量

```
# export PATH=/opt/kong-bin/kong/bin:$PATH
#                                                                    export
LUA_PATH="/opt/kong-bin/luarocks/share/lua/5.1/?.lua;/opt/kong-bin
/kong/?/init.lua;;"
```

配置 postgresql 数据库

```
# yum instal postgresql postgresql-server
```

创建用户 postgres

```
# useradd postgres
# passwd postgres
```

切换用户并进入数据库

```
# su - postgres
```

```
[postgres@localhost ~]$ initdb -D data/pgsql
属于此数据库系统的文件宿主为用户 "postgres".
此用户也必须为服务器进程的宿主.
数据库簇将使用本地化语言 "zh_CN.UTF-8"进行初始化.
默认的数据库编码已经相应的设置为 "UTF8".
initdb: 无法为本地化语言环境"zh_CN.UTF-8"找到合适的文本搜索配置
缺省的文本搜索配置将会被设置到"simple"

禁止为数据页生成校验和.

创建目录 data/pgsql ... 成功
正在创建子目录 ... 成功
选择默认最大联接数 (max_connections) ... 100
选择默认共享缓冲区大小 (shared_buffers) ... 128MB
选择动态共享内存实现 ......posix
创建配置文件 ... 成功
正在运行自举脚本 ...成功
正在执行自举后初始化 ...成功
同步数据到磁盘...成功

警告:为本地连接启动了 "trust" 认证.
你可以通过编辑 pg_hba.conf 更改或你下次
行 initdb 时使用 -A或者--auth-local和--auth-host选项.

Success. You can now start the database server using:

    pg_ctl -D data/pgsql -l logfile start

[postgres@localhost ~]$ pg_ctl -D data/pgsql -l logfile start
等待服务器进程启动 .... 完成
服务器进程已经启动
```

postgresql 创建用户 kong1

```
[postgres@localhost ~]$ psql
psql (10.5)
输入 "help" 来获取帮助信息.

postgres=# CREATE USER kong1;
CREATE ROLE
postgres=# CREATE DATABASE kong OWNER kong1;
CREATE DATABASE
postgres=# ALTER USER kong1 WITH password 'kong1';
ALTER ROLE
postgres=# \q
```

切换回 root 用户

创建/etc/kong 文件夹

```
# mkdir /etc/kong
# cp /opt/kong-bin/kong/kong.conf.default /etc/kong/
# cd /etc/kong
# cp kong.conf.default kong.conf
```

修改配置文件 kong.conf，内容如下

```
database = postgres             # Determines which of PostgreSQL or Cassandra
                                 # this node will use as its datastore.
                                 # Accepted values are `postgres`,
                                 # `cassandra`, and `off`.

pg_host = 127.0.0.1             # Host of the Postgres server.
pg_port = 5432                  # Port of the Postgres server.
pg_timeout = 5000               # Defines the timeout (in ms), for connecting,
                                 # reading and writing.

pg_user = kong1                  # Postgres user.
pg_password =kong1                   # Postgres user's password.
pg_database = kong              # The database name to connect to.
```

初始化数据库

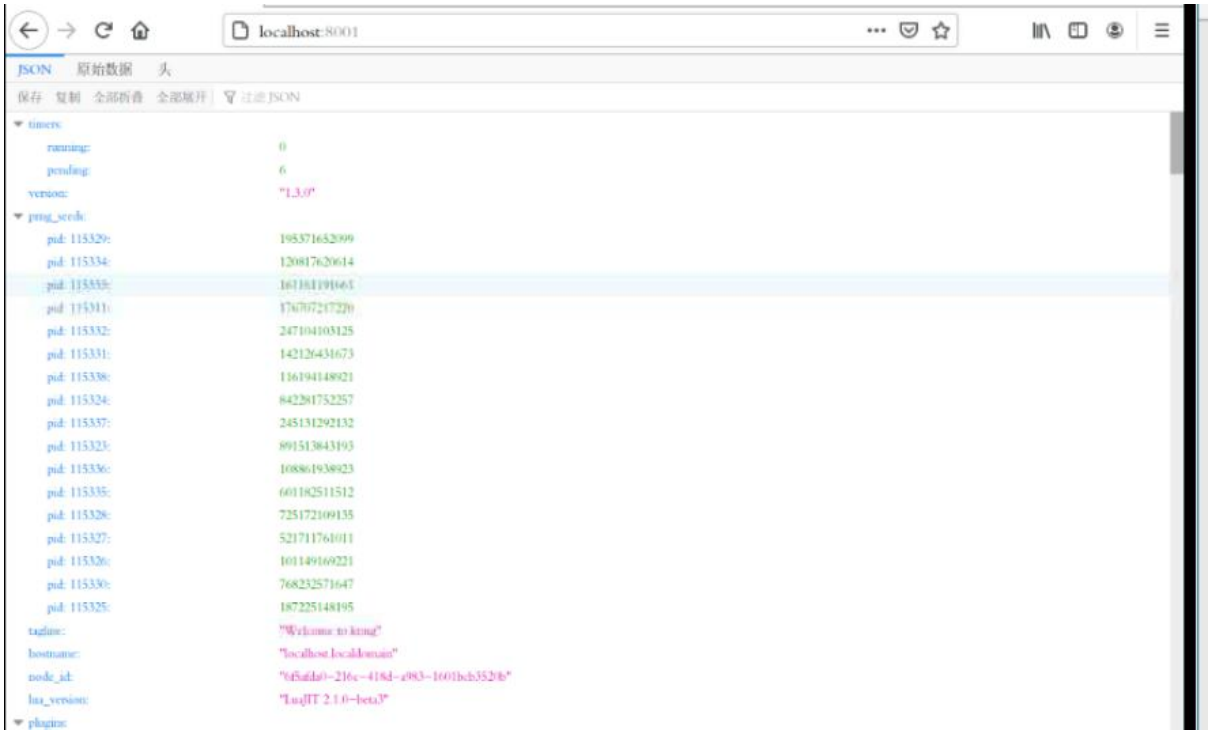
```
[root@localhost kong]# cp kong.conf.default /etc/kong/
[root@localhost kong]# pwd
/opt/kong-bin/kong
[root@localhost kong]# cd /etc/kong
[root@localhost kong]# cp kong.conf.default kong.conf
[root@localhost kong]# vim kong.conf
[root@localhost kong]# kong migrations bootstrap
Bootstrapping database...
migrating core on database 'kong'...
core migrated up to: 000_base (executed)
core migrated up to: 001_14_to_15 (executed)
core migrated up to: 002_15_to_1 (executed)
core migrated up to: 003_100_to_110 (executed)
core migrated up to: 004_110_to_120 (executed)
core migrated up to: 005_120_to_130 (executed)
migrating hmac-auth on database 'kong'...
hmac-auth migrated up to: 000_base_hmac_auth (executed)
hmac-auth migrated up to: 001_14_to_15 (executed)
migrating oauth2 on database 'kong'...
oauth2 migrated up to: 000_base_oauth2 (executed)
oauth2 migrated up to: 001_14_to_15 (executed)
oauth2 migrated up to: 002_15_to_10 (executed)
migrating rate-limiting on database 'kong'...
rate-limiting migrated up to: 000_base_rate_limiting (executed)
rate-limiting migrated up to: 001_14_to_15 (executed)
rate-limiting migrated up to: 002_15_to_10 (executed)
rate-limiting migrated up to: 003_10_to_112 (executed)
migrating jwt on database 'kong'...
jwt migrated up to: 000_base_jwt (executed)
jwt migrated up to: 001_14_to_15 (executed)
migrating key-auth on database 'kong'...
key-auth migrated up to: 000_base_key_auth (executed)
key-auth migrated up to: 001_14_to_15 (executed)
migrating session on database 'kong'...
session migrated up to: 000_base_session (executed)
migrating acl on database 'kong'...
acl migrated up to: 000_base_acl (executed)
acl migrated up to: 001_14_to_15 (executed)
migrating basic-auth on database 'kong'...
basic-auth migrated up to: 000_base_basic_auth (executed)
basic-auth migrated up to: 001_14_to_15 (executed)
migrating response-ratelimiting on database 'kong'...
response-ratelimiting migrated up to: 000_base_response_rate_limiting (executed)
response-ratelimiting migrated up to: 001_14_to_15 (executed)
response-ratelimiting migrated up to: 002_15_to_10 (executed)
27 migrations processed
27 executed
Database is up-to-date
```

启动 kong

```
# kong start
```

```
[root@localhost kong]# kong start
prefix directory /usr/local/kong not found, trying to create it
2023/05/06 13:42:56 [warn] ulimit is currently set to "1024". For better performance set it to at least "4096" using "ulimit -n"
Kong started
```

通过网页查看 localhost:8001



停止 kong

```
[root@localhost kong]# kong stop
Kong stopped
```

3.5.20 如何安装 sysak-1.3.0

3.5.20.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：X86

其他版本或架构可做参考。

3.5.20.2 解决方案

简介：

SysAK（System Analyse Kit）是龙蜥社区系统运维 SIG，通过对过往百万服务器运维经验进行抽象总结，而提供的一个全方位的系统运维工具集，

可以覆盖系统的日常监控、线上问题诊断和系统故障修复等常见运维场景。

下载源码

```
# git clone -b v1.3.0 https://gitee.com/anolis/sysak.git
```

安装依赖

```
# yum install elfutils-devel perf llvm
```

将源换成 sp2 的源，安装高版本 clang，sp1 自带 clang 版本过低，会编译报错

```
# vim /etc/yum.repos.d/kylin_x86_64.repo
```

```
[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl = http://update.cs2c.com.cn:8080/NS/V10/V10SP2/os/adv/lic/base/$basearch/
gpgcheck = 0
enabled = 1
```

```
# yum install clang
```

安装完后记得换回 sp1 的源

修改文件

```
# vim /root/sysak/source/tools/detect/mem/podmem/memcache/Makefile
```

修改第 3 行，删除-static-libstdc++参数

```
1 target = memcache
2 LIBS += libbpf.a -lelf -lz
3 #LDFLAGS += -Wall -static-libstdc++ -L./ $(LIBS)
4 LDFLAGS += -Wall -L./ $(LIBS)
5 mods := memread.o memcg.o offset.o
6 include $(SRC)/mk/cc.mk
```

```
# vim /root/sysak/rpm/sysak-build-nodep.sh
```

修改第 86 行，删除--enable-static 参数

```
79 main() {
80         export BASE=`pwd`
81         export RPM_VERSION=$1
82         export RELEASE=$2
83
84         export LINUX_VERSION=$(uname -r)
85
86         #TARGET_LIST="--enable-target-all --enable-static --disable-target-rtrace --disable-target-PingTrace"
87         TARGET_LIST="--enable-target-all --disable-target-rtrace --disable-target-PingTrace"
88
89         build_rpm
90 }
91
92 main $1 $2
```

编译打包

```
# cd /root/sysak/rpm
# ./sysak-build-nodep.sh 1.3.0 1
```

脚本后面两个参数分别为，版本号（1.3.0）和更新版本（1），需根据实际源码版本修改

编译完成的包位于/root/sysak/rpm/RPMS/x86_64/目录下

```
[root@localhost x86_64]# ls
sysak-1.3.0-1.ky10.x86_64.rpm
```

安装验证

```
# rpm -ivh sysak-1.3.0-1.ky10.x86_64.rpm
```

```
# systemctl start sysak.service
# systemctl status sysak.service
```

```
[root@localhost ~]# systemctl status sysak.service
● sysak.service - SysAK monitor service
   Loaded: loaded (/usr/lib/systemd/system/sysak.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2023-02-01 11:04:29 CST; 9s ago
 Main PID: 12597 (sysak)
    Tasks: 10
   Memory: 18.6M
   CGroup: /system.slice/sysak.service
           ├─12597 /usr/bin/sysak mservice -S &
           ├─12598 sh -c export SYSAK_WORK_PATH=/usr/local/sysak/.sysak_components;/usr/local/sysak/.sysak_components/tools/mservice "-S" "&"
           ├─12600 /usr/local/sysak/.sysak_components/tools/mservice -S &
           ├─12616 sysak runqslower -S sysak_mservice_jitter_shm 40
           ├─12617 sysak irqoff -S sysak_mservice_jitter_shm 10
           └─12618 sysak nosched -S sysak_mservice_jitter_shm -t 10

2月 01 11:04:29 localhost.localdomain systemd[1]: Started SysAK monitor service.
[root@localhost ~]#
```

3.5.21 如何安装 vernemq-1.11.0

3.5.21.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP3）

适用架构：X86

其他版本或架构可做参考。

3.5.21.2 解决方案

源码获取

```
# git clone https://ghproxy.com/https://github.com/vernemq/vernemq.git
```

切换版本

```
# cd vernemq/
# git checkout tags/1.11.0
```

编译步骤

安装依赖

```
# yum install erlang snappy-devel
```

其中 erlang 版本需要>=21.2

配置 github 代理，否则无法拉取软件包

```
#vim .gitconfig      添加如下内容

[url "https://ghproxy.com/https://github.com"]
        insteadOf = https://github.com

[url "https://ghproxy.com/https://github.com"]
        insteadOf = git://github.com
```

编译

```
# cd vernemq
# make rel
```

安装完成后目录在如下路径：

```
# /opt/vernemq/_build/default/rel/vernemq
```

```
[root@localhost vernemq]# ls
bin  data  erts-10.3.2  etc  lib  log  releases  share
```

安装验证

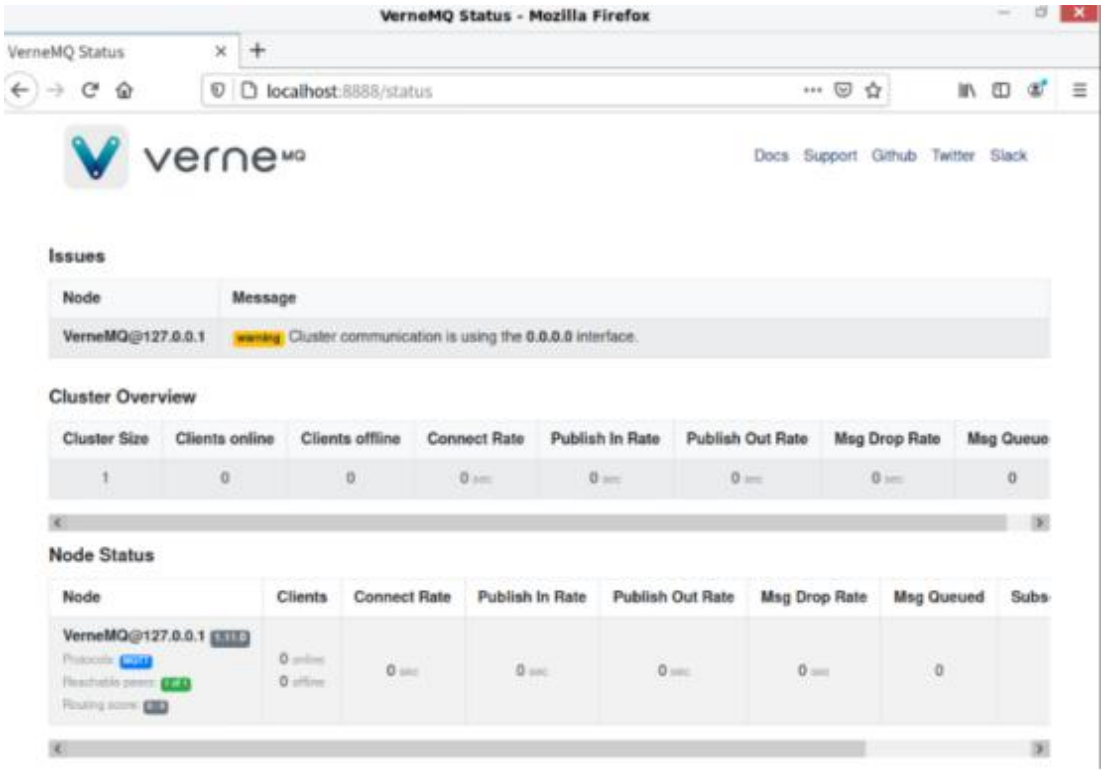
启动

```
# cd /opt/vernemq/_build/default/rel/vernemq
# ./bin/vernemq start
```

```
[root@localhost vernemq]# ./bin/vernemq start
!!!!
!!!! WARNING: ulimit -n is 1024; 65536 is the recommended minimum.
!!!!
vmq_cluster_node_sup
```

通过网页查看 status

访问 http://localhost:8888/status



停止：

```
# ./bin/vernemq stop
```

```
[root@localhost vernemq]# ./bin/vernemq stop
ok
=INFO REPORT==== 9-May-2023::16:23:35.113755 ===
Administrative stop
```

3.5.22 如何安装 emqx-4.4.0

3.5.22.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：X86、AARCH64

其他版本或架构可做参考。

3.5.22.2 解决方案

源码获取

```
# git clone https://github.com/emqx/emqx.git
# cd emqx/
# git checkout v4.4.0
```

编译步骤

此次以 V10-SP1-0518-aarch64 为例

查看依赖的 erlang 版本

```
# vim scripts/fail-on-old-otp-version.escript
```

```
#!/usr/bin/env escript

main(_) ->
    OtpRelease = list_to_integer(erlang:system_info(otp_release)),
    case OtpRelease < 21 of
        true ->
            io:format(standard_error, "ERROR: Erlang/OTP version ~p found. required_min=21, recommended=23~n" [OtpRelease]),
            halt(1);
        false ->
            ok
    end.
```

需要 erlang 版本最低 21，建议 23

此处使用 23 版本的

编译 erlang-23 软件包

使用 openEuler 中的 src.rpm 包编译打包

https://repo.openeuler.org/openEuler-22.03-LTS-SP2/source/Packages/erlang-23.3.4.9-1.oe2203sp2.src.rpm

编译完成后安装 erlang

配置 github 加速

```
# git config --global url."https://ghproxy.com/https://github.com".insteadOf "https://github.com"
```

修改文件（为了提高下载速度）

```
# vim scripts/ensure-rebar3.sh
```

```
VERSION="3.14.3-emqx-8"

# ensure dir
cd -P -- "$(dirname -- "${BASH_SOURCE[0]}")/.."

DOWNLOAD_URL='https://ghproxy.com/https://ghproxy.com/https://github.com/emqx/rebar3/releases/download'

download() {
    echo "downloading rebar3 ${VERSION}"
    curl -f -L "${DOWNLOAD_URL}/${VERSION}/rebar3" -o ./rebar3
}

version_gte() {
    test "$(printf '%s\n' "$1" "$2" | sort -V | head -n 1)" = "$2"
}

```

```
#vim scripts/get-dashboard.sh
```

```
RELEASE_ASSET_FILE="emqx-dashboard.zip"

if [ -f 'EMQX_ENTERPRISE' ]; then
    VERSION="${EMQX_EE_DASHBOARD_VERSION}"
    DASHBOARD_PATH='lib-ee/emqx_dashboard/priv'
    DASHBOARD_REPO='emqx-dashboard-web'
    DIRECT_DOWNLOAD_URL="https://ghproxy.com/https://github.com/emqx/${DASHBOARD_REPO}/releases/download/${VERSION}/${RELEASE_ASSET_FILE}"
else
    VERSION="${EMQX_CE_DASHBOARD_VERSION}"
    DASHBOARD_PATH='lib-ce/emqx_dashboard/priv'
    DASHBOARD_REPO='emqx-dashboard-frontend'
    DIRECT_DOWNLOAD_URL="https://ghproxy.com/https://github.com/emqx/${DASHBOARD_REPO}/releases/download/${VERSION}/${RELEASE_ASSET_FILE}"
fi

```

编译

```
make
```

编译完成的文件在如下路径： _build/emqx/rel/emqx

```
[root@localhost emqx]# ls
bin  data  erts-11.2.2.8  etc  lib  log  releases
```

安装验证

启动

```
# _build/emqx/rel/emqx/bin/emqx start
```

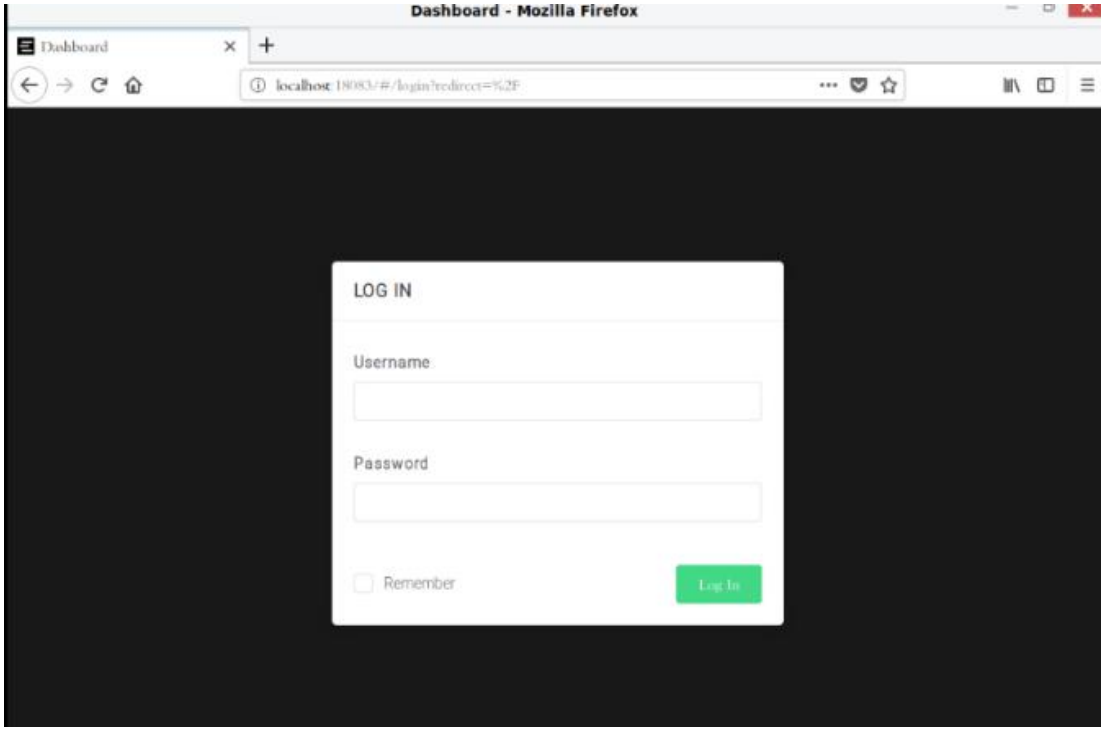
查看状态

```
# _build/emqx/rel/emqx/bin/emqx _ctl status
```

```
[root@localhost emqx]# _build/emqx/rel/emqx/bin/emqx start
EMQ X Broker 4.4.0 is started successfully!
[root@localhost emqx]# _build/emqx/rel/emqx/bin/emqx_ctl status
Node 'emqx@127.0.0.1' 4.4.0 is started
```

通过网页查看

访问 http://localhost:18083



停止

```
# _build/emqx/rel/emqx/bin/emqx stop
```

```
[root@localhost emqx]# _build/emqx/rel/emqx/bin/emqx stop
ok
```

3.5.23 如何安装 fastjson-1.2.79

3.5.23.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：X86、AARCH64

其他版本或架构可做参考。

3.5.23.2 解决方案

源码获取

fastjson 是阿里巴巴的开源 JSON 解析库，它可以解析 JSON 格式的字符串，支持将 Java Bean 序列化为 JSON 字符串，也可以从 JSON 字符串反序列化到 JavaBean。

fastjson 的优点：

速度快

fastjson 相对其他 JSON 库的特点是快，从 2011 年 fastjson 发布 1.1.x 版本之后，其性能从未被其他 Java 实现的 JSON 库超越。

使用广泛

fastjson 在阿里巴巴大规模使用，在数万台服务器上部署，fastjson 在业界被广泛接受。在 2012 年被开源中国评选为最受欢迎的国产开源软件之一。

```
# git clone https://gitee.com/mirrors/fastjson.git
# cd fastjson/
# git checkout 1.2.79
```

编译步骤

安装依赖

```
# yum install java-1.8.0-openjdk-devel
```

配置 maven

V10-SP1 由于系统没有自带 maven，需要从如下路径下载，然后配置环境变量

http://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz

V10-SP2&V10-SP3 可以直接使用 yum 安装 maven 软件

修改配置文件（只有 aarch64 需要，x86_64 不需要），避免使用的 jar 包中包含 x86 的相关文件

```
# vim pom.xml
```

```
                        </executions>
                    </plugin>
                </plugins>
            </build>
        </profile>

</profiles>
        <repositories>
    <repository>
        <id>kunpengmaven</id>
        <name>kunpeng maven</name>
        <url>https://mirrors.huaweicloud.com/kunpeng/maven</url>
    </repository>
</repositories>
</project>
```

编译打包

生成对应的 jar 包

```
# mvn clean package -Dmaven.test.skip=true
```

然后生成的 jar 包在 target 文件夹中

```
[root@localhost target]# ls
classes  fastjson-1.2.79.jar  fastjson-1.2.79-sources.jar  maven-archiver  maven-status
[root@localhost target]#
```

直接安装到本地

```
# mvn clean install -Dmaven.test.skip=true
```

```
[root@localhost 1.2.79]# find / -name fastjson-1.2.79.jar
/root/.m2/repository/com/alibaba/fastjson/1.2.79/fastjson-1.2.79.jar
/root/fastjson/target/fastjson-1.2.79.jar
[root@localhost 1.2.79]# pwd
/root/.m2/repository/com/alibaba/fastjson/1.2.79
[root@localhost 1.2.79]# ls
fastjson-1.2.79.jar  fastjson-1.2.79.pom  fastjson-1.2.79-sources.jar  _remote.repositories
```

安装验证

把 jar 包安装到本地仓库方法：（默认在本地~/.m2/repository 下）

```
# mvn install:install-file -Dfile=/root/fastjson/target/fastjson-1.2.79.jar -DgroupId=com.alibaba -DartifactId=fastjson -Dversion=1.2.79 -Dpackaging=jar
```

-Dfile　需要安装的文件路径

-DgroupId　安装在哪个 group 下面

-DartifactId　安装在哪个项目下面

-Dversion　jar 包版本

-Dpackageing　文件格式

```
[root@localhost target]# mvn install:install-file -Dfile=/root/fastjson/target/fastjson-1.2.79.jar -DgroupId=com.alibaba -DartifactId=fastjson -Dversion=1.2.79 -Dpackaging=jar
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ standalone-pom ---
[INFO] Installing /root/fastjson/target/fastjson-1.2.79.jar to /root/.m2/repository/com/alibaba/fastjson/1.2.79/fastjson-1.2.79.jar
[INFO] Installing /tmp/mvninstall4127740316644413478.pom to /root/.m2/repository/com/alibaba/fastjson/1.2.79/fastjson-1.2.79.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.520 s
[INFO] Finished at: 2023-09-01T16:13:08+08:00
[INFO] ------------------------------------------------------------------------
```

安装需要有项目，步骤参考如下：

https://github.com/alibaba/fastjson/wiki/Quick-Start-CN

3.5.24 如何安装 FATE-1.9.0

3.5.24.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP1）

适用架构：AARCH64

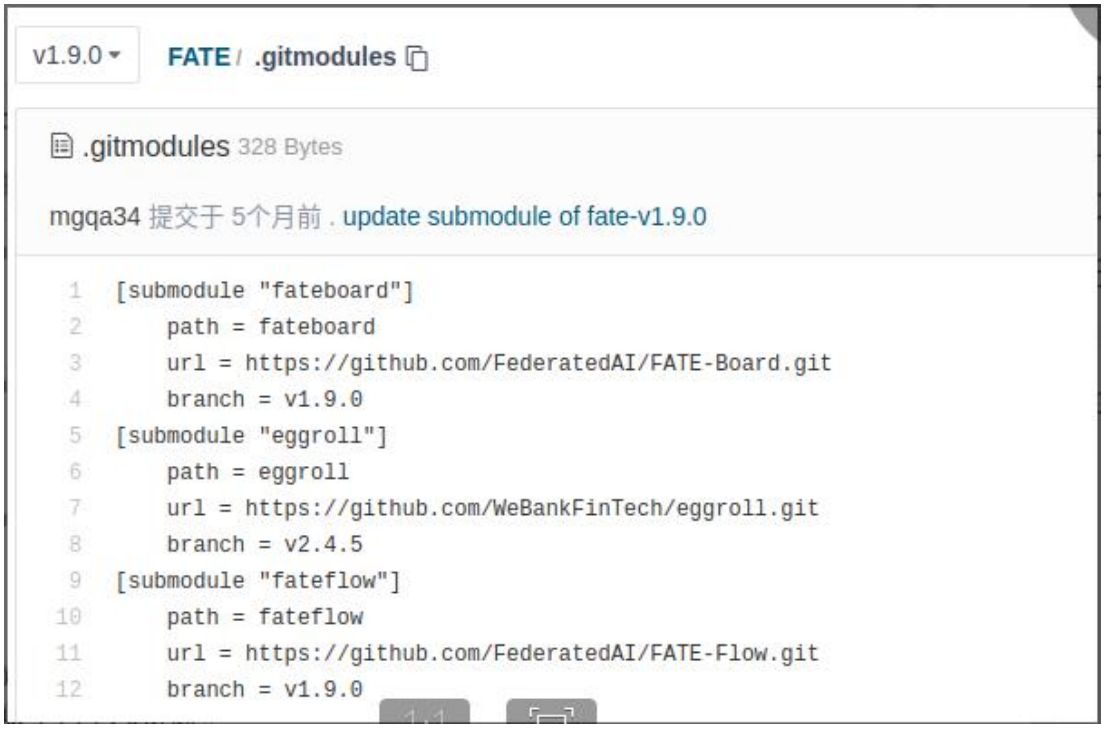
其他版本或架构可做参考。

3.5.24.2 解决方案

源码获取

由于 github 无法正常访问问题，使用 gitee 同步的代码

```
# git clone https://gitee.com/mirrors/FATE.git
# cd FATE
# git checkout tags/v1.9.0
```



申请 gitee 账号，同步上面 3 个子模块，把上面文件中的 github.com 的代码地址修改成自己 gitee 账号中同步的地址。

```
# git submodule sync
# git submodule update --init --recursive
```

```
[root@localhost FATE]# git submodule update --init --recursive
子模组 'eggroll' (https://gitee.com/sun-zhigang/eggroll.git) 已对路径 'eggroll' 注册
子模组 'fateboard' (https://gitee.com/sun-zhigang/FATE-Board.git) 已对路径 'fateboard' 注册
子模组 'fateflow' (https://gitee.com/sun-zhigang/FATE-Flow.git) 已对路径 'fateflow' 注册
正克隆到 '/root/FATE/eggroll'...
Username for 'https://gitee.com': 13986280463
Password for 'https://13986280463@gitee.com':
正克隆到 '/root/FATE/fateboard'...
Username for 'https://gitee.com': 13986280463
Password for 'https://13986280463@gitee.com':
正克隆到 '/root/FATE/fateflow'...
Username for 'https://gitee.com': 13986280463
Password for 'https://13986280463@gitee.com':
子模组路径 'eggroll': 检出 '7bf1abba6221ca13d8c071827dfa25b2d73362e6'
子模组路径 'fateboard': 检出 'c199bc311315d472a289fcb81adf2328e8233ef0'
子模组路径 'fateflow': 检出 '3afbc3e5d335ac96634eadfc493c4c697ecbfc19'
```

通过以上方法获取完整的源码

然后把源码包打包并命名为 FATE-1.9.0-with-submodules-src.tar.gz

编译步骤

获取所有 python 模块

获取官网做好的单机安装软件包

https://webank-ai-1251170195.cos.ap-guangzhou.myqcloud.co

m/fate/1.9.0/release/standalone_fate_install_1.9.0_release.tar.gz

解压此文件，在 standalone_fate_install_1.9.0_release/env/pypi 目录下可以看的需要 python 模块的名称和版本。

把这些模块在 kylin 系统上面做出来，其中部分由于各种原因更换版本，列表参考如下：

fate-1.9.0-python-list.et

做好的 python 模块文件夹重新命名为 whl-for-fate-1.9.0

离线安装

获取源码

把源码包 FATE-1.9.0-with-submodules-src.tar.gz 和 python 模块包 whl-for-fate-1.9.0 上次到对应主机

解压 FATE-1.9.0-with-submodules-src.tar.gz

```
# tar -zxf FATE-1.9.0-with-submodules-src.tar.gz
```

设置部署所需环境变量(注意，通过以下方式设置的环境变量仅在当前终端会话有效，若打开新的终端会话，如重新登录或者新窗口，请重新设置)

```
# cd FATE
# export FATE_PROJECT_BASE=$PWD
# export version=`grep "FATE=" ${FATE_PROJECT_BASE}/fate.env | awk -F "=" '{print $2}'`
```

```
[root@localhost FATE]# echo $version
1.9.0
[root@localhost FATE]#
```

部署前环境检查

本地 8080，9360，9380 端口是否被占用

```
# netstat -apln| grep 8080
# netstat -apln| grep 9360
# netstat -apln| grep 9380
```

安装 python 环境

使用系统自带的 python3.7.9

安装 python 环境

```
# yum install python3-devel python3 python3-pip python3-wheel
```

为 FATE 配置虚拟环境

```
# cd whl-for-fate-1.9.0
# pip3 install pip-23.0-py3-none-any.whl
virtualenv-20.16.2-py2.py3-none-any.whl
distlib-0.3.6-py2.py3-none-any.whl filelock-3.3.1-py3-none-any.whl
importlib_metadata-4.12.0-py3-none-any.whl
platformdirs-2.5.2-py3-none-any.whl
six-1.16.0-py2.py3-none-any.whl zipp-3.8.1-py3-none-any.whl
typing_extensions-4.3.0-py3-none-any.whl
```

创建虚拟环境的根目录

```
# mkdir /opt/virtualenv-python
# cd /opt/virtualenv-python
# python3 -m venv fate-test-for-kylin
# export
FATE_VENV_BASE=/opt/virtualenv-python/fate-test-for-kylin
# source ${FATE_VENV_BASE}/bin/activate
```

```
root@localhost opt]# mkdir /opt/virtualenv-python
root@localhost opt]# cd /opt/virtualenv-python/
root@localhost virtualenv-python]# python3 -m venv fate-test-for-kylin
root@localhost virtualenv-python]# ls
fate-test-for-kylin
root@localhost virtualenv-python]# export FATE_VENV_BASE=/opt/virtualenv-python/fate-test-for-kylin
root@localhost virtualenv-python]# source ${FATE_VENV_BASE}/bin/activate
fate-test-for-kylin) [root@localhost virtualenv-python]#
```

PS：其中 virtualenv-python 为放置虚拟环境的根目录，fate-test-for-kylin 为虚拟环境名称

安装 FATE 所需要的 python 依赖包

```
# cd ${FATE_PROJECT_BASE}
# bash bin/install_os_dependencies.sh
```

```
(fate-test-for-kylin) [root@localhost FATE]# bash bin/install_os_dependencies.sh
Kylin Linux Advanced Server
Not support this system.
```

修改 bin/install_os_dependencies.sh

```
#system=$(sed -e '/"/s/"//g' /etc/os-release | awk -F= '/^NAME/{print $2}')
system="CentOS Linux"
echo ${system}

function ln_lib() {
  local lib_name=$1
  echo "[INFO] deal $lib_name lib"
  so_name="lib$lib_name.so"
  cd /usr/lib/x86_64-linux-gnu
  if [ ! -f $so_name ] && [ ! -L $so_name ]; then
    so_file=$(ldd /usr/bin/openssl | grep $so_name | awk '{print $1}')
    if [[ -n $so_file ]]; then
      $command ln -s $so_file $so_name
      echo "[INFO] ln $so_file"
    else
      echo "[INFO] can not found openssl $so_name"
      #$command apt-get install -y libssl1.0.0 || echo ""
    fi
  fi
}
```

再次执行 bash bin/install_os_dependencies.sh

```
(fate-test-for-kylin) [root@localhost FATE]# bash bin/install_os_dependencies.sh
CentOS Linux
CentOS System
Last metadata expiration check: 0:30:04 ago on 2023年02月07日 星期二 09时37分14秒.
Package gcc-7.3.0-20190804.h30.ky10.aarch64 is already installed.
Package gcc-c++-7.3.0-20190804.h30.ky10.aarch64 is already installed.
Package make-1:4.2.1-15.ky10.aarch64 is already installed.
No match for argument: supervisor
Package libaio-0.3.111-5.p03.ky10.aarch64 is already installed.
Package autoconf-2.69-30.ky10.noarch is already installed.
Package automake-1.16.1-6.ky10.noarch is already installed.
Package libtool-2.4.6-32.ky10.aarch64 is already installed.
Package libffi-devel-3.3-7.ky10.aarch64 is already installed.
Package snappy-1.1.7-10.ky10.aarch64 is already installed.
Package zlib-1.2.11-17.1.ky10.aarch64 is already installed.
Package zlib-devel-1.2.11-17.1.ky10.aarch64 is already installed.
Package bzip2-1.0.8-3.ky10.aarch64 is already installed.
Package bzip2-devel-1.0.8-3.ky10.aarch64 is already installed.
Package lsof-4.93.2-3.ky10.aarch64 is already installed.
Error: Unable to find a match: supervisor
```

提示缺少依赖包

安装以下依赖包

python3-meld3-1.0.2-8.ky10.noarch.rpm

supervisor-4.0.4-3.ky10.noarch.rpm

获取路径：

http://172.30.12.238/kojifiles/packages/supervisor/4.0.4/3.ky10/noarch/supervisor-4.0.4-3.ky10.noarch.rpm

http://172.30.12.238/kojifiles/packages/python-meld3/1.0.2/8.ky10/noarch/python3-meld3-1.0.2-8.ky10.noarch.rpm

安装完依赖包后再次执行 bash bin/install_os_dependencies.sh

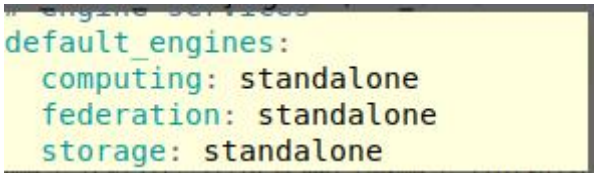
```
# source ${FATE_VENV_BASE}/bin/activate
# pip3 install /opt/whl-for-fate-1.9.0/pip-23.0-py3-none-any.whl
# pip install /opt/whl-for-fate-1.9.0/*.whl
```

配置 FATE

编辑 bin/init_env.sh 环境变量文件

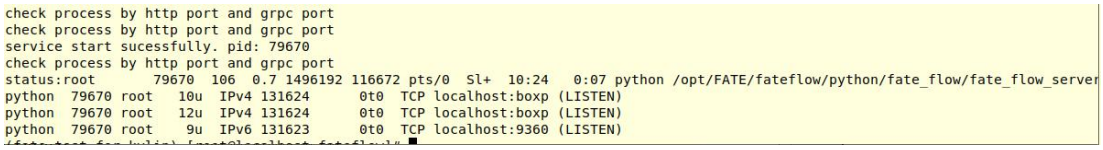
```
# cd ${FATE_PROJECT_BASE}
# sed -i.bak "s#PYTHONPATH=.*#PYTHONPATH=$PWD/python:$PWD/fateflow/python#g" bin/init_env.sh
# sed -i.bak "s#venv=.*#venv=${FATE_VENV_BASE}#g" bin/init_env.sh
```

检查 conf/service_conf.yaml 全局配置文件中是否将基础引擎配置为单机版，若 default_engines 显示如下，则为单机版



启动 fate flow server

```
# cd ${FATE_PROJECT_BASE}
# source bin/init_env.sh
# cd fateflow
# bash bin/service.sh status
# bash bin/service.sh start
```



安装 fate client

```
# cd ${FATE_PROJECT_BASE}
# source bin/init_env.sh
# cd python/fate_client/
# python setup.py install
```

初始化 fate flow client

```
# cd ../..
# flow init -c conf/service_conf.yaml
```

```
(fate-test-for-kylin) [root@localhost fate_client]# cd ../..
(fate-test-for-kylin) [root@localhost FATE]# flow init -c conf/service_conf.yaml
{
    "retcode": 0,
    "retmsg": "Fate Flow CLI has been initialized successfully."
}
```

Toy 测试

```
# flow test toy -gid 10000 -hid 10000
```

```
(fate-test-for-kylin) [root@localhost FATE]# flow test toy -gid 10000 -hid 10000
toy test job 202302071029312152160 is waiting
toy test job 202302071029312152160 is waiting
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is running
toy test job 202302071029312152160 is success
[INFO] [2023-02-07 10:29:37,830] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:96]: begin to init parameters of secure add example guest
[INFO] [2023-02-07 10:29:37,831] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:100]: begin to make guest data
[INFO] [2023-02-07 10:29:37,853] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:103]: split data into two random parts
[INFO] [2023-02-07 10:29:37,962] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:106]: share one random part data to host
[INFO] [2023-02-07 10:29:38,014] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:109]: get share of one random part data from host
[INFO] [2023-02-07 10:29:38,121] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:112]: begin to get sum of guest and host
[INFO] [2023-02-07 10:29:38,143] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:115]: receive host sum from guest
[INFO] [2023-02-07 10:29:38,151] [202302071029312152160] [80067:281466269097760] - [secure_add_guest.run] [line:122]: success to calculate secure sum, it is 2000.0
```

单元测试

```
# cd ${FATE_PROJECT_BASE}
# bash ./python/federatedml/test/run_test.sh
```

```
try to save session record for manager 55bb6ff4-a690-11ed-a360-5254005557c4, computing STANDALONE test_sample_weight_559d4560-a690-11ed-a360-5254005557c4
save session record for manager 55bb6ff4-a690-11ed-a360-5254005557c4, computing STANDALONE test_sample_weight_559d4560-a690-11ed-a360-5254005557c4 successfully
.
----------------------------------------------------------------------
Ran 4 tests in 1.486s

OK
there are 0 failed test
```

源码安装 fateboard

参 考 如 下 ：

https://gitee.com/sun-zhigang/FATE-Board/blob/v1.9.0/deploy/FATE-Board_deploy_guide_CN.md

获取并配置 maven

```
#
wget https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz --no-check-certificate
```

把 apache-maven-3.6.3-bin.tar.gz 放置在/opt 目录下

```
# tar -zxf apache-maven-3.6.3-bin.tar.gz
```

vim /etc/profile 添加如下内容：

```
# export MAVEN_HOME=/opt/apache-maven-3.6.3
# export PATH=$MAVEN_HOME/bin:$PATH
```

生效环境变量

```
# source /etc/profile
```

获取 fateboard 源码包

```
# git clone https://github.com/FederatedAI/FATE-Board.git
# cd FATE-Board
# git checkout tags/v1.9.0
```

编译打包

```
# cd FATE-Board
# yum install java-1.8.0-openjdk-devel
# mvn clean package -DskipTests
```

编译完成后把 FATE-Board 文件夹 copy 到 FATE 文件夹下面

整合 FATE 和 FATE-Board

```
# cd ${FATE_PROJECT_BASE}
# vim fateboard/bin/service.sh
```

修改内容如下：

```
basepath=$(
  cd $(dirname $0)
  pwd
)
configpath=$(
  #cd $basepath/conf
  cd $basepath/..
  pwd
)
fatepath=$(
  #cd $basepath/..
  cd $basepath/../..
  pwd
)

if test -f "${fatepath}/bin/init_env.sh";then
  source ${fatepath}/bin/init_env.sh
  echo "JAVA_HOME=$JAVA_HOME"
else
  echo "file not found:${fatepath}/bin/init_env.sh"
  exit
fi


#module=fateboard
module=fateboard-1.9.0

getpid() {
  #pid=$(ps -ef | grep java | grep ${basepath}/fateboard.jar | grep -v grep | awk '{print $2}')
  pid=$(ps -ef | grep java | grep fateboard-1.9.0.jar | grep -v grep | awk '{print $2}')

  if [[ -n ${pid} ]]; then
    return 1
```

```
start() {
  getpid
  if [[ $? -eq 0 ]]; then
    mklogsdir
    if [[ $1 == "front" ]]; then
      #exec $JAVA_HOME/bin/java -Dspring.config.location=$configpath/application.properties -Dssh_config_file=$basepath/ssh/ -Xmx2048m -Xms2048m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError -jar $basepath/${module}.jar >/dev/null 2>&1
      exec $JAVA_HOME/bin/java -Dspring.config.location=$configpath/src/main/resources/application.properties -Dssh_config_file=$fatepath/FATE-Board/src/main/resources/ -Xmx2048m -Xms2048m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError -jar $fatepath/FATE-Board/target/${module}.jar >/dev/null 2>&1
    else
      #nohup $JAVA_HOME/bin/java -Dspring.config.location=$configpath/application.properties -Dssh_config_file=$basepath/ssh/ -Xmx2048m -Xms2048m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError -jar $basepath/${module}.jar >/dev/null 2>&1 &
      nohup $JAVA_HOME/bin/java -Dspring.config.location=$configpath/src/main/resources/application.properties -Dssh_config_file=$fatepath/FATE-Board/src/main/resources/ -Xmx2048m -Xms2048m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError -jar $fatepath/FATE-Board/target/${module}.jar >/dev/null 2>&1 &
    fi
```

获取 java 路径：（版本可能不一样，但是方法一样）

```
# yum install java-1.8.0-openjdk-devel
```

```
(fate-test-for-kylin) [root@localhost FATE]# which java
/usr/bin/java
(fate-test-for-kylin) [root@localhost FATE]# ll /usr/bin/java
lrwxrwxrwx 1 root root 22  5月 16  2022 /usr/bin/java -> /etc/alternatives/java
(fate-test-for-kylin) [root@localhost FATE]# ll /etc/alternatives/java
lrwxrwxrwx 1 root root 76  5月 16  2022 /etc/alternatives/java -> /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-1.h5.ky10.aarch64/jre/bin/java
```

```
# cd ${FATE_PROJECT_BASE}
# vim bin/init_env.sh
```

添加 java 路径

```
venv=/opt/virtualenv-python/fate-test-for-kylin
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-1.h5.ky10.aarch64
export PATH=$PATH:$JAVA_HOME/bin
source ${venv}/bin/activate
```

启动 fateboard

```
# cd fateboard/bin
# bash service.sh status
```

```
(fate-test-for-kylin) [root@localhost bin]# bash service.sh status
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-1.h5.ky10.aarch64
service not running
```

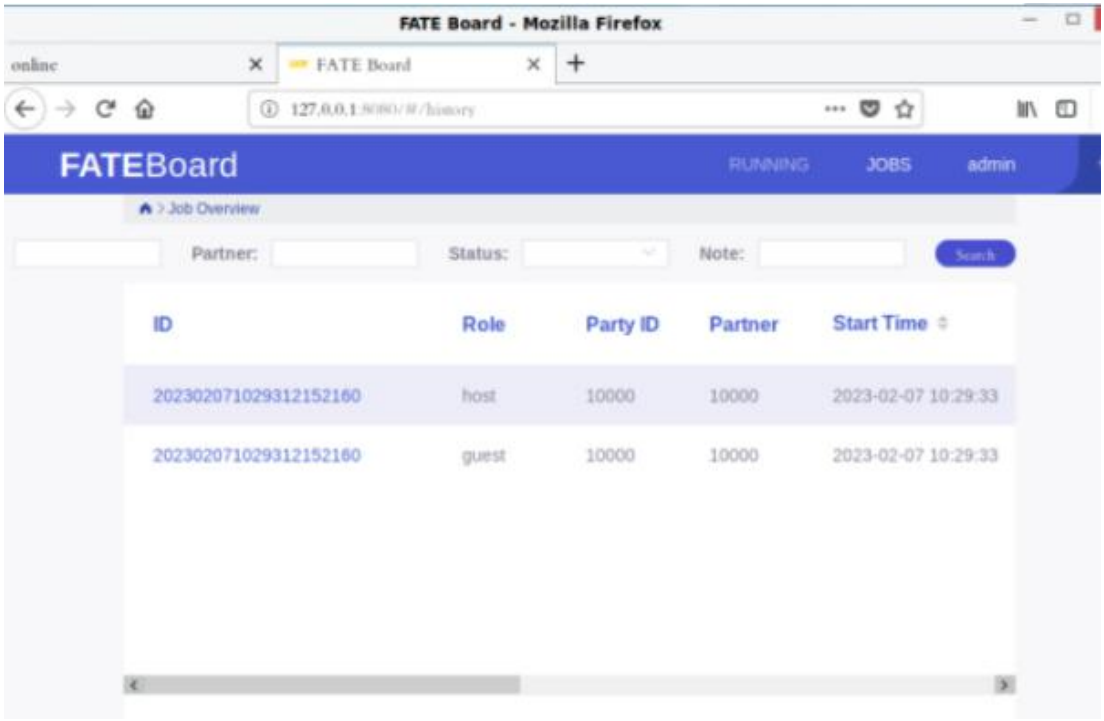
```
# bash service.sh start
```

```
(fate-test-for-kylin) [root@localhost bin]# bash service.sh start
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-1.h5.ky10.aarch64
service start sucessfully. pid: 86788
status:
        root       86788  334  2.3 5998080 356864 pts/0  Sl+  13:51   0:06 /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-1.h5.ky10.aarch64/bin/java -Dspring.config.location=/opt/FATE/fateboard/src/main/resources/application.properties -Dssh_config_file=/opt/FATE/FATE-Board/src/main/resources/ -Xmx2048m -Xms2048m -XX:+PrintGCDetails -XX:+PrintGCDateStamps -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError -jar /opt/FATE/FATE-Board/target/fateboard-1.9.0.jar
```

访问 fateboard

打开网页，输入地址：http://ip:8080,其中 ip 为 127.0.0.1 或本机时间 ip

账号密码均为 admin，由 FATE-Board/src/main/resources/application.properties 文件配置



关闭 fateboard

```
# bash service.sh stop
```

3.5.25 如何安装 tars-2.4.12

3.5.25.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP2）

适用架构：AARCH64

其他版本或架构可做参考。

3.5.25.2 解决方案

源码获取

```
# git clone -b v2.4.12 https://github.com/TarsCloud/TarsFramework.git --recursive
```

编译步骤

1.安装依赖

```
# yum install glibc-devel gcc gcc-c++ bison flex cmake which psmisc ncurses-devel zlib-devel psmisc telnet net-tools wget unzip mariadb
```

```
npm
```

2.进入源码目录进行编译安装

```
# cd TarsFramework
# mkdir build && cd build
# cmake ..
# make && make install
```

编译完成后会在/usr/local 下生成 tars 目录

```
[root@172-18-1-245 cpp]# pwd
/usr/local/tars/cpp
[root@172-18-1-245 cpp]# ls
deploy  include  lib  makefile  script  thirdparty  tools
[root@172-18-1-245 cpp]#
```

安装验证

1.配置 mariadb 数据库

```
#systemctl start mariadb

#mysql -uroot -p   //首次登陆没有密码
```

设置 mysql 密码

```
MariaDB [(none)] > ALTER USER 'root'@'localhost' IDENTIFIED BY '123123';
```

2.下载 TarsWeb 源码，并上传至/usr/local/tars/cpp/deploy 目录下，更改目录名为 web

```
#cd /usr/local/tars/cpp/deploy
#git clone -b v2.4.12 https://github.com/TarsCloud/TarsWeb.git
#mv TarsWeb web
```

3.安装 TarsWeb

```
#cd web
#npm install
#npm install pm2 -g
```

4.安装 tars

```
# ./tars-install.sh 127.0.0.1 123123 127.0.0.1 tars2015 enp3s0 root 3306 false false
```

参数从左到右分别为：数据库地址，数据库密码，安装 tars 地址，网卡名，

数据库端口，数据库用户，是否从节点，是否重建数据库

安装完成后会启动 3000 端口

5.网页访问 ip:3000

首次登陆设置密码

登录

3.5.26 如何安装 greenplum-6.20.0

3.5.26.1 系统版本

适用系统：银河麒麟高级服务器 V10（SP3）

适用架构：AARCH64

其他版本或架构可做参考。

3.5.26.2 解决方案

1.源码获取

http://github.com/greenplum-db/gpdb/releases/download/6.20.0/6.20.0-src-full.tar.gz

2.编译步骤

解压

```
# tar -zxf 6.20.0-src-full.tar.gz
# cd gpdb_src
```

查看 README.md 文件

需要 xerces 3.1 或 gp-xerces

```
## xerces

ORCA requires xerces 3.1 or gp-xerces. For the most up-to-date way of
building gp-xerces, see the README at the following repository:

* https://github.com/greenplum-db/gp-xerces

### Build the database
```

此处使用 xerces3.1

源码获取地址：

http://vault.centos.org/7.9.2009/os/Source/SPackages/xerces-c-3.1.1-10.el7_7.src.rpm

可以正常编译打包安装

```
# yum install xerces-c-3.1.1-10.ky10.ky10.aarch64.rpm xerces-c-devel-3.1.1-10.ky10.ky10.aarch64.rpm
```

r2ec（使用 koji 中的包）

https://kojipkgs.fedoraproject.org//packages/re2c/2.0.3/1.fc33/src/re2c-2.0.3-1.fc33.src.rpm

```
# rpm -ivh re2c-2.0.3-1.fc33.src.rpm
# yum-builddep ~/rpmbuild/SPECS/re2c.spec
# rpmbuild -ba ~/rpmbuild/SPECS/re2c.spec
```

查看依赖包需求：

通过 README.CentOS.bash 查看

```
# Install needed packages. Please add to this list if you discover additional prerequisites.
sudo yum install -y epel-release
sudo yum install -y \
    apr-devel \
    bison \
    bzip2-devel \
    cmake3 \
    flex \
    gcc \
    gcc-c++ \
    krb5-devel \
    libcurl-devel \
    libevent-devel \
    libkadm5 \
    libyaml-devel \
    libxml2-devel \
    libzstd-devel \
    openssl-devel \
    perl-ExtUtils-Embed \
    python-devel \
    python-pip \
    readline-devel \
    xerces-c-devel \
    zlib-devel

sudo pip install conan
sudo pip install -r python-dependencies.txt
sudo pip install -r python-developer-dependencies.txt
```

```
# yum install apr-devel bison bzip2-devel cmake3 flex gcc gcc-c++ krb5-devel libcurl-devel libevent-devel libkadm5 libyaml-devel libxml2-devel libzstd-devel openssl-devel perl-ExtUtils-Embed python2-devel python2-pip readline-devel xerces-c-devel zlib-devel python2-wheel
```

配置 pip 国内源

```
# pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
# pip install conan
# pip install -r python-dependencies.txt
```

报错：

解决方法：

编码错误，把 logilab-common-0.50.1.tar.gz 的源码包在 centos7 上编译成 whl 文件使用

编译方法如下：

安装 python-devel python-wheel

解压进入压缩包，执行 python setup.py bdist_wheel

parse-type 安装 0.3.4 版本

编译安装：

修改 aarch64 支持的指令

```
# vim gpdb_src/src/backend/gporca/libgpos/include/gpos/utils.h
```

```
18 #include "gpos/error/CException.h"
19 #include "gpos/io/COstreamBasic.h"
20 #include "gpos/types.h"
21
22 //#define GPOS_ASMFP asm volatile("movq %%rbp, %0" : "=g"(ulp));
23 //#define GPOS_ASMSP asm volatile("movq %%rsp, %0" : "=g"(ulp));
24 #define GPOS_ASMFP asm volatile("mov %0,fp" : "=g"(ulp));
25 #define GPOS_ASMSP asm volatile("mov %0,fp" : "=g"(ulp));
26
27 #define ALIGNED_16(x) \
28         (((ULONG_PTR) x >> 1) << 1 == (ULONG_PTR) x)  // checks 16-bit alignment
29 #define ALIGNED_32(x) \
30         (((ULONG_PTR) x >> 2) << 2 == (ULONG_PTR) x)  // checks 32-bit alignment
31 #define ALIGNED_64(x) \
32         (((ULONG_PTR) x >> 3) << 3 == (ULONG_PTR) x)  // checks 64-bit alignment
```

创建安装目录：

```
# mkdir /root/greenplum-6.20.0-bin

# ./configure --enable-orca --with-perl --with-python --with-libxml --with-gssapi
--prefix=/root/greenplum-6.20.0-bin

# make -j `nproc`
# make install
```

如果把编译好的二进制文件移到其他机器使用，需要安装软件包

xerces-c-3.1.1-10.ky10.ky10.aarch64.rpm

3.安装验证

关闭防火墙

```
# systemctl stop firewalld.service
# systemctl disable firewalld.service
```

创建组和用户

```
# groupadd gpadmin
# useradd -g gpadmin gpadmin
# passwd gpadmin
```

创建数据库的数据目录

```
# mkdir -p /data/gpdb/segdata
```

```
# mkdir -p /data/gpdb/master
# chown -R gpadmin:gpadmin /data/gpdb/segdata/
# chown -R gpadmin:gpadmin /data/gpdb/master/
```

修改主机名

```
# hostnamectl set-hostname master
# echo "IP  master" >> /etc/hosts      #IP 为主机实际 ip
```

切换到 gpadmin 用户

```
# su - gpadmin
```

配置环境变量

```
# vim .bash_profile
```

添加如下内容：

```
source /opt/greenplum-6.20.0-bin/greenplum_path.sh
export PGPORT=5432
export MASTER_DATA_DIRECTORY=/data/gpdb/master/gpseg-1
```

生效环境变量

```
# source .bash_profile
```

配置节点互信

```
# gpssh-exkeys -h 'master'
```

```
[gpadmin@master ~]$ gpssh-exkeys -h 'master'
[STEP 1 of 5] create local ID and authorize on local host

[STEP 2 of 5] keyscan all hosts and update known_hosts file

[STEP 3 of 5] retrieving credentials from remote hosts

[STEP 4 of 5] determine common authentication file content

[STEP 5 of 5] copy authentication files to all remote hosts

[INFO] completed successfully
```

切换到 root 用户，创建集群主机配置文件

```
# exit
# echo master > /data/gpdb/hostfile
```

设置打开文件数量

```
# ulimit -n 65535
```

切换到 gpadmin 用户，创建初始化配置文件 init.config

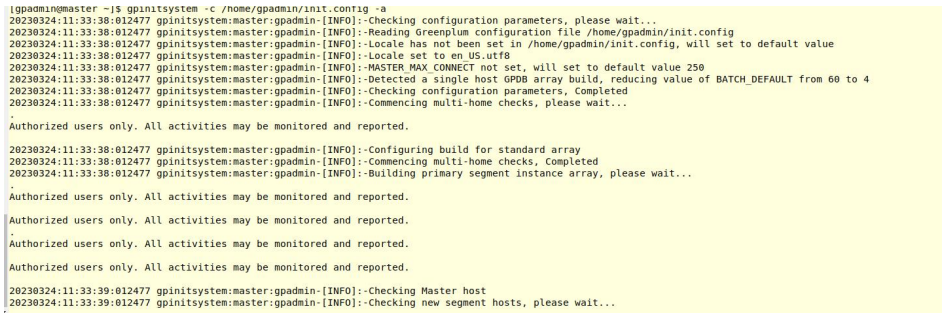
```
# su - gpadmin
# vim /home/gpadmin/init.config
```

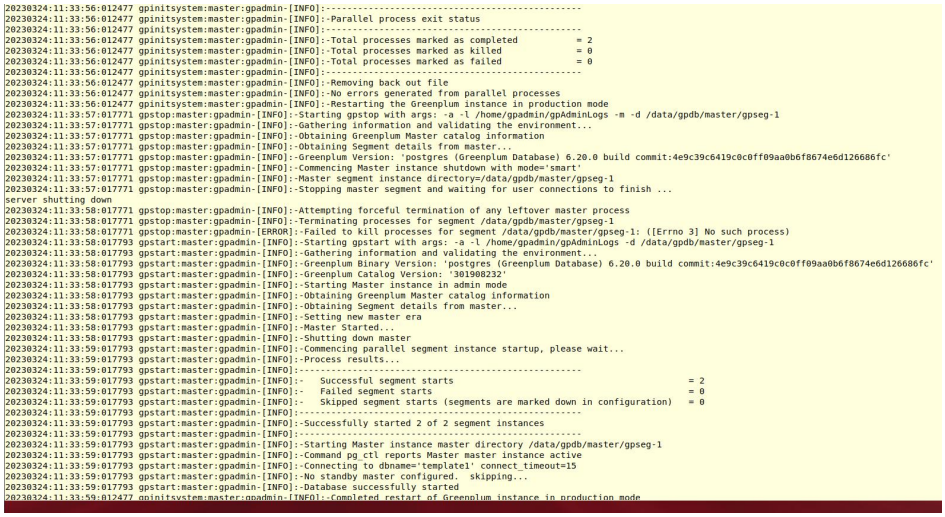
文件添加如下内容：

```
ARRAY_NAME="Greenplum Cluster"
SEG_PREFIX=gpseg #segment 数据库前缀名
PORT_BASE=40000 #segment 数据库起始端口号
declare -a DATA_DIRECTORY=(/data/gpdb/segdata /data/gpdb/segdata)
#segments数据目录,有几个DATA_DIRECTORY，每个节点上便会启动几个segment，本例为 2 个 segment
MASTER_HOSTNAME=master #master 主机名
MASTER_DIRECTORY=/data/gpdb/master #master 数据目录
MASTER_PORT=5432 #master 端口号，也是对外业务端口号
TRUSTED_SHELL=ssh
CHECK_POINT_SEGMENTS=8
ENCODING=UNICODE
DATABASE_NAME=gpdb
MACHINE_LIST_FILE=/data/gpdb/hostfile #集群配置文件，hostfile 对应上面创建的文件名
```

初始化数据库

```
# gpinitsystem -c /home/gpadmin/init.config -a
```





查询数据库进程

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

```
[gpadmin@master ~]$ ps -ef | grep postgres
gpadmin     17931       1  0 11:33 ?        00:00:00 /opt/greenplum-6.20.0-bin/bin/postgres -D /data/gpdb/segdata/gpseg1 -p 40001
gpadmin     17932       1  0 11:33 ?        00:00:00 /opt/greenplum-6.20.0-bin/bin/postgres -D /data/gpdb/segdata/gpseg0 -p 40000
gpadmin     17933   17932  0 11:33 ?        00:00:00 postgres: 40000, logger process
gpadmin     17934   17931  0 11:33 ?        00:00:00 postgres: 40001, logger process
gpadmin     17937   17932  0 11:33 ?        00:00:00 postgres: 40000, checkpointer process
gpadmin     17938   17932  0 11:33 ?        00:00:00 postgres: 40000, writer process
gpadmin     17939   17931  0 11:33 ?        00:00:00 postgres: 40001, checkpointer process
gpadmin     17940   17932  0 11:33 ?        00:00:00 postgres: 40000, wal writer process
gpadmin     17941   17931  0 11:33 ?        00:00:00 postgres: 40001, writer process
gpadmin     17942   17932  0 11:33 ?        00:00:00 postgres: 40000, stats collector process
gpadmin     17943   17931  0 11:33 ?        00:00:00 postgres: 40001, wal writer process
gpadmin     17944   17931  0 11:33 ?        00:00:00 postgres: 40001, stats collector process
gpadmin     17945   17932  0 11:33 ?        00:00:00 postgres: 40000, bgworker: sweeper process
gpadmin     17946   17931  0 11:33 ?        00:00:00 postgres: 40001, bgworker: sweeper process
gpadmin     17951       1  0 11:33 ?        00:00:00 /opt/greenplum-6.20.0-bin/bin/postgres -D /data/gpdb/master/gpseg-1 -p 5432 -E
gpadmin     17952   17951  0 11:33 ?        00:00:00 postgres:  5432, master logger process
gpadmin     17954   17951  0 11:33 ?        00:00:00 postgres:  5432, checkpointer process
gpadmin     17955   17951  0 11:33 ?        00:00:00 postgres:  5432, writer process
gpadmin     17956   17951  0 11:33 ?        00:00:00 postgres:  5432, wal writer process
gpadmin     17957   17951  0 11:33 ?        00:00:00 postgres:  5432, stats collector process
gpadmin     17958   17951  0 11:33 ?        00:00:00 postgres:  5432, bgworker: dtx recovery process
gpadmin     17959   17951  0 11:33 ?        00:00:00 postgres:  5432, bgworker: ftsprobe process
gpadmin     17966   17951  0 11:33 ?        00:00:00 postgres:  5432, bgworker: sweeper process
gpadmin     18381   10316  0 11:35 pts/0    00:00:00 grep postgres
```

运行数据库

停止数据库

```
# gpstop
```

```
[gpadmin@master ~]$ gpstop
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Starting gpstop with args:
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Gathering information and validating the environment...
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Obtaining Greenplum Master catalog information
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Obtaining Segment details from master...
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Greenplum Version: 'postgres (Greenplum Database) 6.20.0 build commit:4e9c39c6419c0c0ff09aa0b6f8674e6d126686fc'
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-------------------------------------------------
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Master instance parameters
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-------------------------------------------------
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Master Greenplum instance process active PID   = 17951
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Database                                       = template1
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Master port                                    = 5432
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Master directory                               = /data/gpdb/master/gpseg-1
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Shutdown mode                                  = smart
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Timeout                                        = 120
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Shutdown Master standby host                   = Off
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-------------------------------------------------
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-Segment instances that will be shutdown:
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-------------------------------------------------
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   Host     Datadir                 Port    Status
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   master   /data/gpdb/segdata/gpseg0   40000   u
20230324:11:35:30:018382 gpstop:master:gpadmin-[INFO]:-   master   /data/gpdb/segdata/gpseg1   40001   u

Continue with Greenplum instance shutdown Yy|Nn (default=N):
> y
20230324:11:35:39:018382 gpstop:master:gpadmin-[INFO]:-Commencing Master instance shutdown with mode='smart'
20230324:11:35:39:018382 gpstop:master:gpadmin-[INFO]:-Master segment instance directory=/data/gpdb/master/gpseg-1
20230324:11:35:39:018382 gpstop:master:gpadmin-[INFO]:-Stopping master segment and waiting for user connections to finish ...
server shutting down
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Attempting forceful termination of any leftover master process
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Terminating processes for segment /data/gpdb/master/gpseg-1
20230324:11:35:40:018382 gpstop:master:gpadmin-[ERROR]:-Failed to kill processes for segment /data/gpdb/master/gpseg-1: ([Errno 3] No such process)
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-No standby master host configured
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Targeting dbid [2, 3] for shutdown
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Commencing parallel segment instance shutdown, please wait...
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-0.00% of jobs completed
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-100.00% of jobs completed
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-----------------------------------------------------
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-   Segments stopped successfully      = 2
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-   Segments with errors during stop   = 0
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-----------------------------------------------------
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Successfully shutdown 2 of 2 segment instances
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Database successfully shutdown with no errors reported
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Cleaning up leftover gpmmon process
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-No leftover gpmmon process found
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Cleaning up leftover gpsmon processes
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-No leftover gpsmon processes on some hosts. not attempting forceful termination on these hosts
20230324:11:35:40:018382 gpstop:master:gpadmin-[INFO]:-Cleaning up leftover shared memory
```

启动数据库

```
[gpadmin@master ~]$ gpstart
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Starting gpstart with args:
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Gathering information and validating the environment...
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Greenplum Binary Version: 'postgres (Greenplum Database) 6.20.0 build commit:4e9c39c6419c0c0ff09aa0b6f8674e6d126686fc'
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Greenplum Catalog Version: '301908232'
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Starting Master instance in admin mode
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Obtaining Greenplum Master catalog information
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Obtaining Segment details from master...
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Setting new master era
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Master Started...
20230324:11:36:57:018705 gpstart:master:gpadmin-[INFO]:-Shutting down master
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:---------------------------
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Master instance parameters
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:---------------------------
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Database                 = template1
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Master Port              = 5432
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Master directory         = /data/gpdb/master/gpseg-1
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Timeout                  = 600 seconds
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Master standby           = Off
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:---------------------------------------
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-Segment instances that will be started
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:---------------------------------------
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-   Host     Datadir                 Port
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-   master   /data/gpdb/segdata/gpseg0   40000
20230324:11:36:58:018705 gpstart:master:gpadmin-[INFO]:-   master   /data/gpdb/segdata/gpseg1   40001

Continue with Greenplum instance startup Yy|Nn (default=N):
> y
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-Commencing parallel segment instance startup, please wait...
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-Process results...
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-----------------------------------------------------
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-   Successful segment starts                                            = 2
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-   Failed segment starts                                                = 0
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-   Skipped segment starts (segments are marked down in configuration)   = 0
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-----------------------------------------------------
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-Successfully started 2 of 2 segment instances
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-----------------------------------------------------
20230324:11:37:02:018705 gpstart:master:gpadmin-[INFO]:-Starting Master instance master directory /data/gpdb/master/gpseg-1
20230324:11:37:03:018705 gpstart:master:gpadmin-[INFO]:-Command pg_ctl reports Master master instance active
20230324:11:37:03:018705 gpstart:master:gpadmin-[INFO]:-Connecting to dbname='template1' connect_timeout=15
20230324:11:37:03:018705 gpstart:master:gpadmin-[INFO]:-No standby master configured.  skipping...
20230324:11:37:03:018705 gpstart:master:gpadmin-[INFO]:-Database successfully started
```

验证数据库

使用 gpadmin 用户登录数据库

```
# psql -d postgres
```

创建数据库

```
# create database isoft;
# \l
```

```
[gpadmin@master ~]$ psql -d postgres
psql (9.4.26)
Type "help" for help.

postgres=# create database isoft;
CREATE DATABASE
postgres=# \l
                                List of databases
   Name    |  Owner  | Encoding |  Collate   |   Ctype    |  Access privileges
-----------+---------+----------+------------+------------+---------------------
 gpdb      | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 isoft     | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
 template1 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
(5 rows)
```

创建用户

```
# create role isfot password 'isoft' createdb login;
# select rolname,oid from pg_roles;
```

```
postgres=# create role isfot password 'isoft' createdb login;
NOTICE:  resource queue required -- using default resource queue "pg_default"
CREATE ROLE
postgres=#
postgres=# select rolname,oid from pg_roles;
 rolname |  oid
---------+-------
 gpadmin |    10
 isfot   | 16386
(2 rows)
```

创建表

```
# create table isoftt(id int, name varchar(10));
# select * from isoftt;
```

```
postgres=# create table isoftt(id int, name varchar(10));
NOTICE:  Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'id' as the Greenplum Database data distribution key for this table.
HINT:  The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s) chosen are the optimal data distribution key to minimize skew.
CREATE TABLE
postgres=# select * from isoftt;
 id | name
----+------
(0 rows)
```

表的增删改查

```
postgres=# insert into isoftt values(1, 'isoftstone');
INSERT 0 1
postgres=# select * from isoftt;
 id |    name
----+------------
  1 | isoftstone
(1 row)

postgres=# delete from isoftt where id = 1;
DELETE 1
postgres=# select * from isoftt;
 id | name
----+------
(0 rows)

postgres=# insert into isoftt values(1, 'isoftstone');
INSERT 0 1
postgres=# select * from isoftt;
 id |    name
----+------------
  1 | isoftstone
(1 row)

postgres=# update isoftt set name='ISOFTSTONE' where id =1;
UPDATE 1
postgres=# select * from isoftt;
 id |    name
----+------------
  1 | ISOFTSTONE
(1 row)

postgres=# delete from isoftt where id = 1;
DELETE 1
postgres=# select * from isoftt;
 id | name
----+------
(0 rows)
```

3.6 虚拟机问题

3.6.1 如何扩展 swap 交换空间

3.6.1.1 系统版本

适用系统：V10(SP1)、V10(SP2)

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.6.1.2 解决方案

扩展 swap 空间

1) 查看内存空间

```
# free –h
```

```
[root@localhost mongo-4.2]# free -h
              total        used        free      shared  buff/cache   available
Mem:           14Gi       622Mi        13Gi       3.0Mi       394Mi        12Gi
Swap:         8.0Gi       401Mi       7.6Gi
```

2) 创建物理分区

```
# dd if=/dev/zero of=/mnt/swap bs=1024 count=40960000
```

注：of：物理分区位置，以实际路径为准

count：设置 swap 大小，以 KB 为单位，40960000 设置 swap 空间为 40G

bs=bytes：同时设置读入/输出的块大小为 bytes 个字节。

```
[root@localhost mongo-4.2]# dd if=/dev/zero of=/opt/swap bs=1024 count=40960000

记录了40960000+0 的读入
记录了40960000+0 的写出
41943040000字节（42 GB，39 GiB）已复制，243.841 s，172 MB/s
[root@localhost mongo-4.2]#
```

3) 启用交换分区文件

```
# swapon /mnt/swap
```

4) 再次查看内存

```
[root@localhost mongo-4.2]# free -h
              total        used        free      shared  buff/cache   available
Mem:           14Gi       672Mi       1.4Gi       8.0Mi        12Gi        11Gi
Swap:          47Gi       393Mi        46Gi
```

3.6.2 KVM 虚拟机如何进行磁盘扩容

3.6.2.1 系统版本

适用系统：V10（SP1）、V10（SP2）

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.6.2.2 问题描述

给 kvm 虚拟机扩容（原来 kylin10.0-1.qcow2 只有 100G，现在增加 100G，总容量 200G）。

3.6.2.3 问题分析

给虚拟机扩容时需要将其关机。

3.6.2.4 解决方案

1) 关闭虚拟机前先用 fdisk -l 看下虚拟机容量。

```
# fdisk -l
```

```
[root@localhost ~]# fdisk -l
Disk /dev/vda: 100 GiB, 107374182400 字节，209715200 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
磁盘标识符：0C98C52C-2DD5-4020-A204-1C858430C779

设备          起点      末尾      扇区  大小 类型
/dev/vda1     2048    411647    409600  200M EFI 系统
/dev/vda2   411648   2508799   2097152    1G Linux 文件系统
/dev/vda3  2508800 209713151 207204352 98.8G Linux LVM
```

2) 关闭物理机，在物理机上查看需要扩容的虚拟机硬盘容量信息。

```
# qemu-img info /var/lib/libvirt/images/kylin10.0-1.qcow2
```

```
[root@localhost images]# qemu-img info /var/lib/libvirt/images/kylin10.0-1.qcow2
image: /var/lib/libvirt/images/kylin10.0-1.qcow2
file format: qcow2
virtual size: 100 GiB (107374182400 bytes)
disk size: 104 GiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: true
    refcount bits: 16
    corrupt: false
```

3) 给虚拟硬盘扩容

```
# qemu-img resize /var/lib/libvirt/images/kylin10.0-1.qcow2 +100G
```

```
[root@localhost ~]# qemu-img resize /var/lib/libvirt/images/kylin10.0-1.qcow2 +100G
Image resized.
```

4) 再次查看当前虚拟机硬盘容量有没有增加

```
[root@localhost ~]# qemu-img info /var/lib/libvirt/images/kylin10.0-1.qcow2
image: /var/lib/libvirt/images/kylin10.0-1.qcow2
file format: qcow2
virtual size: 200 GiB (214748364800 bytes)
disk size: 104 GiB
cluster_size: 65536
Format specific information:
    compat: 1.1
    lazy refcounts: true
    refcount bits: 16
    corrupt: false
```

5) 开启虚拟机，进入虚拟机，用 fdisk -l 查看容量有没有增加

```
[root@localhost ~]# fdisk -l
GPT PMBR 大小不符 (209715199 != 419430399)，将用写入予以更正。
The backup GPT table is not on the end of the device. This problem will be corrected by write.
Disk /dev/vda: 200 GiB, 214748364800 字节，419430400 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
磁盘标识符：0C98C52C-2DD5-4020-A204-1C858430C779
```

出现以上报错，可以通过重新写入解决。

```
# fdisk /dev/vda
```

```
[root@localhost ~]# fdisk /dev/vda

欢迎使用 fdisk (util-linux 2.35.2)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。

GPT PMBR 大小不符 (209715199 != 419430399)，将用写入予以更正。
The backup GPT table is not on the end of the device. This problem will be corrected by write.

命令(输入 m 获取帮助): w

分区表已调整。
正在同步磁盘。

[root@localhost ~]# fdisk -l
Disk /dev/vda: 200 GiB, 214748364800 字节，419430400 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
磁盘标识符：0C98C52C-2DD5-4020-A204-1C858430C779

设备          起点      末尾      扇区  大小 类型
/dev/vda1     2048    411647    409600  200M EFI 系统
/dev/vda2   411648   2508799   2097152    1G Linux 文件系统
/dev/vda3  2508800 209713151 207204352 98.8G Linux LVM
```

6) 用 lsblk 查看分区情况，关键是找出需要扩容的分区（本文以 vda3 作为示例）

```
# lsblk
```

```
[root@localhost ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0              11:0    1 1024M  0 rom
vda             253:0    0  200G  0 disk
├─vda1          253:1    0  200M  0 part /boot/efi
├─vda2          253:2    0    1G  0 part /boot
└─vda3          253:3    0 98.8G  0 part
  ├─klas-root   252:0    0 59.7G  0 lvm  /
  ├─klas-swap   252:1    0   10G  0 lvm  [SWAP]
  └─klas-backup 252:2    0 29.1G  0 lvm
```

7) 安装 growpart 命令

a.

```
# yum install cloud-utils-growpart -y
```

b. 执行 growpart 命令对需要扩容的分区进行容量扩展操作。

```
# growpart /dev/vda 3
```

但是出现了以下错误：

```
[root@localhost ~]# growpart /dev/vda 3
unexpected output in sfdisk --version [sfdisk, 来自 util-linux 2.35.2]
```

解决该错误：

①查看服务器当前的语言类型:

```
# echo $LANG
```

```
[root@localhost ~]# echo $LANG
zh_CN.UTF-8
```

②定义全局变量，语言修改为英文并再次查看语言类型：

```
# export LANG=en_US.UTF-8
# echo $LANG
```

```
[root@localhost ~]# export LANG=en_US.UTF-8
[root@localhost ~]# echo $LANG
en_US.UTF-8
```

③重新使用 growpart 命令对分区扩容:

```
# growpart /dev/vda 3
```

```
[root@localhost ~]# growpart /dev/vda 3
CHANGED: partition=3 start=2508800 old: size=207204352 end=209713152 new: size=416921567 end=41943036
```

④lsblk 再次查看有没有给 vda3 扩容成功:

```
# lsblk
```

```
[root@localhost ~]# lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0              11:0    1  1024M  0 rom
vda             253:0    0   200G  0 disk
├─vda1          253:1    0   200M  0 part /boot/efi
├─vda2          253:2    0     1G  0 part /boot
└─vda3          253:3    0 198.8G  0 part
  ├─klas-root   252:0    0  59.7G  0 lvm  /
  ├─klas-swap   252:1    0    10G  0 lvm  [SWAP]
  └─klas-backup 252:2    0  29.1G  0 lvm
```

8) vda3 总分区扩容成功后，再给其下的某个分区扩容（用 df -h 查看其分区下某个分区的全路径，例如/dev/mapper/klas-root

```
[root@localhost ~]# df -h
Filesystem              Size  Used Avail Use% Mounted on
devtmpfs                 16G     0   16G   0% /dev
tmpfs                    16G  256K   16G   1% /dev/shm
tmpfs                    16G   71M   16G   1% /run
tmpfs                    16G     0   16G   0% /sys/fs/cgroup
/dev/mapper/klas-root    60G   44G   16G  74% /
tmpfs                    16G  150M   16G   1% /tmp
/dev/vda2              1014M  217M  798M  22% /boot
/dev/vda1               200M  5.8M  195M   3% /boot/efi
tmpfs                   3.1G  704K  3.1G   1% /run/user/0
```

```
# pvresize /dev/vda3
# pvdisplay
# vgdisplay
# lvextend -L +100G /dev/mapper/klas-root
```

```
[root@localhost ~]# pvresize /dev/vda3
  Physical volume "/dev/vda3" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

```
[root@localhost ~]# pvdisplay
  --- Physical volume ---
  PV Name               /dev/vda3
  VG Name               klas
  PV Size               198.80 GiB / not usable 1.98 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              50893
  Free PE               25600
  Allocated PE          25293
  PV UUID               875kxR-S41B-5W2s-uqhf-IRYu-dN5K-Fdfnkn

[root@localhost ~]# vgdisplay
  --- Volume group ---
  VG Name               klas
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  9
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                3
  Open LV               2
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               198.80 GiB
  PE Size               4.00 MiB
  Total PE              50893
  Alloc PE / Size       25293 / 98.80 GiB
  Free  PE / Size       25600 / 100.00 GiB
  VG UUID               7C9EDa-hokb-kD6c-KQry-1GH0-9ejj-Sz7zwP

[root@localhost ~]# lvextend -L +100G /dev/mapper/klas-root
  Size of logical volume klas/root changed from <59.67 GiB (15275 extents) to <159.67 GiB (40875 extents).
  Logical volume klas/root successfully resized.
```

此时用 lsblk 查看 klas-root 这个分区发现容量已经增加，但用 df -h 查看并没增加，此时用 xfs_growfs /dev/mapper/klas-root 刷新下即可。

```
# lsblk
# df -h
# xfs_growfs /dev/mapper/klas-root
```

```
[root@localhost ~]# lsblk
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0              11:0    1  1024M  0 rom
vda             253:0    0   200G  0 disk
├─vda1          253:1    0   200M  0 part /boot/efi
├─vda2          253:2    0     1G  0 part /boot
└─vda3          253:3    0 198.8G  0 part
  ├─klas-root   252:0    0 159.7G  0 lvm  /
  ├─klas-swap   252:1    0    10G  0 lvm  [SWAP]
  └─klas-backup 252:2    0  29.1G  0 lvm
[root@localhost ~]# df -h
Filesystem              Size  Used Avail Use% Mounted on
devtmpfs                 16G     0   16G   0% /dev
tmpfs                    16G  256K   16G   1% /dev/shm
tmpfs                    16G  111M   16G   1% /run
tmpfs                    16G     0   16G   0% /sys/fs/cgroup
/dev/mapper/klas-root   160G   45G  116G  28% /
tmpfs                    16G  149M   16G   1% /tmp
/dev/vda2              1014M  217M  798M  22% /boot
/dev/vda1               200M  5.8M  195M   3% /boot/efi
tmpfs                   3.1G  704K  3.1G   1% /run/user/0
```

```
[root@localhost ~]# xfs_growfs /dev/mapper/klas-root
meta-data=/dev/mapper/klas-root  isize=512    agcount=4, agsize=3910400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=15641600, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=7637, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 15641600 to 41856000
```

如果提示如下报错，则使用分区“/”方式刷新：

```
[root@localhost ~]# xfs_growfs /dev/mapper/klas-root
xfs_growfs: /dev/mapper/klas-root is not a mounted XFS filesystem
```

使用分区“/”方式刷新：

```
# xfs_growfs /
[root@localhost ~]# xfs_growfs /
meta-data=/dev/mapper/klas-root  isize=512    agcount=4, agsize=4587264 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=0
data     =                       bsize=4096   blocks=18349056, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=8959, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 18349056 to 31456256
```

3.6.3 如何使用 virt-install 安装虚拟机

3.6.3.1 系统版本

适用系统：V10(SP1)、V10(SP2)

适用架构：X86、AARCH

其他版本和架构可作参考。

3.6.3.2 问题描述

如何使用 virt-instal 安装虚拟机?

3.6.3.3 问题分析

virt-install 是一个命令行工具,它能够为 KVM、Xen 或其它支持 libvrit API 的 hypervisor 创建虚拟机并完成 GuestOS 安装；此外，它能够基于串行控制台、VNC 或支持文本或图形安装界面。

3.6.3.4 解决方案

1) 安装软件包

```
# yum install qemu-kvm libvirt virt-install
```

2) 配置桥接网络

a.

```
# yum install bridge-utils
```

b. 假设默认物理机网卡为 enp4s0f0

备份 enp4s0f0 网卡配置信息

```
# cp /etc/sysconfig/network-scripts/ifcfg-enp4s0f0 /etc/sysconfig/network-scripts/ifcfg-enp4s0f0.bak
```

c. 配置新的 br0 和 enp4s0f0 网卡信息

```
# cp /etc/sysconfig/network-scripts/ifcfg-enp4s0f0 /etc/sysconfig/network-scripts/ifcfg-br0
```

在 enp4s0f0 配置信息中添加网桥名

BRIDGE=br0

```
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp4s0f0
UUID=9d901aad-dbc1-429d-b6e6-2872d8cb6955
DEVICE=enp4s0f0
ONBOOT=yes
IPADDR=[redacted]
PREFIX=[redacted]
DNS1=[redacted]
DNS2=[redacted]
IPV6_PRIVACY=no
HWADDR=14:58:D0:5F:19:9C
GATEWAY=[redacted]
BRIDGE=br0
```

配置 br0 的信息(ip 重新配一个，如果是 dhcp 的话需要安装 dhcp 的方法配置)

```
TYPE=Bridge
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=br0
#UUID=9d901aad-dbc1-429d-b6e6-2872d8cb6955
DEVICE=br0
ONBOOT=yes
IPADDR=[redacted]
PREFIX=[redacted]
DNS1=[redacted]
DNS2=[redacted]
IPV6_PRIVACY=no
#HWADDR=14:58:D0:5F:19:9C
GATEWAY=[redacted]
~
~
```

d. 重启网络

```
# systemctl restart NetworkManager
```

e. 用 brctl show 查看网桥

```
# brctl show
```

```
[root@19cnode1 network-scripts]# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.1458d05f199c       no              enp4s0f0
                                                        vnet0
                                                        vnet1
virbr0          8000.5254005919ed       yes             virbr0-nic
```

f. 用 ip a 查看网络（其中 enp4s0f0 已经无 ip，但是 br0 的 ip 有了，可以通过这个链接机器）

```
# ip a
```

```
[root@19cnode1 network-scripts]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp4s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state UP group default qlen 1000
    link/ether 14:58:d0:5f:19:9c brd ff:ff:ff:ff:ff:ff
3: enp4s0f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 14:58:d0:5f:19:9d brd ff:ff:ff:ff:ff:ff
4: enp4s0f2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 14:58:d0:5f:19:9e brd ff:ff:ff:ff:ff:ff
5: enp4s0f3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 14:58:d0:5f:19:9f brd ff:ff:ff:ff:ff:ff
6: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:59:19:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
7: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:59:19:ed brd ff:ff:ff:ff:ff:ff
8: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 14:58:d0:5f:19:9c brd ff:ff:ff:ff:ff:ff
    inet 172.17.31.209/24 brd 172.17.31.255 scope global noprefixroute br0
       valid_lft forever preferred_lft forever
    inet6 fe80::8857:9bce:3501:aac5/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
9: vnet0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UNKNOWN group default qlen 1000
    link/ether fe:54:00:5f:eb:74 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fc54:ff:fe5f:eb74/64 scope link
       valid_lft forever preferred_lft forever
18: vnet1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master br0 state UNKNOWN group default qlen 1000
    link/ether fe:54:00:9d:ec:d5 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::fc54:ff:fe9d:ecd5/64 scope link
       valid_lft forever preferred_lft forever
```

3) 用 virt-install 安装虚拟机

```
# virt-install --virt-type=kvm --name=test1 --vcpus=4 --memory=4096
--location=/root/Kylin-Server-10-SP1-x86-Release-Build04-20200711.iso --disk
path=/root/test1.qcow2,size=50,format=qcow2 --network bridge=br0 ,
model=virtio --graphics none --extra-args='console=ttyS0' --force
```

然后就进入命令行安装界面，根据提示操作即可。

3.7 远程连接服务器问题

3.7.1 VNC 界面如何启动中文输入法

3.7.1.1 系统版本

适用系统：V10(SP1)、V10(SP2)

适用架构：X86、AARCH、MIPS64el

其他版本和架构可作参考。

3.7.1.2 问题描述

VNC 远程连接银河麒麟高级服务器无法启动中文输入法。

3.7.1.3 问题分析

查看银河服务器操作系统，只有英文输入法，没有中文输入法，无法输入中文。操作系统缺少中文输入法的相关软件，如果源中有相关软件包，直接利用 yum 进行安装即可，然后再修改 VNC 相关配置文件，将中文输入法的变量添

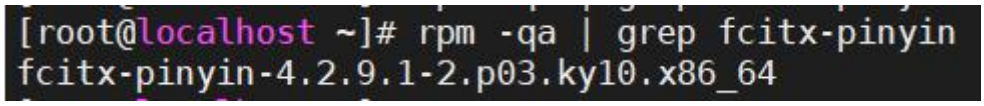
加进去。

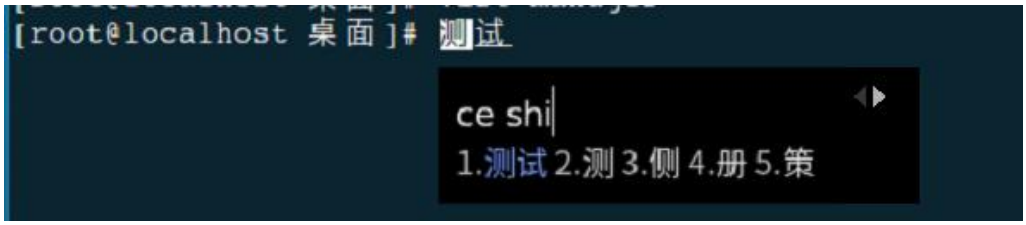
3.7.1.4 解决方案

1） 利用 yum 安装

```
# yum install fcitx-pinyin
```

2） 重启系统使输入法生效并进行验证





3)

```
# vim /root/.vnc/xstartup
```

添加以下内容

```
export GTK_IM_MODULE="fcitx"
export QT_IM_MODULE="fcitx"
export XMODIFIERS="@im=fcitx"
```

3.7.2 如何使用密钥连接服务器

3.7.2.1 系统版本

适用系统：V10(SP1)、V10(SP2)、V10(SP3)

适用架构：全架构

其他版本和架构可作参考。

3.7.2.2 问题描述

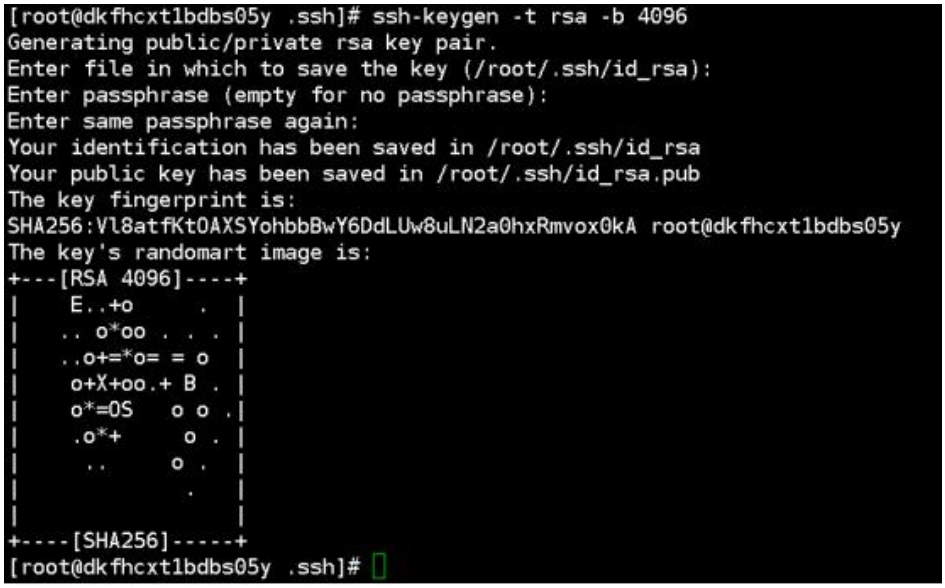
如何使用密钥连接银河麒麟高级服务器系统。

3.7.2.3 解决方案

1）在本地机器使用 ssh-keygen 命令生成密钥对（公钥和私钥）

```
# ssh-keygen -t rsa -b 4096
```

将在 ~/.ssh/ 目录下生成 id_rsa（私钥）和 id_rsa.pub（公钥）文件：



2）将公钥上传到目标服务器

使用 ssh-copy-id 命令将公钥上传到 Linux 服务器：

```
# ssh-copy-id user@server_ip
```

或者手动将公钥内容添加到目标服务器上 ~/.ssh/authorized_keys 文件中。

3）配置目标服务器 SSH 服务启用密钥认证

在 SSH 配置文件 /etc/ssh/sshd_config 中，确保以下设置：

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      ./ssh/authorized_keys
```

（均为系统默认的配置文件）

4）重启 SSH 服务

```
# systemctl restart sshd
```

5）连接进行验证

3.7.3 如何部署虚拟多用户访问 ftp

3.7.3.1 系统版本

适用系统：V10(SP1)、V10(SP2)、V10(SP3)

适用架构：全架构

其他版本和架构可作参考。

3.7.3.2 问题描述

如何使用虚拟用户连接 vsftp。

3.7.3.3 解决方案

1）安装 vsftpd

```
# yum install vsftpd
```

2）创建 ftp 用户

```
# useradd -d /home/data/ftp/ vuser -s /sbin/nologin
# mkdir /home/data/ftp/test
# mkdir /home/data/ftp/test2
```

3）更新权限和组权限

更新权限：

```
# chmod 755 -R /home/data/ftp/
```

赋予组权限：

```
# chown -R vuser.root /home/data/ftp/
```

4）创建虚拟用户[用户名和密码]文件

```
# gdbmtool /etc/vsftpd/vuser-list.pag
```

```
[root@localhost vuser_config]# gdbmtool /etc/vsftpd/vuser-list.pag

Welcome to the gdbm tool.  Type ? for help.

gdbmtool> store test
DATA? 112233
gdbmtool> store test2
DATA? 112233
gdbmtool> q
[root@localhost vuser_config]#
```

5）创建认证 PAM 文件

```
# vim /etc/pam.d/vsftpd.vu
```

加入：

```
auth required /lib64/security/pam_userdb.so db=/etc/vsftpd/vuser-list
account required /lib64/security/pam_userdb.so db=/etc/vsftpd/vuser-list
```

```
auth      required    /lib64/security/pam_userdb.so   db=/etc/vsftpd/vuser-list
account   required    /lib64/security/pam_userdb.so   db=/etc/vsftpd/vuser-list
```

6）修改配置文件

备份原配置文件

```
# mv /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.bak
# vim /etc/vsftpd/vsftpd.conf
```

加入：

```
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES
pam_service_name=vsftpd.vu
#userlist_enable=YES

#修改 ftp 默认目录到/data/ftp 下面
chroot_local_user=YES
local_root=/home/data/ftp
anon_root=/home/data/ftp
```

```
#虚拟用户权限配置目录
user_config_dir=/etc/vsftpd/vuser_config
allow_writeable_chroot=YES
one_process_model=NO

#开启虚拟用户
guest_enable=YES
guest_username=vuser
```

7）创建存放虚拟用户权限目录

```
# mkdir /etc/vsftpd/vuser_config
```

切换到权限目录

```
# cd /etc/vsftpd/vuser_config/
```

创建 test 拥有所有权限

```
# vim test
```

加入：

```
local_root=/home/data/ftp/test
write_enable=yes
anon_world_readable_only=no
anon_upload_enable=yes
anon_mkdir_write_enable=yes
anon_other_write_enable=yes
```

创建 test2 用户，只有上传下载的权限：

```
# vim test2
```

```
local_root=/home/data/ftp/test2
anon_upload_enable=yes
anon_world_readable_only=no
```

8）重启 vsftpd 服务并连接

```
[root@localhost vuser_config]# systemctl restart vsftpd.service
[root@localhost vuser_config]# systemctl status vsftpd.service
● vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Mon 2024-08-12 16:27:10 CST; 5s ago
  Process: 28928 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
 Main PID: 28929 (vsftpd)
    Tasks: 1
   Memory: 1.8M
   CGroup: /system.slice/vsftpd.service
           └─28929 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

8月 12 16:27:10 localhost.localdomain systemd[1]: Starting Vsftpd ftp daemon...
8月 12 16:27:10 localhost.localdomain systemd[1]: Started Vsftpd ftp daemon.
[root@localhost vuser_config]#
```

```
root@kylin-pc:/home/kylin/桌面# ftp 10.41.4.57
Connected to 10.41.4.57.
220 (vsFTPd 3.0.3)
Name (10.41.4.57:kylin): test
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

3.7.4 xrdp 如何使用 xorg 登录

3.7.4.1 系统版本

适用系统：V10(SP1)、V10(SP2)、V10(SP3)

适用架构：全架构

其他版本和架构可作参考。

3.7.4.2 问题描述

xrdp 如何使用 xorg 登录。

3.7.4.3 问题分析

xrdp 默认搭配使用的是 xvnc，查看/etc/xrdp/xrdp.ini 配置文件里默认是没有打开 xorg 的，只有 xvnc 是打开的，手动将其打开，连接失败，验证 centos 也是同样的问题。

银河麒麟桌面操作系统默认 xorg 是打开的，可以连接。

3.7.4.4 解决方案

对比桌面操作系统和服务器操作系统安装的包的差异，

服务器操作系统安装的包只有三个：xrdp、xrdp-devel、xrdp-selinux

桌面操作系统安装的包有：xrdp、xorgxrdp

服务器操作系统若想 xrdp 使用 xorg 登录，也需要安装 xorgxrdp 这个包。

以下是详细配置步骤：

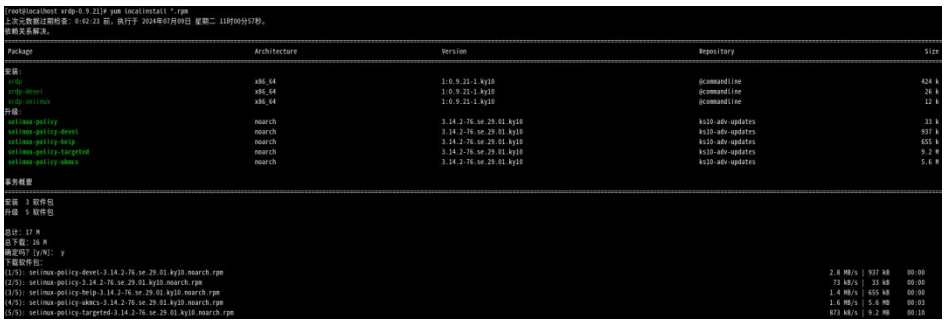
1）关闭防火墙

```
# systemctl stop firewalld
```

2）将已适配的 xrdp 软件包上传到服务器并解压

3）安装 xrdp

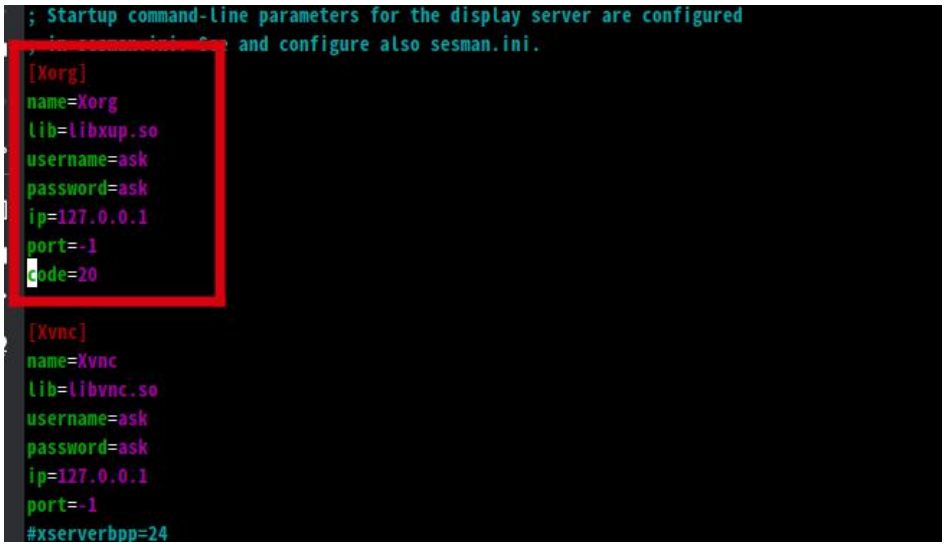
```
# rpm -ivh *.rpm
```



4）更改配置文件

```
# vim /etc/xrdp/xrdp.ini
```

将 Xorg 相关配置的注释去掉



5）编译安装 xorgxrdp

①下载源码

https://codeload.github.com/neutrinolabs/xorgxrdp/tar.gz/refs/tags/v0.9.19

②上传到服务器并解压

③安装依赖包

```
# yum install xorg-x11-server-devel nasm
```

④编译

```
# cd xorgxrdp-0.9.19
# ./bootstrap
```

```
[root@localhost ~]# cd xorgxrdp-0.9.19/
[root@localhost xorgxrdp-0.9.19]# ./bootstrap
/usr/bin/autoconf
/usr/bin/automake
/usr/bin/libtool
/usr/bin/pkg-config
autoreconf: Entering directory `.'
autoreconf: configure.ac: not using Gettext
autoreconf: running: aclocal --force -I m4
autoreconf: configure.ac: tracing
autoreconf: running: libtoolize --copy --force
libtoolize: putting auxiliary files in '.'.
libtoolize: copying file './ltmain.sh'
libtoolize: putting macros in AC_CONFIG_MACRO_DIRS, 'm4'.
libtoolize: copying file 'm4/libtool.m4'
libtoolize: copying file 'm4/ltoptions.m4'
libtoolize: copying file 'm4/ltsugar.m4'
libtoolize: copying file 'm4/ltversion.m4'
libtoolize: copying file 'm4/lt~obsolete.m4'
autoreconf: running: /usr/bin/autoconf --force
autoreconf: running: /usr/bin/autoheader --force
autoreconf: running: automake --add-missing --copy --force-missing
configure.ac:15: installing './compile'
configure.ac:17: installing './config.guess'
configure.ac:17: installing './config.sub'
configure.ac:13: installing './install-sh'
configure.ac:13: installing './missing'
module/Makefile.am: installing './depcomp'
parallel-tests: installing './test-driver'
autoreconf: Leaving directory `.'
```

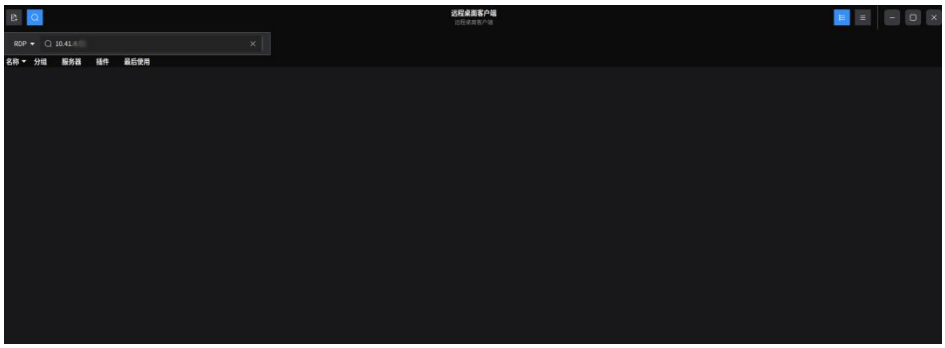
```
# ./configure
```

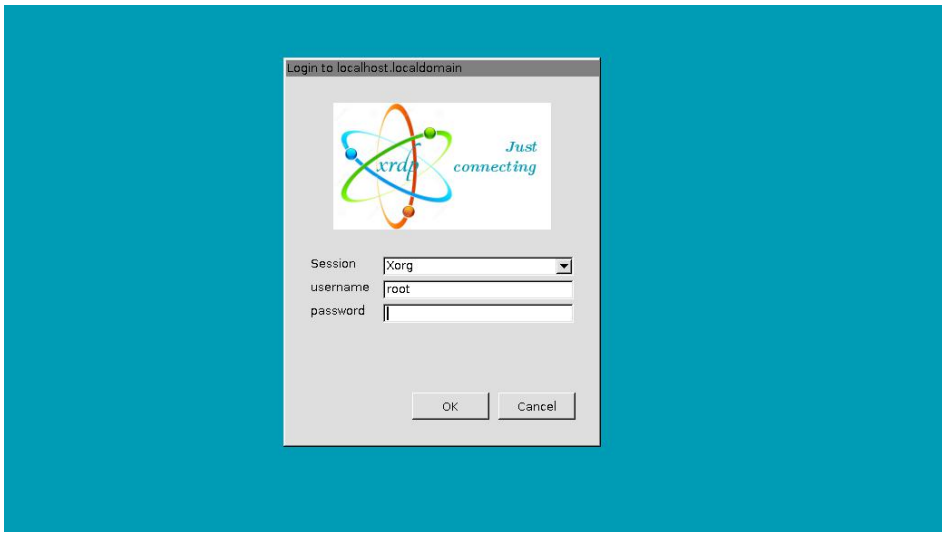
```
# make
# make install
```

6）启动 xrdp、xrdp-sesman 服务

```
[root@localhost xorgxrdp-0.9.19]# systemctl start xrdp xrdp-sesman
[root@localhost xorgxrdp-0.9.19]# systemctl status xrdp xrdp-sesman
●xrdp.service - xrdp daemon
   Loaded: loaded (/usr/lib/systemd/system/xrdp.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-07-09 11:15:04 CST; 4s ago
     Docs: man:xrdp(8)
           man:xrdp.ini(5)
 Main PID: 19912 (xrdp)
    Tasks: 1
   Memory: 920.0K
   CGroup: /system.slice/xrdp.service
           └─19912 /usr/sbin/xrdp --nodaemon

7月 09 11:15:04 localhost.localdomain systemd[1]: Started xrdp daemon.
7月 09 11:15:04 localhost.localdomain xrdp[19912]: [INFO ] starting xrdp with pid 19912
7月 09 11:15:04 localhost.localdomain xrdp[19912]: [INFO ] address [0.0.0.0] port [3389] mode 1
7月 09 11:15:04 localhost.localdomain xrdp[19912]: [INFO ] listening to port 3389 on 0.0.0.0
7月 09 11:15:04 localhost.localdomain xrdp[19912]: [INFO ] xrdp_listen_pp done

●xrdp-sesman.service - xrdp session manager
   Loaded: loaded (/usr/lib/systemd/system/xrdp-sesman.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2024-07-09 11:15:04 CST; 4s ago
     Docs: man:xrdp-sesman(8)
           man:sesman.ini(5)
 Main PID: 19911 (xrdp-sesman)
    Tasks: 1
   Memory: 924.0K
   CGroup: /system.slice/xrdp-sesman.service
           └─19911 /usr/sbin/xrdp-sesman --nodaemon

7月 09 11:15:04 localhost.localdomain systemd[1]: Started xrdp session manager.
7月 09 11:15:04 localhost.localdomain xrdp-sesman[19911]: [INFO ] starting xrdp-sesman with pid 19911
```

7）远程连接





4 联系我们

如有需要帮助，可联系我们：

咨询热线	400-089-1870