

# H3C 百业灵犀大模型&推理引擎

安装部署指导

新华三技术有限公司  
<http://www.h3c.com>

资料版本：5W100-20241219

Copyright © 2024 新华三技术有限公司及其许可者 版权所有，保留一切权利。

未经本公司书面许可，任何单位和个人不得擅自摘抄、复制本书内容的部分或全部，并不得以任何形式传播。

除新华三技术有限公司的商标外，本手册中出现的其它公司的商标、产品标识及商品名称，由各自权利人拥有。

由于产品版本升级或其他原因，本手册内容有可能变更。H3C 保留在没有任何通知或者提示的情况下对本手册的内容进行修改的权利。本手册仅作为使用指导，H3C 尽全力在本手册中提供准确的信息，但是 H3C 并不确保手册内容完全没有错误，本手册中的所有陈述、信息和建议也不构成任何明示或暗示的担保。

# 前言

本手册详细介绍了 H3C 百业灵犀大模型&推理引擎的安装部署步骤。

前言部分包含如下内容：

- [读者对象](#)
- [本书约定](#)
- [资料意见反馈](#)

## 读者对象

本手册主要适用于如下工程师：

- 网络规划人员
- 现场技术支持与维护人员
- 负责网络配置和维护的网络管理员

## 本书约定

### 1. 图形界面格式约定

格 式	意 义
<>	带尖括号“<>”表示按钮名，如“单击<确定>按钮”。
[]	带方括号“[]”表示窗口名、菜单名和数据表，如“弹出[新建用户]窗口”。
/	多级菜单用“/”隔开。如[文件/新建/文件夹]多级菜单表示[文件]菜单下的[新建]子菜单下的[文件夹]菜单项。

### 2. 各类标志

本书还采用各种醒目标志来表示在操作过程中应该特别注意的地方，这些标志的意义如下：

 注意	提醒操作中应注意的事项，不当的操作可能会导致数据丢失或者设备损坏。
 说明	对操作内容的描述进行必要的补充和说明。

## 资料意见反馈

如果您在使用过程中发现产品资料的任何问题，可以通过以下方式反馈：

E-mail: [info@h3c.com](mailto:info@h3c.com)

感谢您的反馈，让我们做得更好！

# 目 录

<b>1 概述</b> .....	<b>1-1</b>
1.1 文档目的.....	1-1
1.2 常用词.....	1-1
<b>2 安装前的准备工作</b> .....	<b>2-2</b>
2.1 部署环境确认.....	2-2
2.2 部署模式规划.....	2-3
2.3 组网规划.....	2-5
2.3.1 3 机或 3+N 机组网规划.....	2-5
2.3.2 分布式推理组网规划.....	2-6
2.3.3 单机组网规划.....	2-7
2.4 IP 地址规划 .....	2-8
2.5 应用安装包准备.....	2-8
<b>3 LinSeer RT 部署流程</b> .....	<b>3-9</b>
<b>4 安装 NingOS 操作系统</b> .....	<b>4-11</b>
4.1 登录远程控制台.....	4-11
4.1.1 配置 Java 环境 .....	4-11
4.1.2 登录 HDM Web 页面.....	4-12
4.1.3 连接远程控制台并挂载镜像 .....	4-13
4.2 设置 BIOS 从 CD/DVD 启动 .....	4-18
4.3 安装操作系统.....	4-22
4.3.1 基本设置.....	4-23
4.3.2 磁盘分区.....	4-28
4.3.3 用户及网络设置 .....	4-31
4.3.4 操作系统安装.....	4-37
<b>5 安装部署 NingOS Matrix 集群</b> .....	<b>5-39</b>
5.1 安装 Matrix 依赖包 .....	5-39
5.2 关闭防火墙.....	5-39
5.3 设置时区时间.....	5-40
5.4 配置本地 YUM 源.....	5-40
5.5 安装 Matrix 依赖包 .....	5-41
5.6 安装 Matrix.....	5-42

5.7 创建 Matrix 集群 .....	5-44
5.7.1 登录 Matrix .....	5-44
5.7.2 配置集群参数 .....	5-45
5.7.3 创建集群 .....	5-46
5.7.4 创建网络 .....	5-49
<b>6 安装 KylinOS 操作系统 .....</b>	<b>6-51</b>
6.1 登录远程控制台 .....	6-51
6.1.1 配置 Java 环境 .....	6-51
6.1.2 登录 HDM Web 页面 .....	6-52
6.1.3 连接远程控制台并挂载镜像 .....	6-53
6.2 设置 BIOS 从 CD/DVD 启动 .....	6-58
6.3 安装操作系统 .....	6-62
6.3.1 基本设置 .....	6-63
6.3.2 磁盘分区 .....	6-66
6.3.3 用户及网络设置 .....	6-69
6.3.4 操作系统安装 .....	6-74
<b>7 安装部署 KylinOS Matrix 集群 .....</b>	<b>7-76</b>
7.1 关闭防火墙 .....	7-76
7.2 设置时区时间 .....	7-77
7.3 安装 kylin Matrix .....	7-78
<b>8 安装 GPU 驱动及相关工具包 .....</b>	<b>8-79</b>
8.1 安装 NVIDIA GPU 驱动及工具包 .....	8-80
8.1.1 查看显卡型号 .....	8-80
8.1.2 下载显卡驱动 .....	8-81
8.1.3 禁用 Linux 自带的显卡驱动 .....	8-83
8.1.4 安装显卡驱动 .....	8-84
8.2 安装 nvidia-container-runtime .....	8-85
8.2.1 NingOS 操作系统安装 nvidia-container-runtime .....	8-85
8.2.2 KylinOS 操作系统安装 nvidia-container-runtime .....	8-86
8.3 安装 nvidia-device-plugin .....	8-86
8.3.1 修改默认运行时 .....	8-86
8.3.2 部署 NVIDIA 设备插件（在线部署方式） .....	8-87
8.3.3 部署 NVIDIA 设备插件（离线部署方式） .....	8-90
8.3.4 确认 GPU 识别正常 .....	8-93
8.4 安装 nvidia-fabricmanager .....	8-94
8.5 安装 nv_peer_memory .....	8-96

8.5.1 下载 nv_peer_memory.....	8-96
8.5.2 安装 nv_peer_memory.....	8-96
8.6 安装昆仑芯驱动.....	8-97
8.6.1 驱动版本信息 .....	8-97
8.6.2 查看 GPU 状态 .....	8-97
8.6.3 安装显卡驱动.....	8-98
8.7 安装天数智铠 MR-V100 驱动.....	8-100
8.7.1 驱动版本信息 .....	8-100
8.7.2 安装显卡驱动 .....	8-100
8.8 安装天数天核 BI-V150 驱动 .....	8-105
8.8.1 驱动版本信息 .....	8-105
8.8.2 安装显卡驱动 .....	8-105
8.9 配置 NFS 存储服务器 .....	8-109
8.9.1 配置 NFS 存储服务端.....	8-109
8.9.2 验证 NFS 存储服务端配置.....	8-110
<b>9 安装网卡驱动及 ROCE 配置.....</b>	<b>9-111</b>
9.1 安装网卡驱动.....	9-111
9.1.1 安装依赖包.....	9-112
9.1.2 安装网卡驱动.....	9-112
9.2 参数网卡 ROCE 配置.....	9-113
<b>10 安装 k8s-rdma-shared-dev-plugin.....</b>	<b>10-114</b>
10.1.1 配置 ConfigMap .....	10-114
10.1.2 部署 k8s-rdma-shared-dev-plugin .....	10-115
<b>11 (可选) 部署 PolarDB 数据库.....</b>	<b>11-116</b>
<b>12 安装部署 LinSeer RT .....</b>	<b>12-116</b>
12.1 登录 Matrix .....	12-117
12.2 部署 LinSeer RT 组件安装包.....	12-118
12.2.1 上传安装包 .....	12-118
12.2.2 解析安装包 .....	12-119
12.2.3 部署 LinSeer RT 安装包.....	12-120
<b>13 访问 LinSeer RT .....</b>	<b>13-124</b>
<b>14 增加模型 .....</b>	<b>14-124</b>
<b>15 软件授权 .....</b>	<b>15-126</b>
<b>16 集群扩缩容 .....</b>	<b>16-126</b>
16.1 集群扩容 Worker 节点 .....	16-126

16.2 集群缩容 Worker 节点 .....	16-130
17 卸载组件 .....	17-131
18 常见问题解答 .....	18-132
18.1 安装操作系统时，各磁盘分区分别有什么作用？ .....	18-132
18.2 为什么要禁用 Nouveau？ .....	18-132

# 1 概述

## 1.1 文档目的

本文档主要介绍 H3C 百业灵犀推理引擎（以下简称 LinSeer RT）的安装、部署、卸载等功能。



提示

- 本文档中的设备显示信息和 Web 页面截图均为举例，请以实际环境为准。
- 为方便您的使用，本文提供了部分组件安装包的下载网址。部分网址可能会变更，如网址失效，请尝试到对应组件官网搜索下载，如无法解决，请联系技术支持。

## 1.2 常用词

- NingOS：H3C 自研的基于 Linux 的 NingOS 操作系统。
- Kylin OS：基于 Linux 的麒麟操作系统。
- Matrix：基于 Kubernetes，实现了对 Docker 容器的编排调度。主要用于 Kubernetes 集群的搭建，微服务的部署，以及系统、Docker 容器、微服务等的运维监控。
- Kubernetes：简称 K8s，是一个开源的容器编排引擎，用于对容器化应用进行自动化部署、扩缩和管理。
- Docker：是一个开源的容器编排引擎，用于对容器化应用进行自动化部署、扩缩和管理。
- GPU：图形处理器（Graphics Processing Unit），又称显示核心、视觉处理器、显示芯片，是一种专门在个人电脑、工作站、游戏机和一些移动设备上进行图像和图形相关运算工作的微处理器。
- GUI：图形用户界面（Graphical User Interface），是指采用图形方式显示的计算机操作用户界面。
- CPU 服务器：CPU 服务器是指以中央处理器（CPU）为核心的服务器，主要用于串行计算和处理单个线程的计算任务。
- GPU 服务器：GPU 服务器是指配备了图形处理器（GPU）的服务器。GPU 服务器专注于高度并行的计算任务，如图形渲染、机器学习、深度学习和科学计算等。GPU 服务器价格较为昂贵，适用于对高性能计算要求较高的场景，如大规模数据分析、深度学习等。
- Master 节点：集群主节点。一个集群必须包含 3 个 Master 节点（单机模式下仅需一个 Master 节点），集群将从 3 个 Master 节点中自动选举出一台作为主用 Master 节点，其它 Master 节点为备用 Master 节点。当集群的主用 Master 节点故障时，将在备用 Master 节点中重新选出新的主用 Master 节点，接管原主用 Master 节点的业务，避免业务中断。主用 Master 和备用 Master 的作用如下：
  - 主用 Master 节点：负责管理和监控集群中的所有节点，北向业务虚 IP 会下发至主用 Master 节点上，所有 Master 节点共同承担业务的运行。

- 备用 Master 节点：仅承担业务的运行，不负责管理和监控其它节点。
- Pod：Kubernetes 最小的调度和管理单位。Pod 是一组运行在同一节点上的容器集合，它们共享相同的网络命名空间、存储资源和其他资源。
- Worker 节点：集群业务节点。Worker 节点仅承担业务的运行，不参与主用 Master 节点的选举。Worker 节点为可选节点，当 Master 上的业务已经达到资源和性能瓶颈，如 CPU 和内存使用率居高不下，业务响应慢，节点上 Pod 个数达到或接近 300 个限制等，可通过增加 Worker 节点提升集群的性能和业务处理能力。
- YUM 源：Linux 操作系统中的软件包管理器 YUM（Yellowdog Updater, Modified）所使用的软件仓库。YUM 源是包含一系列软件包的在线仓库，提供各种软件和工具的安装包。通过配置 YUM 源，用户可以方便地使用 YUM 命令来管理软件包，从在线仓库中下载并安装需要的软件包。
- nvidia-container-runtime（NVIDIA 运行时工具）：在容器化环境中提供 GPU 加速和管理功能的运行时工具，它的主要作用是提供对 NVIDIA GPU 的容器级别的访问和管理。
- nvidia-device-plugin（NVIDIA 设备插件）：Kubernetes 容器编排系统的一个插件，用于在 Kubernetes 集群中管理和调度 NVIDIA GPU 设备。

## 2 安装前的准备工作

### 2.1 部署环境确认

请确认安装部署环境，确保安装部署 LinSeer RT 的条件已经具备。

表2-1 安装部署环境确认

配置要求		检测标准
服务器	硬件检查	<ul style="list-style-type: none"> <li>● 检查硬件是否符合要求（包括 CPU、GPU、内存、硬盘、网卡等），请联系技术支持了解硬件的详细配置要求。</li> <li>● 建议使用如下 H3C 自研服务器：           <ul style="list-style-type: none"> <li>○ R5350 G6</li> <li>○ R5500 G6</li> <li>○ R5500 G5</li> <li>○ R5200 G3</li> <li>○ R5300 G5</li> </ul> </li> <li>● 请确认已配置 RAID。RAID 模式可选 RAID1、RAID5 或 RAID10，磁盘分区大小均为配置 RAID 后的磁盘大小</li> </ul>
	软件检查	<p>请检查如下几点：</p> <ul style="list-style-type: none"> <li>● 待安装的操作系统版本是否正确。对于 NingOS 操作系统，请使用 Ning OS-v3-1.0.2403-x86_64；对于 KylinOS 操作系统，请使用 Kylin-Server-10-SP2-x86-Release-Build09</li> <li>● 请检查系统时间已配置完成</li> </ul>
客户端		用户不需要安装客户端软件，使用浏览器即可访问LinSeer RT。推荐使用的浏览器为Google Chrome98、Firefox 78及以上版本，最低分辨率显示宽度

配置要求	检测标准
	为1600



### 注意

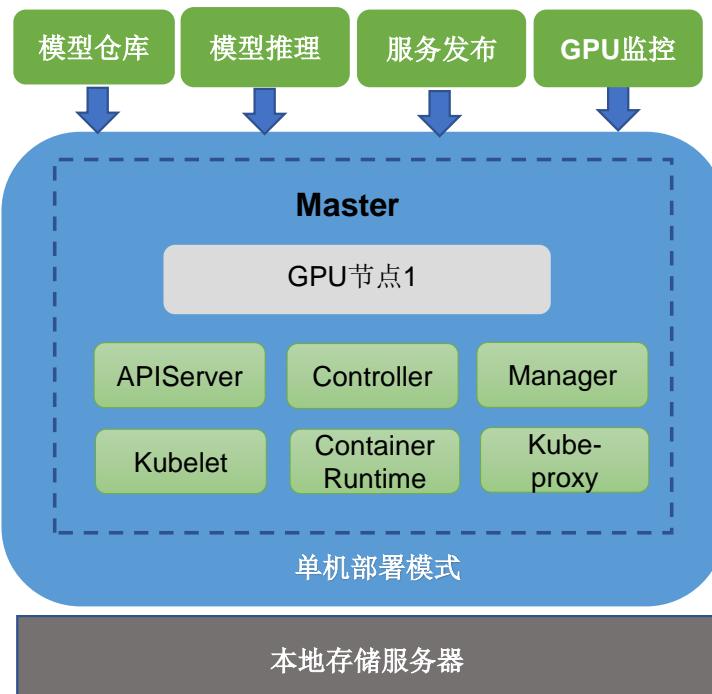
- 推荐将服务器的下一次启动模式配置为 UEFI 模式。
- 安装操作系统时，请勿同时使用 KVM 给多个服务器安装同一镜像。
- H5 KVM 性能不稳定，可能出现加载镜像停止或缓慢等问题，建议使用 Java KVM 挂载镜像安装操作系统，即 NingOS 或 KylinOS 操作系统。
- 安装过程中禁止输入 Scroll Lock 键，否则可能会导致安装失败。
- 安装过程中异常断电会导致部分服务安装失败，请重新安装以保证功能完整性。
- Matrix 安装部署过程中，请勿进行开启或关闭防火墙的操作。
- 集群部署完成后，请勿修改系统时间，否则可能导致集群异常。

## 2.2 部署模式规划

LinSeer RT 支持三种部署模式：

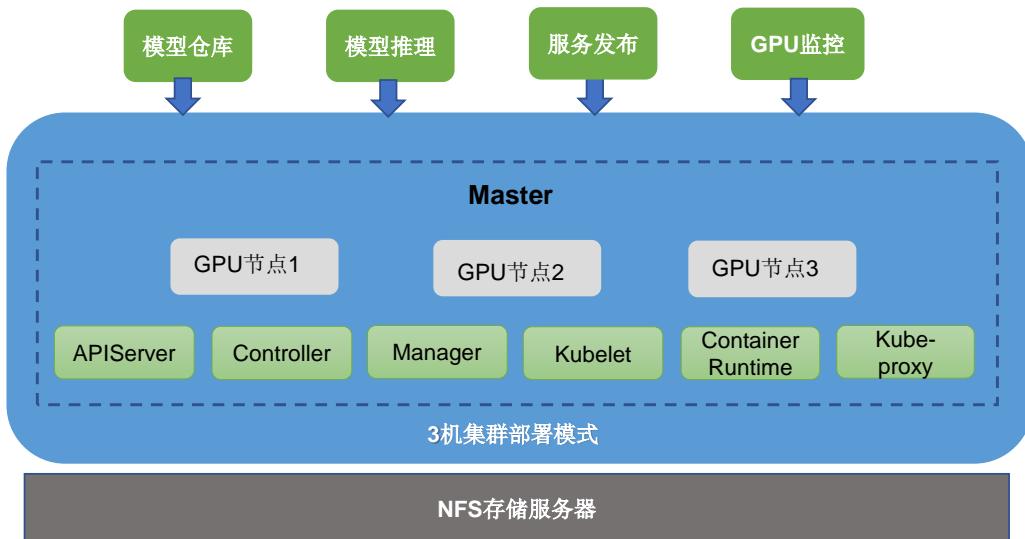
- 单机部署模式：使用 GPU 服务器部署 1 个 Master 节点，即可部署 Matrix 集群，业务规模较小且可靠性要求不高时可以采用本模式。

图2-1 单机部署模式示意图



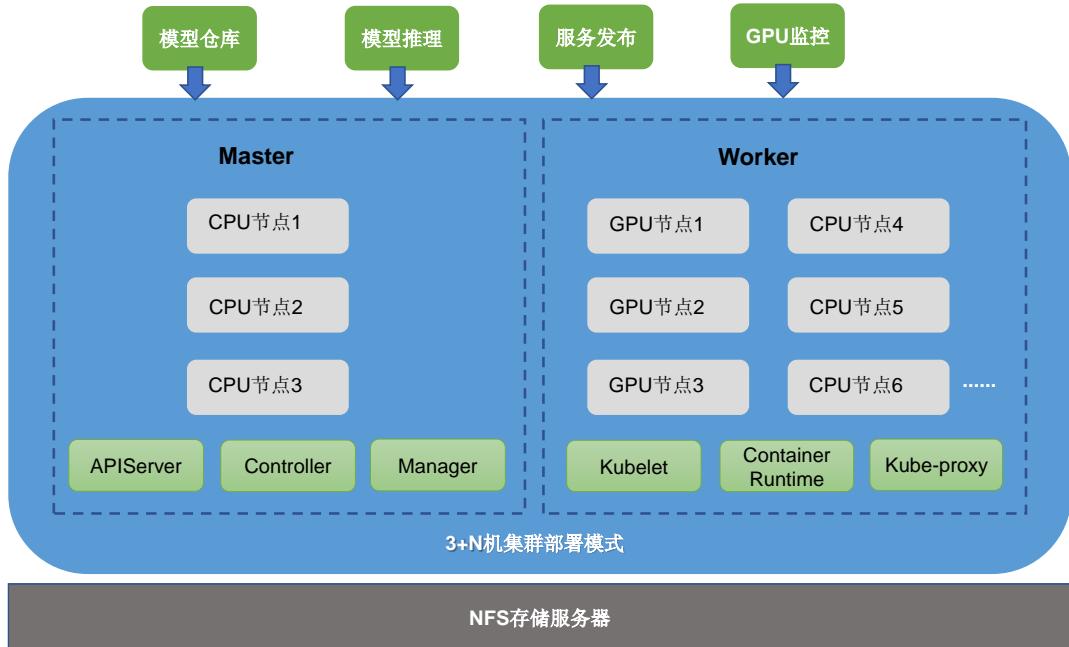
- 3 机集群部署模式：使用 GPU 服务器部署 3 个 Master 节点，再部署 Matrix 集群，可实现负载均衡和高可用性；适用中小型业务场景。

图2-2 3 机集群部署模式示意图



- 3+N 机集群部署模式：需要部署 3 个 Master 节点和 N 个 Worker 节点 ( $N \geq 1$ )。请使用 GPU 服务器部署 Worker 节点，建议选用 CPU 服务器部署 Master 节点的服务器。该模式下，支持按需增加 Worker 节点以实现扩容。
  - 该部署模式可将不同用途的业务模块调度到合适的服务器节点上执行，即 Worker 节点可以分摊 Master 节点的压力。Master 节点主要负责任务调度并处理指令复杂的任务（如集群资源的管理和调度），Worker 节点负责处理高度并行的计算子任务（如 AI 大模型的推理）。
  - 该部署模式可支持 CPU 和 GPU 服务器混合部署，实现更高的性能和可扩展性，适用大规模业务场景。

图2-3 3+N 机集群部署模式示意图



单机部署模式、3 机集群部署模式和 3+N 机集群部署模式差异如图 2-4 所示。

图2-4 三种部署模式差异对比表

部署模式	节点个数	推荐组网	推荐场景
单机部署模式	1个Master节点（1台GPU服务器）	单机组网，详细介绍请参见 <a href="#">2.3.3 单机组网规划</a>	业务规模较小且可靠性要求不高时，可以采用本模式
3机集群部署模式	3个Master节点（3台GPU服务器）	3机或3+N机组网，详细介绍请参见 <a href="#">2.3.1 3机或3+N机组网规划</a>	可实现负载均衡和高可靠性，适用中小型业务场景
3+N机集群部署模式	3个Master节点（3台CPU服务器）和N个Worker节点（N台GPU服务器，N≥1）	3机或3+N机组网，详细介绍请参见 <a href="#">2.3.1 3机或3+N机组网规划</a>	可实现负载均衡和高可靠性，支持按需增加或减少Worker节点以实现灵活部署，适用各种规模业务场景

## 2.3 组网规划

### 2.3.1 3 机或 3+N 机组网规划

3 机或 3+N 机组网是指通过将多个计算节点的网络打通，形成统一的计算资源池，以实现更大规模数据处理和复杂计算任务的高效、可靠和灵活执行。其中 3 机或 3+N 机组网使用外置的存储服务器作为 NFS 共享存储。3 机组网如图 2-5 所示。

图2-5 3机组网示意图

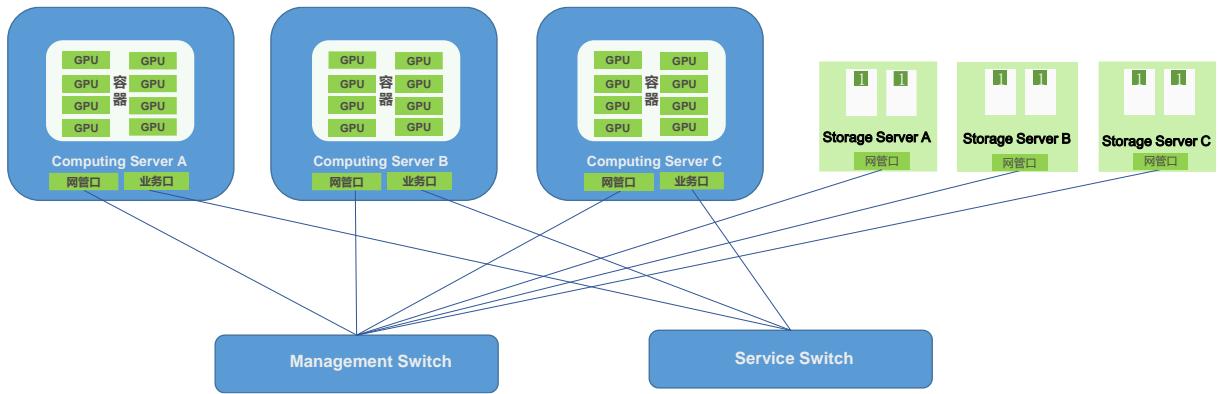


表2-2 IP 地址规划表

设备	接口	IP 地址	设备	接口	IP 地址
Compute Server A	ens1f1np1 (网管口)	192.168.0.1	Storage Sserver A	ens1f1np1 (网管口)	192.168.0.4
	ens2f1np1 (业务口)	20.20.1.1/24		ens1f1np1 (网管口)	192.168.0.5
Compute Server B	ens1f1np1 (网管口)	192.168.0.2	Storage Server B	ens1f1np1 (网管口)	192.168.0.6
	ens2f1np1 (业务口)	20.20.1.2/24			
Compute Server C	ens1f1np1 (网管口)	192.168.0.3	Storage Server C	ens1f1np1 (网管口)	192.168.0.7
	ens2f1np1 (业务口)	20.20.1.3/24			

### 2.3.2 分布式推理组网规划

分布式推理组网采用 3+N 机集群部署模式，通过将模型和数据分散存储在多个计算节点上，实现计算资源和内存容量的扩展。这样，百业灵犀推理引擎便能部署更大规模的模型，从而提升服务质量。

- 部署要求：请确保网络中服务器型号及网卡型号均相同，建议优先使用 R5500 G5 或 R5500 G6 或 R5350 G6 服务器。
- 组网概述：如图 2-6 所示，每台 GPU 服务器（以 R5350 G6 为例，服务器拓扑结构为 AI 并联）上安装有 2 张 RDMA 网卡。建议把不同 GPU 服务器上相同编号的 RDMA 网卡加入同一个网络，提供高速度、低延迟的通信通道，从而实现快速数据传输和协同计算。

图2-6 分布式推理组网

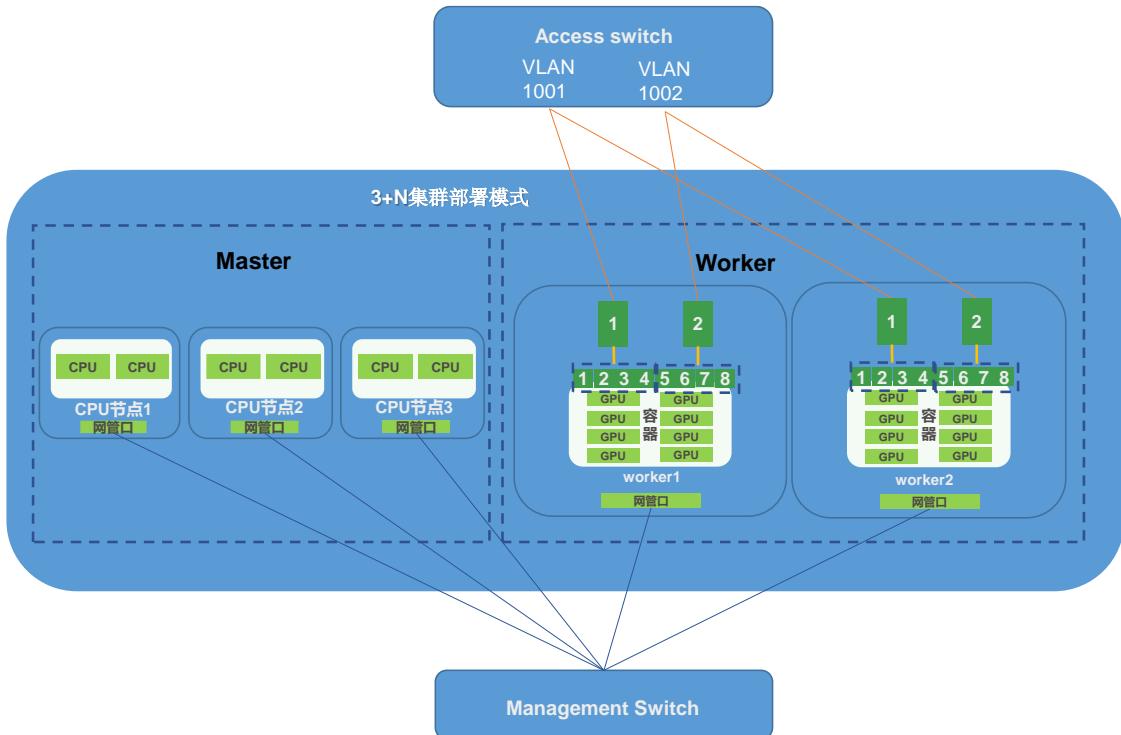


表2-3 IP 地址规划表

设备	接口	IP 地址	设备	接口	IP 地址
CPU节点1	ens1f1np1 (网管口)	192.168.0.1	worker1	ens1f1np1 (网管口)	192.168.0.4
CPU节点2	ens1f1np1 (网管口)	192.168.0.2		ens2f1np1 (业务口)	20.20.1.1/24
CPU节点3	ens1f1np1 (网管口)	192.168.0.3		ens3f1np1 (业务口)	20.20.2.1/24
			worker2	ens1f1np1 (网管口)	192.168.0.6
				ens2f1np1 (业务口)	20.20.1.2/24
				ens3f1np1 (业务口)	20.20.2.2/24

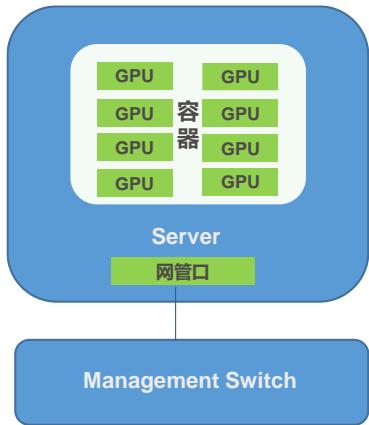


不同型号的服务器，采用不同的服务器拓扑结构，具体结构请参照服务器技术白皮书。

### 2.3.3 单机组网规划

单机组网是指将所有的数据和计算资源都集中在单个计算节点上进行处理，能够有效处理规模较小的模型推理任务，可以更好地利用单个计算节点的计算资源，简化管理和维护工作。

图2-7 单机组网示意图



## 2.4 IP地址规划

安装部署 LinSeer RT 的 IP 地址，规则如[表 2-4](#) 所示

表2-4 IP 地址规划表

IP 地址规划	用途	备注
Master节点1IP地址	为安装了操作系统的Master节点分配的IP地址	必选 加入同一集群的所有Master节点的IP地址必须处于同一网段 单机部署模式只需要一个Master节点
Master节点2IP地址	为安装了操作系统的Master节点分配的IP地址	
Master节点3IP地址	为安装了操作系统的Master节点分配的IP地址	
北向业务虚IP地址	通过Matrix页面部署的应用的统一外部访问地址	必选 北向业务虚IP必须在Master节点所处的网段内
Worker节点IP地址	为Worker节点分配的IP地址	可选 Worker节点的IP地址必须与加入同一集群的Master节点的IP地址处于同一网段

## 2.5 应用安装包准备

LinSeer RT 需要准备的安装包如[表 2-5](#) 所示。其中 *version* 为版本号，*platform* 为 CPU 架构类型。



### 注意

不同 GPU 环境下，执行安装包的顺序有所不同：

- NVIDIA GPU 环境：操作系统安装包->Matrix 集群安装包->linseer-rt-base 基础安装包->linseer-rt-inference 安装包。
  - 昆仑芯 GPU 环境：操作系统安装包->Matrix 集群安装包->linseer-rt-base 安装包->linseer-rt-inference 安装包->linseer-rt-kunlunxin-inference 安装包。
  - 天数 GPU 环境：操作系统安装包->Matrix 集群安装包->linseer-rt-base 安装包->linseer-rt-inference 安装包->linseer-rt-tianshu-inference 安装包。
- 

表2-5 应用安装包说明

安装包名称	功能说明	说明
NingOS-version.iso	H3C自研的NingOS操作系统的安装包	必选 NingOS或KylinOS 二选一
Kylin-version.iso	麒麟操作系统的安装包	
Matrix_version_platform.zip	Matrix集群安装包	必选
linseer-rt-base_version.zip	基础安装包，包括监控、日志管理、License 等功能	必选
linseer-rt-inference_version.zip	百业灵犀推理引擎组件安装包，提供模型仓库、模型推理等功能	必选
linseer-rt-kunlunxin-inference_version.zip	昆仑芯推理组件安装包，提供昆仑芯推理等功能	可选
linseer-rt-tianshu-inference_version.zip	天数推理组件安装包，提供天数推理等功能	可选
LinSeer-Model-2.1-AWQ-INT4-version.zip	LinSeer2.1量化模型包	可选
LinSeer-Model-2.1-Lite-version.zip	LinSeer2.1-Lite模型包	可选
LinSeer-Model-2.2-version.zip	LinSeer2.2模型包	可选
LinSeer-Model-Ex-version.zip	LinSeerEx模型包	可选
LinSeer-Model-Code-Lite-version.zip	LinSeer Code Lite模型包	可选
LinSeer-Model-TS-Gauss-version.zip	时序模型	可选
LICENSE_SERVER-version.zip	License Server服务安装包	必选

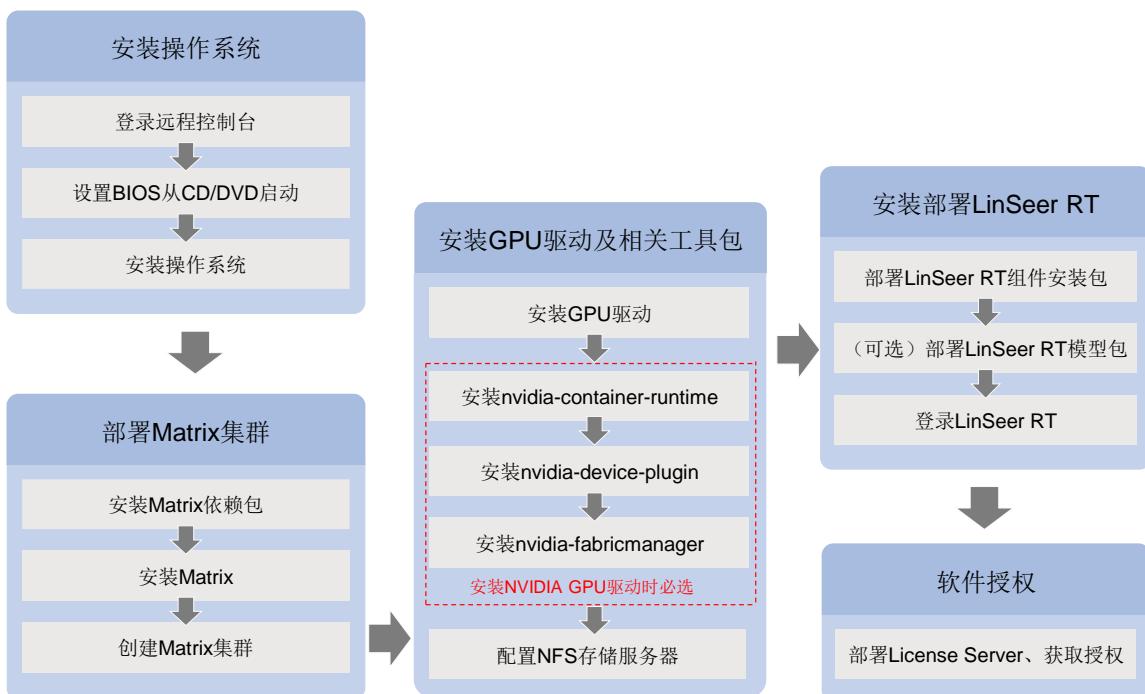
## 3 LinSeer RT 部署流程

部署 LinSeer RT，需要包含以下几个主要部署步骤：

- 安装操作系统：为安装和配置其他软件和程序提供基础环境。LinSeer RT 支持安装 H3C 自研的 NingOS 操作系统和 kylinOS 操作系统（二选一）。
- 部署 Matrix 集群：提供一个大规模计算资源的集群环境，以满足高性能高可靠计算和大规模数据处理的需求。不同操作系统，Matrix 集群部署步骤不同。部署时，请参见操作系统对应的 Matrix 集群部署章节。
- 安装 GPU 驱动及相关工具包：可以确保 GPU 资源被正确地识别和调用，从而支持 AI 模型推断等计算密集型任务，可充分发挥 GPU 的计算能力。
- （可选）安装网卡驱动及 ROCE 配置。仅分布式推理组网须要安装。
- （可选）安装 k8s-rdma-shared-dev-plugin。仅分布式推理组网须要安装。
- （可选）部署 PolarDB 数据库。
- 安装部署 LinSeer RT：可实现对已部署模型的监控、管理与升级，帮助管理者更好地监控和管理 LinSeer RT 的部署状态，确保其正常运行。
- 软件授权：可确保 LinSeer RT 安装部署后的正常使用。LinSeer RT 提供 License client 功能，License Server 可为 LinSeer RT 管理授权。

LinSeer RT 主要部署流程如图 3-1 所示。

图3-1 LinSeer RT 主要部署流程



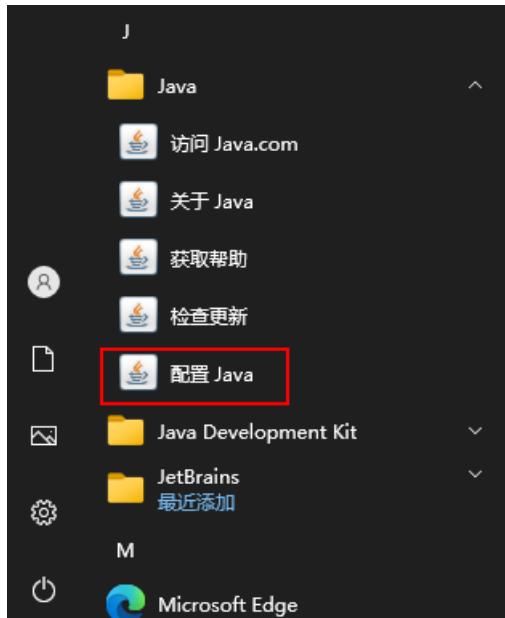
# 4 安装 NingOS 操作系统

## 4.1 登录远程控制台

### 4.1.1 配置 Java 环境

- (1) 在本地 PC 机上安装 JDK，推荐版本为 8u181。
- (2) 如图 4-1 所示，单击[开始]菜单选择“配置 Java”，进入 JAVA 控制面板。

图4-1 配置 Java



- (3) 把 HDM 的 URL 加入“例外站点”列表。如图 4-2 所示。
  - a. 选择“安全”页签，单击<编辑站点列表>按钮进入“例外站点”列表。
  - b. 在“例外站点”列表对话框中单击<添加>按钮，。
  - c. 输入 HDM 的 URL，单击<确定>按钮完成添加。
  - d. 单击 JAVA 控制面板上的<确定>按钮完成设置。

图4-2 更新例外站点列表



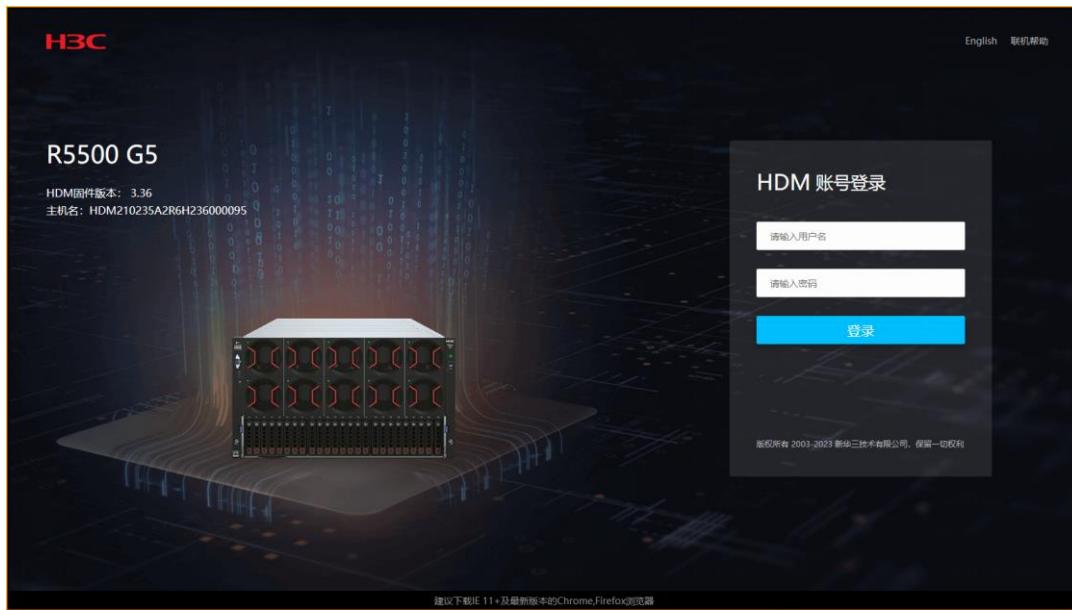
#### 4.1.2 登录 HDM Web 页面

HDM (Hardware Device Management, 设备管理系统) 用于远程管理服务器。

登录 HDM 操作步骤如下：

- (1) 将网线连接到服务器的 HDM 专用网口，确保本地 PC 与服务器之间网络可达。
- (2) 在本地 PC 浏览器地址栏中输入“<https://HDM 专用网口 IP 地址>”，按<Enter>键打开登录页面。输入用户名和密码，单击<登录>按钮登录服务器 HDM Web 页面，如图 4-3 所示。

图4-3 登录服务器 HDM Web 页面示意图



#### 说明

- HDM Web 操作页面、默认 HDM 专用网口 IP 地址、默认用户名/密码等与服务器型号和 HDM 版本有关，详细信息请参见对应版本的服务器 HDM 用户指南。
- 对于本文示例版本，默认 HDM 专用网口 IP 地址为 192.168.1.2/24，默认用户名/密码为 admin/Password@\_。
- 若已修改默认 HDM 专用网口 IP 地址、用户名/密码，请以修改后的 HDM 专用网口 IP 地址、用户名/密码登录。

### 4.1.3 连接远程控制台并挂载镜像

本步骤用于使用 Java 集成远程控制台连接服务器。通过 JVViewer 或 KVM 远程连接软件，您可以在远程进行配置管理服务器、安装操作系统等操作。

对于 R5300 G5/R5500 G5/R5500 G6/R5350 G6 服务器，连接远程控制台的操作步骤如下：

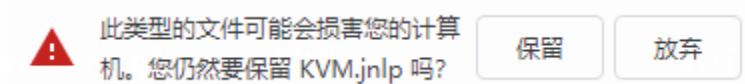
- (1) 在 HDM Web 页面单击顶部[远程服务]菜单项，再单击左侧导航树中的[远程控制台]进入远程控制台页签。单击<启动 KVM>按钮下载 KVM 软件，如图 4-4 所示。

图4-4 下载远程连接软件



- (2) 如图 4-5 所示, 在弹出的提示框中单击<保留>按钮, 开始下载 KVM 软件。

图4-5 远程连接软件下载提示信息



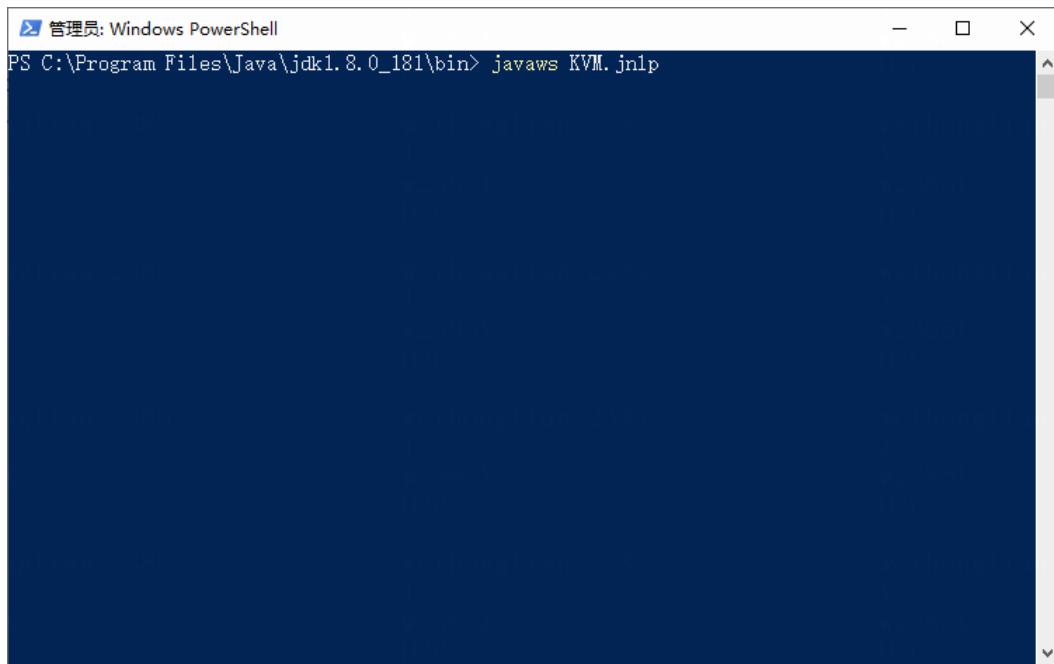
- (3) 将已下载的 KVM.Jnlp 文件拷贝至 JDK 的安装目录“C:\Program Files\Java\jdk1.8.0\_181\bin”下。
- (4) 按键盘<Shift>键, 同时鼠标右键单击文件夹空白处, 在下拉菜单中选择“在此处打开 Powershell 窗口”, 如图 4-6 所示。

图4-6 打开命令窗口



- (5) 在命令窗口执行命令 `javaws KVM.jnlp` 并按回车键, 如图 4-7 所示。

图4-7 执行命令



A screenshot of a Windows PowerShell window titled "管理员: Windows PowerShell". The command entered is "PS C:\Program Files\Java\jdk1.8.0\_181\bin> javaws KVM.jnlp". The window has a dark blue background and standard Windows-style controls at the top.

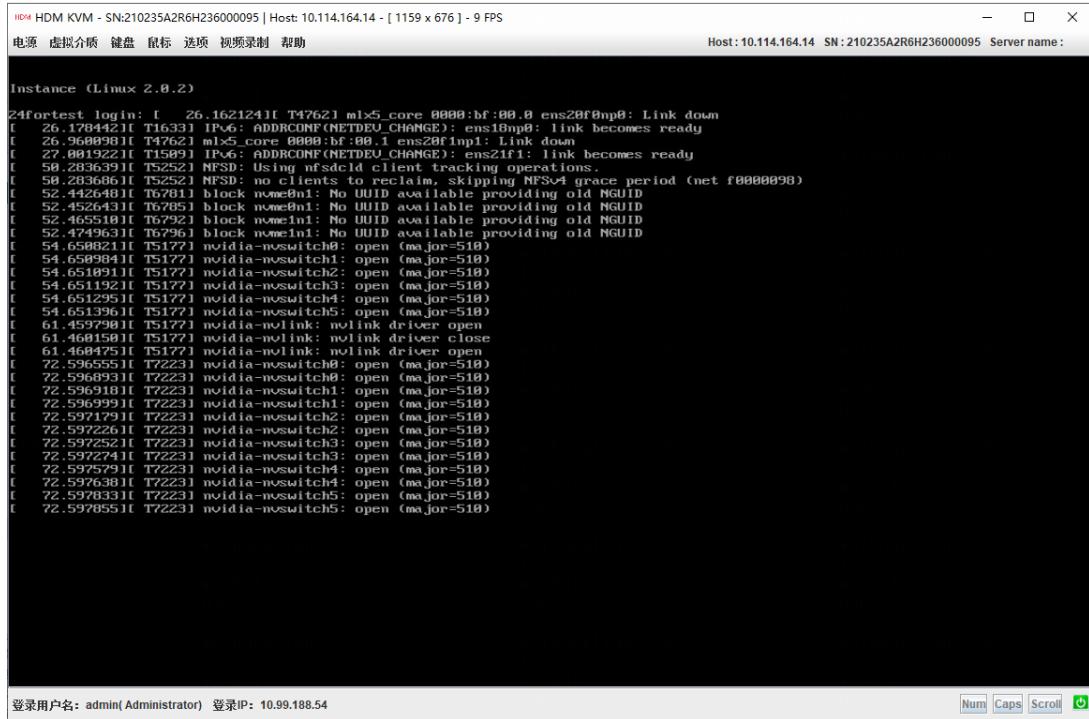
- (6) 在安全警告窗口中点击<继续>按钮。如图 4-8 所示。

图4-8 安全警告窗口



- (7) 成功连接远程控制台，可显示远程控制台窗口，如图 4-9 所示。

图4-9 连接远程控制台



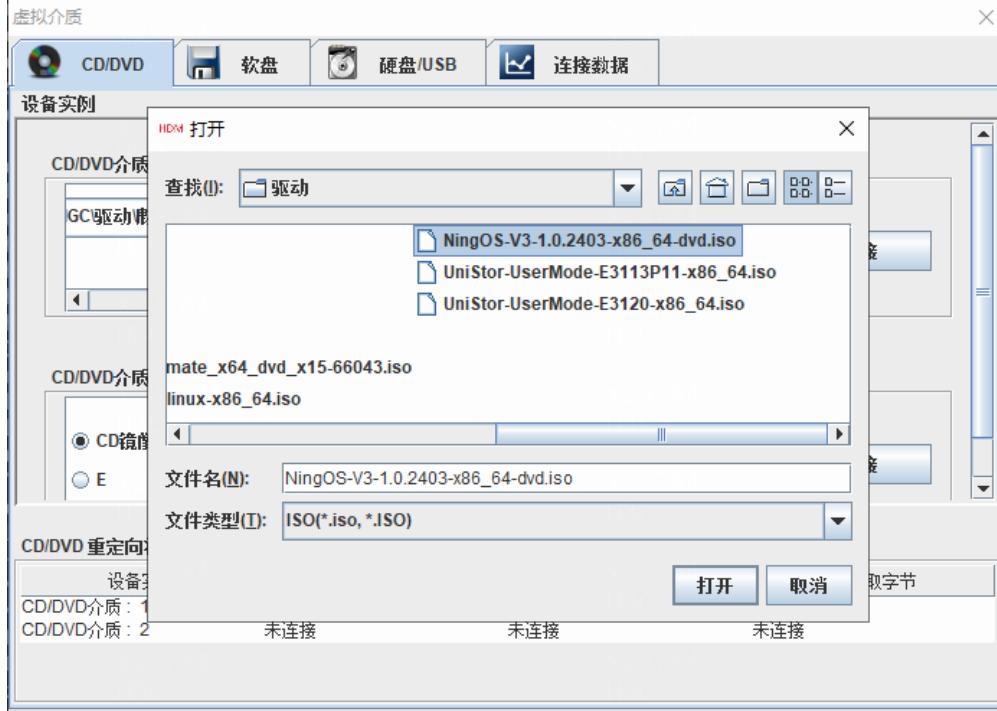
- (8) 在远程控制台窗口单击[虚拟介质->虚拟介质向导]菜单项，弹出虚拟介质对话框，如图 4-10 所示，单击<浏览>按钮。

图4-10 虚拟介质对话框



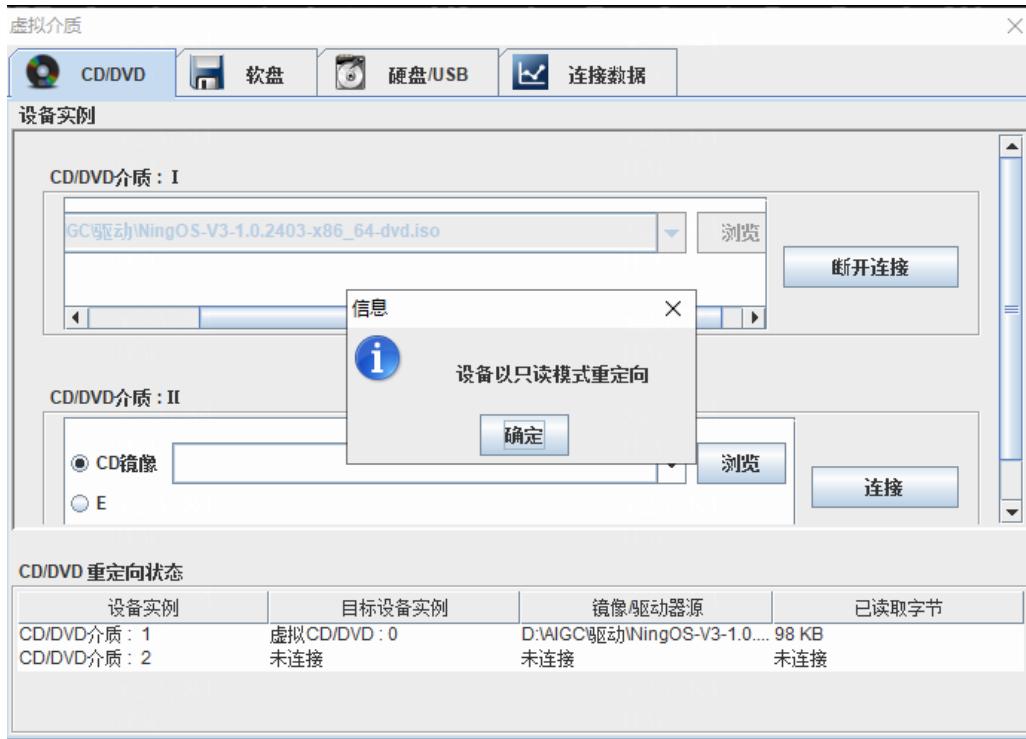
(9) 选择本地的操作系统镜像文件 (.iso 格式)，单击<打开>按钮。如图 4-11 所示。

图4-11 打开对话框



(10) 在弹出的提示信息对话框内单击<确定>按钮，即可成功挂载镜像文件，如图 4-12 所示。

图4-12 确认提示信息



## 4.2 设置BIOS从CD/DVD启动

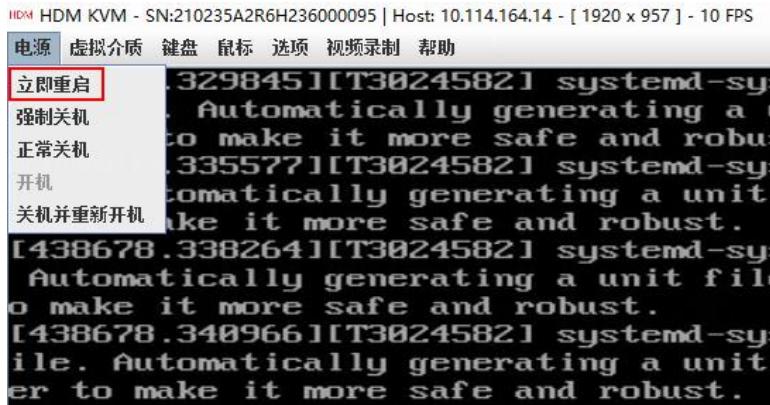


说明

- 设置 BIOS 从 CD/DVD 启动需通过键盘操作。
- 不同型号服务器参数配置界面有所不同，本节仅以 R5500 G5 服务器为例。
- 参数配置界面可能会不定期更新，请以设备实际显示为准。

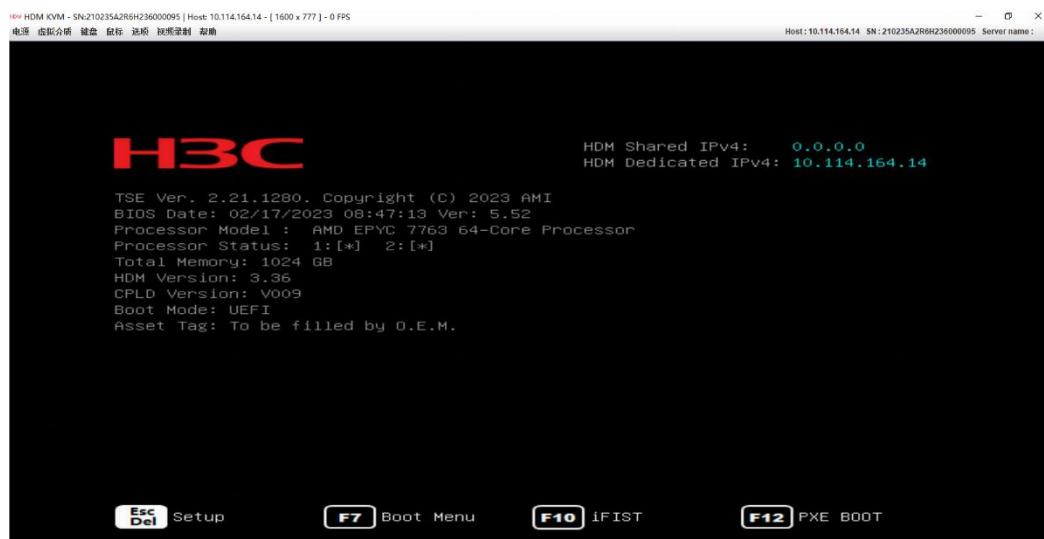
(1) 单击选择[电源 -> 立即重启]菜单项，重启服务器，如图 4-13 所示。

图4-13 重启服务器



- (2) 服务器重启后，按键盘~~DEL~~键或~~ESC~~键进入 BIOS 设置界面，如图 4-14 所示。

图4-14 进入 BIOS 设置界面



- (3) 通过按键盘~~<--><-->~~键移动选择[Boot]菜单项，按~~Enter~~键设置 BIOS，如图 4-15 所示。

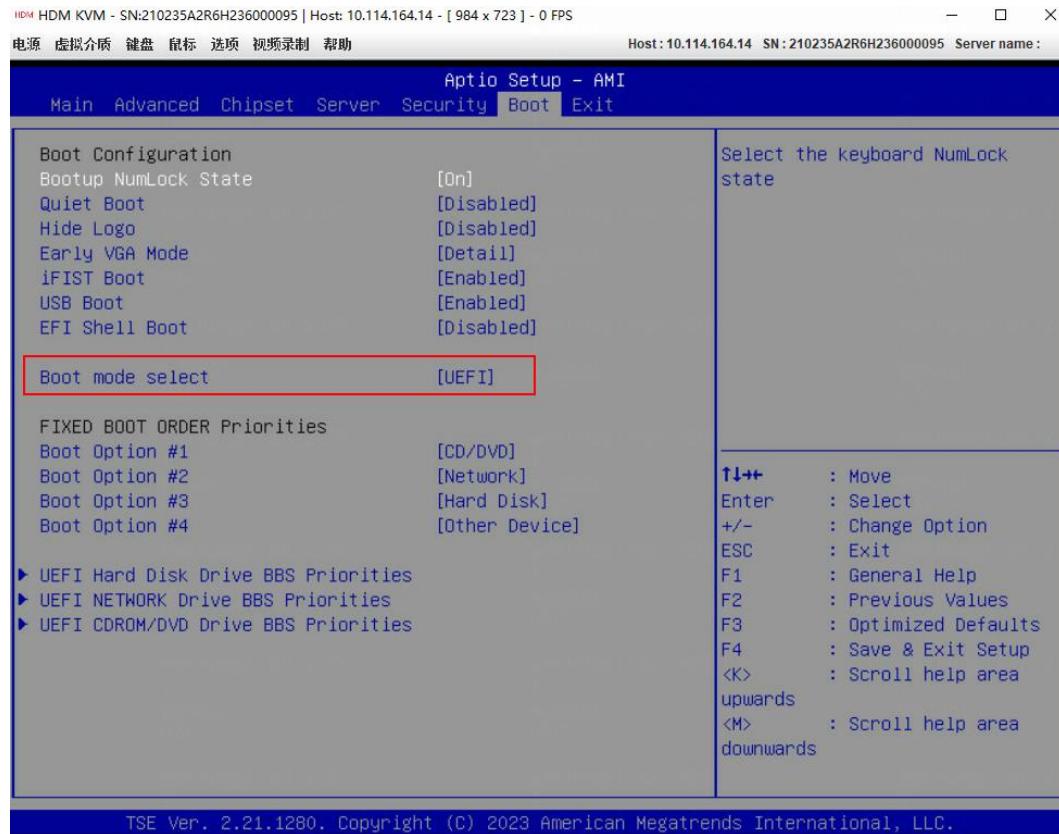
图4-15 设置 BIOS



(4) 设置 Boot 启动模式。

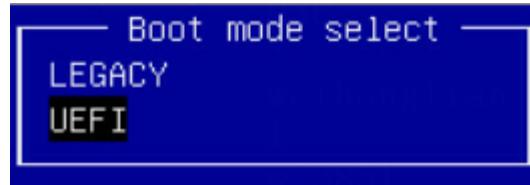
- a. 通过按键盘 $\uparrow$  $\downarrow$ 键移动选择“Boot mode select”配置项，按 $<\text{Enter}>$ 键设置 Boot 启动模式，如图 4-16 所示。

图4-16 设置 Boot 启动模式 (1)



- b. 在弹出的“Boot mode select”对话框中，按 $<\text{Enter}>$ 键选择“UEFI”。如图 4-17 所示。

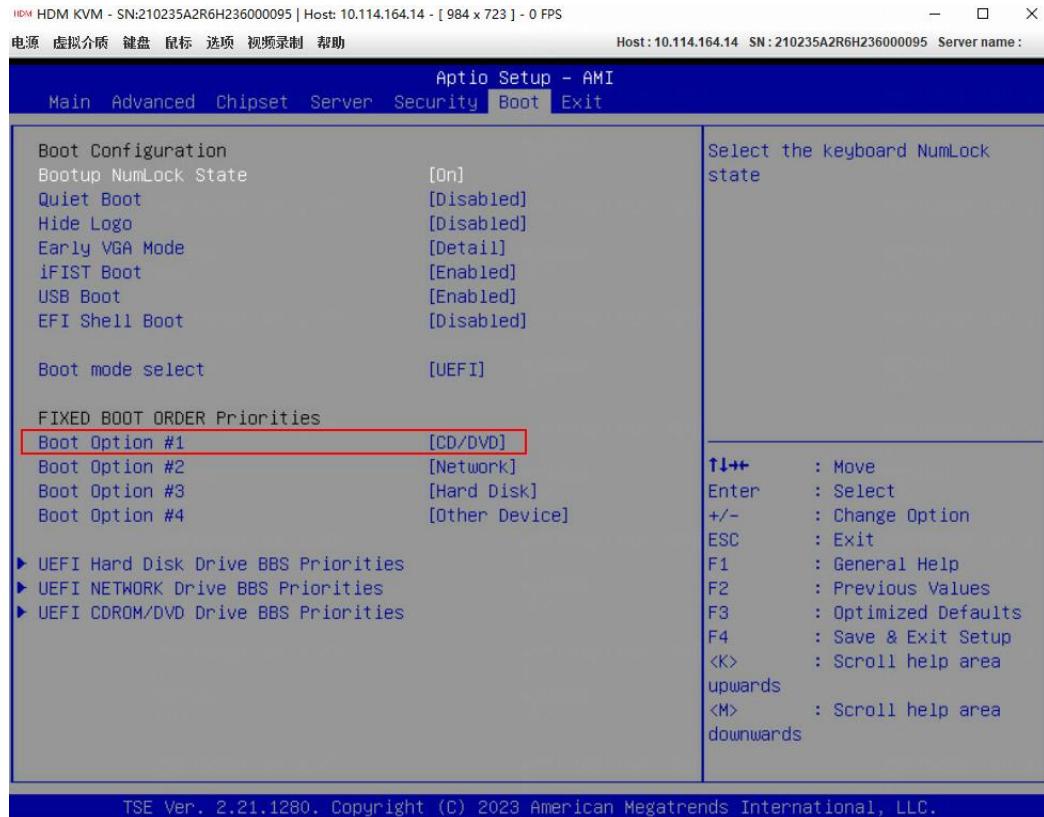
图4-17 设置启动 Boot 模式 (2)



(5) 设置 Boot 启动项。

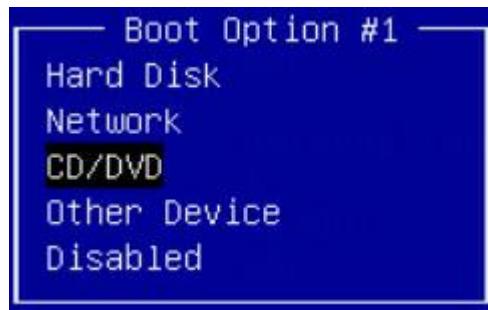
- a. 通过按键盘 $\uparrow$  $\downarrow$ 键移动选择“Boot Option #1”配置项，按 $<\text{Enter}>$ 键设置 Boot 启动项，如图 4-18 所示。

图4-18 设置 Boot 启动项（1）



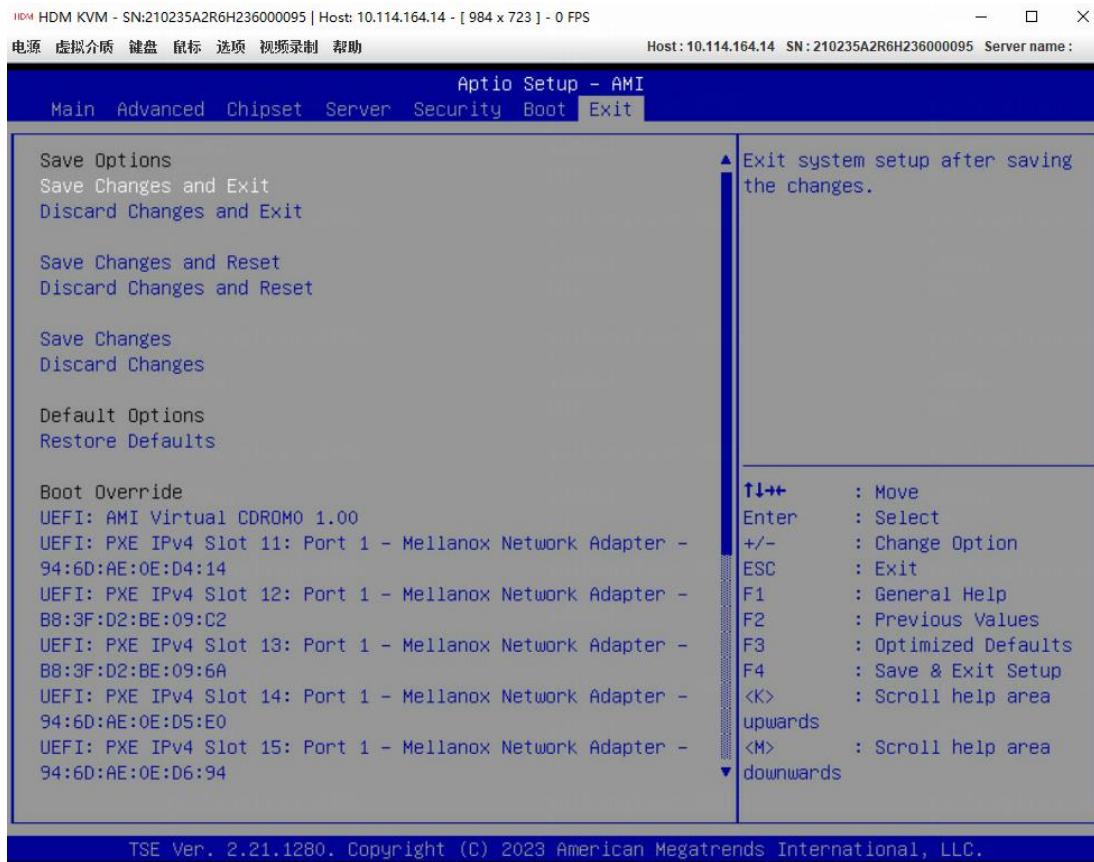
- b. 在弹出的“Boot Option #1”对话框中，按<Enter>键选择“CD/DVD”，如图 4-19 所示。

图4-19 设置 Boot 启动项（2）



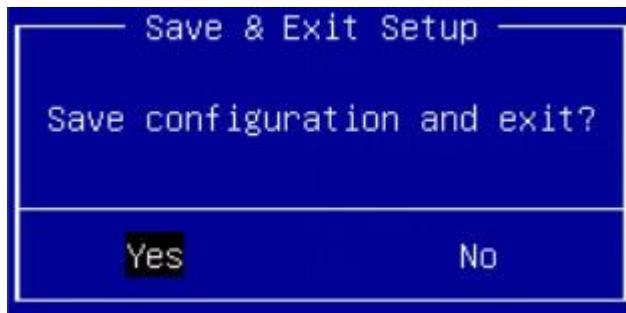
- (6) 通过键盘<--><-->键选择[Exit]菜单项，按<Enter>键保存设置，如图 4-20 所示。

图4-20 保存设置



- (7) 在弹窗的对话框中选择“YES”，保存设置并退出，如图4-21所示。

图4-21 保存设置并退出



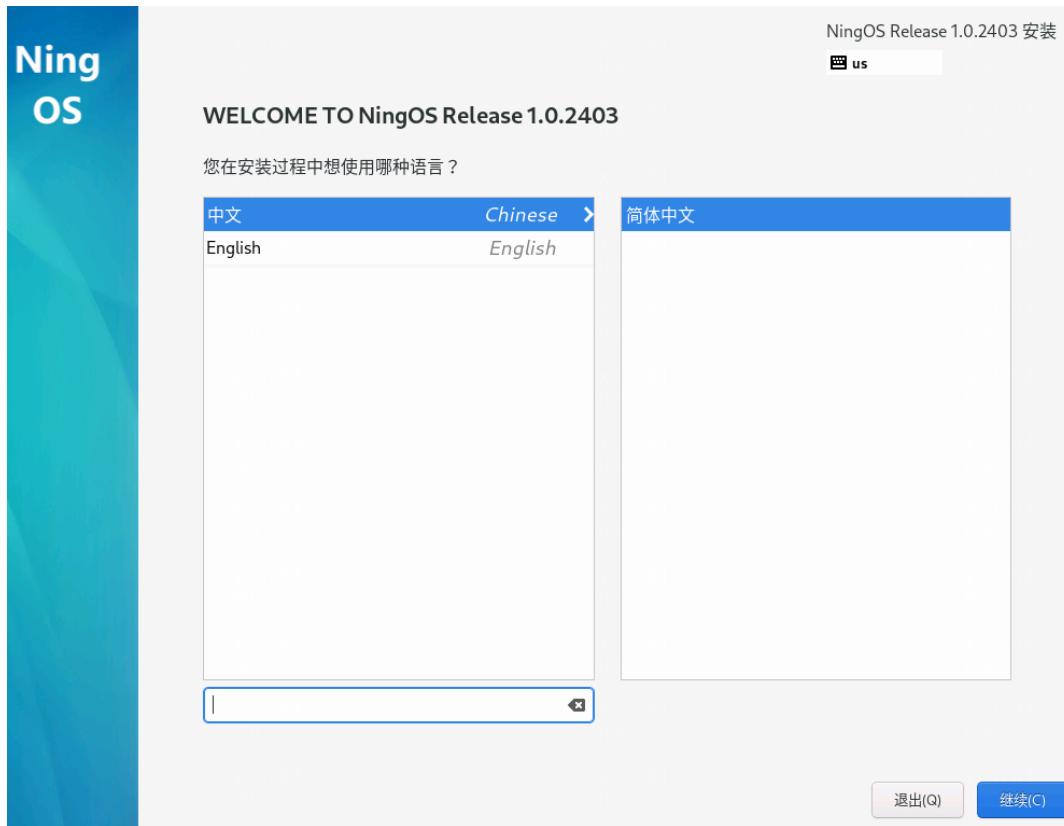
## 4.3 安装操作系统

本节以从未安装过任何操作系统的服务器为例，请使用 LinSeer RT 配套的 NingOS-v3-1.0.2403 ISO 文件安装操作系统。

#### 4.3.1 基本设置

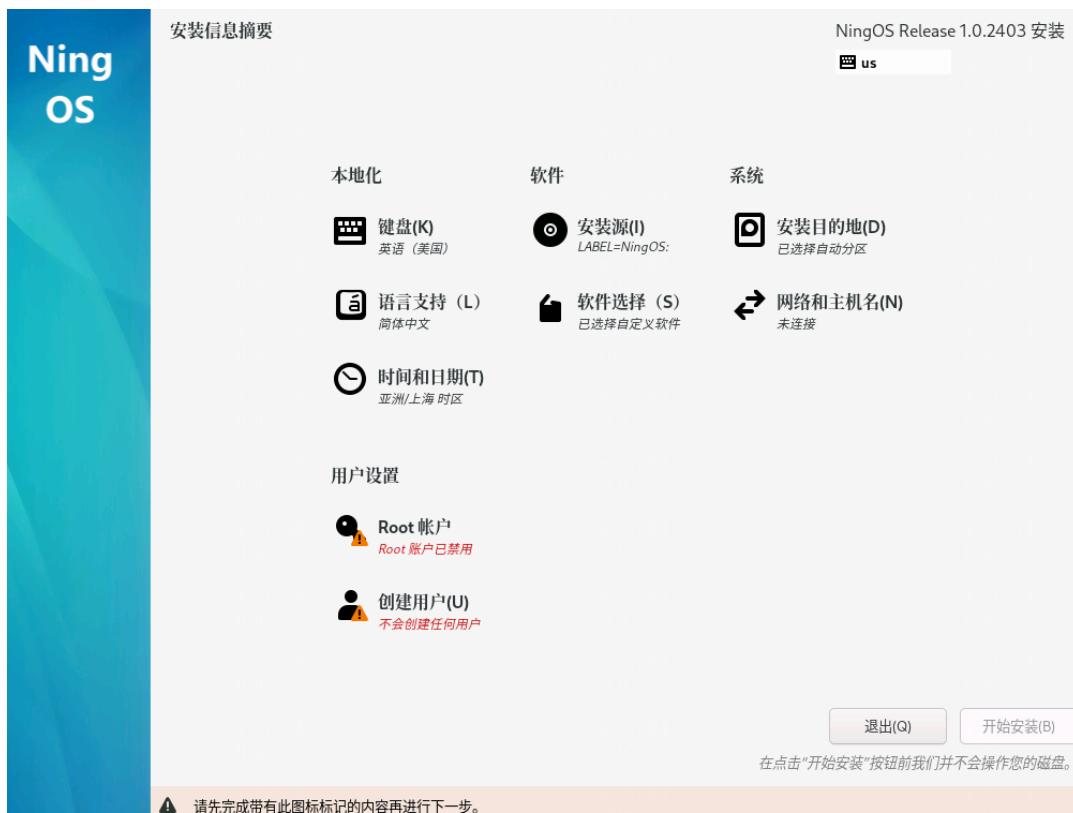
- (1) ISO 镜像文件加载完成后，进入语言选择界面。
- (2) 选择安装语言，此处以“中文”为例，单击<继续(C)>按钮，进入安装信息摘要界面，如图 4-22 所示。

图4-22 语言选择界面



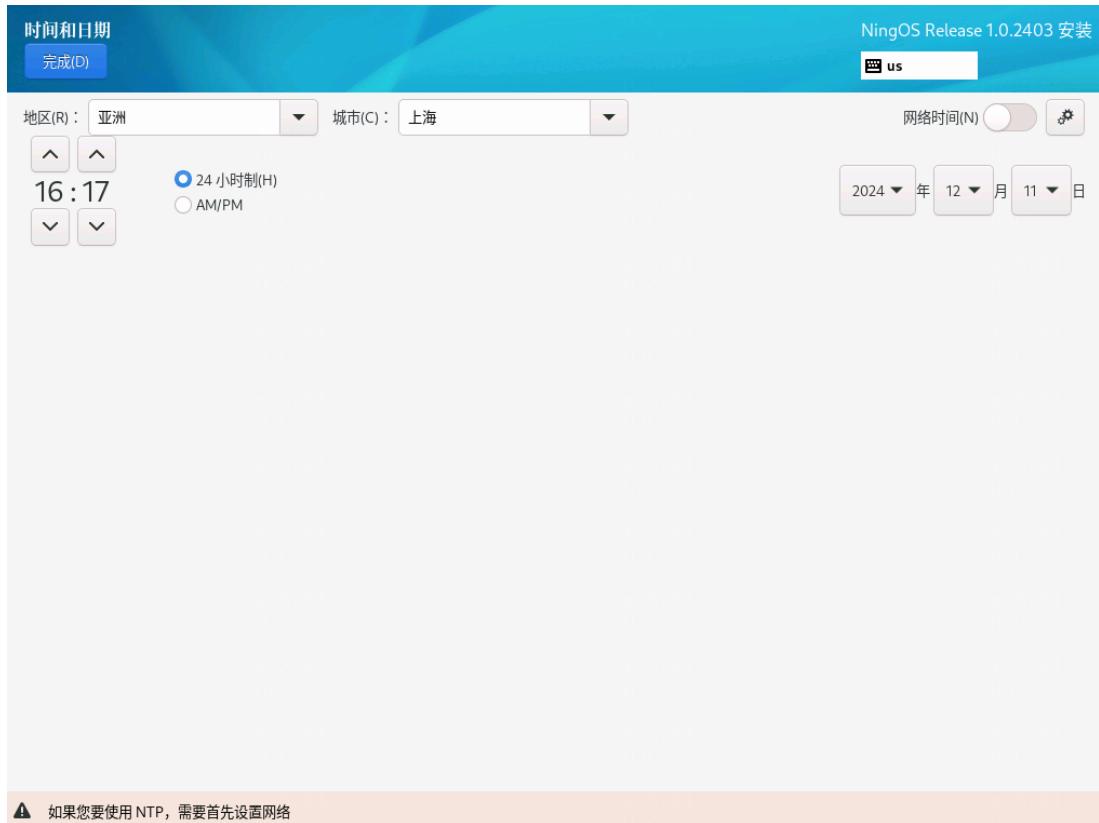
- (3) 在安装信息摘要界面的本地化区域单击“日期和时间(T)”链接，进入日期和时间界面，如图 4-23 所示。

图4-23 安装信息摘要界面



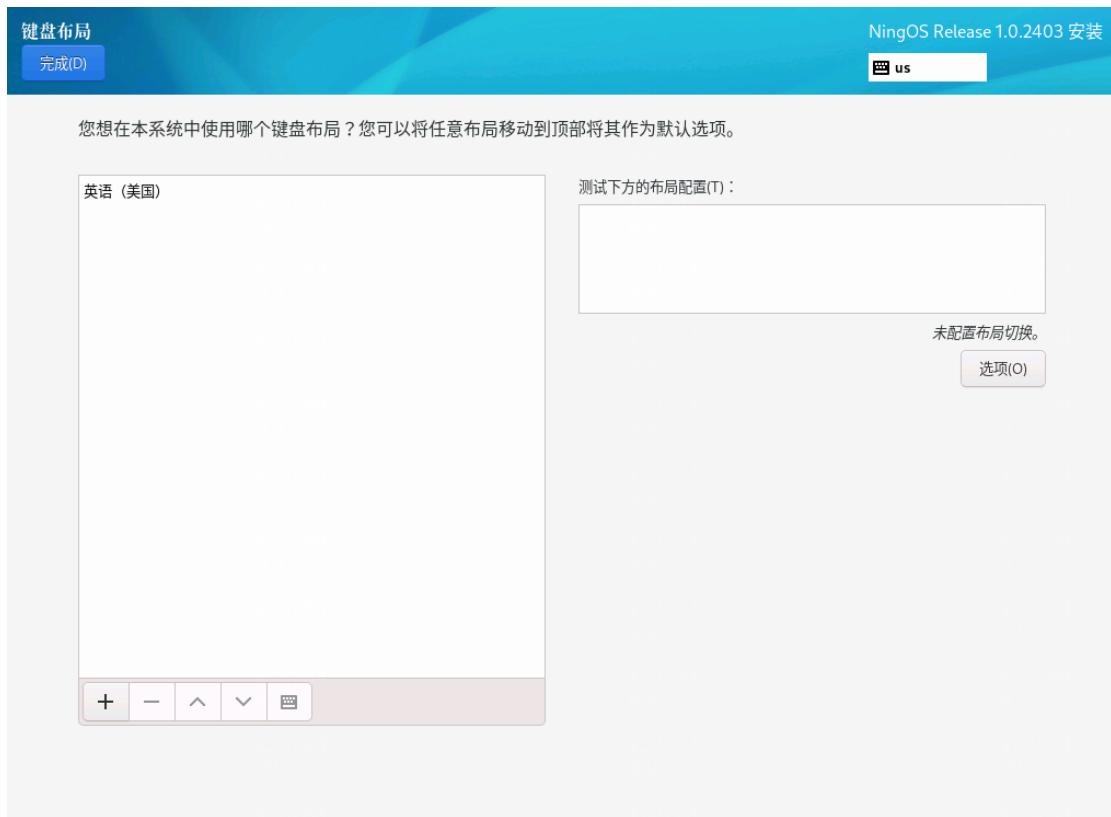
- (4) 设置系统的日期和时间，如图 4-24 所示。选择地区为“亚洲”，城市为“上海”，单击<完成>按钮，返回安装信息摘要界面。

图4-24 设置系统的日期和时间



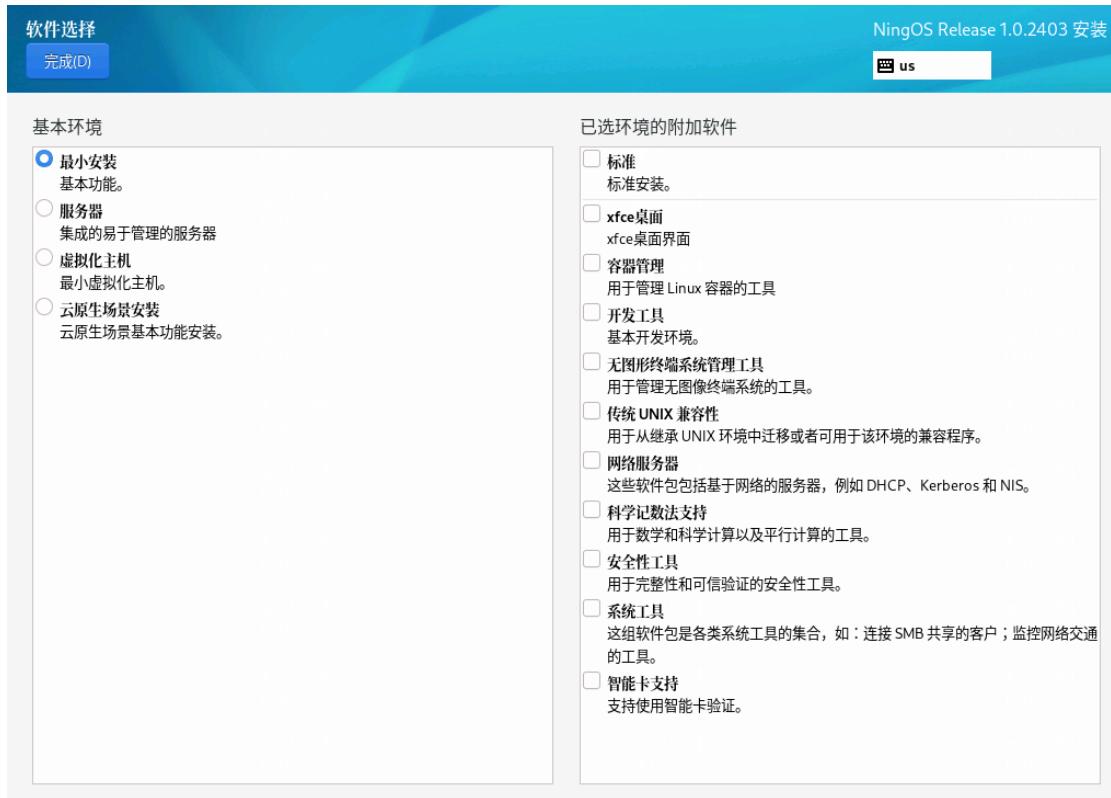
(5) 在安装信息摘要界面的本地化区域单击“键盘(K)”链接，进入键盘布局界面，如图 4-25 所示。

图4-25 设置键盘布局



(6) 安装信息摘要界面的软件区域“软件选择(S)”选择基本环境为“最小安装”，如图 4-26 所示。

图4-26 选择基本环境为“最小安装”



- (7) 在安装信息摘要界面的系统区域，单击“安装目的地(D)”链接，进入安装目标位置界面。设置安装目标位置，如图 4-27 所示。在本地标准磁盘区域选择目标磁盘。在存储配置区域选择“自定义(C)”，单击<完成>按钮，进入手动分区界面。

图4-27 安装目标位置界面



### 4.3.2 磁盘分区

在“新挂载点将使用以下分区方案(N)”下拉列表中选择分区方案，支持选择“标准分区”和“LVM”，建议选择“标准分区”。

- 使用“LVM”分区。  
在“设备类型”下拉列表中选择“LVM”，并依次创建各分区。
- 使用“标准分区”。  
手动创建磁盘分区，如图 4-28 所示。磁盘分区的具体信息如表 4-1 所示，支持手动修改配置信息。
  - 设备类型：选择“标准分区”，并单击 + 图标，在弹出窗口中进行如下配置后，单击<添加挂载点>按钮。
  - 挂载点：输入挂载点目录名称。
  - 期望容量：输入磁盘容量并指定容量单位，例如“GiB”、“MiB”。
  - 文件系统(Y)：在下拉列表中选择推荐的文件系统。
  - 设备：单击<修改(M)...>按钮，在弹出的配置挂载点窗口中，选择分区所挂载的磁盘后，单击<选择>按钮。



注意

boot 相关分区不能设置为 LVM，否则分区将会配置失败。

图4-28 手动进行磁盘分区（标准分区）

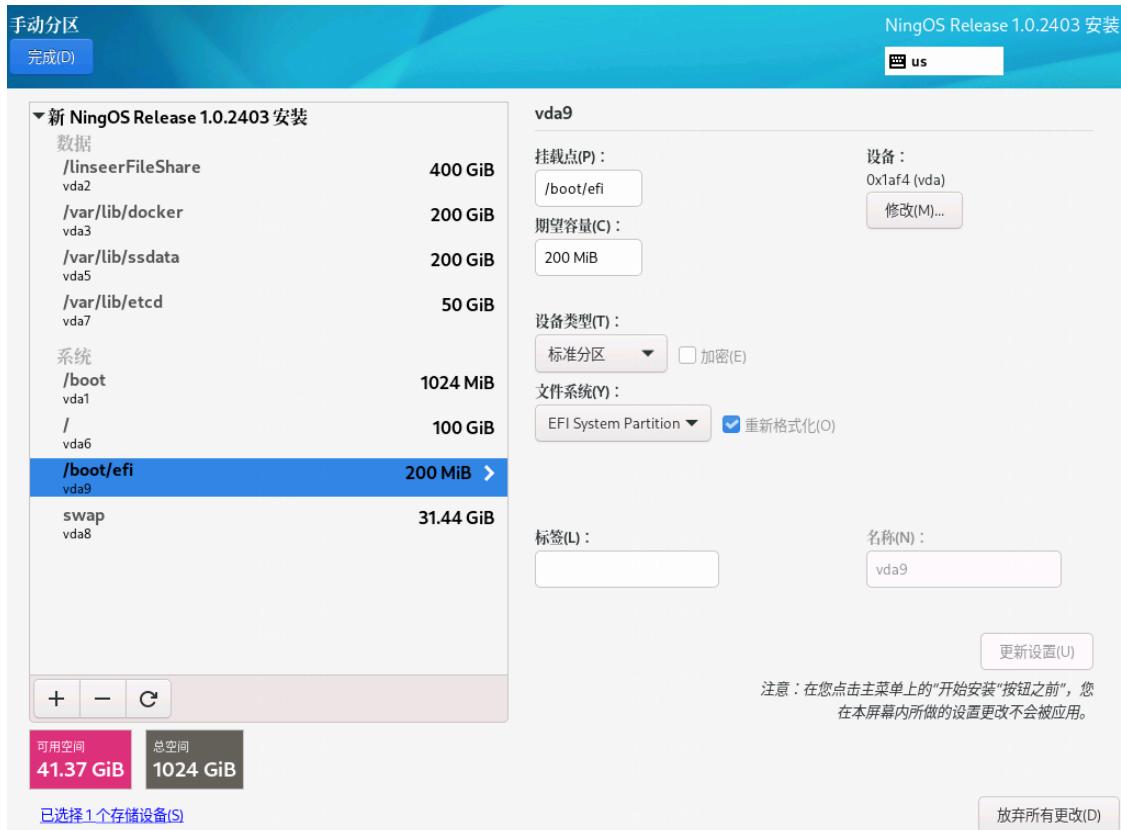


表4-1 磁盘分区要求

分区名称	预设容量	适用模式	文件系统	备注
/	1 TB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/docker	350 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/ssdata	200 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/etcdd	50 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/boot	1024 MiB	BIOS模式/UEFI模式	ext4	
swap	1024 MiB	BIOS模式/UEFI模式	swap	
/boot/efi	200 MiB	UEFI模式	EFI System Partition	根据服务器的BIOS启动模型选择创建对应分区，Legacy: /biosboot UEFI: /boot/efi
/linseerFileShare (默)	1 TB	BIOS模式/UEFI模式	ext4	单机环境需要创建该分区，磁

认名称)				盘空间充足时，可适当增大 集群环境使用NFS存储，不需要划分该分区
------	--	--	--	--------------------------------------

各主要分区的作用如下：

- **/:** Matrix 使用，包括 K8s、Harbor 和各组件上传的安装包，该分区的容量和各组件上传的镜像大小有关，需要确定各组件占用的磁盘容量大小，在此基础上扩缩容。
- **/var/lib/docker/:** 与 Docker 的运行相关，需要根据实际运行情况确定容量大小。
- **/var/lib/ssdata/:** 供 Kafka、ZooKeeper 使用。
- **/linseerFileShare**（默认名称）：推荐预设容量为 1TB。
  - 单机部署模式下，**/linseerFileShare** 供推理组件使用，作为模型仓库的共享目录。
  - 3 机集群部署模式、3+N 机集群部署模式下，使用外置的 NFS 存储空间供推理组件使用。

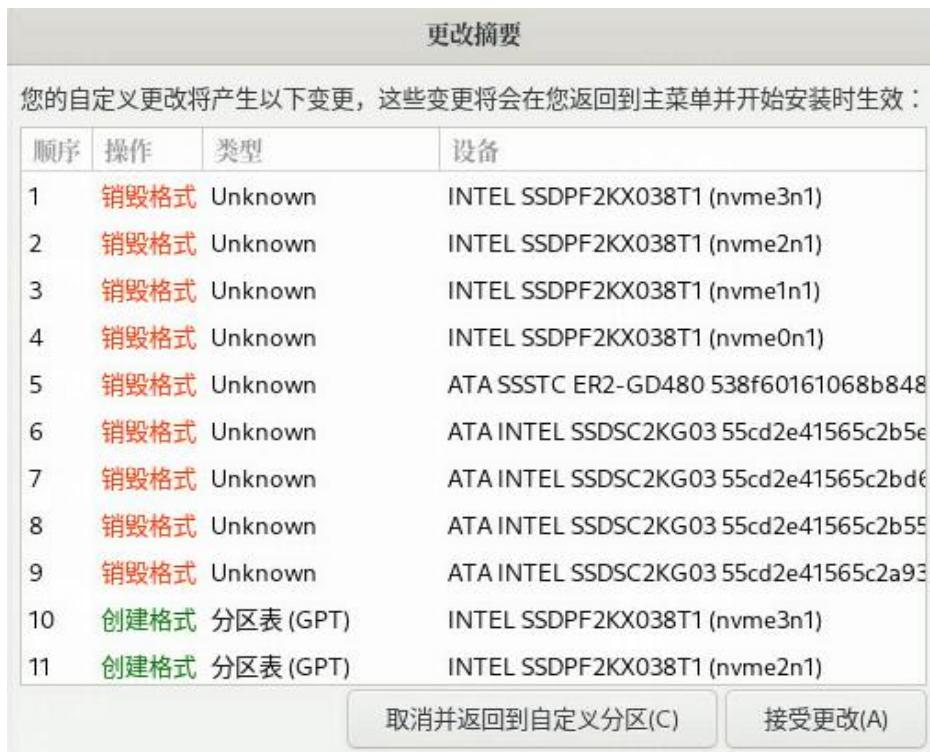
各部署模式下的“/linseerFileShare”磁盘空间分配详情，如[表 4-2](#) 所示。

**表4-2 “/linseerFileShare” 磁盘空间分配**

组件	目录名	存储空间大小计算方式	默认大小
模型仓库	\$(sharePath)/model_preset, 默认名称为“/linseerFileShare”	<ul style="list-style-type: none"> <li>● LinSeer_Model 文件: 132 GiB</li> <li>● LinSeer_Model_Lite 文件: 15 GiB</li> </ul>	300 GiB

在弹出的更改摘要窗口中，单击<接受更改>按钮，返回安装信息摘要界面，如[图 4-29](#) 所示。

图4-29 更改摘要窗口

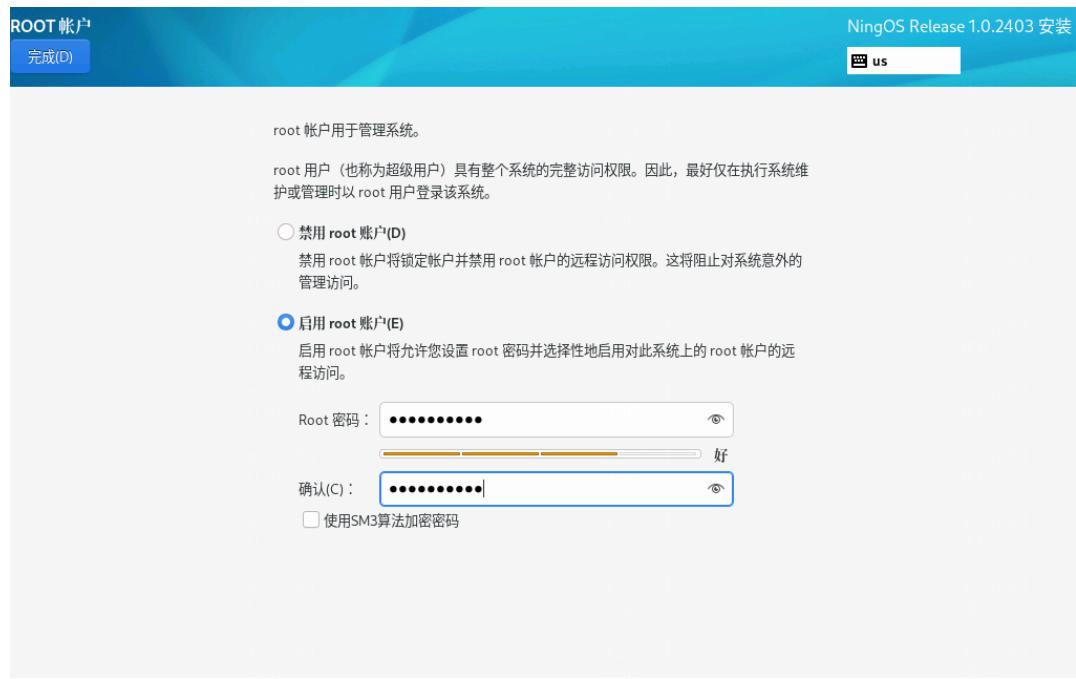


### 4.3.3 用户及网络设置

- (1) 选择管理员帐户设置。使用 **root** 用户作为管理员帐户，无需创建新用户。
  - a. 在安装信息摘要界面用户设置区域单击“Root 账户”配置项设置 **root** 用户作为管理员帐户，如图 4-23 所示。

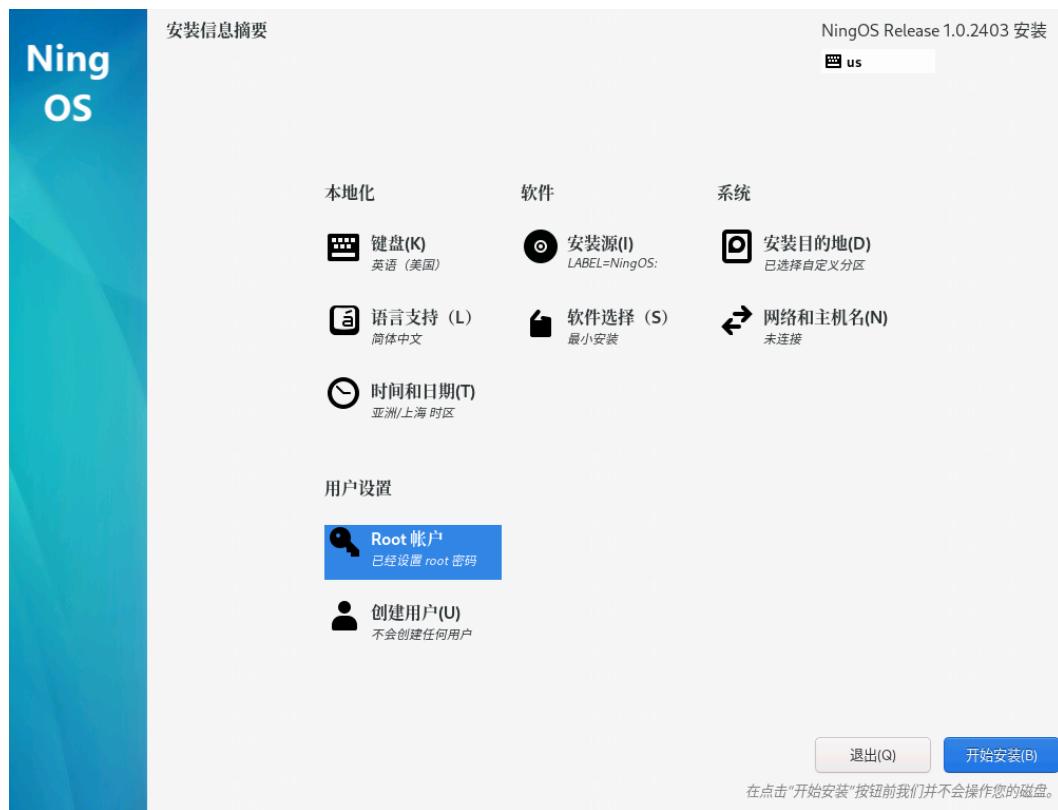
该功能用于选择安装 Matrix 并创建集群时使用的用户名，如需部署 Matrix 集群，需为集群中的所有节点选择相同的用户名。
  - b. 选择“启用 root 账户”并设置 **root** 用户密码，单击<完成>按钮，返回安装信息摘要界面，如图 4-30 所示。

图4-30 设置 root 用户密码



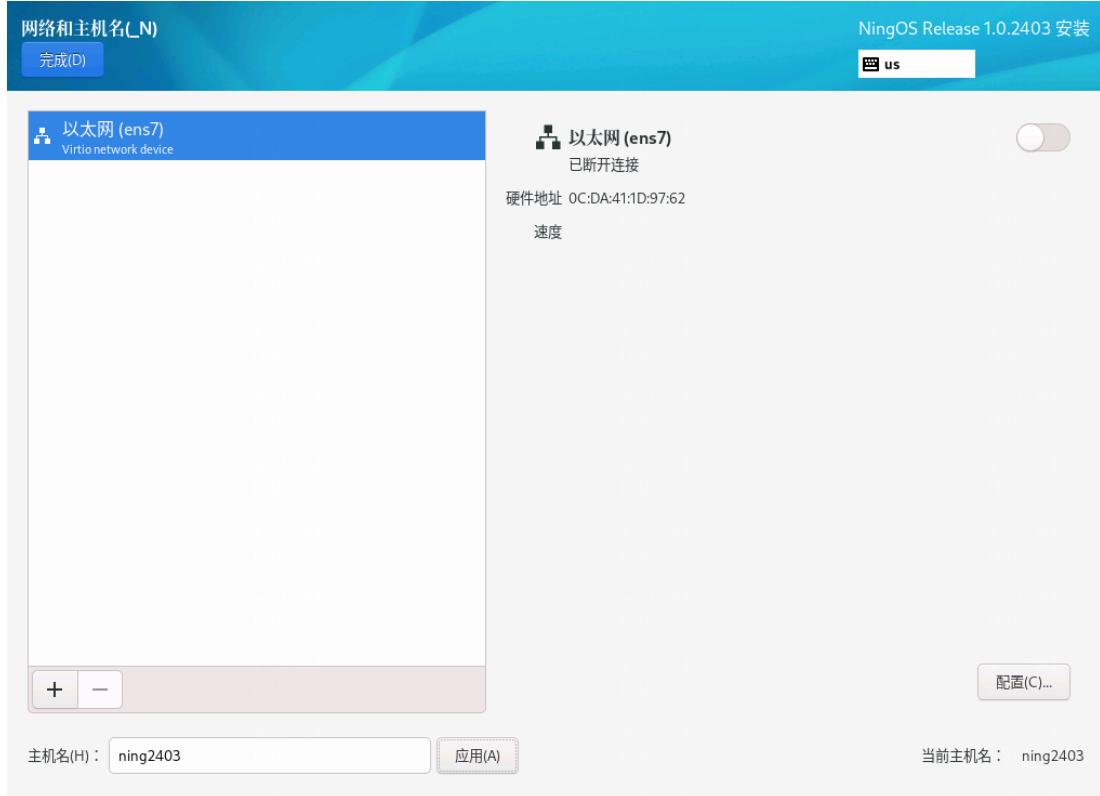
- c. 在安装信息摘要界面可查看 root 用户已作为管理员帐户，该用户拥有所有功能的操作权限，如图 4-31 所示。

图4-31 使用 root 用户作为管理员帐户



- (2) 在系统区域单击“网络和主机名(N)”链接，进入网络和主机名界面。
- (3) 在主机名文本框中输入主机名后，单击<应用>按钮，如图 4-32所示。

图4-32 网络和主机名配置界面



### 说明

- 请勿使用默认主机名（localhost、localhost.localdomain、localhost4、localhost4.localdomain4、localhost6、localhost6.localdomain6）。主机名称最长 63 个字符，仅支持小写字母、数字、连字符和小数点，不能以 0 开头且全为数字，不能以 0x、连字符、小数点开头，以连字符、小数点结尾。
- 建立 Matrix 集群时，必须保证集群内各个节点的主机名互不相同，且符合主机名的命名规则，否则将会导致集群建立失败。
- Matrix 集群部署前，若需要修改节点的主机名，可在节点操作系统的命令行界面，通过 `hostnamectl set-hostname hostname` 命令进行修改，其中 `hostname` 为修改后的主机名。新主机名将在节点重启后生效。
- Matrix 集群部署完成后，请不要再对操作系统的主机名进行修改。

- (4) 在网络和主机名配置界面可配置网卡。单击<配置>按钮，在弹出的网络配置窗口中进行网卡配置。单击“常规”页签，勾选“自动以优先级连接(A)”项，“所有用户都可以连接这个网络(U)”项保持默认勾选，如图 4-33 所示。

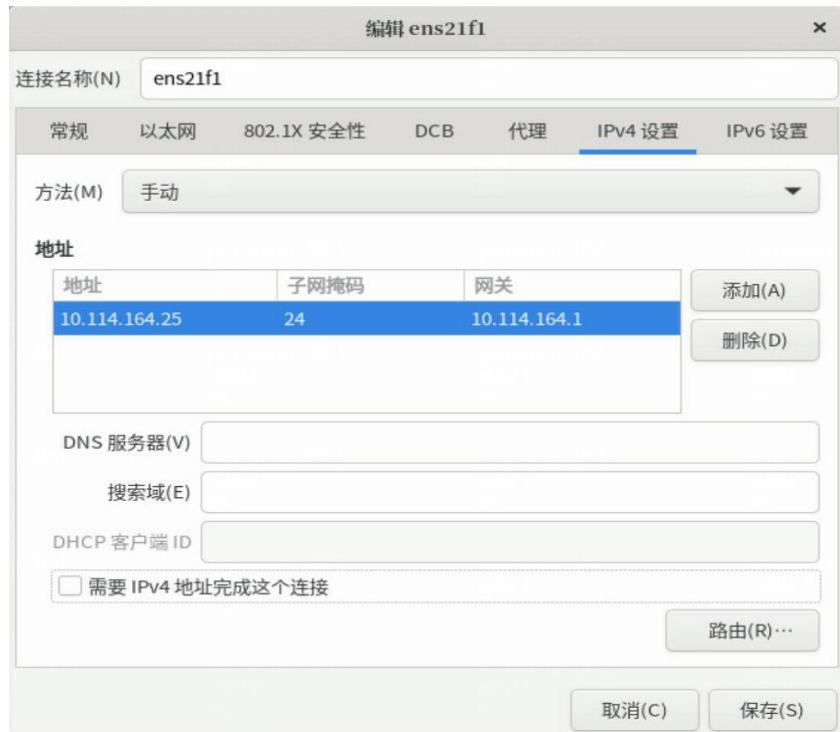
图4-33 常规页签



(5) 配置 Matrix 节点的 IPv4 地址。

单击“IPv4 设置”页签，在“方法(M)”下拉框中选择“手动”，在地址区域单击<添加(A)>按钮，配置服务器的 IPv4 地址（规划的 Matrix 节点 IP），配置完成后，单击<保存>按钮保存配置，如图 4-34 所示。

图4-34 配置服务器的 IPv4 地址



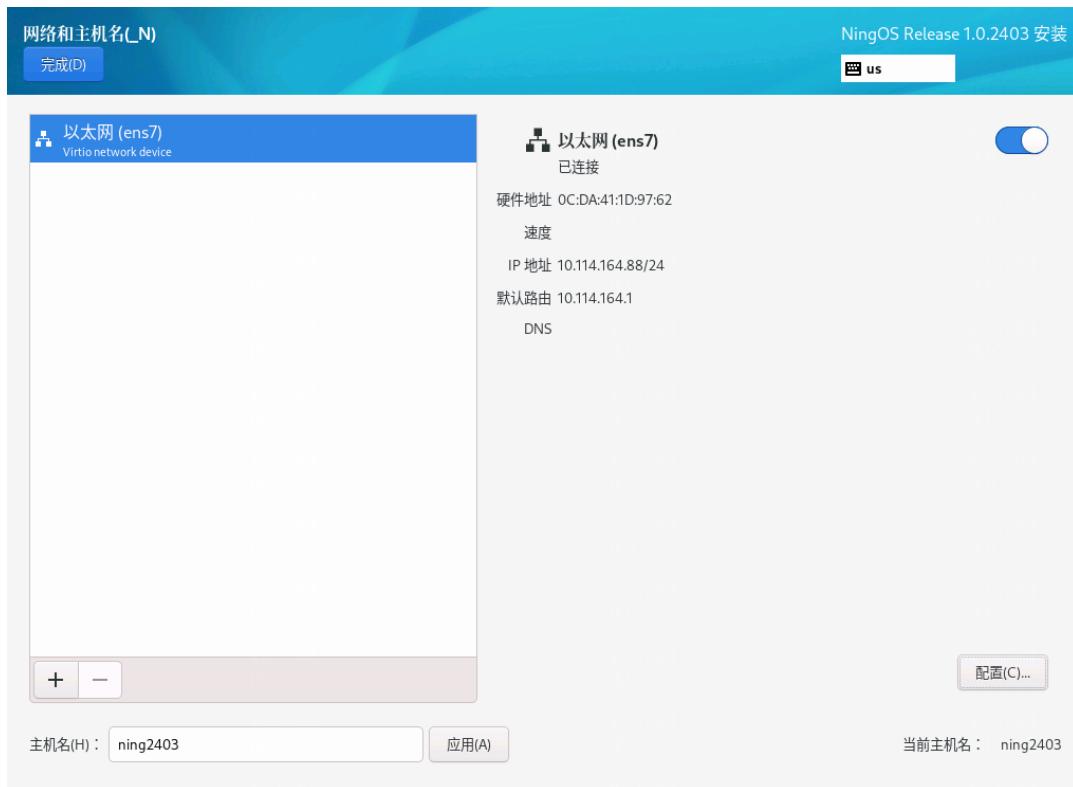
---

说明

- 配置 IPv4 地址时必须指定网关，否则在创建集群时可能出现问题。
  - Matrix 单独使用一个网口，不允许在此网口上配置子接口及子 IP。
  - Matrix 节点其它网口的 IP 地址，不能和建立集群使用的 IP 处于同一网段。
  - 不允许在操作系统中配置 DNS 服务器。
- 

- (6) 网络配置完成后，手动启用指定物理网卡，如图 4-35 所示。单击<完成>按钮，返回安装信息摘要界面。

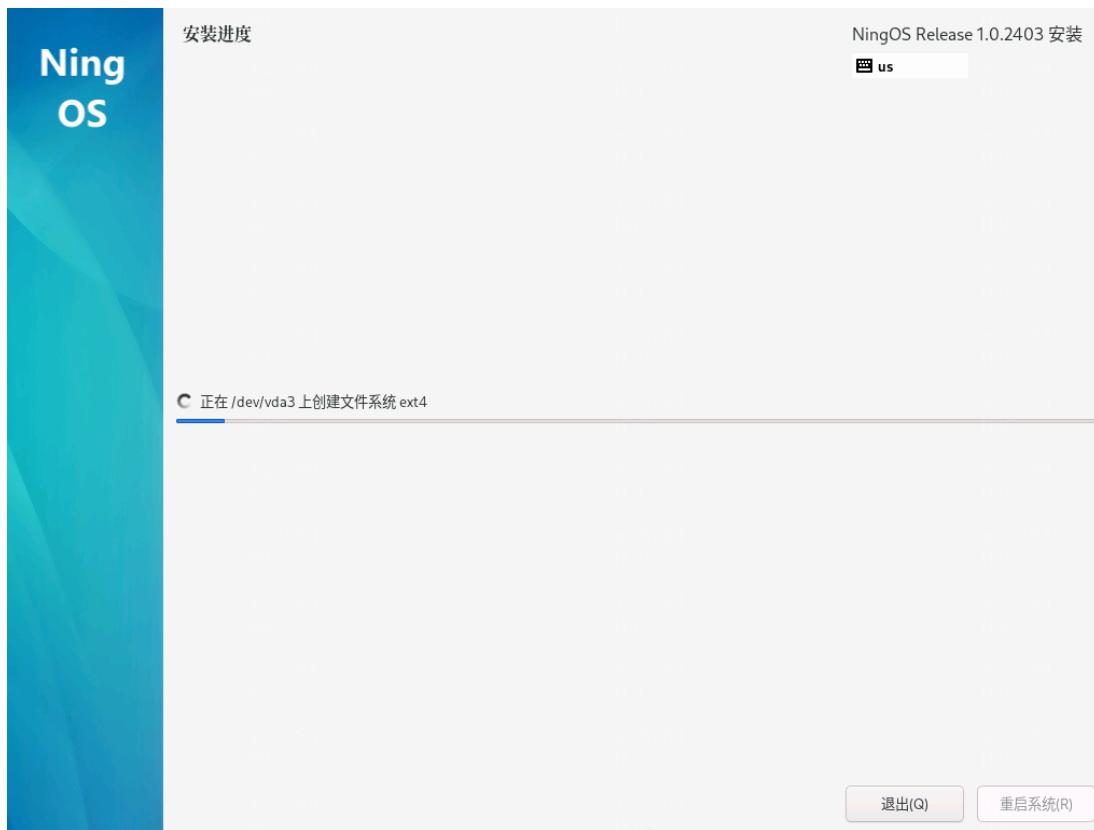
图4-35 配置服务器的 IPv4 地址



#### 4.3.4 操作系统安装

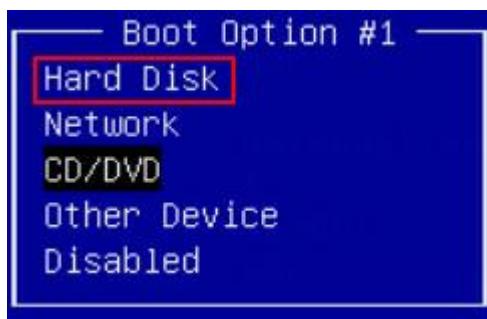
- (1) 在本地电脑控制端窗口执行命令 `ping ip_address`, 检查配置的 IP 地址是否连通。其中, `ip_address` 为 IPv4 设置页签下配置的 IP 地址。若可 ping 通, 则继续进行后续步骤, 否则返回 IPv4 设置页签, 检查掩码网关等信息是否配置正确。
- (2) 在图 4-36 所示的安装信息摘要界面中单击<开始安装>按钮, 开始安装操作系统。

图4-36 安装操作系统



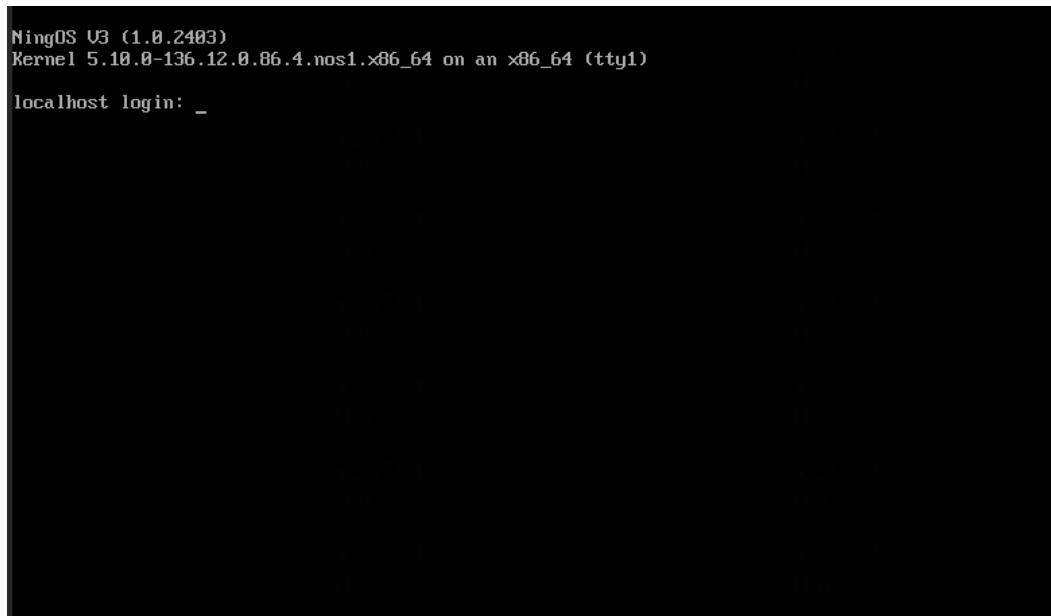
- (3) 安装完成后，单击<重启系统(R)>按钮。进入 BIOS 设置界面，设置 BIOS 从 Hard Disk 启动，即设置 Boot 启动项时，在弹出的“Boot Option #1”对话框中，按<Enter>键选择“Hard Disk”，如图 4-37 所示。具体步骤与设置 BIOS 从 CD/DVD 启动类似，请参见“[4.2 设置 BIOS 从 CD/DVD 启动](#)”。

图4-37 设置 Boot 启动项



- (4) 保存退出后，服务器再次自动重启，重启后的界面如图 4-38 所示。

图4-38 安装完成界面



## 5 安装部署 NingOS Matrix 集群

### 5.1 安装Matrix依赖包

### 5.2 关闭防火墙

- (1) 在服务器上执行 **systemctl stop firewalld.service** 命令关闭防火墙。

```
[root@localhost ~]# systemctl stop firewalld.service
```

- (2) 执行 **systemctl status firewalld.service** 命令查看防火墙状态，确保防火墙已关闭。

```
[root@localhost ~]# systemctl status firewalld.service
○ firewalld.service - firewalld - dynamic firewall daemon
    Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor
    preset: enabled)
    Active: inactive (dead) //防火墙已关闭
      Docs: man:firewalld(1)
```

- (3) 执行 **systemctl disable firewalld.service** 命令设置防火墙开机不启动。

```
[root@localhost ~]# systemctl disable firewalld.service
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
```

## 5.3 设置时区时间

- (1) 使用命令 **timedatectl** 查看当前时区及对应时间，若与当前本地时间不一致，需要手动修改时间。时区设置为 **CST,Shanghai**，时间设置为本地实际时间。

```
[root@localhost ~]#timedatectl
    Local time: 三 2024-04-17 08:36:11 CST
    Universal time: 三 2024-04-17 00:36:11 UTC
        RTC time: 三 2024-04-17 08:36:12
        Time zone: Asia/Shanghai (CST, +0800)

System clock synchronized: yes
          NTP service: active
    RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
This mode cannot be fully supported. It will create various problems
with time zone changes and daylight saving time adjustments. The RTC
time is never updated, it relies on external facilities to maintain it.
If at all possible, use RTC in UTC by calling
'timedatectl set-local-rtc 0'.
```

- (2) 使用命令 **sudo timedatectl set-ntp off** 关闭自动时间同步功能，

```
[root@beijing-ning25 ~]# sudo timedatectl set-ntp off
```

- (3) 使用命令 **sudo timedatectl set-time '2024-04-17 16:44:00'** 设置正确时间。

```
[root@beijing-ning25 ~]# sudo timedatectl set-time '2024-04-17 16:44:00'
```

```
[root@beijing-ning25 ~]# timedatectl
    Local time: 三 2024-04-17 16:44:05 CST
    Universal time: 三 2024-04-17 08:44:05 UTC
        RTC time: 三 2024-04-17 16:44:05
        Time zone: Asia/Shanghai (CST, +0800)

System clock synchronized: no
          NTP service: inactive
    RTC in local TZ: yes
```

```
Warning: The system is configured to read the RTC time in the local time zone.
This mode cannot be fully supported. It will create various problems
with time zone changes and daylight saving time adjustments. The RTC
time is never updated, it relies on external facilities to maintain it.
If at all possible, use RTC in UTC by calling
'timedatectl set-local-rtc 0'.
```

- (4) 使用命令 **sudo timedatectl set-ntp on** 手动开启自动时间同步功能。

```
[root@beijing-ning25 ~]# sudo timedatectl set-ntp on
```

## 5.4 配置本地YUM源

通过挂载操作系统 ISO 镜像的方式来配置本地 YUM 源。

- (1) 登录服务器的 HDM 管理平台，挂载操作系统 ISO 镜像文件，详细操作步骤请参见“[4.1 登录远程控制台](#)”。

- (2) 在服务器上执行 **mount /dev/cdrom/mnt** 命令将 ISO 镜像挂载至/mnt 目录。

```
[root@localhost ~]# mount /dev/cdrom /mnt  
mount: /mnt: WARNING: source write-protected, mounted read-only.//挂载成功
```



#### 说明

服务器重启后，ISO 镜像目录可能会被取消挂载，如果需要继续使用本地 YUM 源的方式安装依赖包，请重新挂载 ISO 镜像文件。

- (3) 修改服务器/etc/yum.repos.d/路径下 NingOS.repo 文件内容。

- a. 进入服务器的目录/etc/yum.repos.d/。

```
[root@localhost ~]# cd /etc/yum.repos.d/  
[root@localhost ~]# vi NingOS.repo
```

- b. 通过键盘输入“i”进入编辑界面，增加灰显内容：

```
[NingOS_repo]  
name=NingOS_repo  
baseurl=file:///mnt  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NingOS  
enabled=1  
[NingOS_repo_updates]  
name=NingOS_repo  
baseurl=file:///mnt  
gpgcheck=0  
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NingOS  
enabled=1
```

- c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

- (4) 安装 GPU 驱动依赖包

安装 GPU 驱动时需要部署 GPU 驱动所需依赖包，详细操作步骤请参见“[8.1.4 安装显卡驱动](#)”，避免反复挂载 yum 镜像源。

- (5) 执行 **yum clean all && yum makecache** 命令清空 YUM 源缓存。

```
[root@localhost ~]# yum clean all && yum makecache  
6 files removed  
NingOS repo 369 MB/s | 3.4 MB 00:00  
元数据缓存已建立。
```

## 5.5 安装Matrix依赖包

需要安装的 Matrix 依赖包如[表 5-1](#) 所示。

表5-1 Matrix 依赖包版本列表

依赖包名称	版本号
java openjdk	1.8.0

docker-ce-cli	20.10.7
docker-ce	20.10.7
docker-ce-rootless-extras	20.10.7
docker-scan-plugin	0.8.0
chrony	4.1-3
python	3.9.9
ntpdate	4.2.8p15

(1) 安装 Matrix 依赖包。

```
[root@localhost ~]# yum install -y docker-ce-20.10.7* docker-ce-cli-20.10.7*
docker-ce-rootless-extras-20.10.7* docker-scan-plugin-0.8.0*
[root@localhost ~]# yum install java-1.8.0-openjdk
[root@localhost ~]# yum install chrony
[root@localhost ~]# yum install python
[root@localhost ~]# yum install ntpdate
```

(2) 执行 **rpm -qa | grep** 命令依次查询各依赖包是否安装成功（以查询 docker 依赖包为例）。

```
[root@localhost ~]# rpm -qa | grep docker
docker-scan-plugin-0.8.0-3.h1202.x86_64
docker-ce-cli-20.10.7-3.h1202.x86_64
docker-ce-rootless-extras-20.10.7-3.h1202.x86_64
docker-ce-20.10.7-3.h1202.x86_64
```

(3) 取消挂载本地 YUM 源。

```
[root@localhost ~]# umount /mnt
```

## 5.6 安装Matrix

- (1) 获取 Matrix 软件安装包，并将软件包通过 **FTP** 工具上传至服务器的待安装目录（例如/**/root**）下。
- (2) 进入 Matrix 软件包（.zip 文件）的存放路径，解压缩并安装 Matrix。软件包的名称格式为 **Matrix\_version-platform.zip**，其中 **version** 为版本号，**platform** 为 CPU 架构类型。下面以 **x86\_64** 版本为例进行安装。

```
[root@localhost ~]# unzip Matrix_V900R001B07D014SP03_x86_64.zip
[root@localhost ~]# cd Matrix-V900R001B07D014SP03-x86_64
[root@localhost Matrix-V900R001B07D014SP03_x86_64]# ./install.sh
Installing...
[install] -----
[install]   Matrix-V100R001B01D014SP03-x86_64
[install]   NingOS-v3 1.0.2404
[install]   Linux 5.10.0-136.12.0.86.4.nos1.x86_64
[install] -----
NingOS-v3 1.0.2403
[install] WARNING: To avoid unknown error, do not interrupt this installation procedure.
[install] Checking environment...
```

```

[install] Done.
[install] Checking current user permissions...
[install] Done.
[install] Decompressing matrix package...
[install] Done.
[install] Installing dependent software...
[install] Done.
[install] Starting and checking matrix service...
[install] /opt/matrix/config/navigator_config.json exist
[install] Done.
Complete!

```

- (3) 通过命令 **systemctl status matrix** 验证 Matrix 服务是否安装成功。若安装成功，则将在 Active 字段后显示运行信息为 active (running)。剩余节点执行同样操作即可。

```

[root@localhost ~]# systemctl status matrix.service
● matrix.service - Matrix Server
   Loaded: loaded (/usr/lib/systemd/system/matrix.service; enabled; vendor preset: disabled)
     Active: active (running) since Tue 2023-10-24 08:01:46 CST; 2 weeks 5 days ago
       Main PID: 702359 (karaf)
          Tasks: 549 (limit: 3355442)
        Memory: 2.3G
      CGroup: /system.slice/matrix.service
              └─ 702359 /bin/sh /opt/matrix/bin/karaf server
                  ├─ 703187 /usr/bin/java -XX:+UnlockDiagnosticVMOptions
                  -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/matrix/heapdumps/
                  -XX:OnOutOfMemoryError=/opt/matrix/k8s/disaster-reco>

Nov 13 02:56:35 localhost sudo[916772]: pam_unix(sudo:session): session closed for user
root

Nov 13 02:56:39 localhost sudo[917311]:      root : PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile;
/opt/matrix/k8s/monitorscript/check_node_service_2m.sh

Nov 13 02:56:39 localhost sudo[917311]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

Nov 13 02:56:39 localhost sudo[917311]: pam_unix(sudo:session): session closed for user
root

Nov 13 02:57:14 localhost sudo[920771]:      root : PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile;
/opt/matrix/k8s/monitorscript/check_config_dir_file_3m.sh

Nov 13 02:57:14 localhost sudo[920771]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

Nov 13 02:57:14 localhost sudo[920771]: pam_unix(sudo:session): session closed for user
root

Nov 13 02:57:15 localhost sudo[920998]:      root : PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile;
/opt/matrix/k8s/monitorscript/check_docker_config.sh

Nov 13 02:57:15 localhost sudo[920998]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

Nov 13 02:57:15 localhost sudo[920998]: pam_unix(sudo:session): session closed for user
root

```



### 说明

Matrix 所有节点使用一个安装包进行安装，在配置页面进行配置时再区分 Master 节点与 Worker 节点。

## 5.7 创建Matrix集群



### 说明

- 当使用内置 NTP 服务器作为时钟同步源时，在部署集群之前，需确保所有节点的系统时间和当前时间保持一致。
- 当使用外置 NTP 服务器作为时钟同步源时，在部署集群之前，需确保外置 NTP 服务器的时间与当前时间保持一致。
- 如果 NTP 服务器网络故障或时间不准确，则可能会导致 Matrix 集群部署失败。

### 5.7.1 登录 Matrix

- (1) 在浏览器中输入 Matrix 的登录地址，进入如图 5-1 所示的登录页面。登录地址格式为：  
[https://ip\\_address:8443/matrix/ui/](https://ip_address:8443/matrix/ui/)。其中 ip\_address 为 Master 节点 IP 地址，8443 为缺省端口号。



### 说明

未部署集群之前，ip\_address 可以是任意一个规划为 Master 节点的 IP 地址。

图5-1 Matrix 登录界面



- (2) 输入用户名和密码(默认用户名为 admin, 密码为 Pwd@12345, 若安装操作系统设置过密码, 则按设置的填写) 后, 单击<登录>按钮, 默认进入 Matrix 的集群部署向导页面。

### 5.7.2 配置集群参数

在 Matrix 的集群部署向导页面中配置 Matrix 集群参数, 如图 5-2 所示, 集群参数详细介绍如表 5-2 所示。

图5-2 Matrix 集群设置（一）

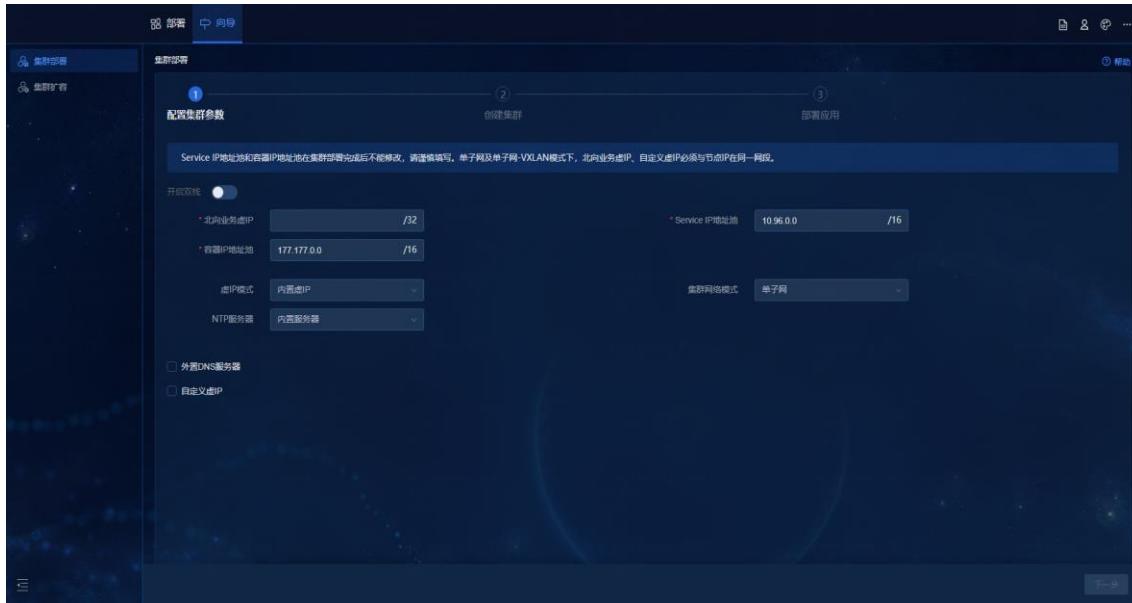


表5-2 集群参数说明

参数	说明
北向业务虚IP	集群对外提供服务的IP地址, 该地址必须在Master节点所处的网段内
Service IP地址池	用于为Service分配IP地址, 不能与部署环境中的其它网段冲突。默认地址为10.96.0.0/16, 一般保持默认值
容器IP地址池	用于为容器分配IP地址, 不能与部署环境中的其它网段冲突。默认地址为177.177.0.0/16, 一般保持默认值
虚IP模式	取值为内置虚IP、外置虚IP。内置模式下虚IP由Matrix下发到集群内, 并由Matrix管理虚IP在集群节点间的漂移; 外置模式下, 虚IP由第三方平台或软件下发到集群外, 不再由Matrix管理。默认为内置模式
集群网络模式	集群网络模式有两种: 单子网: 集群内所有节点、虚IP必须在相同网段内, 否则将无法互相通信。如果使用本地环境部署LinSeer RT, 请选择本模式 单子网-VXLAN: 集群内所有节点、虚IP必须在相同网段内, 否则将无法互相通信。如果使用云上环境部署LinSeer RT, 请选择本模式, 否则Pod之间可能无法通信, 会导致部署失败
NTP服务器	用于保证集群内各节点系统时间的一致性, 支持选择内置服务器和外置服务器。选择外置服务器时, 需要配置NTP服务器地址, 且该地址不可与集群内

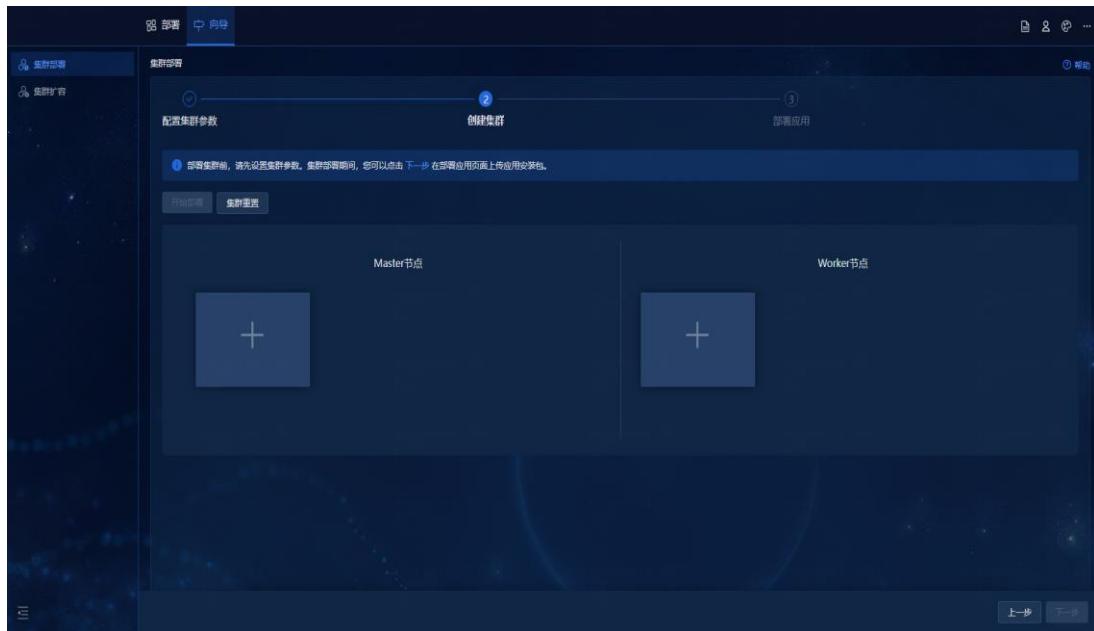
	<p>各节点的IP地址冲突</p> <p>本文档使用内置服务器作为NTP服务器，则部署集群时会首先进行时间同步，集群部署完成后，三台Master节点会定时同步时间，从而保证集群内各节点的系统时间保持一致</p>
外置DNS服务器	<p>用于解析K8s集群外部的域名，格式为IP:Port，可根据实际需要配置外置DNS服务器。本文档中不配置此项</p> <p>容器解析域名时，集群外部的域名无法被内置DNS服务器解析，本系统将把需要解析的外部域名随机转发给一台外置DNS服务器来解析</p> <p>外置DNS服务器最多可以配置10个，各外置DNS服务器要求具有相同的DNS解析能力，并可以独立满足外部域名解析需求、无主备之分、无先后顺序之分</p> <p>建议所有的DNS服务器都能够访问根域，可使用命令行nslookup -port={port}-q=ns . {ip}查看是否可以访问</p>
自定义虚IP	<p>用于在南北隔离的情况下，设备需要使用固定的IP和集群通信</p> <p>不能与部署环境中的其它网段冲突</p> <p>精简模式下该参数无需配置。</p>

### 5.7.3 创建集群

单机部署模式下，仅需增加一个Master节点即可部署集群。集群部署模式下，需要增加三个Master节点后，再部署集群。

(1) 配置集群参数后，单击<下一步>按钮，进入创建集群页面，如图5-3所示。

图5-3 集群部署界面



(2) 单击Master节点区域的增加图标 $\text{+}$ ，弹出增加节点窗口。

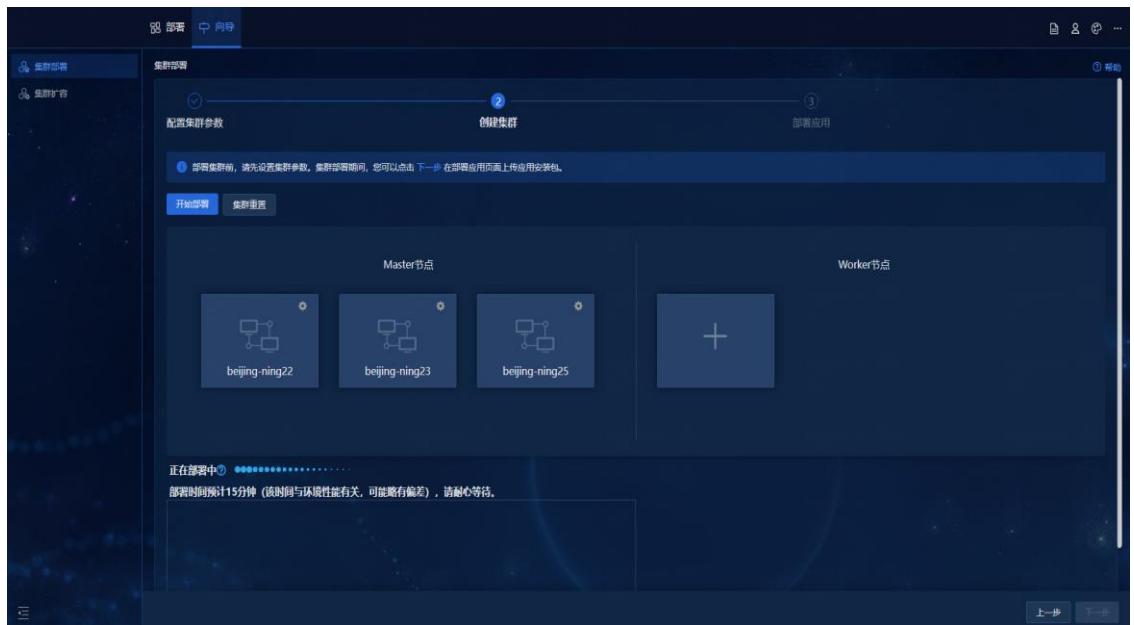
- (3) 在弹出的节点窗口中，配置节点参数，如图 5-4 所示配置参数后，单击<应用>按钮，完成增加 Master 节点操作。
- 类型：显示为“Master”不可修改。
  - IP 地址：规划的 Master 节点的 IP 地址。
  - 用户名：节点操作系统的用户名，为安装 NingOS 系统时设置的 root 用户名。
  - 密码：节点操作系统的用户密码，为安装 NingOS 系统时设置的 root 用户密码。

图5-4 增加节点窗口



- (4) 单击<开始部署>按钮，开始部署集群，如图 5-5 所示。

图5-5 集群部署



- (5) 部署成功后，会显示各 Master 节点。

- 单机部署成功页面如图 5-6 所示，Master 节点左上角显示标记 。

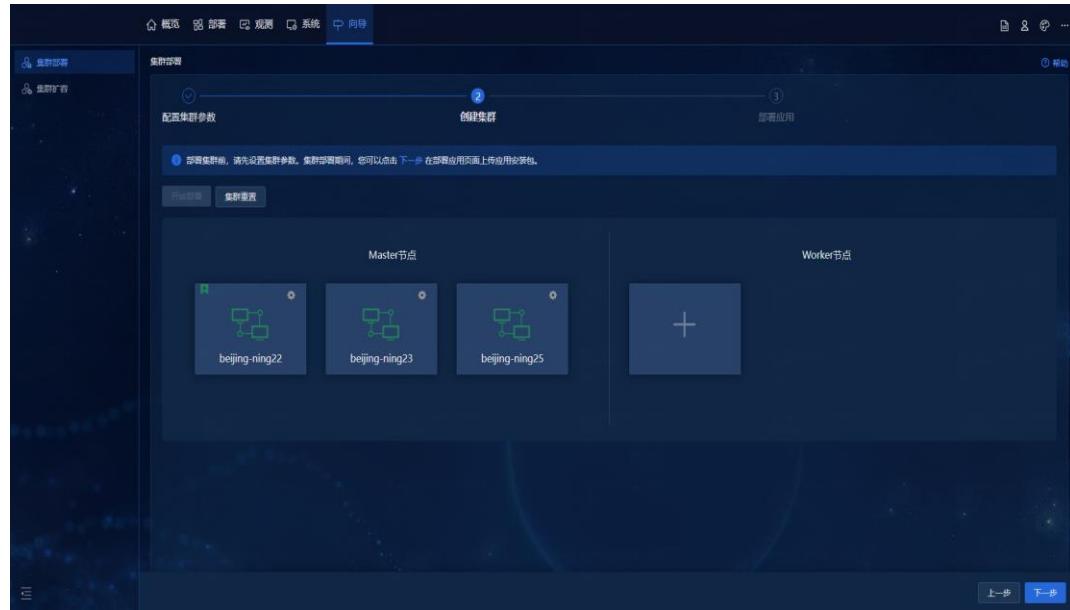
图5-6 单机部署成功页面



- 集群中所有节点的进度达到 100% 时，表示集群部署成功，如图 5-7 所示。

集群部署成功后，主 Master 节点左上角显示标记 ，其余未被标记的 Master 节点为备用 Master 节点。

图5-7 集群部署成功页面



#### 5.7.4 创建网络

仅分布式推理组网需要配置 MAC VLAN 网络。

3 机或 3+N 机组网下，需要配置网络完成多机资源统一，以实现更大规模数据处理和复杂计算任务的高效、可靠和灵活执行。

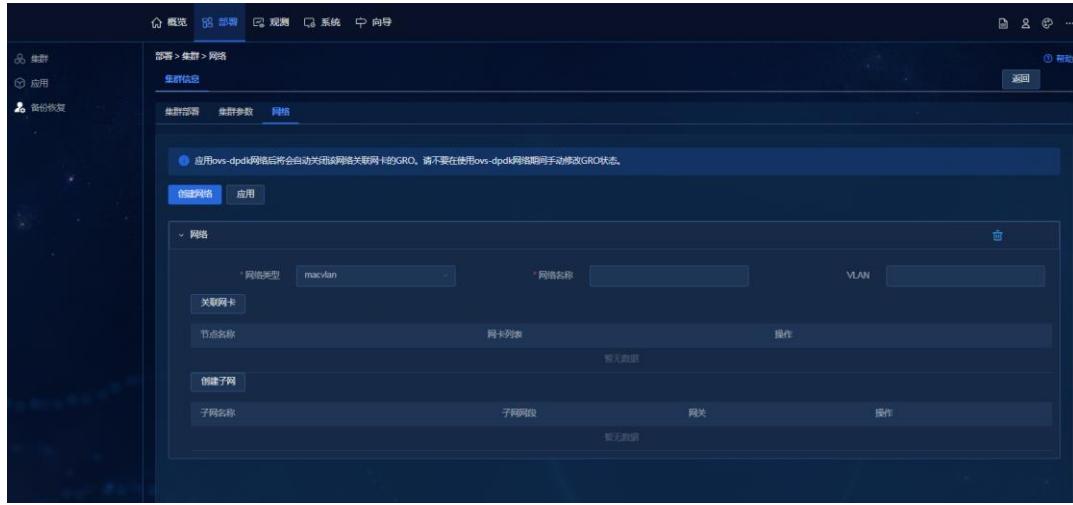
(1) 如图 5-8 所示，在 Matrix 集群部署页面单击操作列的详情按钮 ，进入“网络”页签。

图5-8 Matrix 集群部署页面



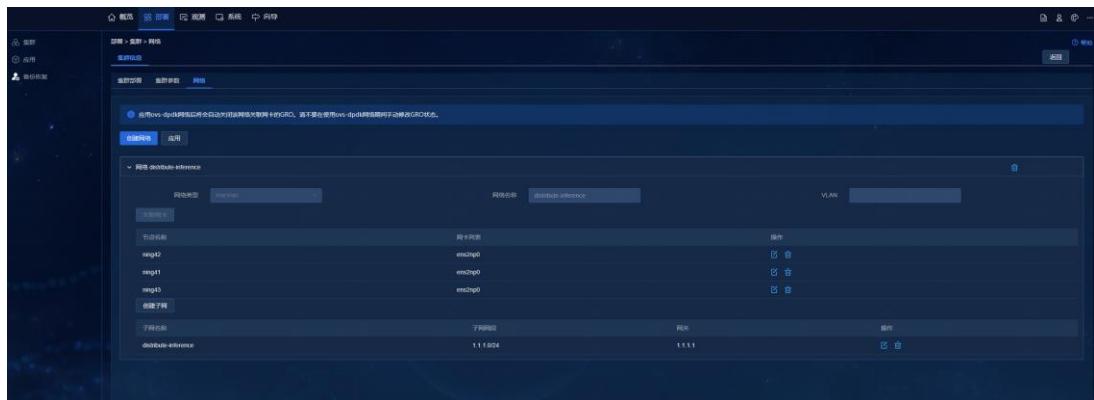
(2) 单击<创建网络>按钮新建网络。根据网络规划设置参数，选择网络类型为“macvlan”，如图 5-9 所示。

图5-9 创建网络



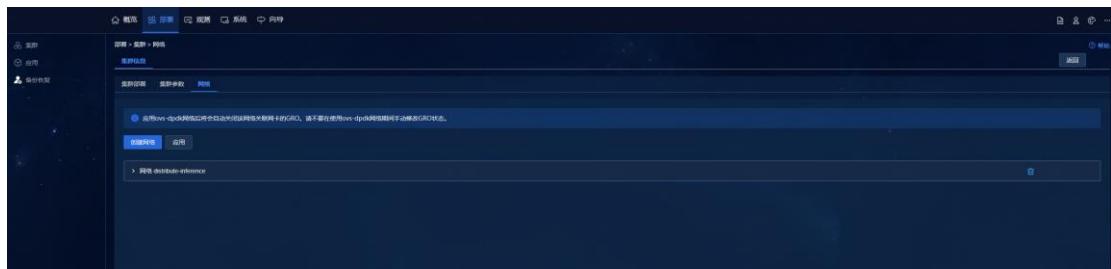
- (3) 如图 5-9 所示，单击<关联网卡>按钮添加物理网卡。
- 选择“节点名称”为 GPU 服务器名称。
  - 选择“网卡列表”为对应 GPU 服务器的业务网卡（例如 ens2np0）。
  - 重复上述步骤，关联所有 GPU 服务器的物理网卡。
- (4) 单击<创建子网>按钮为 MAC VLAN 网络创建子网。如图 5-10 所示。
- 设置分布式推理子网名称。必须使用“distribute-inference”。
  - 输入子网网段。必须与网关处于同一网段。
  - 输入网关 IP 地址。该网关地址必须与参数网交换机上地址一致。

图5-10 创建子网



- (5) 网络创建完成后，单击<应用>按钮，完成配置，如图 5-11 所示。

图5-11 完成网络创建



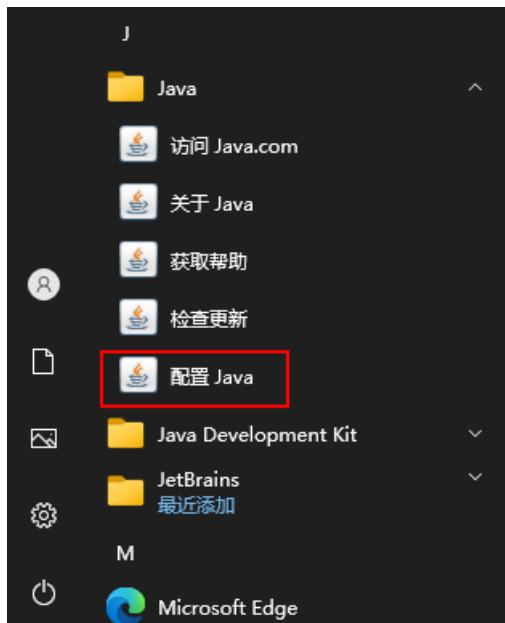
## 6 安装 KylinOS 操作系统

### 6.1 登录远程控制台

#### 6.1.1 配置 Java 环境

- (1) 在本地 PC 机上安装 JDK，推荐版本为 8u181。
- (2) 如图 6-1 所示，单击[开始]菜单选择“配置 Java”，进入 JAVA 控制面板。

图6-1 配置 Java



- (3) 把 HDM 的 URL 加入“例外站点”列表。如图 6-2 所示。
  - a. 选择“安全”页签，单击<编辑站点列表>按钮进入“例外站点”列表。
  - b. 在“例外站点”列表对话框中单击<添加>按钮，。
  - c. 输入 HDM 的 URL，单击<确定>按钮完成添加。

d. 单击 JAVA 控制面板上的<确定>按钮完成设置。

图6-2 更新例外站点列表



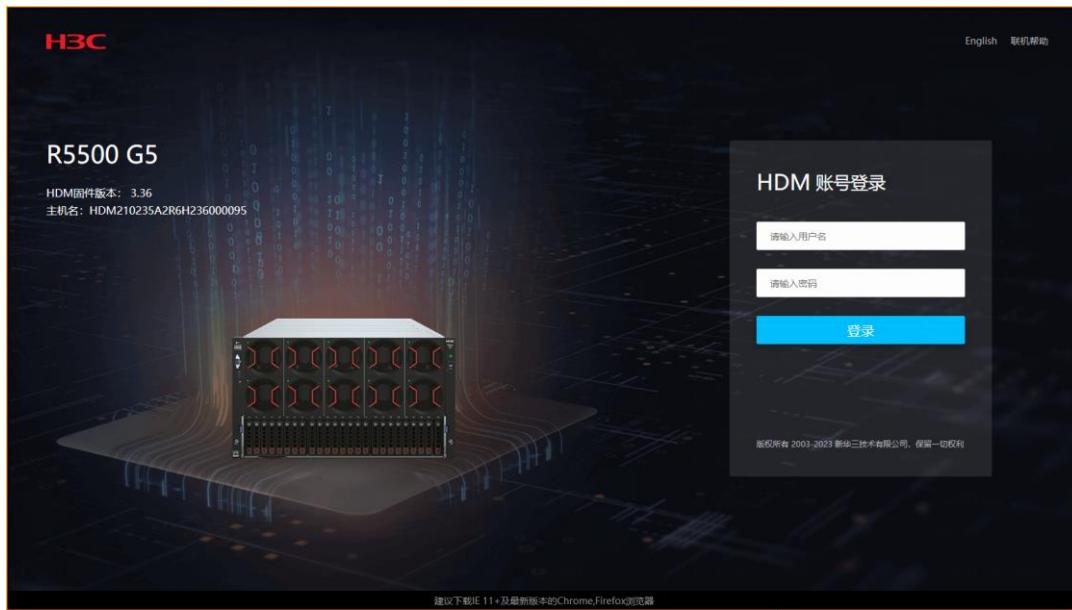
### 6.1.2 登录 HDM Web 页面

HDM (Hardware Device Management, 设备管理系统) 用于远程管理服务器。

登录 HDM 操作步骤如下：

- (1) 将网线连接到服务器的 HDM 专用网口，确保本地 PC 与服务器之间网络可达。
- (2) 在本地 PC 浏览器地址栏中输入“<https://HDM 专用网口 IP 地址>”，按<Enter>键打开登录页面。输入用户名和密码，单击<登录>按钮登录服务器 HDM Web 页面，如图 6-3 所示。

图6-3 登录服务器 HDM Web 页面示意图



#### 说明

- HDM Web 操作页面、默认 HDM 专用网口 IP 地址、默认用户名/密码等与服务器型号和 HDM 版本有关，详细信息请参见对应版本的服务器 HDM 用户指南。
- 对于本文示例版本，默认 HDM 专用网口 IP 地址为 192.168.1.2/24，默认用户名/密码为 admin/Password@\_。
- 若已修改默认 HDM 专用网口 IP 地址、用户名/密码，请以修改后的 HDM 专用网口 IP 地址、用户名/密码登录。

### 6.1.3 连接远程控制台并挂载镜像

本步骤用于使用 Java 集成远程控制台连接服务器。通过 JVViewer 或 KVM 远程连接软件，您可以在远程进行配置管理服务器、安装操作系统等操作。

对于 R5300 G5/R5500 G5/R5500 G6/R5350 G6 服务器，连接远程控制台的操作步骤如下：

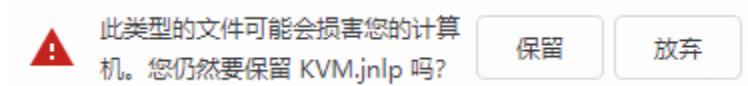
- (1) 在 HDM Web 页面单击顶部[远程服务]菜单项，再单击左侧导航树中的[远程控制台]进入远程控制台页签。单击<启动 KVM>按钮下载 KVM 软件，如图 6-4 所示。

图6-4 下载远程连接软件



- (2) 如图 6-5 所示, 在弹出的提示框中单击<保留>按钮, 开始下载 KVM 软件。

图6-5 远程连接软件下载提示信息



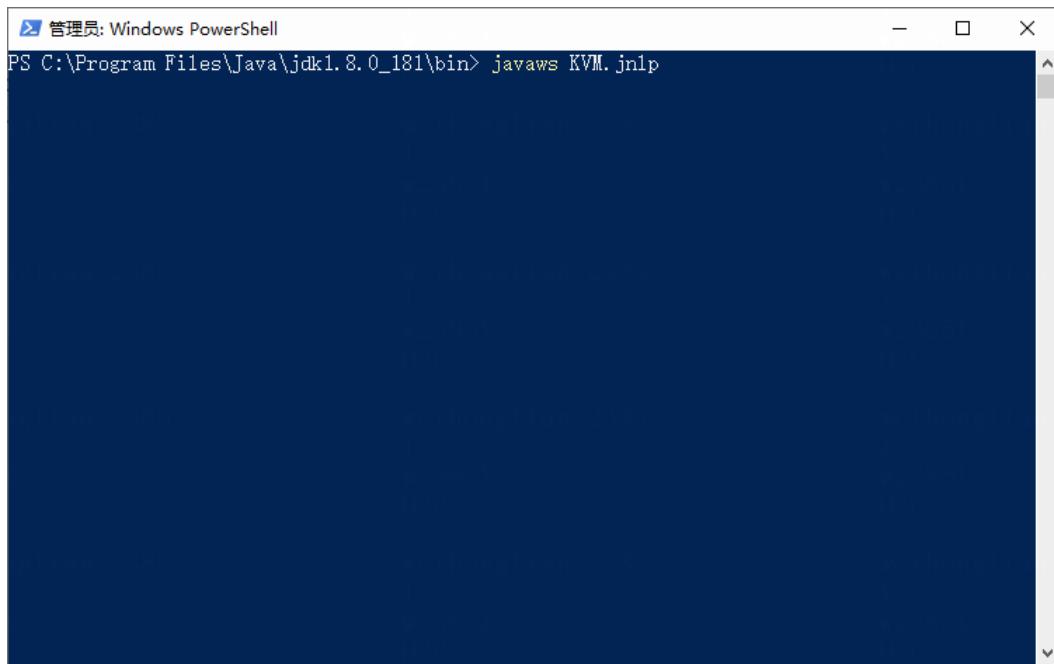
- (3) 将已下载的 KVM.Jnlp 文件拷贝至 JDK 的安装目录“C:\Program Files\Java\jdk1.8.0\_181\bin”下。
- (4) 按键盘<Shift>键, 同时鼠标右键单击文件夹空白处, 在下拉菜单中选择“在此处打开 Powershell 窗口”, 如图 6-6 所示。

图6-6 打开命令窗口



- (5) 在命令窗口执行命令 `javaws KVM.jnlp` 并按回车键, 如图 6-7 所示。

图6-7 执行命令



```
PS C:\Program Files\Java\jdk1.8.0_181\bin> javaws KVM.jnlp
```

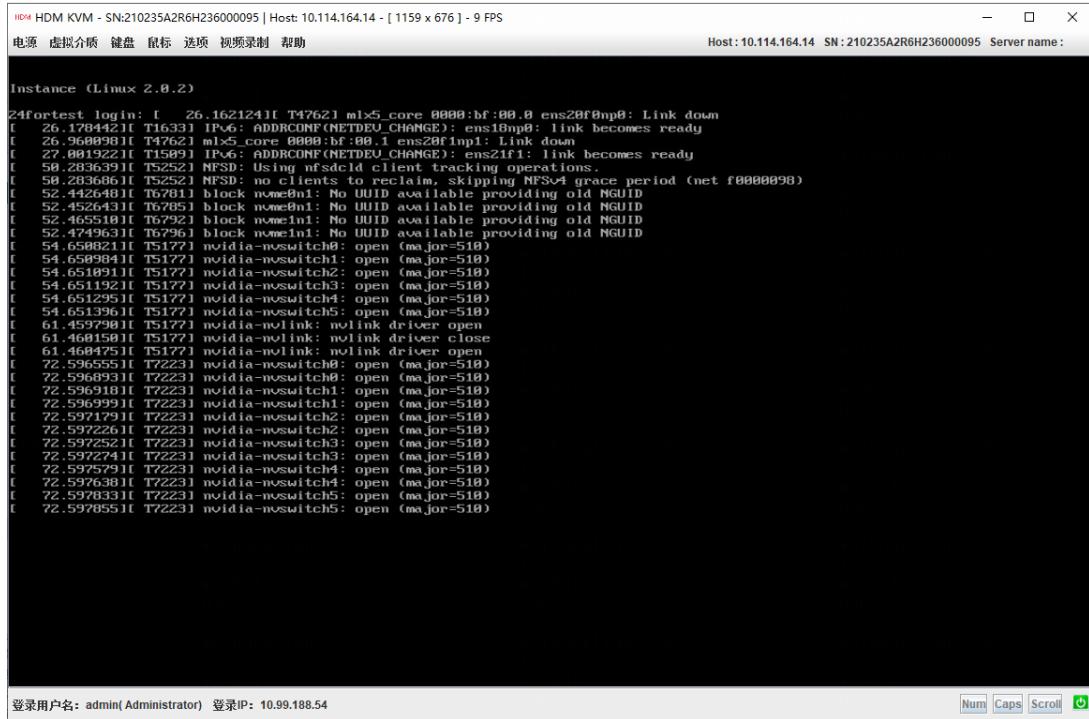
- (6) 在安全警告窗口中点击<继续>按钮。如图 6-8 所示。

图6-8 安全警告窗口



- (7) 成功连接远程控制台, 可显示远程控制台窗口, 如图 6-9 所示。

图6-9 连接远程控制台



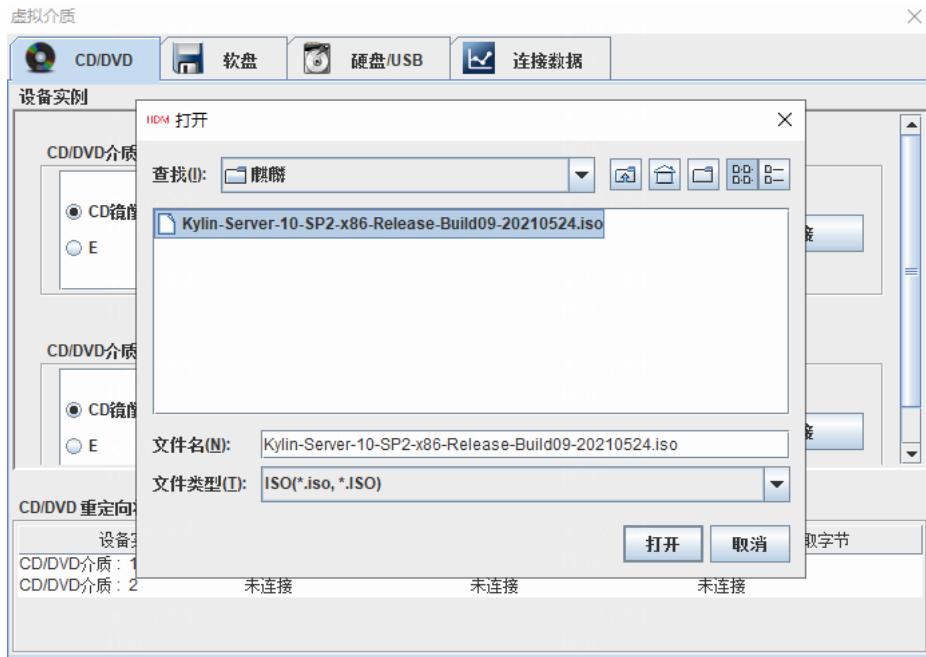
- (8) 在远程控制台窗口单击[虚拟介质 -> 虚拟介质向导]菜单项，弹出虚拟介质对话框，如图 6-10 所示，单击<浏览>按钮。

图6-10 虚拟介质对话框



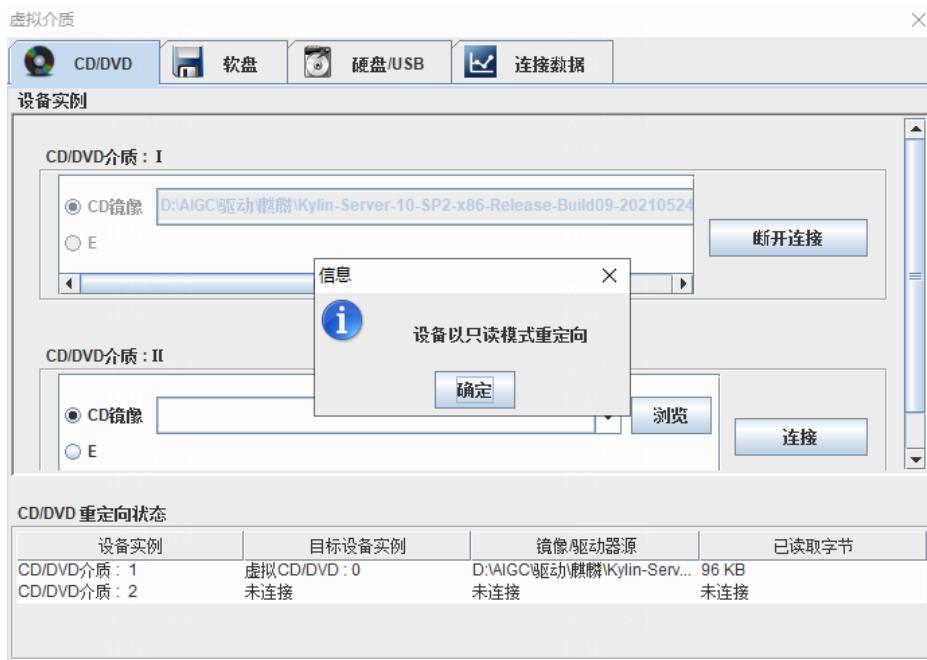
(9) 选择本地的操作系统镜像文件 (.iso 格式)，单击<打开>按钮。如图 6-11 所示。

图6-11 打开对话框



(10) 在弹出的提示信息对话框内单击<确定>按钮，即可成功挂载镜像文件，如图 6-12 所示。

图6-12 确认提示信息



## 6.2 设置BIOS从CD/DVD启动

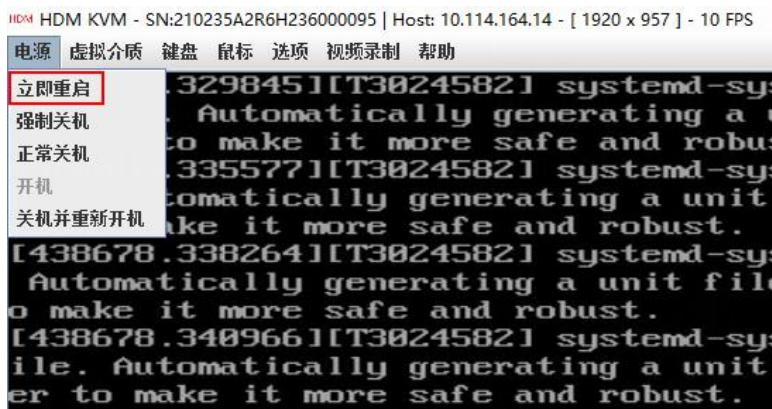


### 说明

- 设置 BIOS 从 CD/DVD 启动需通过键盘操作。
- 不同型号服务器参数配置界面有所不同，本节仅以 R5500 G5 服务器为例。
- 参数配置界面可能会不定期更新，请以设备实际显示为准。

(1) 单击选择[电源 -> 立即重启]菜单项，重启服务器，如图 6-13 所示。

图6-13 重启服务器



(2) 服务器重启后，按键盘~~DEL~~键或~~ESC~~键进入 BIOS 设置界面，如图 6-14 所示。

图6-14 进入 BIOS 设置界面



(3) 通过按键盘~~<-->~~键移动选择[Boot]菜单项，按~~Enter~~键设置 BIOS，如图 6-15 所示。

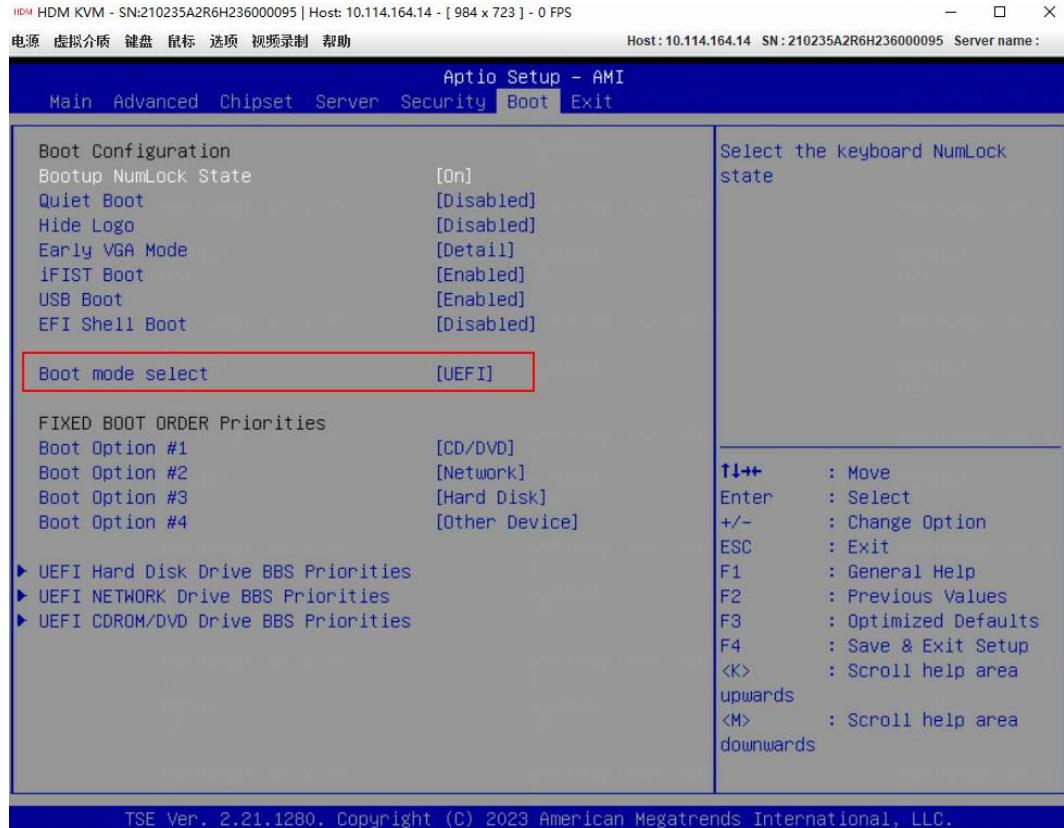
图6-15 设置 BIOS



(4) 设置 Boot 启动模式。

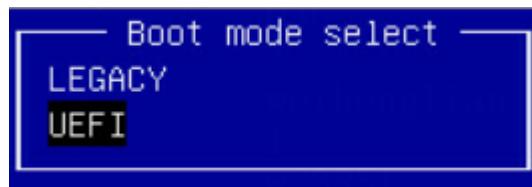
- a. 通过按键盘 $\leftarrow$  $\rightarrow$  $\uparrow$  $\downarrow$ 键移动选择“Boot mode select”配置项，按 $<\text{Enter}>$ 键设置 Boot 启动模式，[图 6-16](#)所示。

图6-16 设置 Boot 启动模式（1）



- b. 在弹出的“Boot mode select”对话框中，按 $<\text{Enter}>$ 键选择“UEFI”。如[图 6-17](#)所示。

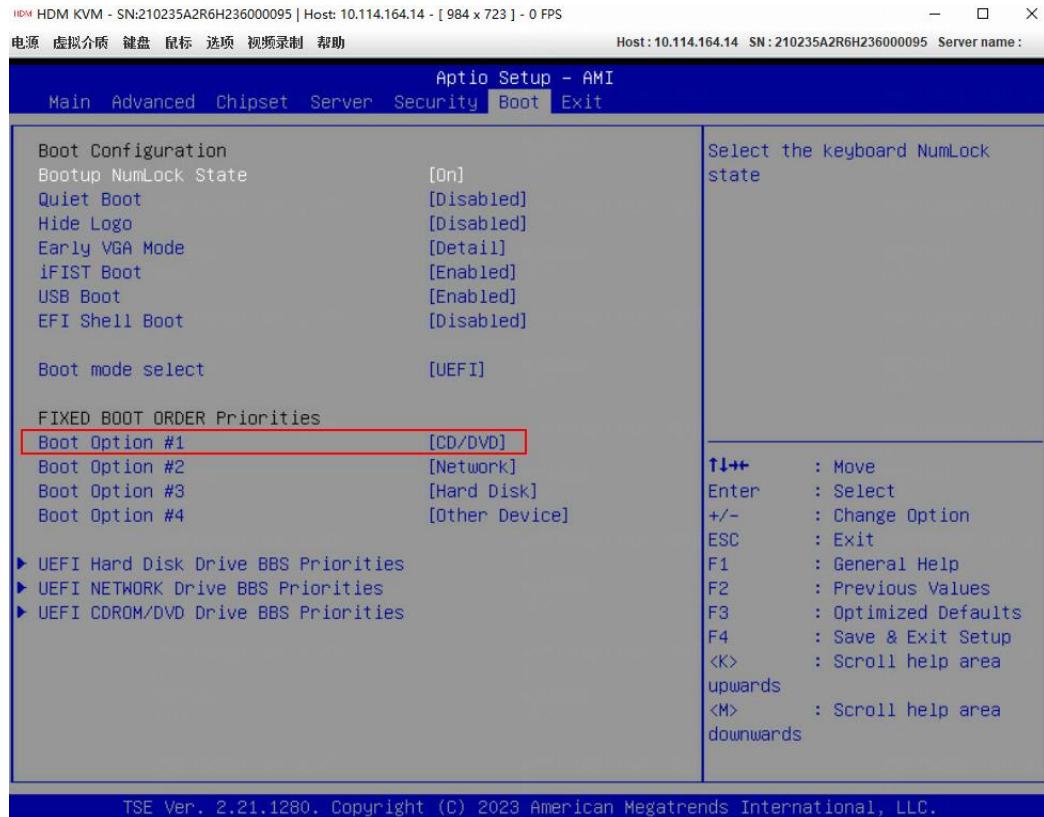
图6-17 设置启动 Boot 模式（2）



(5) 设置 Boot 启动项。

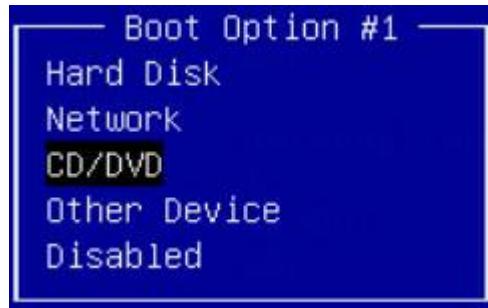
- a. 通过按键盘 $\leftarrow$  $\rightarrow$  $\uparrow$  $\downarrow$ 键移动选择“Boot Option #1”配置项，按 $<\text{Enter}>$ 键设置 Boot 启动项，如[图 6-18](#)所示。

图6-18 设置 Boot 启动项 (1)



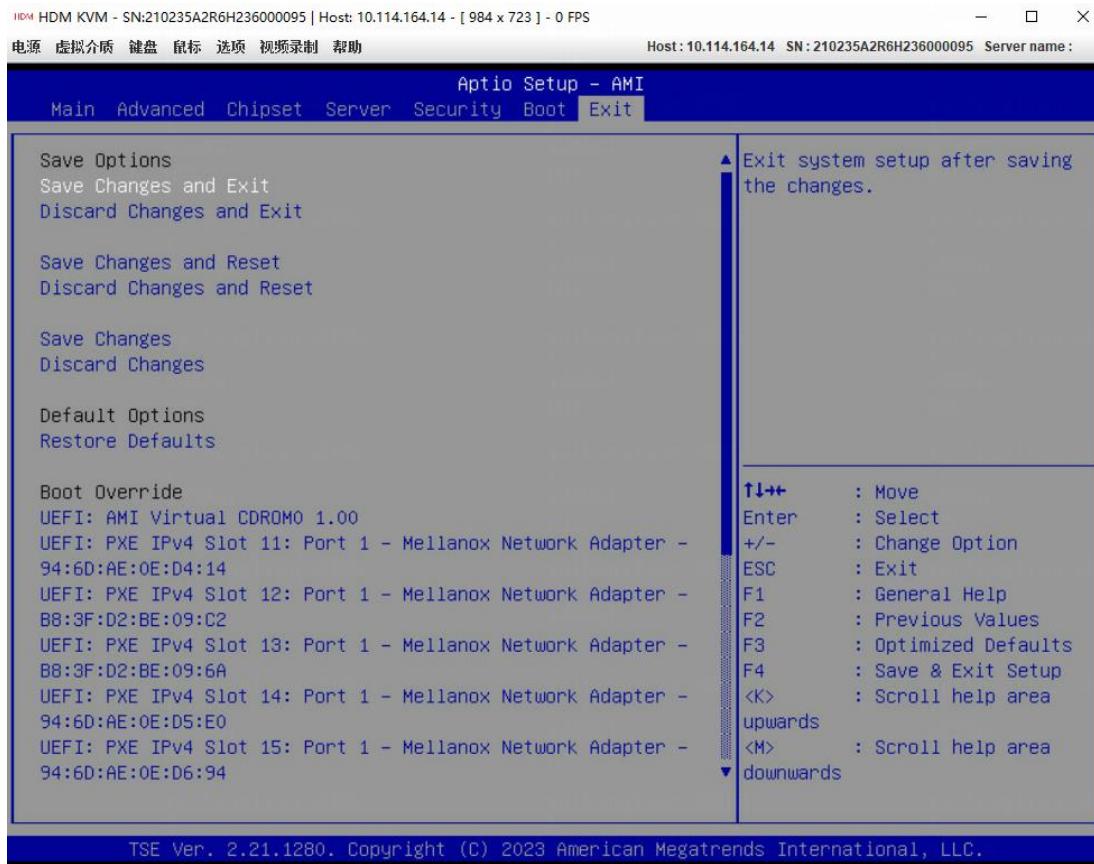
- b. 在弹出的“Boot Option #1”对话框中，按<Enter>键选择“CD/DVD”，如图 6-19 所示。

图6-19 设置 Boot 启动项 (2)



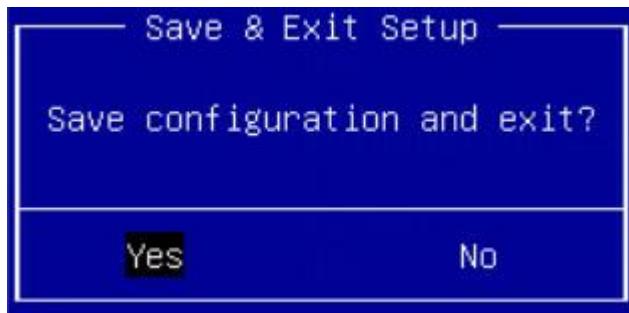
- (6) 通过键盘<--><-->键选择[Exit]菜单项，按<Enter>键保存设置，如图 6-20 所示。

图6-20 保存设置



- (7) 在弹窗的对话框中选择“YES”，保存设置并退出，如图6-21所示。

图6-21 保存设置并退出



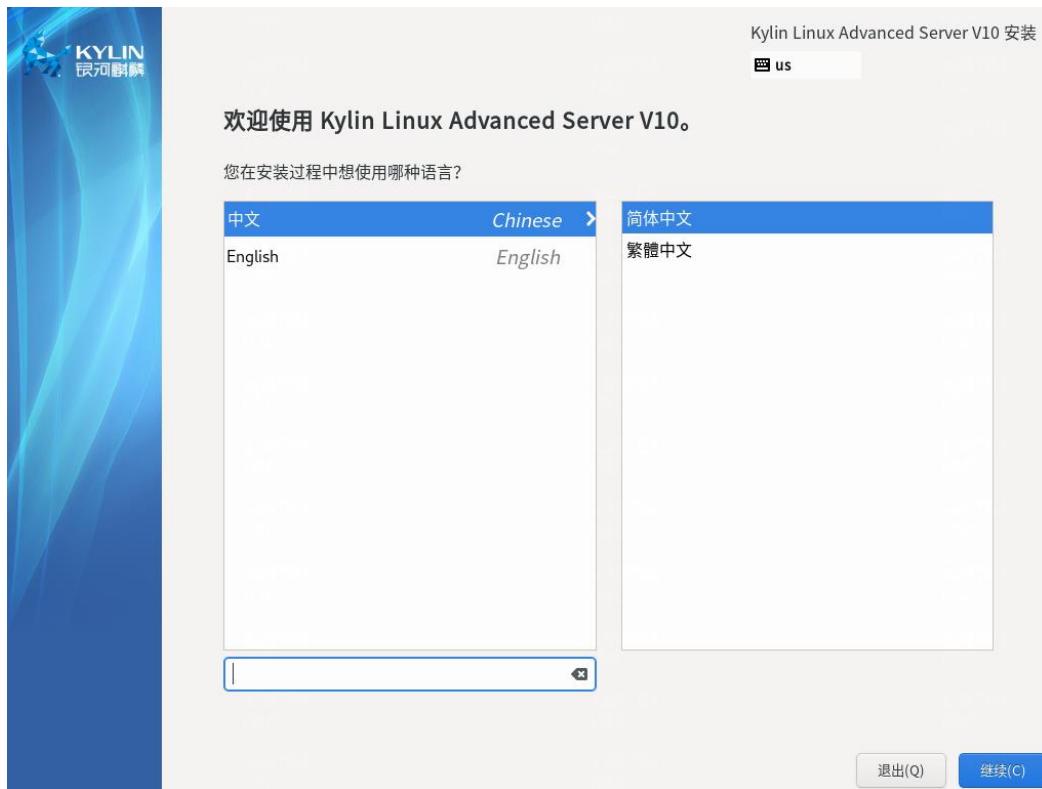
## 6.3 安装操作系统

本节以从未安装过任何操作系统的服务器为例，请使用 LinSeer RT 配套的 Kylin-Server-10-SP2-x86-Release-Build09 ISO 文件安装操作系统。

### 6.3.1 基本设置

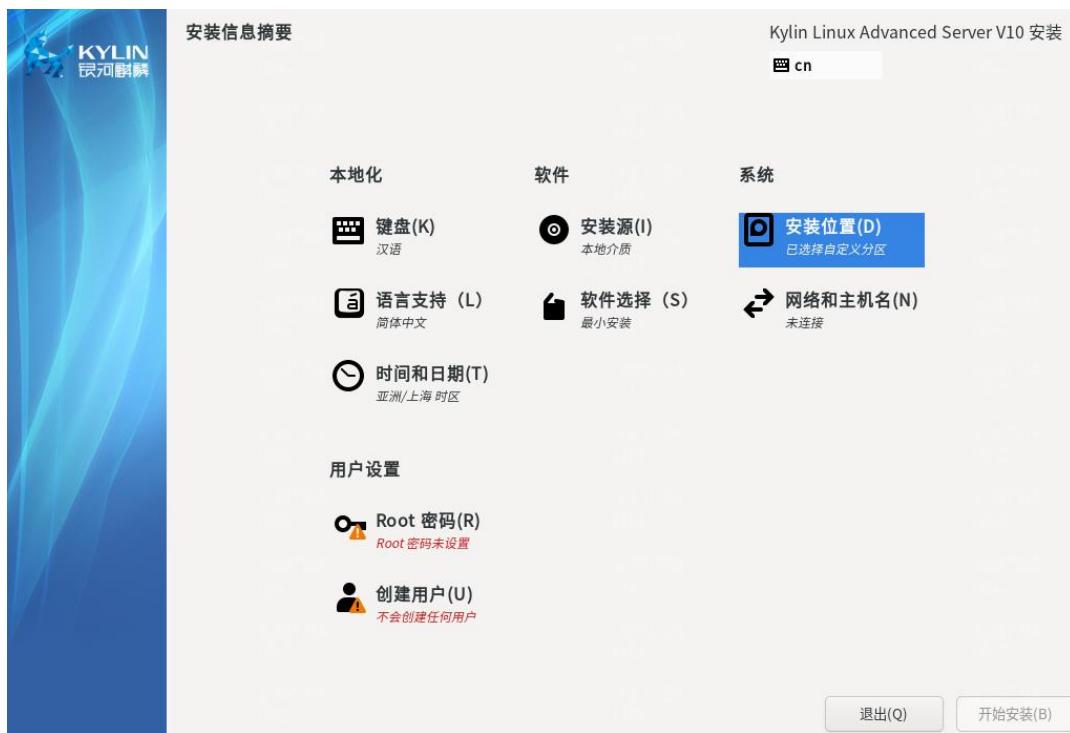
- (1) ISO 镜像文件加载完成后，进入语言选择界面。
- (2) 选择安装语言，此处以“中文”为例，单击<继续(C)>按钮，进入安装信息摘要界面，如图 6-22所示。

图6-22 语言选择界面



- (3) 在安装信息摘要界面的本地化区域单击“日期和时间(T)”链接，进入日期和时间界面，如图 6-23所示。

图6-23 安装信息摘要界面



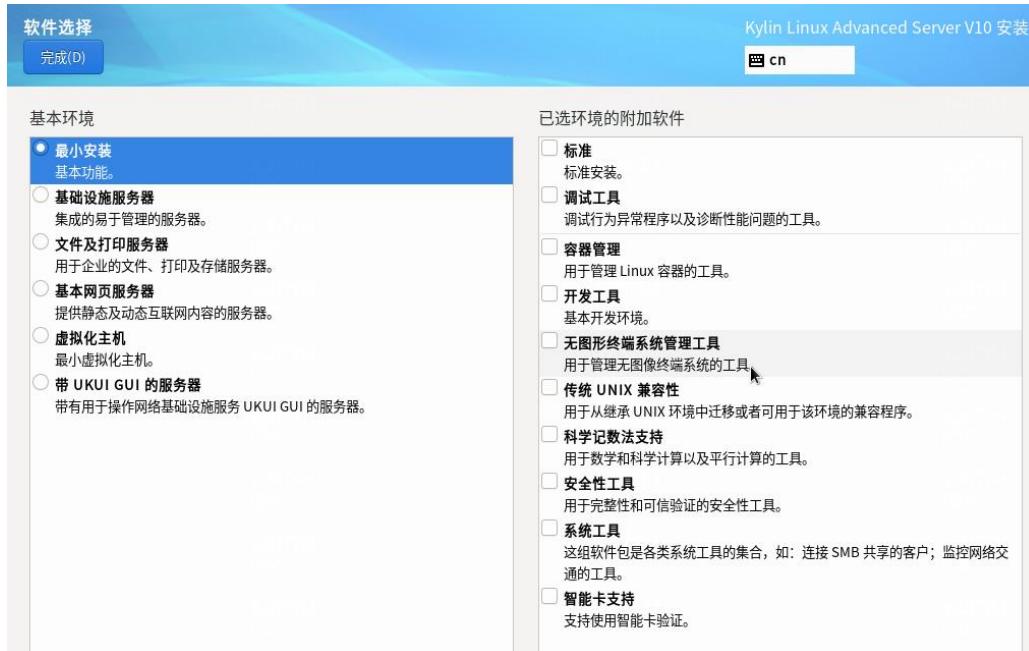
- (4) 设置系统的日期和时间，如图 6-24 所示。选择地区为“亚洲”，城市为“上海”，单击<完成>按钮，返回安装信息摘要界面。

图6-24 设置系统的日期和时间



(5) 安装信息摘要界面的软件区域“软件选择(S)”选择基本环境为“最小安装”，如图 6-25 所示。

图6-25 选择基本环境为“最小安装”



- (6) 在安装信息摘要界面的系统区域，单击“安装位置”链接，进入安装目标位置界面。设置安装目标位置，如图 6-26 所示。在本地标准磁盘区域选择目标磁盘。在存储配置区域选择“自定义(C)”，单击<完成>按钮，进入手动分区界面。

图6-26 安装目标位置界面



### 6.3.2 磁盘分区

在“新挂载点将使用以下分区方案(N)”下拉列表中选择分区方案，支持选择“标准分区”和“LVM”，建议选择“标准分区”。

- 使用“LVM”分区。  
在“设备类型”下拉列表中选择“LVM”，并依次创建各分区。
- 使用“标准分区”。  
手动创建磁盘分区，如图 6-27 所示。磁盘分区的具体信息如表 6-1 所示，支持手动修改配置信息。
  - 设备类型：选择“标准分区”，并单击 + 图标，在弹出窗口中进行如下配置后，单击<添加挂载点>按钮。
  - 挂载点：输入挂载点目录名称。
  - 期望容量：输入磁盘容量并指定容量单位，例如“GiB”、“MiB”。
  - 文件系统(Y)：在下拉列表中选择推荐的文件系统。
  - 设备：单击<修改(M)...>按钮，在弹出的配置挂载点窗口中，选择分区所挂载的磁盘后，单击<选择>按钮。



注意

boot 相关分区不能设置为 LVM，否则分区将会配置失败。

图6-27 手动进行磁盘分区（标准分区）

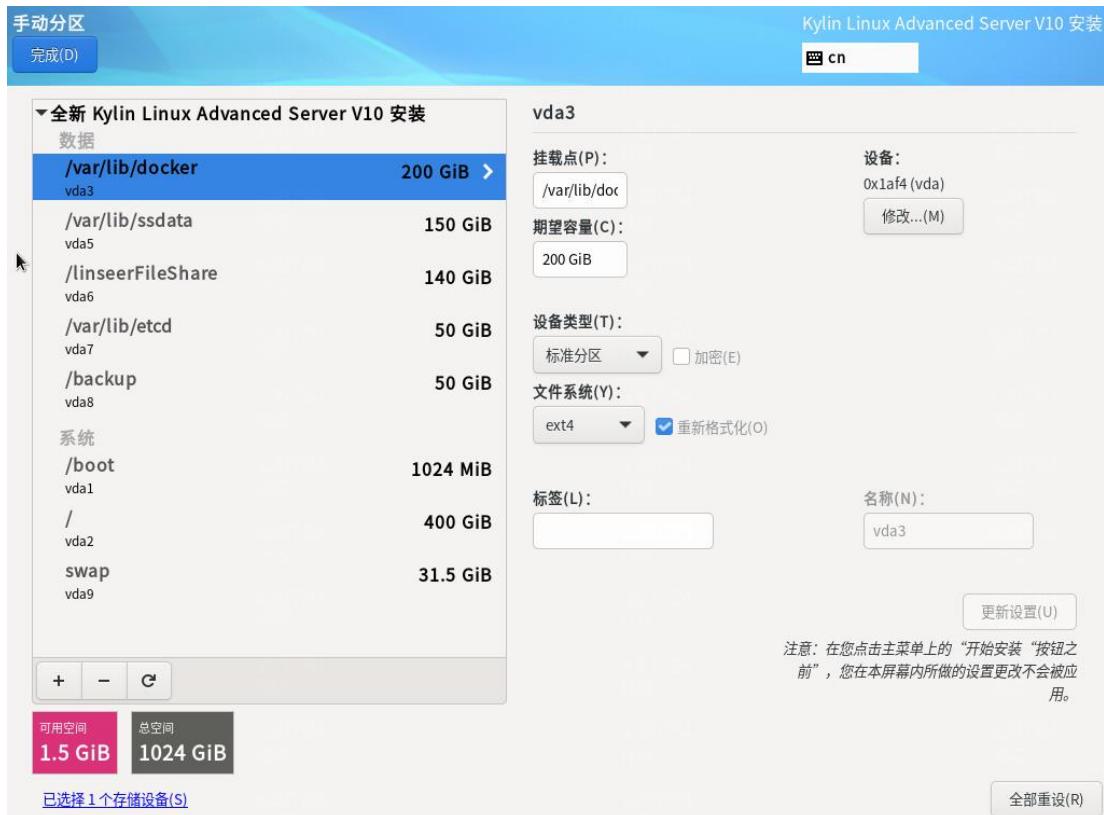


表6-1 磁盘分区要求

分区名称	预设容量	适用模式	文件系统	备注
/	1 TB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/docker	350 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/ssdata	200 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/var/lib/etc	50 GiB	BIOS模式/UEFI模式	ext4	磁盘空间充足时，可适当增大
/boot	1024 MiB	BIOS模式/UEFI模式	ext4	
swap	1024 MiB	BIOS模式/UEFI模式	swap	
/backup	50GiB	BIOS模式/UEFI模式	ext4	麒麟系统中设置的一个备份分区
/boot/efi	200 MiB	UEFI模式	EFI System Partition	根据服务器的BIOS启动模型选择创建对应分区，Legacy: /biosboot

				UEFI: /boot/efi
/linseerFileShare (默认名称)	1 TB	BIOS模式/UEFI模式	ext4	单机环境需要创建该分区，磁盘空间充足时，可适当增大 集群环境使用NFS存储，不需要划分该分区

各主要分区的作用如下：

- **/:** Matrix 使用，包括 K8s、Harbor 和各组件上传的安装包，该分区的容量和各组件上传的镜像大小有关，需要确定各组件占用的磁盘容量大小，在此基础上扩缩容。
- **/var/lib/docker/:** 与 Docker 的运行相关，需要根据实际运行情况确定容量大小。
- **/var/lib/ssddata/:** 供 Kafka、ZooKeeper 使用。
- **/linseerFileShare (默认名称):** 推荐预设容量为 1TB。
  - 单机部署模式下，/linseerFileShare 供推理组件使用，作为模型仓库的共享目录。
  - 3 机集群部署模式、3+N 机集群部署模式下，使用外置的 NFS 存储空间（可自定义名称）供推理组件使用。

各部署模式下的“/linseerFileShare”磁盘空间分配详情，如[表 6-2](#) 所示。

表6-2 “/linseerFileShare” 磁盘空间分配

组件	目录名	存储空间大小计算方式	默认大小
模型仓库	\$(sharePath)/model_preset, 默认名称为“/linseerFileShare”	<ul style="list-style-type: none"> <li>• LinSeer_Model 文件: 132 GiB</li> <li>• LinSeer_Model_Lite 文件: 15 GiB</li> </ul>	300 GiB

(2) 在弹出的更改摘要窗口中，单击<接受更改>按钮，返回安装信息摘要界面，如[图 6-28](#) 所示。

图6-28 更改摘要窗口



### 6.3.3 用户及网络设置

- (1) 选择管理员帐户设置。使用 `root` 用户作为管理员帐户，无需创建新用户。
  - a. 在安装信息摘要界面用户设置区域单击“Root 密码(R)”配置项设置 `root` 用户作为管理员帐户，如图 6-23 所示。
  - b. 设置 `root` 用户密码，单击<完成>按钮，返回安装信息摘要界面，如图 6-29 所示。

图6-29 设置 root 用户密码



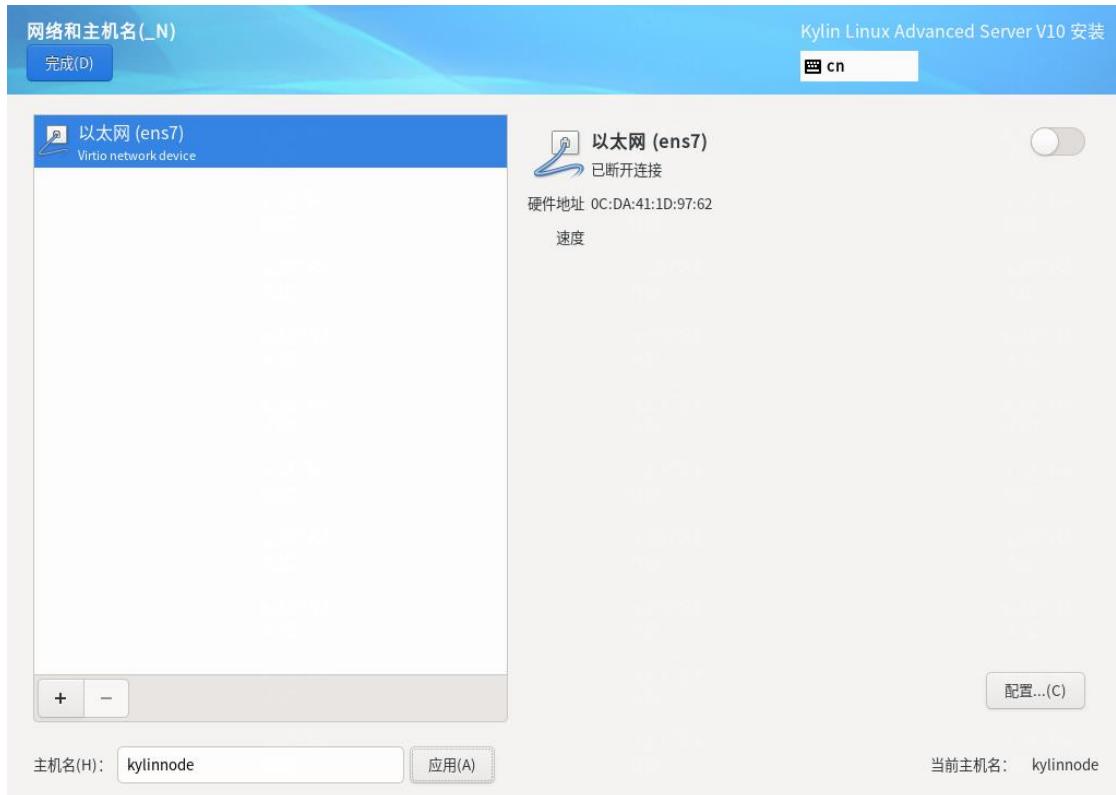
- c. 在安装信息摘要界面可查看 root 用户已作为管理员帐户，该用户拥有所有功能的操作权限，如图 6-30 所示。

图6-30 使用 root 用户作为管理员帐户



- (2) 在系统区域单击“网络和主机名(N)”链接，进入网络和主机名界面。  
(3) 在主机名文本框中输入主机名后，单击<应用>按钮，如图 6-31 所示。

图6-31 网络和主机名配置界面

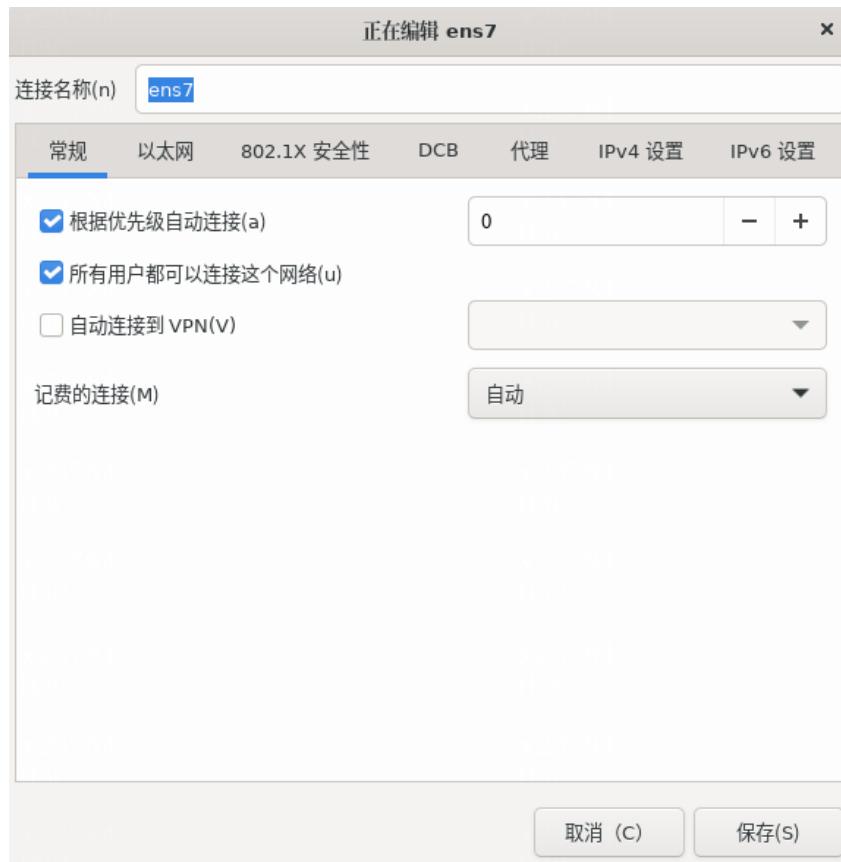


### 说明

- 请勿使用默认主机名（localhost、localhost.localdomain、localhost4、localhost4.localdomain4、localhost6、localhost6.localdomain6）。主机名称最长 63 个字符，仅支持小写字母、数字、连字符和小数点，不能以 0 开头且全为数字，不能以 0x、连字符、小数点开头，以连字符、小数点结尾。
- 建立 Matrix 集群时，必须保证集群内各个节点的主机名互不相同，且符合主机名的命名规则，否则将会导致集群建立失败。
- Matrix 集群部署前，若需要修改节点的主机名，可在节点操作系统的命令行界面，通过 `hostnamectl set-hostname hostname` 命令进行修改，其中 `hostname` 为修改后的主机名。新主机名将在节点重启后生效。
- Matrix 集群部署完成后，请不要再对操作系统的主机名进行修改。

- (4) 在网络和主机名配置界面可配置网卡。单击<配置>按钮，在弹出的网络配置窗口中进行网卡配置。单击“常规”页签，勾选“自动以优先级连接(A)”项，“所有用户都可以连接这个网络(U)”项保持默认勾选，如图 6-32 所示。

图6-32 常规页签



(5) 配置 Matrix 节点的 IPv4 地址。

单击“IPv4 设置”页签，在“方法(M)”下拉框中选择“手动”，在地址区域单击<添加(A)>按钮，配置服务器的 IPv4 地址（规划的 Master 节点 IP），配置完成后，单击<保存>按钮保存配置，如图 6-33 所示。

图6-33 配置服务器的 IPv4 地址



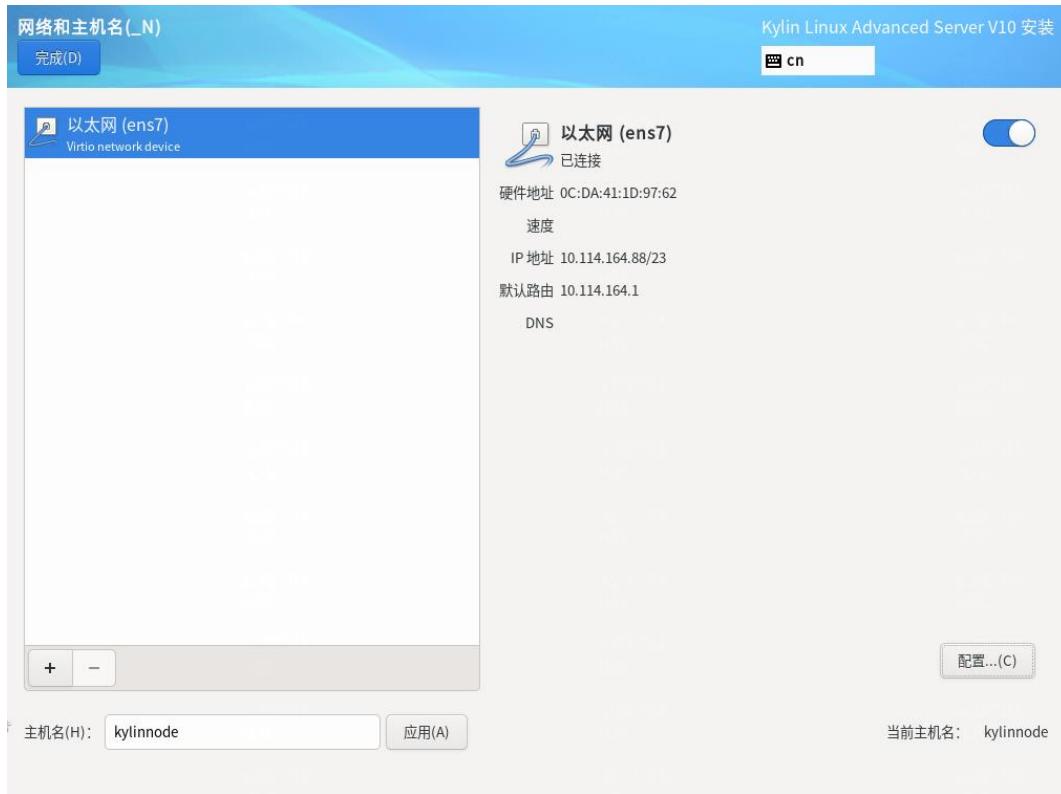
---

 说明

- 配置 IPv4 地址时必须指定网关，否则在创建集群时可能出现问题。
  - Matrix 单独使用一个网口，不允许在此网口上配置子接口及子 IP。
  - Matrix 节点其它网口的 IP 地址，不能和建立集群使用的 IP 处于同一网段。
  - 不允许在操作系统中配置 DNS 服务器。
- 

(6) 网络配置完成后，手动启用指定物理网卡，如图 6-34 所示。单击<完成>按钮，返回安装信息摘要界面。

图6-34 配置服务器的 IPv4 地址



#### 6.3.4 操作系统安装

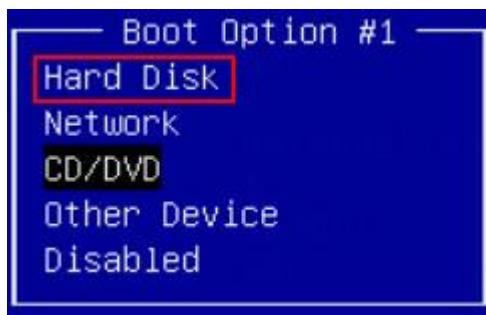
- (1) 在本地电脑控制端窗口执行命令 `ping ip_address`, 检查配置的 IP 地址是否连通。其中, `ip_address` 为 IPv4 设置页签下配置的 IP 地址。若可 ping 通, 则继续进行后续步骤, 否则返回 IPv4 设置页签, 检查掩码网关等信息是否配置正确。
- (2) 在图 6-35 所示的安装信息摘要界面中单击<开始安装>按钮, 开始安装操作系统。

图6-35 安装操作系统



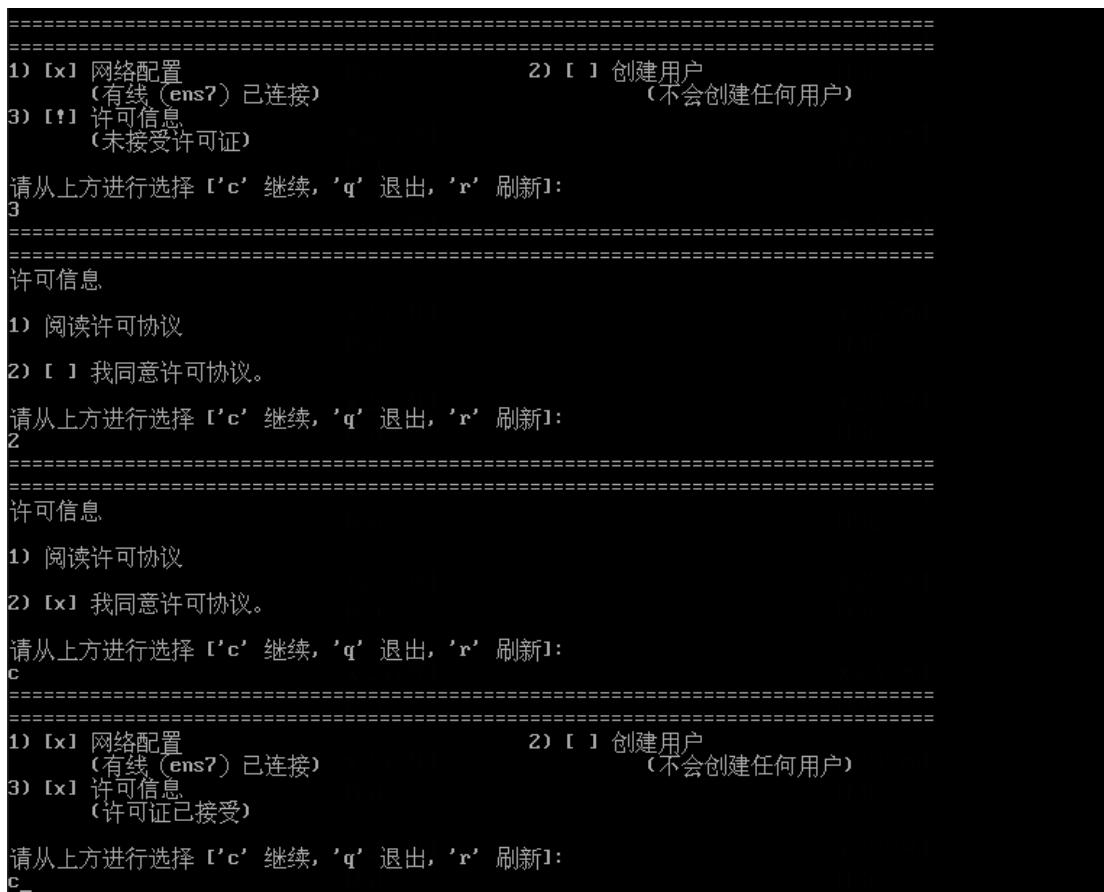
- (3) 安装完成后，单击<重启系统(R)>按钮。进入 BIOS 设置界面，设置 BIOS 从 Hard Disk 启动，即设置 Boot 启动项时，在弹出的“Boot Option #1”对话框中，按<Enter>键选择“Hard Disk”，如图 6-36 所示。具体步骤与设置 BIOS 从 CD/DVD 启动类似，请参见“[6.2 设置 BIOS 从 CD/DVD 启动](#)”。

图6-36 设置 Boot 启动项



- (4) 保存退出后，服务器再次自动重启，重启后的界面如图 6-37 所示。根据提示，输入“3”回车键确认，选择“2”，同意许可协议，回车键确认。输入“c”继续完成选项。再次输入“c”继续，进入 kylin 系统。

图6-37 安装完成界面



## 7 安装部署 KylinOS Matrix 集群

### 7.1 关闭防火墙

(1) 在服务器上执行 **systemctl stop firewalld.service** 命令关闭防火墙。

```
[root@localhost ~]# systemctl stop firewalld.service
```

(2) 执行 **systemctl status firewalld.service** 命令查看防火墙状态，确保防火墙已关闭。

```
[root@localhost ~]# systemctl status firewalld.service
○ firewalld.service - firewalld - dynamic firewall daemon
    Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor
    preset: enabled)
    Active: inactive (dead) //防火墙已关闭
      Docs: man:firewalld(1)
```

(3) 执行 **systemctl disable firewalld.service** 命令设置防火墙开机不启动。

```
[root@localhost ~]# systemctl disable firewalld.service
```

```
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
```

```
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
```

## 7.2 设置时区时间

- (1) 使用命令 **timedatectl** 查看当前时区及对应时间，若与当前本地时间不一致，需要手动修改时间。时区设置为 **CST,Shanghai**，时间设置为本地实际时间。

```
[root@localhost ~]#timedatectl
    Local time: 三 2024-04-17 08:36:11 CST
    Universal time: 三 2024-04-17 00:36:11 UTC
        RTC time: 三 2024-04-17 08:36:12
        Time zone: Asia/Shanghai (CST, +0800)

System clock synchronized: yes
          NTP service: active
      RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
This mode cannot be fully supported. It will create various problems
with time zone changes and daylight saving time adjustments. The RTC
time is never updated, it relies on external facilities to maintain it.
If at all possible, use RTC in UTC by calling
'timedatectl set-local-rtc 0'.
```

- (2) 使用命令 **sudo timedatectl set-ntp off** 关闭自动时间同步功能，

```
[root@beijing-ning25 ~]# sudo timedatectl set-ntp off
```

- (3) 使用命令 **sudo timedatectl set-time '2024-04-17 16:44:00'** 设置正确时间。

```
[root@beijing-ning25 ~]# sudo timedatectl set-time '2024-04-17 16:44:00'
```

```
[root@beijing-ning25 ~]# timedatectl
    Local time: 三 2024-04-17 16:44:05 CST
    Universal time: 三 2024-04-17 08:44:05 UTC
        RTC time: 三 2024-04-17 16:44:05
        Time zone: Asia/Shanghai (CST, +0800)

System clock synchronized: no
          NTP service: inactive
      RTC in local TZ: yes
```

```
Warning: The system is configured to read the RTC time in the local time zone.
This mode cannot be fully supported. It will create various problems
with time zone changes and daylight saving time adjustments. The RTC
time is never updated, it relies on external facilities to maintain it.
If at all possible, use RTC in UTC by calling
'timedatectl set-local-rtc 0'.
```

- (4) 使用命令 **sudo timedatectl set-ntp on** 手动开启自动时间同步功能。

```
[root@beijing-ning25 ~]# sudo timedatectl set-ntp on
```

## 7.3 安装kylin Matrix

- (1) 向技术支持获取 **kylin Matrix** 软件安装包、**kylin** 依赖包及部署脚本，并将软件包通过 **FTP** 工具上传至服务器的待安装目录（例如/**root**）下。

```
[root@ localhost~]# ll  
-rw-r--r-- 1 root root      4090 12月 12 16:48 install.sh  
-rw-r--r-- 1 root root  241974091 12月 12 16:48 matrix_dep_kylin.zip  
-rw-r--r-- 1 root root 1658653207 12月 12 16:48  
Matrix-V100R001B01D014KYLINTESTSP03-x86_64.zip
```

- (2) 进入 **Matrix** 软件包(**.zip**文件)的存放路径，执行命令 **sh install.sh**，完成部署 **kylin matrix**。

```
[root@ localhost~]# sh install.sh  
fs.inotify.max_user_instances = 65000  
kernel.sysrq = 0  
net.ipv4.ip_forward = 0  
net.ipv4.conf.all.send_redirects = 0  
net.ipv4.conf.default.send_redirects = 0  
net.ipv4.conf.all.accept_source_route = 0  
...  
Installing...  
[install] -----  
[install]   Matrix-V100R001B01D014KYLINTESTSP03-x86_64  
[install]   Kylin Linux Advanced Server release V10 (Sword)  
[install]   Linux 4.19.90-24.4.v2101.ky10.x86_64  
[install] -----  
Kylin Linux Advanced Server release V10 (Sword)  
[install] WARNING: To avoid unknown error, do not interrupt this installation procedure.  
[install] Checking environment...  
[install] Done.  
[install] Checking current user permissions...  
[install] Done.  
[install] Decompressing matrix package...  
[install] Done.  
[install] Installing dependent software...  
...  
Complete!
```

- (3) 通过命令 **systemctl status matrix** 验证 **Matrix** 服务是否安装成功。若安装成功，则将在 **Active** 字段后显示运行信息为 **active (running)**。剩余节点执行同样操作即可。

```
[root@localhost ~]# systemctl status matrix
```

```
● matrix.service - Matrix Server
```

```

Loaded: loaded (/usr/lib/systemd/system/matrix.service; enabled; vendor preset:
disabled)

Active: active (running) since Thu 2024-12-05 09:41:52 CST; 3 days ago
Main PID: 108490 (karaf)
Tasks: 357
Memory: 2.3G
CGroup: /system.slice/matrix.service
        └─108490 /bin/sh /opt/matrix/bin/karaf server
          └─ 108789 /usr/bin/java -XX:+UnlockDiagnosticVMOptions
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=/opt/matrix/heapdumps>

12月 08 15:54:01 kylintg sudo[94582]: pam_unix(sudo:session): session closed for user
root

12月 08 15:54:04 kylintg sudo[95962]: root : TTY=unknown ; PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile; >

12月 08 15:54:04 kylintg sudo[95962]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

12月 08 15:54:05 kylintg sudo[95962]: pam_unix(sudo:session): session closed for user
root

12月 08 15:54:08 kylintg sudo[96325]: root : TTY=unknown ; PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile; >

12月 08 15:54:08 kylintg sudo[96325]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

12月 08 15:54:09 kylintg sudo[96325]: pam_unix(sudo:session): session closed for user
root

12月 08 15:54:11 kylintg sudo[96599]: root : TTY=unknown ; PWD=/opt/matrix ; USER=root ;
COMMAND=/bin/bash -c source /etc/profile; >

12月 08 15:54:11 kylintg sudo[96599]: pam_unix(sudo:session): session opened for user
root(uid=0) by (uid=0)

12月 08 15:54:12 kylintg sudo[96599]: pam_unix(sudo:session): session closed for user
root

lines 1-20/20 (END)

```

## 8 安装 GPU 驱动及相关工具包

所有 GPU 的服务器均需要安装 GPU 驱动及相关工具包，请联系技术支持获取所需的软件安装包。安装适配的 GPU 驱动程序及工具包，能确保显卡在操作系统中被正确识别，可提高深度学习速度、提升推理性能，并能提供多任务并行处理、大规模数据处理等功能。



说明

单台 GPU 服务器不支持不同型号的 GPU 混插。

---

## 8.1 安装NVIDIA GPU驱动及工具包

### 8.1.1 查看显卡型号

在服务器上执行 **lspci -k | grep -A 2 -E "(VGA|3D)"** 命令查看显卡型号。

```
[root@localhost ~]# lspci -k | grep -A 2 -E "(VGA|3D)"
25:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

2b:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

63:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

68:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

74:00.0 VGA compatible controller: ASPEED Technology, Inc. ASPEED Graphics Family (rev 41)
    DeviceName: Onboard Video
    Subsystem: ASPEED Technology, Inc. ASPEED Graphics Family
--

9f:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

a4:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

e2:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
--

e8:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
    Subsystem: NVIDIA Corporation Device 1463
    Kernel driver in use: nvidia
```

## 8.1.2 下载显卡驱动

登录 NVIDIA 官网，并进入“驱动程序下载”栏目下载显卡驱动。

- (1) 登录 NVIDIA 官网: <https://www.nvidia.cn/Download/index.aspx?lang=cn>
- (2) 请设置驱动参数。
  - a. 参见表 8-1 设置 NVIDIA 驱动程序下载参数。
  - b. 请根据实际环境设置操作系统参数，NingOS 操作系统选择“Linux 64-bit”，KylinOS 操作系统选择“Linux 64-bit KylinOS 10”。
  - c. 单击<搜索>按钮。

图8-1 设置 NingOS 驱动参数

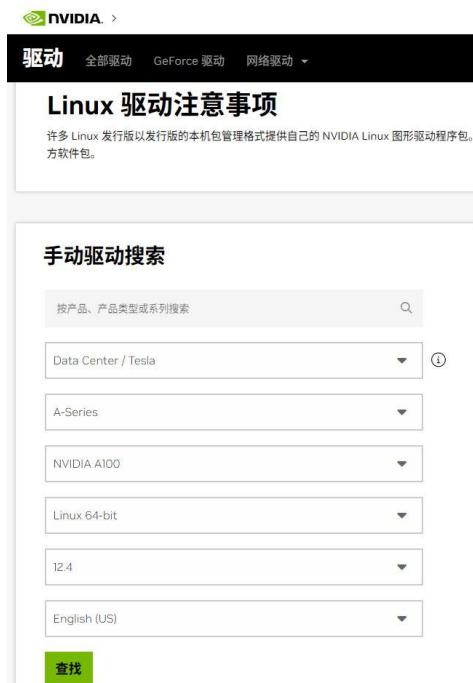


图8-2 设置 KylinOS 驱动参数

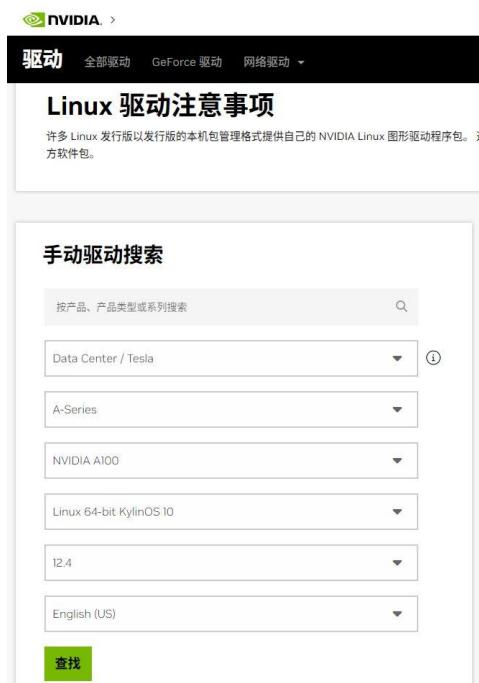
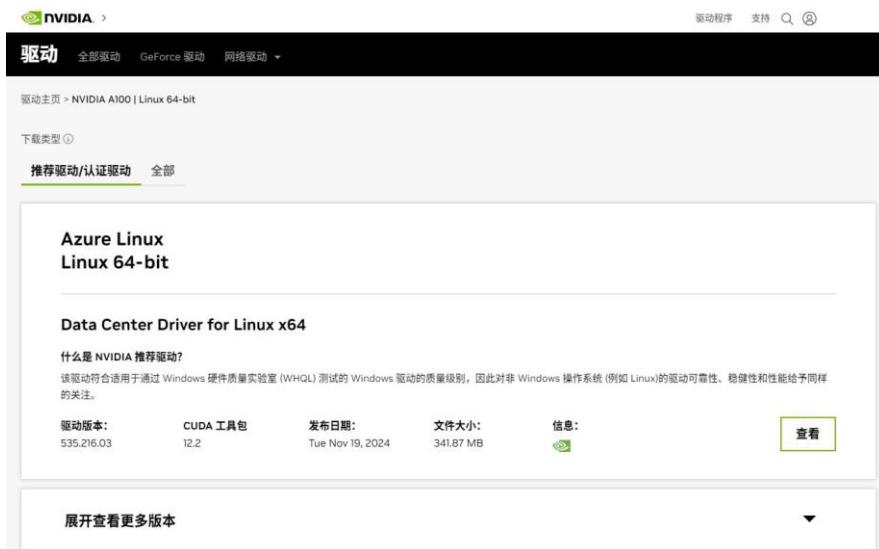


表8-1 显卡与 CUDA Toolkit 版本对应关系表

产品类型	产品系列	产品家族 (显卡型号)	驱动版本	CUDA Toolkit 版本
Tesla	L-Series	NVIDIA L20	NVIDIA-SMI 550.54.15	CUDA 12.4
Tesla	L-Series	NVIDIA L40	NVIDIA-SMI 550.127.08	CUDA 12.4
Tesla	A-Series	NVIDIA A30	NVIDIA-SMI 550.54.15	CUDA 12.4
Tesla	A-Series	NVIDIA A100	NVIDIA-SMI 535.54.03	CUDA 12.2
Tesla	A-Series	NVIDIA A800	NVIDIA-SMI 535.54.03	CUDA 12.2
Tesla	HGX-Series	NVIDIA H800	NVIDIA-SMI 550.54.15	CUDA 12.4
Tesla	HGX-Series	NVIDIA H20	NVIDIA-SMI 535.161.08	CUDA 12.2

(3) 如图 8-3 所示，搜索到目标显卡驱动后，单击<下载>按钮下载显卡驱动。

图8-3 下载显卡驱动



### 8.1.3 禁用 Linux 自带的显卡驱动

Nouveau 是 Linux 自带的显卡驱动，需要禁用，否则安装 NVIDIA 显卡驱动时会提示冲突。

(1) 输入 `lsmod | grep nouveau` 命令，若显示如下信息则表示 Nouveau 已被启用，需要禁用 Nouveau。

```
[root@localhost ~]# lsmod | grep nouveau
nouveau          1869689  0
mxm_wmi          13021   1 nouveau
wmi              21636   2 mxm_wmi,nouveau
video            24538   1 nouveau
i2c_algo_bit     13413   1 nouveau
ttm              114635  2 cirrus,nouveau
drm_kms_helper  179394  2 cirrus,nouveau
drm              429744  5 ttm,drm_kms_helper,cirrus,nouveau
```

(2) 禁用 Nouveau。

a. 执行 `vi` 命令分别创建如下两个文件。

```
[root@localhost ~]# vi /etc/modprobe.d/nvidia-installer-disable-nouveau.conf
[root@localhost ~]# vi /lib/modprobe.d/nvidia-installer-disable-nouveau.conf
```

b. 通过键盘输入“i”进入编辑界面，输入如下内容：

```
blacklist nouveau//屏蔽 nouveau
options nouveau modeset=0//如果不加本行，字符界面下，可能导致显示器黑屏，无法安装驱动
```

c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

(3) 更新 initramfs 镜像 (img 格式) 文件

a. 通过 `mv` 命令重命名镜像文件（作为备份）。

```
[root@localhost ~]# mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).img.bak
```

b. 通过 **dracut** 命令更新镜像文件信息。

```
[root@localhost ~]# dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

- (4) 通过 **reboot** 命令重启设备后，再次输入 **lsmod | grep nouveau** 命令，显示信息为空，则表示 **Nouveau** 已被禁用。

```
[root@localhost ~]# lsmod | grep nouveau  
[root@localhost ~]#
```

### 8.1.4 安装显卡驱动

- (1) 配置本地 YUM 源，分别执行如下命令安装相关依赖包。

```
[root@localhost ~]# yum install gcc  
[root@localhost ~]# yum install kernel-devel  
[root@localhost ~]# yum install make
```

- (2) 将显卡驱动安装包通过 **FTP** 工具上传至服务器的待安装目录（例如/**root**）下。

- (3) 在服务器上执行 **chmod +x NVIDIA-Linux-x86\_64-515.105.01.run** 命令将显卡驱动安装包的权限改为可执行。

```
[root@localhost ~]# chmod +x NVIDIA-Linux-x86_64-515.105.01.run
```

- (4) 安装显卡驱动

a. 执行 **./NVIDIA-Linux-x86\_64-515.105.01.run** 命令安装显卡驱动。

```
[root@localhost ~]# ./NVIDIA-Linux-x86_64-515.105.01.run
```

b. 安装过程中各步骤均点击<YES>或者<OK>按钮，直到显卡驱动安装完成。

- (5) 执行 **nvidia-smi** 命令，若显示信息如下说明显卡驱动安装成功。

```
[root@localhost ~]# nvidia-smi  
Mon Nov 13 03:39:19 2023  
+-----+  
| NVIDIA-SMI 515.105.01 Driver Version: 515.105.01 CUDA Version: 11.7 |  
+-----+-----+-----+  
| GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC | | | |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |  
| | | | | | MIG M. |  
+=====+=====+=====+=====+=====+=====+  
| 0 NVIDIA A100-SXM... On | 00000000:25:00.0 Off | 0 | | | |
| N/A 31C P0 62W / 400W | 0MiB / 81920MiB | 0% Default |  
| | | | | | Disabled |  
+-----+-----+-----+  
| 1 NVIDIA A100-SXM... On | 00000000:2B:00.0 Off | 0 | | | |
| N/A 34C P0 61W / 400W | 0MiB / 81920MiB | 0% Default |  
| | | | | | Disabled |  
+-----+-----+-----+  
| 2 NVIDIA A100-SXM... On | 00000000:63:00.0 Off | 0 | | | |
| N/A 34C P0 63W / 400W | 0MiB / 81920MiB | 0% Default |  
| | | | | | Disabled |  
+-----+-----+-----+  
| 3 NVIDIA A100-SXM... On | 00000000:68:00.0 Off | 0 | | | |
| N/A 32C P0 62W / 400W | 0MiB / 81920MiB | 0% Default |  
| | | | | | Disabled |
```

```

+-----+-----+
| 4 NVIDIA A100-SXM... On | 00000000:9F:00.0 Off | 0 | | |
| N/A 32C P0 62W / 400W | 0MiB / 81920MiB | 0% Default |
| | | | | Disabled |
+-----+-----+
| 5 NVIDIA A100-SXM... On | 00000000:A4:00.0 Off | 0 | | |
| N/A 35C P0 65W / 400W | 0MiB / 81920MiB | 0% Default |
| | | | | Disabled |
+-----+-----+
| 6 NVIDIA A100-SXM... On | 00000000:E2:00.0 Off | 0 | | |
| N/A 35C P0 64W / 400W | 0MiB / 81920MiB | 0% Default |
| | | | | Disabled |
+-----+-----+
| 7 NVIDIA A100-SXM... On | 00000000:E8:00.0 Off | 0 | | |
| N/A 33C P0 62W / 400W | 0MiB / 81920MiB | 0% Default |
| | | | | Disabled |
+-----+-----+
...

```

## 8.2 安装nvidia-container-runtime

### 8.2.1 NingOS 操作系统安装 nvidia-container-runtime

- (1) 向技术支持获取离线安装包“nvidiaRuntimeOffline.zip”。
- (2) 将 NVIDIA 运行时工具离线安装包，通过 FTP 工具上传至服务器的待安装目录（例如/root）下。
- (3) 执行 **unzip** 命令解压缩离线安装包。

```
[root@localhost ~]# unzip NvidiaRuntimeOffline.zip
Archive:  NvidiaRuntimeOffline.zip
  creating: NvidiaRuntimeOffline/
  creating: NvidiaRuntimeOffline/rpm/
...

```

- (4) 执行 **cd** 命令进入离线安装包文件夹。
- (5) 执行 **rpm -Uvh --force --nodeps \*.rpm** 命令安装软件包。
- (6) 执行 **systemctl restart docker** 命令重启 docker 服务，使 NVIDIA 运行时生效。
- (7) 分别执行如下命令检查 NVIDIA 运行时工具安装情况。如显示如下信息，则表示该工具安装成功。

```
[root@localhost ~]# whereis nvidia-container-runtime
nvidia-container-runtime: /usr/bin/nvidia-container-runtime
/etc/nvidia-container-runtime
[root@localhost ~]# whereis nvidia-container-runtime-hook
nvidia-container-runtime-hook: /usr/bin/nvidia-container-runtime-hook
```

## 8.2.2 KylinOS 操作系统安装 nvidia-container-runtime

- (1) 向技术支持获取离线安装包“**rpm.zip**”。
- (2) 将 NVIDIA 运行时工具离线安装包，通过 **FTP** 工具上传至服务器的待安装目录（例如/**root**）下。
- (3) 执行 **unzip** 命令解压缩离线安装包。

```
[root@localhost ~]# unzip rpm.zip
inflating: rpm/libnvidia-container1-1.13.1-1.x86_64.rpm
inflating: rpm/nvidia-container-toolkit-1.13.1-1.x86_64.rpm
inflating: rpm/libnvidia-container-tools-1.13.1-1.x86_64.rpm
inflating: rpm/nvidia-container-toolkit-base-1.13.1-1.x86_64.rpm
inflating: rpm/nvidia-container-runtime-3.13.0-1.noarch.rpm
```

- (4) 执行 **cd** 命令进入离线安装包文件夹。

```
[root@localhost ~]# cd /rpm/
```

- (5) 执行 **rpm -Uvh --force --nodeps \*.rpm** 命令安装软件包。

```
[root@localhost rpm]# rpm -Uvh --force --nodeps *.rpm
```

- (6) 执行 **systemctl restart docker** 命令重启 **docker** 服务，使 NVIDIA 运行时生效。

```
[root@localhost ~]# systemctl restart docker
```

- (7) 分别执行如下命令检查 NVIDIA 运行时工具安装情况。如显示如下信息，则表示该工具安装成功。

```
[root@localhost ~]# whereis nvidia-container-runtime
nvidia-container-runtime: /usr/bin/nvidia-container-runtime
/etc/nvidia-container-runtime
[root@localhost ~]# whereis nvidia-container-runtime-hook
nvidia-container-runtime-hook: /usr/bin/nvidia-container-runtime-hook
```

## 8.3 安装nvidia-device-plugin

### 8.3.1 修改默认运行时

集群每个节点都需要将容器默认运行时修改为 NVIDIA 运行时。

- (1) 修改/etc/docker/daemon.json 文件。

- a. 执行 **vi** 命令修改文件。

```
[root@localhost ~]# vi /etc/docker/daemon.json
```

- b. 通过键盘输入“i”进入编辑界面，在文件开头增加配置（见灰显信息）。

```
{
  "default-runtime": "nvidia",
  "runtimes": {
    "nvidia": {
      "path": "/usr/bin/nvidia-container-runtime",
      "runtimeArgs": []
    }
  },
  "live-restore": true,
```

```

    "exec-opts": ["native.cgroupdriver=systemd"],
    "insecure-registries": ["matrix-registry.h3c.com:8088",
    "matrix-registry-master1.h3c.com:8088"],
    "bridge": "none",
    "log-driver": "json-file",
    "log-opt": {"max-size": "50m", "max-file": "3"}
}

```

c. 按键盘上的`<ESC>`键，再输入“`:wq`”，退出配置并保存文件。

```
:wq
```

(2) 执行 `systemctl restart docker` 命令重启 docker 服务，使 NVIDIA 设备插件生效。

```
[root@localhost ~]# systemctl restart docker
```

### 8.3.2 部署 NVIDIA 设备插件（在线部署方式）

(1) 在任一 Master 节点的服务器上执行 `wget` 命令下载 `nvidia-device-plugin.yml` 文件。

```

[root@localhost ~]# wget
https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.14.1/nvidia-device-pl
ugin.yml
--2023-11-03 09:55:36--
https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.14.1/nvidia-device-pl
ugin.yml
正在解析主机 raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.111.133,
185.199.108.133, 185.199.109.133, ...
正在连接 raw.githubusercontent.com (raw.githubusercontent.com) |185.199.111.133|:443...
已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度: 1894 (1.8K) [text/plain]
正在保存至: “nvidia-device-plugin.yml”

nvidia-device-plugin.yml
100%[=====] 1.85K --.KB/s 用时 0s

```

2023-11-03 09:55:36 (50.1 MB/s) - 已保存 “nvidia-device-plugin.yml” [1894/1894]

(2) 在各 GPU 服务器上执行 `docker pull` 命令下载 `nvidia-device-plugin` 镜像。

```

[root@localhost ~]# docker pull nvcr.io/nvidia/k8s-device-plugin:v0.14.1
v0.14.1: Pulling from nvidia/k8s-device-plugin
56e0351b9876: Pull complete
50292f59408b: Pull complete
bac1f8a1c195: Pull complete
0b0815e859ed: Pull complete
8abd77f1cc9e: Pull complete
d473781522df: Pull complete
76e0ddc91db3: Pull complete
1597e60bbfb: Pull complete
86dc2aec77cc: Pull complete
c9ab9a167444: Pull complete
Digest: sha256:15c4280d13a61df703b12d1fd1b5b5eec4658157db3cb4b851d3259502310136

```

```
Status: Downloaded newer image for nvcr.io/nvidia/k8s-device-plugin:v0.14.1
nvcr.io/nvidia/k8s-device-plugin:v0.14.1
```

- (3) 根据集群部署环境的实际情况选择合适的方式部署 nvidia-device-plugin.yml。
- 如果集群所有服务器都支持 GPU，按照步骤(4)部署。
  - 如果集群只有部分服务器支持 GPU，按照步骤(5)部署。
- (4) 集群所有服务器都支持 GPU 时，请在任一个 Master 节点上执行 **kubectl apply -f** 命令部署 nvidia-device-plugin.yml 文件，其他节点无需重复部署。例如：
- ```
[root@localhost ~]# kubectl apply -f nvidia-device-plugin.yml
```
- (5) 如果集群只有部分服务器支持 GPU 时，请为每个 GPU 服务器节点创建 yml 文件，通过 yml 文件内的 name 字段区分。
- a. 在任一个 Master 节点上，为各个 GPU 服务器节点创建 nvidia-device-plugin.yml 文件，文件名称不能重复。
  - b. 执行 **kubectl get node --show-labels** 命令查看集群 GPU 服务器节点的 nodeid（灰显信息）。

```
[root@localhost ~]# kubectl get node --show-labels
NAME      STATUS    ROLES           AGE     VERSION   LABELS
localhost  Ready     control-plane,master  47d    v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=localhost,kubernetes.io/os=linux,master=master1,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernetes.io/exclude-from-external-load-balancers=,node=node1,platform=plat,role=master
localhost3  Ready     control-plane,master  48d    v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=localhost3,kubernetes.io/os=linux,master=master2,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernetes.io/exclude-from-external-load-balancers=,node=node2,platform=plat,role=master
localhost5  Ready     control-plane,master  48d    v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=localhost5,kubernetes.io/os=linux,master=master3,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernetes.io/exclude-from-external-load-balancers=,node=node3,platform=plat,role=master
localhost24 Ready     <none>          4d22h   v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=localhost24,kubernetes.io/os=linux,node=node4,role=worker,worker=worker1
```

- c. 修改各个 yml 文件的 name 字段，修改后的文件示例如下（灰显信息为修改后的内容），确认无误后保存。

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nvidia-device-plugin-daemonset-node1 //各 GPU 服务器节点名称不要重复
  namespace: kube-system
spec:
  selector:
    matchLabels:
      name: nvidia-device-plugin-ds
  updateStrategy:
```

```

type: RollingUpdate
template:
  metadata:
    labels:
      name: nvidia-device-plugin-ds
spec:
  nodeSelector:
    node: node1 //GPU 服务器节点的 nodeid, 通过上一步骤可查询
  tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
  # Mark this pod as a critical add-on; when enabled, the critical add-on
  # scheduler reserves resources for critical add-on pods so that they can
  # be rescheduled after a failure.
  # See
  https://kubernetes.io/docs/tasks/administer-cluster/guaranteed-scheduling-critical-addon-pods/
  priorityClassName: "system-node-critical"
  containers:
  - image: nvcr.io/nvidia/k8s-device-plugin:v0.14.0
    name: nvidia-device-plugin-ctr
    env:
    - name: FAIL_ON_INIT_ERROR
      value: "false"
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: ["ALL"]
    volumeMounts:
    - name: device-plugin
      mountPath: /var/lib/kubelet/device-plugins
    volumes:
    - name: device-plugin
      hostPath:
        path: /var/lib/kubelet/device-plugins

```

- d. 在任意一个 Master 节点上执行 **kubectl apply -f** 命令部署各个 GPU 服务器节点的 yml 文件。例如：

```
[root@localhost ~]# kubectl apply -f nvidia-device-plugin-node2.yml
```

- (6) 执行 **kubectl get pod -n kube-system -o wide | grep plugin** 命令查询 Pod 运行状态，如果各 GPU 服务节点的 Pod 均为 running 状态，则表示 nvidia-device-plugin 插件安装成功。

```
[root@localhost ~]# kubectl get pod -n kube-system -o wide | grep plugin
nvidia-device-plugin-daemonset-9jrb8     1/1   Running   2          47d   177.177.173.14
localhost       <none>           <none>
```

### 8.3.3 部署 NVIDIA 设备插件（离线部署方式）

#### 1. 修改默认运行时

集群中用到 GPU 的节点都需要修改默认运行时。

- (1) 修改/etc/docker/daemon.json 配置文件，增加 default-runtime,runtimes 配置（灰显信息）。

```
{
    "default-runtime": "nvidia",
    "runtimes": {
        "nvidia": {
            "path": "/usr/bin/nvidia-container-runtime",
            "runtimeArgs": []
        }
    },
    "live-restore": true,
    "exec-opt": ["native.cgroupdriver=systemd"],
    "insecure-registries": ["matrix-registry.h3c.com:8088",
    "matrix-registry-master1.h3c.com:8088"],
    "bridge": "none",
    "log-driver": "json-file",
    "log-opt": {"max-size": "50m", "max-file": "3"}
}
```

- (2) 执行 **systemctl restart docker** 命令重启 docker 服务使配置生效。

```
[root@localhost ~]# systemctl restart docker
```

- (3) 执行 **docker info** 命令查看配置是否生效。如果能看到“Default Runtime: nvidia”，则表示配置生效。

```
[root@localhost ~]# docker info
Client:
  Context:    default
  Debug Mode: false
  Plugins:
    app: Docker App (Docker Inc., v0.9.1-beta3)
    buildx: Build with BuildKit (Docker Inc., v0.5.1-docker)
    scan: Docker Scan (Docker Inc., 0.8.0)

Server:
  Containers: 141
  Running: 121
  Paused: 0
  Stopped: 20
  Images: 91
  Server Version: 20.10.7
  Storage Driver: overlay2
    Backing Filesystem: extfs
    Supports d_type: true
    Native Overlay Diff: true
  userxattr: false
```

```

Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 1
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: io.containerd.runc.v2 io.containerd.runtime.v1.linux nvidia runc
Default Runtime: nvidia

```

## 2. 部署 nvidia-device-plugin

- (1) 通过 FTP 等方式将 k8s-device-plugin.tar 包上传至服务器的某个目录下。在该目录下执行命令 **docker load -i k8s-device-plugin.tar** 加载 k8s-device-plugin 镜像。

```
[root@localhost ~]# docker load -i k8s-device-plugin.tar
6021993d84a2: Loading layer 75.16MB/75.16MB
35fafba09344: Loading layer 18.85MB/18.85MB
22530106c24f: Loading layer 149.7MB/149.7MB
2f38cda98e92: Loading layer 3.072kB/3.072kB
0c90cb26ece1: Loading layer 17.92kB/17.92kB
75465ad0bfcd: Loading layer 554.5kB/554.5kB
e25701965449: Loading layer 18.94kB/18.94kB
98772958eeee: Loading layer 26.09MB/26.09MB
901527cbba58: Loading layer 15.35MB/15.35MB
a5fddbad8d3a: Loading layer 1.528MB/1.528MB
Loaded image: nvcr.io/nvidia/k8s-device-plugin:v0.14.0
```

- (2) 根据集群部署环境的实际情况选择合适的方式部署 nvidia-device-plugin.yml。

- 如果集群所有服务器都支持 GPU，按照步骤(3)部署。
- 如果集群只有部分服务器支持 GPU，按照步骤(4)部署。

- (3) 集群所有服务器都支持 GPU 时，请在任一个 Master 节点上执行 **kubectl apply -f** 命令部署 nvidia-device-plugin.yml 文件，其他节点无需重复部署。例如：

```
[root@localhost ~]# kubectl apply -f nvidia-device-plugin.yml
```

- (4) 如果集群只有部分服务器支持 GPU 时，请为每个 GPU 服务器节点创建 yml 文件，通过 yml 文件内的 name 字段区分。

- a. 在任一个 Master 节点上，为各个 GPU 服务器节点创建 nvidia-device-plugin.yml 文件，文件名称不能重复。
- b. 执行 **kubectl get node --show-labels** 命令查看集群 GPU 服务器节点的 nodeid（灰显信息）。

```
[root@localhost ~]# kubectl get node --show-labels
NAME      STATUS    ROLES           AGE     VERSION   LABELS
localhost   Ready    control-plane,master   47d    v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=localhost,kubernetes.io/os=linux,master=master1,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernetes.io/exclude-from-external-load-balancers=,node=node1,platform=plat,role=master
```

```

localhost3      Ready    control-plane,master   48d      v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd
64,kubernetes.io/hostname=localhost3,kubernetes.io/os=linux,master=master2,node-
role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernete
s.io/exclude-from-external-load-balancers=,node=node2,platform=plat,role=master
localhost5      Ready    control-plane,master   48d      v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd
64,kubernetes.io/hostname=localhost5,kubernetes.io/os=linux,master=master3,node-
role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.kubernete
s.io/exclude-from-external-load-balancers=,node=node3,platform=plat,role=master
localhost24     Ready    <none>                 4d22h    v1.21.14
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd
64,kubernetes.io/hostname=localhost24,kubernetes.io/os=linux,node=node4,role=wor
ker,worker=worker1

```

- c. 修改各个 yml 文件的 name 字段，修改后的文件示例如下（灰显信息为修改后的内容），确认无误后保存。

```

apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nvidia-device-plugin-daemonset-node1 //各 GPU 服务器节点名称不要重复
  namespace: kube-system
spec:
  selector:
    matchLabels:
      name: nvidia-device-plugin-ds
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        name: nvidia-device-plugin-ds
    spec:
      nodeSelector:
        node: node1 //GPU 服务器节点的 nodeid, 通过上一步骤可查询
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      # Mark this pod as a critical add-on; when enabled, the critical add-on
      # scheduler reserves resources for critical add-on pods so that they can
      # be rescheduled after a failure.
      # See
      https://kubernetes.io/docs/tasks/administer-cluster/guaranteed-scheduling-critic
al-addon-pods/
      priorityClassName: "system-node-critical"
      containers:
        - image: nvcr.io/nvidia/k8s-device-plugin:v0.14.0
          name: nvidia-device-plugin-ctr
      env:

```

```

        - name: FAIL_ON_INIT_ERROR
          value: "false"
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop: ["ALL"]
        volumeMounts:
        - name: device-plugin
          mountPath: /var/lib/kubelet/device-plugins
        volumes:
        - name: device-plugin
          hostPath:
            path: /var/lib/kubelet/device-plugins

```

- d. 在任意一个 Master 节点上执行 **kubectl apply -f** 命令部署各个 GPU 服务器节点的 yml 文件。例如：

```
[root@localhost ~]# kubectl apply -f nvidia-device-plugin-node2.yml
```

- (5) 执行 **kubectl get pod -n kube-system -o wide | grep plugin** 命令查询 Pod 运行状态，如果各 GPU 服务节点的 Pod 均为 running 状态，则表示 nvidia-device-plugin 插件安装成功。

```
[root@localhost ~]# kubectl get pod -n kube-system -o wide | grep plugin
```

### 3. 验证服务是否正确启动

执行 **kubectl get pod -n kube-system -o wide** 命令查询 Pod 运行状态。

```
[root@localhost ~]# kubectl get pod -n kube-system -o wide
NAME                               READY   STATUS    RESTARTS   AGE     IP
NODE      NOMINATED NODE      READINESS GATES
nvidia-device-plugin-daemonset-p7s76   1/1     Running   1          46h    177.177.66.141
beijing-ning25   <none>           <none>
nvidia-device-plugin-daemonset-vgnmt   1/1     Running   1          46h    177.177.236.16
beijing-ning23   <none>           <none>
nvidia-device-plugin-daemonset-w6nlj    1/1     Running   2          46h    177.177.230.82
beijing-ning22   <none>           <none>
```

#### 8.3.4 确认 GPU 识别正常

在支持 GPU 的服务器上执行 **kubectl describe node** 命令，查看 **nvidia.com/gpu** 字段可显示 GPU 数量，则表示 GPU 识别正常。

```
[root@localhost ~]# kubectl describe node localhost //localhost 为服务器主机名称
Name:           localhost
Roles:          control-plane,master
...
Capacity:
  cpu:           255
  ephemeral-storage: 3748905484Ki
  hugepages-1Gi:    0
  hugepages-2Mi:    0
  memory:         1055870908Ki
  nvidia.com/gpu:  8
```

```

pods: 300
rdma/hpc_shared_devices_ens11f1np1: 1k
rdma/hpc_shared_devices_ens12f1np1: 1k
rdma/hpc_shared_devices_ens13f1np1: 1k
rdma/hpc_shared_devices_ens14f1np1: 1k
rdma/hpc_shared_devices_ens15f1np1: 1k
rdma/hpc_shared_devices_ens16f1np1: 1k
rdma/hpc_shared_devices_ens17f1np1: 1k
rdma/hpc_shared_devices_ens18f1np1: 1k

Allocatable:
cpu: 240
ephemeral-storage: 3454991288335
hugepages-1Gi: 0
hugepages-2Mi: 0
memory: 1045282748Ki
nvidia.com/gpu: 8
pods: 300
rdma/hpc_shared_devices_ens11f1np1: 1k
rdma/hpc_shared_devices_ens12f1np1: 1k
rdma/hpc_shared_devices_ens13f1np1: 1k
rdma/hpc_shared_devices_ens14f1np1: 1k
rdma/hpc_shared_devices_ens15f1np1: 1k
rdma/hpc_shared_devices_ens16f1np1: 1k
rdma/hpc_shared_devices_ens17f1np1: 1k
rdma/hpc_shared_devices_ens18f1np1: 1k
...

```

## 8.4 安装nvidia-fabricmanager

GPU 形态分为模组形态和 PCIe 板卡形态，只有模组形态才需要安装 nvidia-fabricmanager。

通过 nvlink 桥或 PCIe 桥互连，如果 GPU 通过 nvlink 桥互连，则必须要安装 NVIDIA 网络设备管理器 nvidia-fabricmanager，否则 GPU 可能无法正常使用。

(1) 执行 `lspci | grep -i nvidia` 命令查询服务器上的 GPU 形态。

- 如果显示信息包含“Bridge”信息，表示 GPU 通过 nvlink 桥互连，即 GPU 为模组形态。  
需要安装 nvidia-fabricmanager，请执行下一步骤安装 nvidia-fabricmanager。

```
[root@localhost ~]# lspci | grep -i nvidia
1c:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
1d:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
1e:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
1f:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
20:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
21:00.0 Bridge: NVIDIA Corporation Device 1af1 (rev a1)
25:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
2b:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
63:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
68:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
9f:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
```

- ```
a4:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
e2:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)
e8:00.0 3D controller: NVIDIA Corporation GA100 [A100 SXM4 80GB] (rev a1)

o 如果显示信息不包含“Bridge”信息，表示 GPU 通过 PCIe 桥互连，即 GPU 为 PCIe 板卡形态，不需要安装 nvidia-fabricmanager。
```

- ```
[root@localhost ~]# lspci | grep -i nvidia
40:00.0 3D controller: NVIDIA Corporation GA100 [A100 PCIe 80GB] (rev a1)
49:00.0 3D controller: NVIDIA Corporation GA100 [A100 PCIe 80GB] (rev a1)
4a:00.0 3D controller: NVIDIA Corporation GA100 [A100 PCIe 80GB] (rev a1)
59:00.0 3D controller: NVIDIA Corporation GA100 [A100 PCIe 80GB] (rev a1)

(2) 登录网址 https://developer.download.nvidia.cn/compute/cuda/repos/rhel7/x86\_64/，下载 nvidia-fabricmanager 或者向技术支持获取安装包。
```



请确保下载的 nvidia-fabricmanager 版本与表 8-1 所示的显卡驱动版本一致。

图8-4 nvidia-fabricmanager 下载页面参考图

|                                                               |       |                  |
|---------------------------------------------------------------|-------|------------------|
| <a href="#">nvidia-fabric-manager-510.39.01-1.x86_64.rpm</a>  | 1.7MB | 2022-01-11 06:10 |
| <a href="#">nvidia-fabric-manager-510.47.03-1.x86_64.rpm</a>  | 1.7MB | 2022-01-25 16:13 |
| <a href="#">nvidia-fabric-manager-510.73.08-1.x86_64.rpm</a>  | 1.7MB | 2022-05-19 15:00 |
| <a href="#">nvidia-fabric-manager-510.84-1.x86_64.rpm</a>     | 1.7MB | 2022-07-07 12:48 |
| <a href="#">nvidia-fabric-manager-510.85.02-1.x86_64.rpm</a>  | 1.7MB | 2022-07-26 19:22 |
| <a href="#">nvidia-fabric-manager-510.108.03-1.x86_64.rpm</a> | 2.2MB | 2022-11-19 02:27 |
| <a href="#">nvidia-fabric-manager-515.43.04-1.x86_64.rpm</a>  | 1.7MB | 2022-05-04 15:43 |
| <a href="#">nvidia-fabric-manager-515.48.07-1.x86_64.rpm</a>  | 1.7MB | 2022-06-02 22:18 |
| <a href="#">nvidia-fabric-manager-515.65.01-1.x86_64.rpm</a>  | 1.7MB | 2022-07-29 18:50 |
| <a href="#">nvidia-fabric-manager-515.65.07-1.x86_64.rpm</a>  | 2.2MB | 2022-09-26 18:46 |
| <a href="#">nvidia-fabric-manager-515.86.01-1.x86_64.rpm</a>  | 2.2MB | 2022-10-26 22:34 |
| <a href="#">nvidia-fabric-manager-515.105.01-1.x86_64.rpm</a> | 2.2MB | 2023-02-27 18:10 |
| <a href="#">nvidia-fabric-manager-520.61.05-1.x86_64.rpm</a>  | 2.3MB | 2022-09-29 18:24 |
| <a href="#">nvidia-fabric-manager-525.60.13-1.x86_64.rpm</a>  | 2.3MB | 2022-11-30 21:57 |
| <a href="#">nvidia-fabric-manager-525.85.12-1.x86_64.rpm</a>  | 2.3MB | 2023-01-28 17:37 |
| <a href="#">nvidia-fabric-manager-525.105.17-1.x86_64.rpm</a> | 2.3MB | 2023-03-29 08:57 |
| <a href="#">nvidia-fabric-manager-525.125.06-1.x86_64.rpm</a> | 2.3MB | 2023-06-02 03:00 |
| <a href="#">nvidia-fabric-manager-525.147.05-1.x86_64.rpm</a> | 2.3MB | 2023-10-26 06:14 |
| <a href="#">nvidia-fabric-manager-530.30.02-1.x86_64.rpm</a>  | 2.3MB | 2023-02-23 05:24 |
| <a href="#">nvidia-fabric-manager-535.54.03-1.x86_64.rpm</a>  | 2.5MB | 2023-06-13 00:03 |
| <a href="#">nvidia-fabric-manager-535.86.10-1.x86_64.rpm</a>  | 2.5MB | 2023-07-27 06:46 |
| <a href="#">nvidia-fabric-manager-535.104.05-1.x86_64.rpm</a> | 2.5MB | 2023-08-21 01:24 |
| <a href="#">nvidia-fabric-manager-535.104.12-1.x86_64.rpm</a> | 2.5MB | 2023-09-22 05:42 |

- (3) 执行 rpm -ivh 命令安装 nvidia-fabricmanager。

```
[root@localhost ~]# rpm -ivh nvidia-fabric-manager-515.105.01-1.x86_64.rpm
警告:nvidia-fabric-manager-515.105.01-1.x86_64.rpm:头 V4 RSA/SHA512 signature, 密钥 ID d420045:NOKE
Verifying... ######[100%]
准备中... ######[100%]
正在升级/安装... ######[100%]
1:nvidia-fabric-manager-515.105.01#####[100%]
```

- (4) 执行如下命令启动 nvidia-fabricmanager 服务，并且设置为开机自动启动。

```
[root@localhost ~]# systemctl start nvidia-fabricmanager
[root@localhost ~]# systemctl enable nvidia-fabricmanager
```

- (5) 执行 `systemctl status nvidia-fabricmanager` 命令查询 nvidia-fabricmanager 服务状态，如果 Active 字段显示为“running”表示服务启动正常。

```
[root@localhost ~]# systemctl status nvidia-fabricmanager
● nvidia-fabricmanager.service - NVIDIA fabric manager service
   Loaded: loaded (/usr/lib/systemd/system/nvidia-fabricmanager.service; disabled;
   vendor preset: disabled)
     Active: active (running) since Wed 2023-10-18 02:38:11 CST; 3 weeks 5 days ago
       Main PID: 3850798 (nv-fabricmanager)
         Tasks: 18 (limit: 3355442)
        Memory: 15.4M
      CGroup: /system.slice/nvidia-fabricmanager.service
              └─ 3850798 /usr/bin/nv-fabricmanager -c
                /usr/share/nvidia/nvswitch/fabricmanager.cfg

Notice: journal has been rotated since unit was started, output may be incomplete.
```

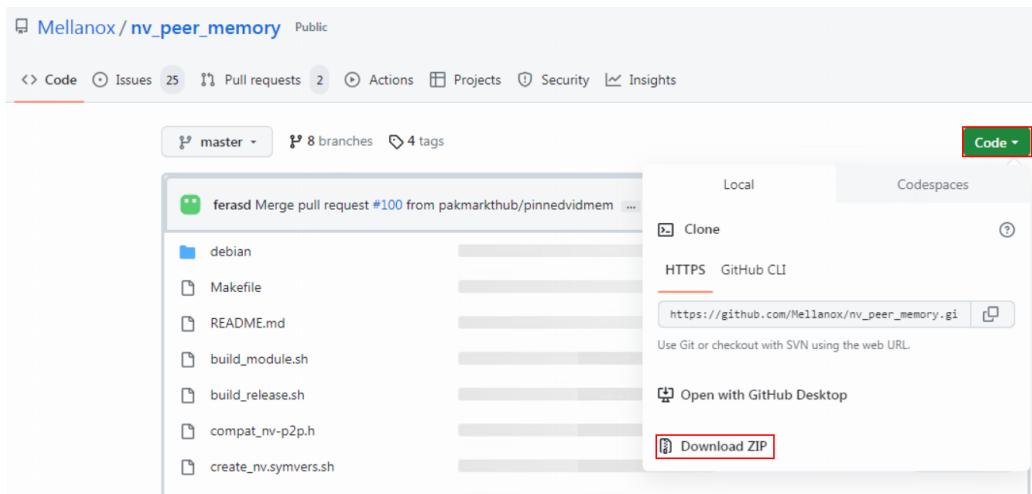
## 8.5 安装nv\_peer\_memory

仅分布式推理网络需要安装 nv\_peer\_memory。

### 8.5.1 下载 nv\_peer\_memory

登录 [https://github.com/Mellanox/nv\\_peer\\_memory](https://github.com/Mellanox/nv_peer_memory) 网站。在导航栏 Code 的页签，单击<Code>按钮后单击<Download ZIP>下载 nv\_peer\_memory 安装包。如图 8-5 所示。

图8-5 下载 nv\_peer\_memory



### 8.5.2 安装 nv\_peer\_memory

- (1) 配置本地 YUM 源，详细步骤请参见“[5.4 配置本地 YUM 源](#)”。

- (2) 执行 `yum` 命令安装依赖包。

```
[root@localhost ~]# yum install rpm-build
```

- (3) 将 nv\_peer\_memory 安装包通过 FTP 等工具上传到服务器（例如/root 目录）。

- (4) 进入目录并执行 **unzip** 命令解压缩安装包。

```
[root@localhost ~]# unzip nv_peer_memory-master.zip
```

- (5) 执行 **cd** 命令进入安装包解压缩后的目录并执行命令安装 **nv\_peer\_memory**。

```
[root@localhost ~]# cd nv_peer_memory-master/  
[root@localhost nv_peer_memory-master]# ./build_module.sh  
[root@localhost nv_peer_memory-master]# rpmbuild --rebuild  
/tmp/nvidia_peer_memory-1.3-0.src.rpm  
[root@localhost nv_peer_memory-master]# rpm -ivh  
/root/rpmbuild/RPMS/x86_64/nvidia_peer_memory-1.3-0.x86_64.rpm  
(6) 执行命令 modprobe nvidia-peermem 启动 nv_peer_memory 服务。  
[root@localhost ~]# modprobe nvidia-peermem  
[root@localhost ~]# lsmod | grep peer  
nvidia_peermem           16384  0  
nvidia                  54063104  1187 nvidia_uvm,nvidia_peermem,nvidia_modeset  
ib_core                 499712   9  
rdma_cm,ib_ipoib,nvidia_peermem,iw_cm,ib_umad,rdma_ucm,ib_uverbs,mlx5_ib,ib_cm
```

## 8.6 安装昆仑芯驱动

### 8.6.1 驱动版本信息

请按需向技术支持获取对应版本的昆仑芯驱动组件安装包，详细版本信息请参见[表 8-2](#)。

表8-2 昆仑芯驱动版本信息

| 产品类型 | 产品家族（显卡型号） | 驱动版本 | 运行时版本 | 固件版本           |
|------|------------|------|-------|----------------|
| 昆仑芯  | RG800      | 4.31 | 4.31  | 0002.0018.0065 |
| 昆仑芯  | R300       | 4.31 | 4.31  | 0002.0018.0065 |

### 8.6.2 查看 GPU 状态

在服务器上执行 **lspci -d 1d22: -nv** 命令查看显卡是否被正常识别。

```
[root@localhost ~]# lspci -d 1d22: -nv  
34:00.0 0780: 1d22:3684 (rev 02)  
    Subsystem: 0002:0007  
Physical Slot: 10  
Flags: bus master, fast devsel, latency 0, IRQ 224, NUMA node 0  
Memory at 25ffd800c000 (64-bit, prefetchable) [size=16K]  
Memory at 25ffd4000000 (64-bit, prefetchable) [size=64M]  
Memory at 25ffd0000000 (64-bit, prefetchable) [size=64M]  
Capabilities: [80] Express Endpoint, MSI 00  
Capabilities: [d0] MSI-X: Enable- Count=64 Masked-  
Capabilities: [e0] MSI: Enable+ Count=1/32 Maskable+ 64bit+  
Capabilities: [f8] Power Management version 3  
Capabilities: [40] Null  
Capabilities: [100] Vendor Specific Information: ID=1556 Rev=1 Len=008 <?>
```

```
Capabilities: [128] Alternative Routing-ID Interpretation (ARI)
Capabilities: [140] Single Root I/O Virtualization (SR-IOV)
Capabilities: [1e0] Data Link Feature <?>
Capabilities: [200] Advanced Error Reporting
Capabilities: [300] Secondary PCI Express
Capabilities: [340] Physical Layer 16.0 GT/s <?>
Capabilities: [378] Lane Margining at the Receiver <?>
Kernel driver in use: kunlun
Kernel modules: kunlun
```

### 8.6.3 安装显卡驱动

- (1) 配置本地 YUM 源，依次执行如下命令安装相关依赖包。

```
[root@localhost ~]# yum install gcc g++
[root@localhost ~]# yum install kernel-devel
[root@localhost ~]# yum install make
```

- (2) 将显卡驱动安装包 **xre-centos8\_x86\_64.tar** 通过 FTP 工具上传至服务器的待安装目录（例如 **/root**）下。
- (3) 执行 **tar -zvxf xre-centos8\_x86\_64.tar.gz** 命令解压缩安装包，并进入解压缩后的目录。

```
[root@localhost ~]# tar -zvxf xre-centos8_x86_64.tar.gz
xre-centos8_x86_64/
xre-centos8_x86_64/doc/
xre-centos8_x86_64/doc/images/
xre-centos8_x86_64/doc/images/xre_faq_0001.png
xre-centos8_x86_64/doc/images/xre_install_0003.png
xre-centos8_x86_64/doc/images/xre_install_0002.png
xre-centos8_x86_64/doc/images/xre_overview_0003.png
xre-centos8_x86_64/doc/images/xre_faq_0002.png
xre-centos8_x86_64/doc/images/xre_overview_0001.png
xre-centos8_x86_64/doc/images/xre_overview_0002.png
xre-centos8_x86_64/doc/images/xre_install_0001.png
xre-centos8_x86_64/doc/topics/
...
[root@localhost ~]# cd xre-centos8_x86_64/
[root@localhost ~]# ll
总用量 100
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 bin
-rw-r--r-- 1 1001 1002 68 10月 30 19:20 Changelog
drwxr-xr-x 4 1001 1002 4096 10月 31 16:41 doc
lrwxrwxrwx 1 1001 1002 13 10月 30 19:20 driver -> kunlun_module
drwxr-xr-x 4 1001 1002 4096 10月 30 19:20 firmware
drwxr-xr-x 3 1001 1002 4096 10月 30 19:20 include
-rwxr-xr-x 1 1001 1002 13212 10月 30 19:20 install_rt.sh
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 kunlun_module
drwxr-xr-x 4 1001 1002 4096 10月 30 19:20 kunlun-module-dkms-4.31.0
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 lib
```

```

-rw-r--r-- 1 1001 1002 8878 10月 30 19:20 LICENSE.CHN.TXT
-rw-r--r-- 1 1001 1002 10605 10月 30 19:20 LICENSE.ENG.TXT
-rw-r--r-- 1 1001 1002 3372 10月 30 19:20 ReleaseNotes.md
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 script
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 so
drwxr-xr-x 4 1001 1002 4096 10月 30 19:20 tools
drwxr-xr-x 2 1001 1002 4096 10月 30 19:20 udev_rules
-rw-r--r-x 1 1001 1002 1716 10月 30 19:20 uninstall_rt.sh
-rw-r--r-- 1 1001 1002 104 10月 30 19:20 version.txt

```

- (4) 执行`./install_rt.sh`命令安装显卡驱动。

```

[root@localhost ~]# xre-centos8_x86_64]# ./install_rt.sh
udev rule installed.

DKMS is not exist, will try to install module directly
/home/xwj/xre-centos8_x86_64/kunlun-module-dkms-4.31.0 /home/xwj/xre-centos8_x86_64
make[1]: 进入目录
“/home/xwj/xre-centos8_x86_64/kunlun-module-dkms-4.31.0/kunlun_module”
make[2]: 进入目录 “/usr/src/kernels/5.10.0-136.12.0.86.4.nos1.x86_64”
...

```

- (5) 安装完成后执行`xpu_smi`命令，若显示信息如下说明显卡驱动安装成功。

```

[root@localhost ~]# xpu_smi
Runtime Version: 4.31
Driver Version: 4.31
DEVICES
-----
| DevID | PCI Addr | Model | SN | INODE | State | UseRate | L3
|       Memory | Power(W) | Temp | Freq(MHz) | Firmware Version | CPLD Version |
-----
|   0 | 0000:34:00.0 | RG800 | 02K06J622CV000C2 | /dev/xpu0 | N | 0 % | 13
/ 63 MB | 29860 / 32768 MB | 71 | 74 | 1300 | 0002.0018.0065 | 209
|
|   1 | 0000:35:00.0 | RG800 | 02K06J622CV000JS | /dev/xpu1 | N | 0 % | 13
/ 63 MB | 29860 / 32768 MB | 69 | 73 | 1300 | 0002.0018.0065 | 209
|
|   2 | 0000:38:00.0 | RG800 | 02K06J622CV00024 | /dev/xpu2 | N | 0 % | 0
/ 63 MB | 0 / 32768 MB | 68 | 67 | 1300 | 0002.0018.0065 | 209
|
|   3 | 0000:3d:00.0 | RG800 | 02K06J622CV0005H | /dev/xpu3 | N | 0 % | 0
/ 63 MB | 0 / 32768 MB | 73 | 72 | 1300 | 0002.0012.0065 | 209
|
|   4 | 0000:3e:00.0 | RG800 | 02K06J622CV00058 | /dev/xpu4 | N | 0 % | 0
/ 63 MB | 0 / 32768 MB | 72 | 72 | 1300 | 0002.0018.0065 | 209
|
|   5 | 0000:41:00.0 | RG800 | 02K06J622CV000CW | /dev/xpu5 | N | 0 % | 0
/ 63 MB | 0 / 32768 MB | 73 | 75 | 1300 | 0002.0018.0065 | 209
|

```

```

|      6 | 0000:45:00.0 | RG800 | 02K06J622CV0000N | /dev/xpu6 |     N |      0 % |  0
/ 63 MB |      0 / 32768 MB |       70 |    67 |      1300 | 0002.0018.0065 |      209
|
|      7 | 0000:4d:00.0 | RG800 | 02K06J622CV00066 | /dev/xpu7 |     N |      0 % |  0
/ 63 MB |      0 / 32768 MB |       72 |    71 |      1300 | 0002.0018.0065 |      209
|
-----  

-----  

VIDEO  

-----  

| DevID | Model |          DEC          |          ENC          |          IMGPROC         |
-----  

0	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
1	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
2	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
3	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
4	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
5	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
6	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
7	RG800	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz	0 %, 0 fps, 800 MHz
-----  

PROCESSES  

-----  

| DevID | PID | Streams | L3 | Memory | Command |
-----  

-----
```

## 8.7 安装天数智铠MR-V100驱动

### 8.7.1 驱动版本信息

请按需向技术支持获取对应版本的天数智铠驱动组件安装包，详细版本信息请参见[表 8-3](#)。

**表8-3 天数智铠显卡驱动版本信息**

| 产品类型 | 产品家族（显卡型号）       | IX-ML 版本 | 驱动版本  | CUDA 版本   |
|------|------------------|----------|-------|-----------|
| 天数智铠 | Iluvatar MR-V100 | 4.1.0.w  | 4.1.1 | CUDA 10.2 |

### 8.7.2 安装显卡驱动

(1) 配置本地 YUM 源，依次执行如下命令安装相关依赖包。

```
[root@localhost ~]# yum install -y elfutils-libelf-devel
[root@localhost ~]# yum install kernel-devel
[root@localhost ~]# yum install make
[root@localhost ~]# yum install -y python3-pip
[root@localhost ~]# yum install -y kmod
[root@localhost ~]# yum install -y make
```

```
[root@localhost ~]# yum install -y ncurses-libs
```

- (2) 通过 FTP 工具上传 **partial\_install\_cuda\_header.tar**，并执行如下命令解压缩。

```
[root@localhost ~]# tar -zvxf partial_install_cuda_header.tar.gz
partial_install_cuda_header/
partial_install_cuda_header/README.MD
partial_install_cuda_header/.git/
partial_install_cuda_header/.git/description
partial_install_cuda_header/.git/index
partial_install_cuda_header/.git/branches/
partial_install_cuda_header/.git/objects/
partial_install_cuda_header/.git/objects/pack/
partial_install_cuda_header/.git/objects/pack/pack-41ad15857b08769635c68468207e266f
951706e1.pack
partial_install_cuda_header/.git/objects/pack/pack-41ad15857b08769635c68468207e266f
951706e1.idx
partial_install_cuda_header/.git/objects/info/
partial_install_cuda_header/.git/hooks/
...
```

- (3) 进入解压目录，执行 **sh install\_cuda\_header.sh** 命令，完成部署 CUDA 工具。

```
[root@localhost ~]# cd partial_install_cuda_header/
[[root@localhost partial_install_cuda_header]# ll
总用量 12
drwxr-xr-x 5 root root 4096 12月 28 2022 cuda
-rw-rxr-xr-x 1 root root 901 12月 28 2022 install-cuda-header.sh
-rw-r--r-- 1 root root 1084 12月 28 2022 README.MD
[root@localhost partial_install_cuda_header]# sh install_cuda-header.sh
```

- (4) 创建并编辑/etc/yum.conf 配置文件，在文件中最后一行添加“**exclude=kernel\***”，禁用系统自动更新。

- a. 执行 **vi** 命令创建文件。

```
[root@localhost ~]# vi /etc/yum.conf
```

- b. 通过键盘输入字符“i”进入编辑界面，输入如下内容：

```
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=False
multilib_policy=all
exclude=kernel*
```

- c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

- (5) 安装软件栈（Docker 安装方式）。

- a. 通过 FTP 工具上传文件：corex-installer-linux64-4.0.0\_x86\_64\_10.2.run。

- b. 进入文件目录，执行 **bash** 命令部署天数智铠驱动。

```
[root@localhost ~]# bash corex-installer-linux64-4.0.0_x86_64_10.2.run
```

...

如果您同意遵守本协议的条件条款，请按“接受”。如果您不同意遵守本协议的条件条款，请按“拒绝”，并且不得使用本软件。

Do you accept the EULA?

accept/decline/quit: \_

c. 输入“accept”完成确认。

Do you accept the EULA?

accept/decline/quit: accept

(6) 安装 Core X: 进入安装界面后，按键盘键勾选“Install driver”和“Toolkit”，按<  
↑><↓>键移动选择选择“Install”，最后按<Enter>键确认操作。

a. 执行部署

```
Corex v4.0.0 Docker Instal  
[ * ] Install driver  
[ * ] Toolkit  
Options Show corex installer options
```

Install

b. 等待部署完成，显示部署

```
Verifying archive integrity... 100% All good.  
Uncompressing Corex Installer 100%
```

Start to install the Corex Driver.

Start to install the Corex Toolkit at /usr/local/corex-4.0.0/...

```
=====  
= Summary =  
=====
```

```
Driver: Installed  
For the Corex Driver uninstallation, please run command:  
    sudo /usr/local/corex-4.0.0/bin/corex-driver-uninstaller  
Logfile is /var/log/iluvatarcorex/driver_installer.log
```

Toolkit: Installed at location '/usr/local/corex-4.0.0/'

```
Please make sure that  
- PATH includes /usr/local/corex-4.0.0/bin  
- LD_LIBRARY_PATH includes /usr/local/corex-4.0.0/lib
```

```
For the Corex Toolkit uninstallation, please run command:  
    sudo /usr/local/corex-4.0.0/bin/corex-uninstaller  
Logfile is /var/log/iluvatarcorex/corex_installer.log
```

(7) 执行 **ll /usr/local** 命令查看 Core X 的版本号。

```
[root@localhost ~]# ll /usr/local  
总用量 52  
drwxr-xr-x. 2 root root 4096 3月 20 14:31 bin
```

```

lrwxrwxrwx 1 root root 23 7月 22 10:35 corex -> /usr/local/corex-4.1.0/
drwxr-xr-x 9 root root 4096 5月 23 22:14 corex-4.1.0
lrwxrwxrwx 1 root root 20 2月 4 08:30 cuda -> /usr/local/cuda-10.2
drwxrwxrwx 5 root root 4096 6月 24 2022 cuda-10.2
drwxr-xr-x. 2 root root 4096 8月 8 2022 etc
drwxr-xr-x. 2 root root 4096 8月 8 2022 games
drwxr-xr-x. 5 root root 4096 1月 22 16:43 h3diag
drwxr-xr-x. 2 root root 4096 8月 8 2022 include
drwxr-xr-x. 3 root root 4096 3月 20 14:30 lib
drwxr-xr-x. 4 root root 4096 3月 20 14:30 lib64
drwxr-xr-x. 2 root root 4096 8月 8 2022 libexec
drwxr-xr-x. 2 root root 4096 8月 8 2022 sbin
drwxr-xr-x. 6 root root 4096 5月 8 19:57 share
drwxr-xr-x. 2 root root 4096 8月 8 2022 src

```

(8) 增加环境变量:

- a. 执行 **vi ~/.bashrc** 命令进入**.bashrc**文件，根据查到的版本号“corex-4.0.0”在最后一行增加灰显内容:

```

# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

export OS_TYPE=1
export PATH=$PATH:/usr/local/corex-4.1.0/bin
export LD_LIBRARY_PATH=/usr/local/corex-4.1.0/lib
export KUBECONFIG=/etc/kubernetes/admin.conf

```

- b. 执行 **source ~/.bashrc** 命令，完成环境变量配置。

```
[root@localhost ~]# source ~/.bashrc
```

(9) 安装完成后执行 **ixsmi** 命令，若有如下显示信息则表示说显卡驱动安装成功。

```

[root@localhost ~]# ixsmi
Timestamp      Wed Mar 27 08:02:20 2024
+-----+
| IX-ML: 4.1.0.w      Driver Version: 4.1.0          CUDA Version: 10.2 |
| -----+-----+-----+-----+-----+-----+-----+-----+-----+-----|
| GPU   Name           | Bus-Id           | Clock-SM   Clock-Mem |
| Fan   Temp   Perf   Pwr:Usage/Cap| Memory-Usage | GPU-Util   Compute M. |
| -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----|
| 0     Iluvatar MR-V100       | 00000000:1B:00.0 | 1500MHz   1600MHz |
| 0%   34C   P0     34W / 150W | 25486MiB / 32768MiB | 0%        Default   |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1     Iluvatar MR-V100       | 00000000:1C:00.0 | 1500MHz   1600MHz |

```

|                                             |          |         |              |                     |             |         |  |
|---------------------------------------------|----------|---------|--------------|---------------------|-------------|---------|--|
| 0%                                          | 34C      | P0      | 35W / 150W   | 25486MiB / 32768MiB | 0%          | Default |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 2                                           | Iluvatar | MR-V100 |              | 00000000:1F:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 3                                           | Iluvatar | MR-V100 |              | 00000000:23:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 4                                           | Iluvatar | MR-V100 |              | 00000000:35:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 5                                           | Iluvatar | MR-V100 |              | 00000000:36:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 6                                           | Iluvatar | MR-V100 |              | 00000000:39:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| 7                                           | Iluvatar | MR-V100 |              | 00000000:3D:00.0    | 1500MHz     | 1600MHz |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| +-----+-----+-----+-----+-----+-----+-----+ |          |         |              |                     |             |         |  |
| Processes:                                  |          |         |              |                     | GPU Memory  |         |  |
| GPU                                         | PID      |         | Process name |                     | Usage (MiB) |         |  |

- (10) 请联系技术支持获取 `set_dvfs.sh` 脚本，并执行 `bash set_dvfs.sh 0` 命令关闭 DVFS 功能。DVFS 是天数智铠 MR-V100 的动态调节频率的功能，建议关闭。

```
[root@localhost ~]# bash set_dvfs.sh 0
echo perf_mode 0 > /sys/bus/pci/devices/0000:1b:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:1c:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:1f:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:23:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:35:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:36:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:39:00.0/itr_debug
Turn off DVFS mode: Success
echo perf_mode 0 > /sys/bus/pci/devices/0000:3d:00.0/itr_debug
Turn off DVFS mode: Success
```

## 8.8 安装天数天垓BI-V150驱动

### 8.8.1 驱动版本信息

请按需向技术支持获取对应版本的天数天垓驱动组件安装包，详细版本信息请参见[表 8-4](#)。

表8-4 天数天垓显卡驱动版本对应关系表

| 产品类型 | 产品家族（含显卡型号）      | IX-ML 版本 | 驱动版本  | CUDA 版本   |
|------|------------------|----------|-------|-----------|
| 天数天垓 | Iluvatar BI-V150 | 4.1.0.w  | 4.1.1 | CUDA 10.2 |

### 8.8.2 安装显卡驱动

- (1) 配置本地 YUM 源，分别执行如下命令安装相关依赖包。

```
[root@localhost ~]# yum install -y elfutils-libelf-devel  
[root@localhost ~]# yum install kernel-devel  
[root@localhost ~]# yum install make  
[root@localhost ~]# yum install -y python3-pip  
[root@localhost ~]# yum install -y kmod  
[root@localhost ~]# yum install -y make  
[root@localhost ~]# yum install -y ncurses-libs
```

- (2) 通过 FTP 工具上传 `partial_install_cuda_header.tar`，并执行如下命令解压缩。

```
[root@localhost ~]# tar -zvxf partial_install_cuda_header.tar.gz  
partial_install_cuda_header/  
partial_install_cuda_header/README.MD  
partial_install_cuda_header/.git/  
partial_install_cuda_header/.git/description  
partial_install_cuda_header/.git/index  
partial_install_cuda_header/.git/branches/  
partial_install_cuda_header/.git/objects/  
partial_install_cuda_header/.git/objects/pack/  
partial_install_cuda_header/.git/objects/pack/pack-41ad15857b08769635c68468207e266f  
951706e1.pack  
partial_install_cuda_header/.git/objects/pack/pack-41ad15857b08769635c68468207e266f  
951706e1.idx  
partial_install_cuda_header/.git/objects/info/  
partial_install_cuda_header/.git/hooks/  
...
```

- (3) 进入解压目录，执行 `sh install_cuda_header.sh` 命令，以完成部署 CUDA 工具。

```
[root@localhost ~]# cd partial_install_cuda_header/  
[[root@localhost partial_install_cuda_header]# ll  
总用量 12  
drwxr-xr-x 5 root root 4096 12月 28 2022 cuda  
-rwxr-xr-x 1 root root 901 12月 28 2022 install-cuda-header.sh  
-rw-r--r-- 1 root root 1084 12月 28 2022 README.MD  
[root@localhost partial_install_cuda_header]# sh install_cuda_header.sh
```

(4) 创建并编辑/etc/yum.conf 配置文件，在文件中最后一行添加“exclude=kernel\*”，禁用系统自动更新。

a. 执行 **vi** 命令创建文件。

```
[root@localhost ~]# vi /etc/yum.conf
```

b. 通过键盘输入字符“i”进入编辑界面，输入如下内容：

```
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=False
multilib_policy=all
exclude=kernel*
```

c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

(5) 安装软件栈（Docker 安装方式）

a. 通过 FTP 工具上传 corex-installer-linux64-3.4.0.20240419.75\_x86\_64\_10.2.run 文件。

b. 进入文件目录，执行 **bash** 命令部署天数天垓驱动。

```
[root@localhost ~]# bash corex-installer-linux64-3.4.0.20240419.75_x86_64_10.2.run
Verifying archive integrity... 100% All good.
Uncompressing Corex Installer 100%
...
如果您同意遵守本协议的条件条款，请按"接受"。如果您不同意遵守本协议的条件条款，请按"拒绝"，并且不得使用本软件。
```

```
Do you accept the EUL A?
accept/decline/quit:
```

c. 输入“accept”完成确认。

```
Do you accept the EUL A?
accept/decline/quit: accept
```

(6) 安装 Core X

a. 进入安装界面后，按键盘<Space>键勾选“Install driver”和“Toolkit”，按<↑><↓>键移动选择选择“Install”，最后按<Enter>键确认操作。

```
Corex v3.4.0 Docker Instal
[ * ] Driver
[ * ] Toolkit
Options Show Corex installer options
Install
```

b. 等待部署完成，显示部署成功。

```
Verifying archive integrity... 100% All good.
Uncompressing Corex Installer 100%
```

```
Start to install the Corex Driver.
```

```
Start to install the Corex Toolkit at /usr/local/corex-3.4.0/...
```

```

=====
= Summary =
=====

Driver:           Installed
For the Corex Driver uninstallation, please run command:
    sudo /usr/local/corex-3.4.0/bin/corex-driver-uninstaller
Logfile is /var/log/iluvatarcorex/driver_installer.log

Toolkit:          Installed at location '/usr/local/corex-3.4.0/'

Please make sure that
- PATH includes /usr/local/corex-3.4.0/bin
- LD_LIBRARY_PATH includes /usr/local/corex-3.4.0/lib

For the Corex Toolkit uninstallation, please run command:
    sudo /usr/local/corex-3.4.0/bin/corex-uninstaller
Logfile is /var/log/iluvatarcorex/corex_installer.log
```

## (7) 设置环境变量

- 执行命令 **ll /usr/local**，确定 Core X 的版本号。

```
[root@localhost ~]# ll /usr/local
总用量 52
drwxr-xr-x. 2 root root 4096 5月 30 17:25 bin
lrwxrwxrwx  1 root root   23 7月  1 17:54 corex -> /usr/local/corex-4.1.0/
drwxr-xr-x  9 root root 4096 7月  1 17:54 corex-4.1.0
lrwxrwxrwx. 1 root root   20 5月 22 11:27 cuda -> /usr/local/cuda-10.2
drwxrwxrwx. 5 root root 4096 4月   3 2023 cuda-10.2
drwxr-xr-x. 2 root root 4096 8月   8 2022 etc
drwxr-xr-x. 2 root root 4096 8月   8 2022 games
drwxr-xr-x. 5 root root 4096 5月 21 16:57 h3diag
drwxr-xr-x. 2 root root 4096 8月   8 2022 include
drwxr-xr-x. 3 root root 4096 5月 30 17:25 lib
drwxr-xr-x. 4 root root 4096 5月 30 17:25 lib64
drwxr-xr-x. 2 root root 4096 8月   8 2022 libexec
drwxr-xr-x. 2 root root 4096 8月   8 2022 sbin
drwxr-xr-x. 6 root root 4096 5月 22 11:32 share
drwxr-xr-x. 2 root root 4096 8月   8 2022 src
```

- 执行 **vi ~/.bashrc** 命令进入**.bashrc**文件，根据查到的版本号“corex-3.4.0”，在最后增加灰显内容：

```
# .bashrc
# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

```

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
export OS_TYPE=1
export PATH=$PATH:/usr/local/corex-4.1.0/bin
export LD_LIBRARY_PATH=/usr/local/corex-4.1.0/lib
export KUBECONFIG=/etc/kubernetes/admin.conf

```

c. 执行 **source ~/.bashrc** 命令，完成环境变量配置。

```
[root@localhost ~]# source ~/.bashrc
```

(8) 安装完成后执行 **ixsmi** 命令，若有如下显示信息则表示显卡驱动安装成功。

```

[root@localhost ~]# ixsmi
Timestamp      Wed Mar 27 08:02:20 2024
+-----+
| IX-ML: 4.1.0.w      Driver Version: 4.1.1          CUDA Version: 10.2   |
|-----+-----+-----+
| GPU  Name           | Bus-Id          | Clock-SM  Clock-Mem  |
| Fan   Temp   Perf  Pwr:Usage/Cap| Memory-Usage   | GPU-Util  Compute M.  |
|-----+-----+-----+-----+-----+-----+-----+
| 0    Iluvatar BI-V150     | 00000000:1B:00.0 | 1500MHz  1600MHz  |
| 0%   34C   P0    34W / 150W | 25486MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 1    Iluvatar BI-V150     | 00000000:1C:00.0 | 1500MHz  1600MHz  |
| 0%   34C   P0    35W / 150W | 25486MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 2    Iluvatar BI-V100     | 00000000:1F:00.0 | 1500MHz  1600MHz  |
| 0%   33C   P0    34W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 3    Iluvatar BI-V150     | 00000000:23:00.0 | 1500MHz  1600MHz  |
| 0%   35C   P0    35W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 4    Iluvatar BI-V150     | 00000000:35:00.0 | 1500MHz  1600MHz  |
| 0%   35C   P0    34W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 5    Iluvatar BI-V150     | 00000000:36:00.0 | 1500MHz  1600MHz  |
| 0%   36C   P0    34W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 6    Iluvatar BI-V150     | 00000000:39:00.0 | 1500MHz  1600MHz  |
| 0%   34C   P0    34W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
| 7    Iluvatar BI-V150     | 00000000:3D:00.0 | 1500MHz  1600MHz  |
| 0%   34C   P0    34W / 150W | 114MiB / 32768MiB | 0%       Default   |
+-----+-----+-----+
+-----+
| Processes:                                     GPU Memory  |
|   GPU        PID      Process name           Usage (MiB)  |

```

## 8.9 配置NFS存储服务器

3机集群部署模式和3+N机集群部署模式下，仅支持NFS文件存储方式。在部署基础安装包之前，必须配置NFS存储服务器。单机部署模式下，仅支持本地磁盘（local）文件存储方式，不涉及本配置。



说明

- 操作系统版本不同，NFS存储服务器配置步骤有所不同。本节仅以操作系统为centos7.9的服务器为例。
- NFS存储空间大小推荐配置为1T，“/linseerFileShare”磁盘空间分配详情见[表4-2](#)。

### 8.9.1 配置NFS存储服务端

- (1) 请确认作为NFS存储服务端的服务器已经安装了centos7.9的操作系统。

```
[root@NFSServer ~]# cat /etc/redhat-release  
CentOS Linux release 7.9.2009 (Core)
```

- (2) 执行**yum install nfs-utils rpcbind**命令安装软件包nfs-utils和rpcbind。

```
[root@NFSServer ~]# yum install nfs-utils rpcbind
```

- (3) 依次执行**mkdir**命令和**chmod 755**命令创建共享目录linseerFileShare并添加读写权限。

```
[root@NFSServer ~]# mkdir /linseerFileShare  
[root@NFSServer ~]# chmod 775 /linseerFileShare
```

- (4) 创建并编辑/etc(exports配置文件。

- a. 执行**vi**命令创建文件。

```
[root@NFSServer ~]# vi /etc/exports
```

- b. 通过键盘输入字符“i”进入编辑界面，输入如下内容：

```
/linseerFileShare *(rw,sync,insecure,no_subtree_check,no_root_squash)
```

- c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

- (5) 执行**systemctl start**命令启动RPC服务，并执行**systemctl enable**命令设置开机自动启动RPC服务。

```
[root@NFSServer ~]# systemctl start rpcbind//rpcbind 为服务名称  
[root@NFSServer ~]# systemctl enable rpcbind//rpcbind 为服务名称
```

- (6) 执行**systemctl start**命令启动NFS服务，并执行**systemctl enable**命令设置开机自动启动NFS服务。

```
[root@NFSServer ~]# systemctl start nfs//nfs 为服务名称  
[root@NFSServer ~]# systemctl enable nfs//nfs 为服务名称
```

- (7) 执行**systemctl status**命令查看RPC服务状态为active。

```
[root@NFSServer ~]# systemctl status rpcbind  
● rpcbind.service - RPC Bind
```

```
Loaded: loaded (/usr/lib/systemd/system/rpcbind.service; enabled; vendor preset: enabled)
      Active: active (running) since Tue 2023-11-07 22:10:32 CST; 5 days ago
        TriggeredBy: ● rpcbind.socket
          Docs: man:rpcbind(8)
        Main PID: 5336 (rpcbind)
          Tasks: 1 (limit: 3355442)
        Memory: 2.5M
        CGroup: /system.slice/rpcbind.service
                  └─ 5336 /usr/bin/rpcbind -w -f

11月 07 22:10:32 NFSServer systemd[1]: Starting RPC Bind...
11月 07 22:10:32 NFSServer systemd[1]: Started RPC Bind.
```

- (8) 执行 **systemctl status** 命令查看 NFS 服务状态为 **active**。

```
[root@NFSServer ~]# systemctl status nfs
● nfs-server.service - NFS server and services
  Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor
  preset: disabled)
  Drop-In: /run/systemd/generator/nfs-server.service.d
            └─order-with-mounts.conf
  Active: active (exited) since Tue 2023-11-07 22:10:46 CST; 5 days ago
    Process: 5870 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Process: 5877 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
  Main PID: 5877 (code=exited, status=0/SUCCESS)

11月 07 22:10:46 NFSServer systemd[1]: Starting NFS server and services...
11月 07 22:10:46 NFSServer systemd[1]: Finished NFS server and services.
```

- (9) 执行 **showmount -e** 命令查看是否已正常加载本地共享目录，以下为本地共享目录加载成功后的显示信息，如果没有该显示信息，请执行 **exportfs -a** 命令重新加载本地共享目录后再次查看。

```
[root@NFSServer ~]# showmount -e
Export list for NFSServer:
/linseerFileShare *
```

- (10) 执行 **systemctl stop firewalld** 命令关闭防火墙。

```
[root@NFSServer ~]# systemctl stop firewalld
```

- (11) 执行 **systemctl disable firewalld** 命令设置开机时禁用防火墙。

```
[root@NFSServer ~]# systemctl disable firewalld
```

## 8.9.2 验证 NFS 存储服务端配置

- (1) 任意选择一台操作系统为 centos7.9 的服务器作为 NFS 客户端，确保该服务器和待验证配置的 NFS 存储服务端网络互通。

- (2) 执行 **yum install nfs-utils** 命令安装软件包 **nfs-utils**。

```
[root@localhost ~]# yum install nfs-utils
```

- (3) 安装完成后执行 **systemctl start nfs** 命令启动 NFS 服务。

```
[root@localhost ~]# systemctl start nfs
```

- (4) NFS 客户端可以使用 IP 地址或域名访问 NFS 服务端。
- 使用域名访问 NFS 服务端, 请确保已正确配置 DNS 服务器。
- ```
# 执行 vi 命令创建文件/etc/resolv.conf。
[root@localhost ~]# vi /etc/resolv.conf

# 通过键盘输入字符 “i” 进入编辑界面, 输入如下内容:
nameserver NFSServer

# 按键盘上的<ESC>键, 再输入 “:wq” , 退出配置并保存文件。
:wq

# 执行 ping NFSServer 命令可以正常通信。
[root@localhost ~]# ping NFSServer
PING NFSServer (10.114.164.43) 56(84) bytes of data.
64 bytes from NFSServer (10.114.164.43): icmp_seq=1 ttl=64 time=0.203 ms
64 bytes from NFSServer (10.114.164.43): icmp_seq=2 ttl=64 time=0.908 ms
64 bytes from NFSServer (10.114.164.43): icmp_seq=3 ttl=64 time=0.272 ms
```
- 使用 IP 地址访问 NFS 服务端, 保证 NFS 客户端和服务端可以 ping 通即可
- (5) 在 NFS 客户端上执行 **mkdir** 命令创建目录 /nfsclient, 并执行 **mount** 命令进行挂载。
- ```
[root@localhost ~]# mkdir /nfsclient

[root@localhost ~]# mount NFSServer:/linseerFileShare /nfsclient //NFSServer 为 NFS 服
务端的域名
```
- (6) 执行 **df -h** 命令可查看已成功挂载目录。
- ```
[root@localhost ~]# df -h /nfsclient
NFSServer:/linseerFileShare      50G        1.3G       49G      3% /nfsclient
```
- (7) 执行 **vi** 命令创建文件。
- ```
[root@localhost ~]# vi /nfsclient/test.txt
```
- (8) 键盘上的<ESC>键, 再输入 “:wq” , 退出配置并保存文件。
- ```
:wq
```
- (9) 在 NFS 服务器端, 执行 **cd** 命令进入 /linseerFileShare 目录, 通过 **ls** 命令可查看 NFS 客户端创建的 test.txt 文件, NFS 客户端可以正常读写共享目录。
- ```
[root@NFSServer ~]# cd /linseerFileShare
[root@NFSServer linseerFileShare]# ls
总用量 0
-rw-r--r--. 1 root root 0 9月 5 16:36 test.txt
```
- (10) 验证完成后, 在 NFS 客户端执行 **umount** 命令取消目录挂载。
- ```
[root@localhost ~]# umount /nfsclient/
```

## 9 安装网卡驱动及 ROCE 配置

### 9.1 安装网卡驱动

仅分布式推理网络须要安装网卡驱动。不同类型网卡, 请参照对应的版本说明书进行部署。

### 9.1.1 安装依赖包

- (1) 配置本地 YUM 源，详细步骤请参见“[5.4 配置本地 YUM 源](#)”。
- (2) 根据提示，执行 **yum** 命令安装依赖包。

```
[root@localhost ~]# yum install automake rpm-build lsof patch autoconf python3-devel  
elfutils-devel
```

- (3) 联系技术支持获取 **fuse-devel-2.9.9-9.h1202.x86\_64** 和 **openEuler-rpm-config-30-18.oe1.x86\_64** 的 RPM 包，进行部署安装。

```
[root@localhost ~]# rpm -ivh fuse-devel-2.9.9-9.h1202.x86_64.rpm  
[root@ning41 home]# rpm -e --nodeps NingOS-rpm-config-30-33.nos1.x86_64  
[root@localhost ~]# rpm -ivh openEuler-rpm-config-30-18.oe1.x86_64.rpm
```

### 9.1.2 安装网卡驱动

- (1) 登录 H3C 官网获取网卡驱动离线安装包。如[图 9-1](#) 所示

图9-1 官网下载网卡驱动

The screenshot shows a search interface for network card drivers. It includes fields for '支持机型' (Supported Models), '关键字' (Key Words), 'REPO定制化工具' (Customization Tools), 'VMware镜像工具' (VMware Image Tools), '服务器出厂配置查询' (Server Out-of-the-box Configuration Query), and '智宁NingOS镜像' (ZINING NingOS Image). There are dropdown menus for '版本类别' (Version Category) set to '驱动' (Driver) and '组件类型' (Component Type) set to '网卡' (Network Card). Below these are '操作系统' (Operating System) and '操作系统子类' (Operating System Sub-class) dropdowns, both set to 'Linux' and 'NingOS V3 1.0.2306-SP1'. There are also dropdowns for '推荐级别' (Recommendation Level) set to '全部' (All) and '版本号' (Version Number) which is empty. A red '清空选择' (Clear Selection) button is located at the bottom right of the search bar. Below the search bar is a table listing two driver packages:

名称	版本	类别	操作系统	操作系统子类	发布日期	操作
IC-ETH-E810XXVDA4G2P5-FH-4P	1.12.6	驱动	linux	NingOS V3 1.0.2306-SP1	2023/10/20	<a href="#">查看</a>
NIC-ETH1060F-LP-2P	1.12.6	驱动	linux	NingOS V3 1.0.2306-SP1	2023/10/20	<a href="#">查看</a>

- (2) 通过 **FTP** 工具将网卡驱动安装包上传到服务器（例如/**/root** 目录）。
- (3) 执行 **tar** 命令进行解压缩。

```
[root@localhost ~]# unzip driver-nic-mellanox-MLNX-OFED-5.8-3.0.7.0\ NingOS\ V3\  
1.0.2403-SP1.zip  
Archive:  driver-nic-mellanox-MLNX-OFED-5.8-3.0.7.0 NingOS V3 1.0.2403-SP1.zip  
extracting: MLNX_OFED_LINUX-5.8-3.0.7.0-openeuler20.03sp3-ext.tgz
```

```
[root@localhost ~]# tar -zxvf MLNX_OFED_LINUX-5.8-3.0.7.0-openeuler20.03sp3-ext.tgz
```

- (4) 进入安装包解压后的文件夹目录，执行如下命令进行安装。

```
[root@localhost ~]# ./mlnxofedinstall --skip-distro-check --distro openeuler20.03sp3  
--add-kernel-support --skip-repo --force
```

- (5) 驱动安装完成后执行 **ofed\_info -s** 命令可以查看驱动版本信息。
- ```
[root@localhost ~]# ofed_info -s
MLNX_OFED_LINUX-5.8-3.0.7.0:
```
- (6) 网卡驱动安装完成后执行 **systemctl start openibd** 命令，启动 **openibd** 服务加载新驱动。
- ```
[root@localhost ~]# systemctl start openibd
```
- (7) 执行 **systemctl enable openibd** 命令设置 **openibd** 服务开机自动启动。
- ```
[root@localhost ~]# systemctl enable openibd
```
- (8) 执行 **reboot** 命令重启服务器。
- ```
[root@localhost ~]# reboot
```
- (9) 服务器重启完成后，执行 **mst start** 命令启动网卡管理服务。
- ```
[root@localhost ~]# mst start
```
- (10) 执行 **ibdev2netdev** 命令查看网卡信息，显示如下信息则表示网卡驱动安装成功。
- ```
[root@localhost]# ibdev2netdev
mlx5_0 port 1 ==> ens2np0 (Up)
mlx5_1 port 1 ==> ens5f0np0 (Up)
mlx5_2 port 1 ==> ens5f1np1 (Up)
mlx5_3 port 1 ==> ens3np0 (Up)
```

## 9.2 参数网卡ROCE配置

所有参数网网卡接口均需配置 **roce**。本次以 **mlx5\_1** 为例。

- (1) 使用 **ibdev2netdev** 命令，查看 **mlx** 设备名称和以太网接口名称对应关系。
- ```
[root@localhost]# ibdev2netdev
mlx5_0 port 1 ==> ens2np0 (Up)
mlx5_1 port 1 ==> ens5f0np0 (Up)
mlx5_2 port 1 ==> ens5f1np1 (Up)
mlx5_3 port 1 ==> ens3np0 (Up)
```
- (2) 设置 RoCE 模式为 V2。
- ```
[root@localhost]# cma_roce_mode -d mlx5_1 -p 1 -m 2
```
- (3) 配置接口信任报文的 DSCP 优先级。
- ```
[root@localhost]# mlnx_qos -i ens5f0np0 --trust dscp
```
- (4) 设置 ToS(DSCP)值。DSCP 值为 160 表示网卡流量映射到 5 队列
- ```
[root@localhost]# cma_roce_tos -d mlx5_1 -t 160
[root@localhost]# echo 160 > /sys/class/infiniband/ mlx5_1 /tc/1/traffic_class
```
- (5) 设置 CNP 的 PFC 优先级。
- ```
[root@localhost]# echo 48 > /sys/class/net/ens5f0np0 /ecn/roce_np/cnp_dscp
[root@localhost]# echo 6 > /sys/class/net/ens5f0np0 /ecn/roce_np/cnp_802p_prio
```
- (6) 开启 RoCE 队列的 PFC 功能。
- ```
[root@localhost]# mlnx_qos -i ens5f0np0 -p 0,1,2,3,4,5,6,7
```
- (7) 开启接口 802.1P 优先级为 5 的报文的 PFC。
- ```
[root@localhost]# mlnx_qos -i ens5f0np0 --pfc 0,0,0,0,0,1,0,0
```

- (8) 网卡相关配置服务器重启后不会生效，建议将网卡的完整配置添加到 `rc.local` 配置文件中，形成启动自动加载并给予可执行权限。

```
[root@localhost ~]# vi /etc/rc.local  
[root@localhost ~]# chmod +x /etc/rc.d/rc.local
```

# 10 安装 k8s-rdma-shared-dev-plugin

仅分布式推理组网须要安装 `k8s-rdma-shared-dev-plugin`。

## 10.1.1 配置 ConfigMap

- (1) 执行 `ibdev2netdev` 命令，查询 RDMA 网卡和物理网卡的对应关系。

```
[root@localhost ~]# ibdev2netdev  
mlx5_0 port 1 ==> ens2np0 (Up)  
mlx5_1 port 1 ==> ens5f0np0 (Up)  
mlx5_2 port 1 ==> ens5f1np1 (Up)  
mlx5_3 port 1 ==> ens3np0 (Up)
```

- (2) 执行 `mst status -v` 命令，查询所有 UP 状态的网卡对应的 PCI 号，例如 `mlx5_11` 网卡对应的 PCI 号为 `ec:00.1`。

```
[root@localhost ~]# mst status -v  
MST modules:  
-----  
MST PCI module is not loaded  
MST PCI configuration module is not loaded  
PCI devices:  
-----  


DEVICE_TYPE	MST	PCI	RDMA	NET	NUMA
ConnectX6(rev:0)	NA	2f:00.0	mlx5_0	net-ens2np0	0
ConnectX4LX(rev:0)	NA	c1:00.0	mlx5_1	net-ens5f0np0	1
ConnectX4LX(rev:0)	NA	c1:00.1	mlx5_2	net-ens5f1np1	1
ConnectX6(rev:0)	NA	a6:00.0	mlx5_3	net-ens3np0	1


```

- (3) 执行 `lspci -n | grep 2e:00.0` 命令。查询各 RDMA 网卡的厂商 ID 和设备 ID。

```
[root@localhost ~]# lspci -n | grep 2e:00.0  
c1:00.1 0200: 15b3:1015 //厂商 ID 是 15b3，设备 ID 是 1015
```

- (4) 创建并编辑 `configmap.yml` 文件。

- a. 执行 `vi` 命令，创建文件。

```
[root@localhost ~]# vi configmap.yml
```

- b. 通过键盘输入 “i” 进入编辑界面，在配置文件中增加如下内容，并根据实际设备信息修改灰显部分内容。本处以配置 8 张 RDMA 网卡为例，请根据实际配置网卡的个数进行修改。

```
apiVersion: v1  
kind: ConfigMap
```

```

metadata:
  name: rdma-devices
  namespace: kube-system
data:
  config.json: |
    {
      "periodicUpdateInterval": 300,
      "configList": [
        {
          "resourceName": "hpc_shared_devices_ens5f0np0", //第 1 张物理网卡名称
          "rdmaHcaMax": 1000,
          "selectors": {
            "vendors": ["15b3"], //第 1 张 RDMA 网卡对应的厂商 ID
            "deviceIDs": ["1015"], //第 1 张 RDMA 网卡对应的设备 ID, 通过上述步骤可查询
            "ifNames": ["ens5f0np0"] //填写第 1 张物理网卡名称
          }
        },
        {
          "resourceName": "hpc_shared_devices_ens17f1np1",
          "rdmaHcaMax": 1000,
          "selectors": {
            "vendors": ["15b3"],
            "deviceIDs": ["1015"],
            "ifNames": ["ens5f1np1"]
          }
        }
      ]
    }

```

c. 按键盘上的<ESC>键，再输入“:wq”，退出配置并保存文件。

```
:wq
```

(5) 执行 **kubectl apply -f** 命令，部署 configmap.yml。

```
[root@localhost ~]# kubectl apply -f configmap.yml
configmap/rdma-devices created
```

## 10.1.2 部署 k8s-rdma-shared-dev-plugin

(1) 在 GPU 服务器上执行 **docker pull** 命令，获取镜像文件。

```
[root@localhost ~]# docker pull mellanox/k8s-rdma-shared-dev-plugin:latest
latest: Pulling from mellanox/k8s-rdma-shared-dev-plugin
a9e23b64ace0: Pull complete
38b71301a1d9: Pull complete
39alf5c62989: Pull complete
2e15b5a001c5: Pull complete
Digest: sha256:fd4d86864647270720354d247441f97deb76caf258869f1148d2bb6d0bf90754
Status: Downloaded newer image for mellanox/k8s-rdma-shared-dev-plugin:latest
docker.io/mellanox/k8s-rdma-shared-dev-plugin:latest
```

(2) 在任一个 Master 节点上执行 **wget** 命令，获取 daemonset.yaml，其他节点无需重复执行。

```
[root@localhost ~]# wget
https://raw.githubusercontent.com/Mellanox/k8s-rdma-shared-dev-plugin/master/deploy
ment/k8s/base/daemonset.yaml
--2023-11-08 12:49:36--
https://raw.githubusercontent.com/Mellanox/k8s-rdma-shared-dev-plugin/master/deploy
ment/k8s/base/daemonset.yaml
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.111.133,
185.199.109.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1344 (1.3K) [text/plain]
Saving to: 'daemonset.yaml'

daemonset.yaml
100%[=====]=====
=====
```

2023-11-08 12:49:36 (227 MB/s) - 'daemonset.yaml' saved [1344/1344]

- (3) 在任一个 Master 节点上执行 **kubectl apply -f** 命令，部署 **daemonset.yaml**，其他节点无需重复执行。

```
[root@localhost ~]# kubectl apply -f daemonset.yaml
```

- (4) 执行如下命令查询 **rdma-shared-dp-ds** pod 状态，如果均为 **Running**，表示部署成功。

```
[root@localhost ~]# kubectl get pod -A | grep rdma
kube-system      rdma-shared-dp-ds-5qznm      1/1      Running      0          22h
kube-system      rdma-shared-dp-ds-g52zx      1/1      Running      0          22h
```

## 11 (可选) 部署 PolarDB 数据库

作为一种云原生的关系型数据库，PolarDB 设计了许多特性以应对现代互联网企业和大型企业的需求。具有高性能、弹性扩展、高可用性、兼容性强、数据安全性高、管理和维护方便等优势。关于 polarDB 数据库部署的详细介绍，请参见《第三方数据库 PolarDB v2.0 安装部署指导-5W100 (H3C 版本)》。

## 12 安装部署 LinSeer RT



说明

在部署 LinSeer RT 组件包时，根目录需要预留 500G 磁盘空间。

---

## 12.1 登录Matrix

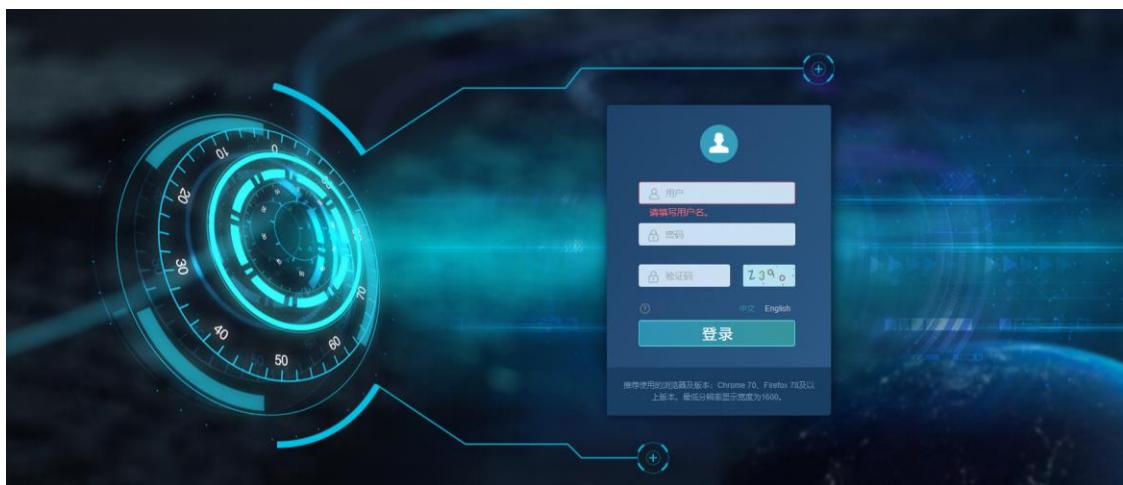
- (1) 在浏览器中输入 Matrix 的登录地址，进入如图 12-1 所示的登录页面。登录地址格式为：[https://ip\\_address:8443/matrix/ui/](https://ip_address:8443/matrix/ui/)。其中 *ip\_address* 为 Master 节点 IP 地址，8443 为缺省端口号。



说明

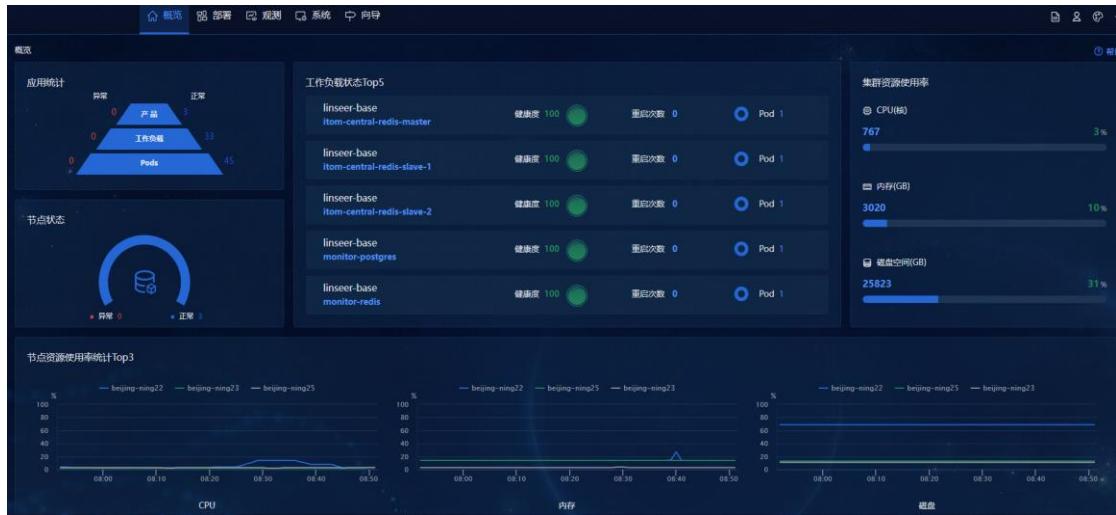
采用集群部署模式时，未部署集群之前，*ip\_address* 可以是任意一个规划为 Master 节点的 IP 地址。

图12-1 Matrix 登录界面



- (2) 输入用户名和密码(默认用户名为 admin, 密码为 Pwd@12345, 若安装操作系统设置过密码，则按设置的填写)后，单击<登录>按钮进入概览页面。

图12-2 概览页面



## 12.2 部署LinSeer RT组件安装包

### 12.2.1 上传安装包



#### 说明

- 对于较大的组件安装包，可以通过 FTP 等方式，上传到集群各个 Master 节点的 /opt/matrix/app/install/packages 目录下，然后在 Matrix 页面进行解析完成安装部署。
- LinSeer RT 默认支持 NVIDIA GPU，无需独立部署。昆仑芯和天数 GPU，需要单独部署对应的组件包。
- LinSeer RT 各组件安装部署步骤类似，且支持同时部署昆仑芯和天数组件包。

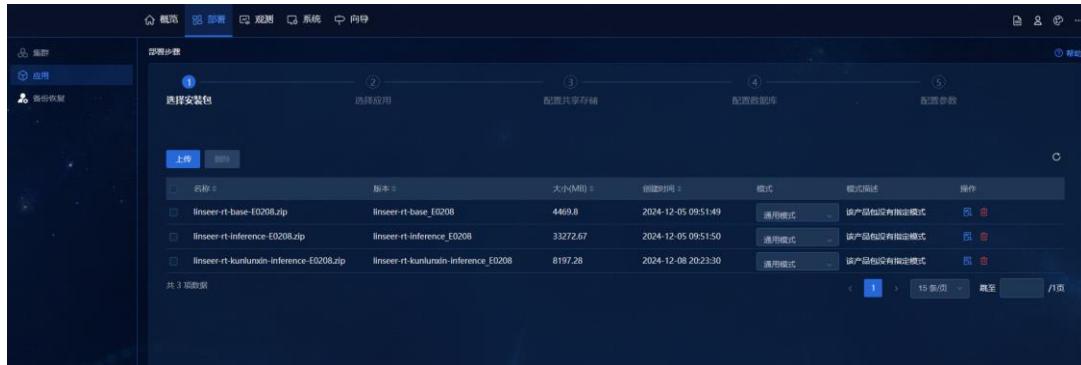
- (1) 在概览页面选择顶部“部署”页签，单击左侧[应用]菜单项进入应用列表页面。单击<部署应用>按钮开始部署，如图 12-3 所示。

图12-3 Matrix 集群应用部署页面



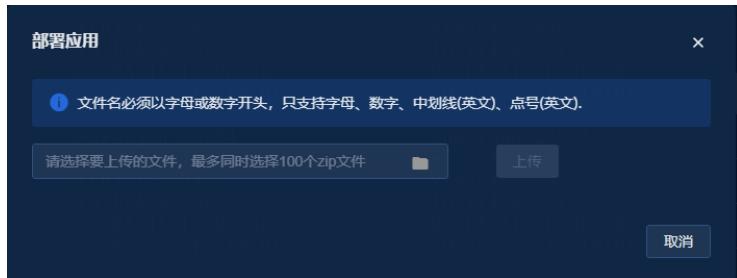
(2) 单击<上传>按钮，弹出部署应用对话框。

图12-4 上传安装包页面



(3) 在部署应用对话框中选择 LinSeer RT 安装包，并单击<上传>按钮上传安装包。

图12-5 上传安装包页面



(4) 重复上述步骤，直到所有组件安装包都上传完毕。

## 12.2.2 解析安装包



### 说明

- 不同 GPU 环境下，解析组件安装包的顺序有所不同：
- NVIDIA GPU 环境：linseer-rt-base 安装包->linseer-rt-inference 安装包。
- 昆仑芯 GPU 环境：linseer-rt-base 安装包->linseer-rt-inference 安装包->linseer-rt-kunlunxin-inference 安装包。
- 天数 GPU 环境：linseer-rt-base 安装包->linseer-rt-inference 安装包->linseer-rt-tianshu-inference 安装包。

(1) 如图 12-6 所示，选择已上传的 LinSeer RT 安装包，单击<下一步>按钮，开始解析安装包。

图12-6 解析安装包



### 12.2.3 部署 LinSeer RT 安装包

LinSeer RT 不同安装包的部署步骤有所不同。

#### 1. 部署 LinSeer RT 基础安装包

- (1) 安装包解析完成后，系统自动选中解析后的组件包，如图 12-7 所示，单击<下一步>按钮跳过配置共享存储和配置数据库两个步骤，进入配置参数页面。

图12-7 部署选中的组件包



- (2) 在配置参数页面，选择“配置项”区域中的“DataStoreConfig”选项，单击“配置项参数”编辑按钮 ，修改共享存储的配置项参数，如图 12-8 所示。

o 对于单机部署模式：

- REDIS PASSWORD: 缓存密码，使用默认值，无需修改。
- STORAGE TYPE: 存储类型，默认为“local”，无需修改。
- STORAGE PATH: 存储服务器路径，默认为“/linseerFileShare”。
- STORAGE IP: 存储服务器 IP 地址，默认为“127.0.0.1”，不可修改。
- DATABASE TYPE: 数据库类型，inner 为默认，选择 PolarDB 需要额外填写连接参数。
- DATABASE IP: 数据库 IP 或域名。默认值 postgres-svc.linseer-service.svc。使用 PolarDB 数据库时，需要填写实际的数据库服务 IP 地址。

- **DATABASE PORT:** 数据库端口。默认值 5432。使用 PolarDB 数据库时，需要填写实际的数据库服务端口号。
- **DATABASE USERNAME:** 数据库管理员用户名。默认值 dba。使用 PolarDB 数据库时，需要填写实际的数据库服务管理员用户名。
- **DATABASE PASSWORD:** 数据库管理员密码。默认值 PassWord。使用 PolarDB 数据库时，需要填写实际的数据库服务管理员密码。
- o 对于 3 机集群部署模式和 3+N 机集群部署模式：
  - **REDIS PASSWORD:** 缓存密码。使用默认值，无需修改。
  - **STORAGE TYPE:** 存储类型，选择“NFS”。
  - **STORAGE PATH:** 存储服务器路径，请根据实际情况填写。
  - **STORAGE IP:** 存储服务器 IP 地址，请根据实际情况填写。
  - **DATABASE TYPE:** 数据库类型，inner 为默认，选择 PolarDB 需要额外填写连接参数。
  - **DATABASE IP:** 数据库 IP 或域名。默认值 postgres-svc.linseer-service.svc。使用 PolarDB 数据库时，需要填写实际的数据库服务 IP 地址。
  - **DATABASE PORT:** 数据库端口。默认值 5432。使用 PolarDB 数据库时，需要填写实际的数据库服务端口号。
  - **DATABASE USERNAME:** 数据库管理员用户名。默认值 dba。使用 PolarDB 数据库时，需要填写实际的数据库服务管理员用户名。
  - **DATABASE PASSWORD:** 数据库管理员密码。默认值 PassWord。使用 PolarDB 数据库时，需要填写实际的数据库服务管理员密码。



### 说明

请确保集群服务器节点与 NFS 服务端网络能正常通信。

请确保 RT 节点与 PolarDB 数据库服务节点网络正常通信

图12-8 配置默认参数



图 10-9 配置 PolarDB 数据库参数



- (3) 编辑完成后，单击<部署>按钮，等待基础安装包部署完成，在弹出的对话框中单击<确定>按钮即可，如图 12-9 所示。

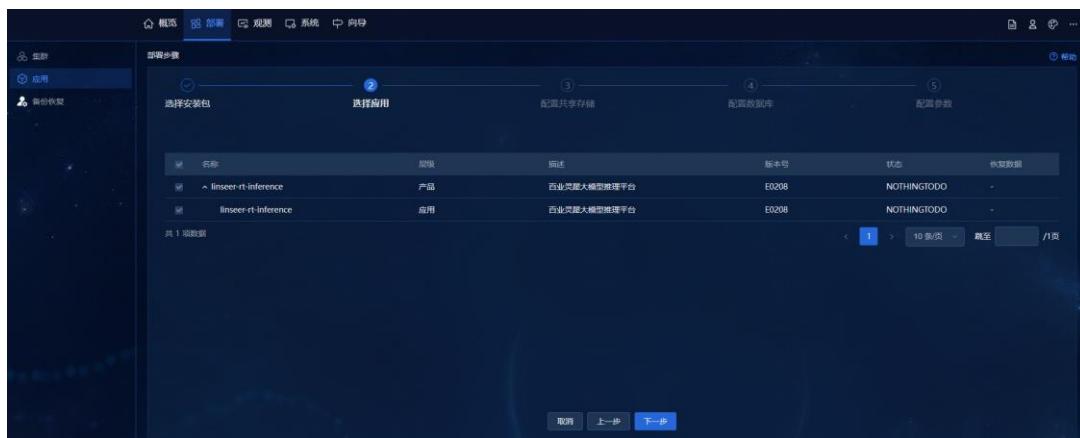
图12-9 部署 LinSeer RT 基础安装包



## 2. 部署 LinSeer RT 其他安装包

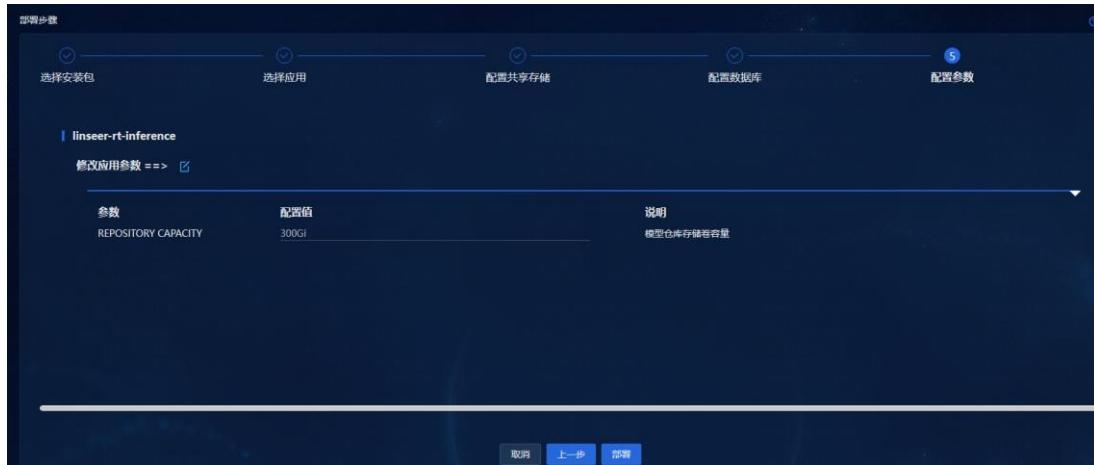
- (1) 安装包解析完成后，系统自动选中解析后的组件包，如图 12-10所示，单击<下一步>按钮部署 LinSeer RT。

图12-10 系统自动选中解析后的组件包



- (2) 连续单击<下一步>按钮跳过中间部署步骤。
- (3) 如图 12-11 所示, 单击<部署>按钮, 等待安装包部署完成即可。

图12-11 部署 LinSeer RT 其他安装包



### 3. 检查部署结果

待所有组件均部署完成后, 在服务器上执行 `kubectl get pod -n linseer-service` 命令查看各组件的 Pod 状态均为“Running”, 表示系统已正常运行。

```
[root@localhost ~]# kubectl get pod -n linseer-service
```

| NAME                                          | READY | STATUS  | RESTARTS | AGE  |
|-----------------------------------------------|-------|---------|----------|------|
| exporter-kernel-kaf-8445b85b6c-skg64          | 1/1   | Running | 2        | 3d3h |
| exporter-redis-57d95ff856-xk8nc               | 1/1   | Running | 0        | 3d3h |
| exporter-zookeeper-864b886c8-26hwj            | 1/1   | Running | 0        | 3d3h |
| filebeat-d254m                                | 1/1   | Running | 0        | 3d3h |
| inference-manager-deployment-54df7f9c7d-7n775 | 1/1   | Running | 0        | 3d2h |
| itom-central-login-256d8                      | 1/1   | Running | 0        | 3d3h |
| itom-central-redis-master-fc4d946c7-kktvn     | 1/1   | Running | 0        | 3d3h |
| itom-central-ucd-nkhbl                        | 1/1   | Running | 0        | 3d3h |
| k-commonresource-ui-5f9cdd867-n4jgn           | 1/1   | Running | 0        | 3d3h |
| k-confcenter-rs-59dcbb5bbd8-lvtdf             | 1/1   | Running | 0        | 3d3h |
| k-confcenter-ui-676c964494-5nq6c              | 1/1   | Running | 0        | 3d3h |
| k-framework-rs-qp5bk                          | 1/1   | Running | 0        | 3d3h |
| k-framework-ui-k4bsv                          | 1/1   | Running | 0        | 3d3h |
| k-gpu-exporter-7t9hj                          | 1/1   | Running | 100      | 3d3h |
| k-gpu-exporter-ts-9qzrg                       | 1/1   | Running | 0        | 3d   |
| k-helpcenter-ui-6758cf9454-vfbvp              | 1/1   | Running | 0        | 3d3h |
| k-kernel-rs-84f744f959-17xg8                  | 1/1   | Running | 0        | 3d3h |
| k-kernel-ui-559588fd96-hgc2n                  | 1/1   | Running | 0        | 3d3h |
| k-license-ui-b6c4464cf-27pl4                  | 1/1   | Running | 0        | 3d3h |
| k-logcenter-rs-gn8h9                          | 1/1   | Running | 0        | 3d3h |
| ...                                           |       |         |          |      |

# 13 访问 LinSeer RT

- (1) 在浏览器中输入 LinSeer RT 的登录地址“`http://ip_address:29000`”，其中 `ip_address` 为北向业务虚 IP，进入 LinSeer RT 登录页面，如图 13-1 所示。

图13-1 LinSeer RT 登录页面



- (2) 输入用户名和密码(默认用户名 admin, 密码 Pwd@12345)，单击<立即登录>按钮，进入 LinSeer RT 系统，如图 13-2 所示。

图13-2 LinSeer RT 系统



# 14 增加模型

- (1) 登录服务器，进入`/linserFileShare` 目录，创建 `model_preset` 目录。

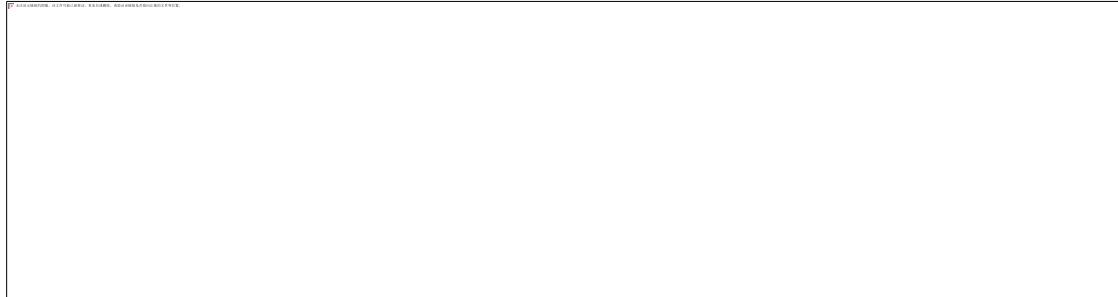
```
[root@localhost ~]# cd /linserFileShare  
[root@localhost ~]# mkdir model_preset
```

- (2) 将模型包（例如表 2-5 中的模型包）放至`/linseerFileShare/model_preset` 目录下，并解压。

```
[root@localhost ~]# unzip LinSeer-Model-2.1-AWQ-INT4-version.zip
```

- (3) 浏览器登录 Linseer RT 页面，进入模型仓库页面，点击<增加>按钮，进入增加模型页面，如图 14-1 所示。

图14-1 增加模型



- **模型名称:** 只允许出现数字、字母、-和.，且长度为 2-64 个字符，模型名称为唯一，不可重复。
- **模型版本:** 只允许出现数字、字母、-和.，且长度为 2-64 个字符，模型版本为唯一，不可重复。
- **模型描述:** 最多支持 255 个字符。
- **模型类型:** 根据不同类型的模型选择对应的模型类型：
  - **LinSeer:** 百业灵犀模型
  - **LinSeer-Lite:** 百业灵犀 Lite 模型
  - **LinSeer-Code:** 百业灵犀 Code 模型
  - **LinSeer-Code-Lite:** 百业灵犀 Code-Lite 模型
  - **LinSeer-LLM:** 第三方开源模型
  - **LinSeer-Diffusion:** 文生图模型
  - **LinSeer-TS-Gauss:** 时序模型
- **模型目录:** 请选择模型目录，列表数据从共享存储目录 model\_preset 下获取。

- (4) 点击确定，完成模型增加。如图 14-2 模型增加成功所示。

图14-2 模型增加成功



# 15 软件授权

LinSeer RT 部署完成后即可试用。LinSeer RT 提供 License 管理功能，可为产品管理授权。关于 License Server 服务部署的详细介绍，请参见《H3C License Server 安装指导》。关于产品授权的详细介绍，请参见《License 支持情况说明》和《License 使用指南》。

# 16 集群扩缩容

## 16.1 集群扩容Worker节点

如需扩容，请先确保待扩容的 Worker 节点已完成上述部署步骤，再进行如下操作。

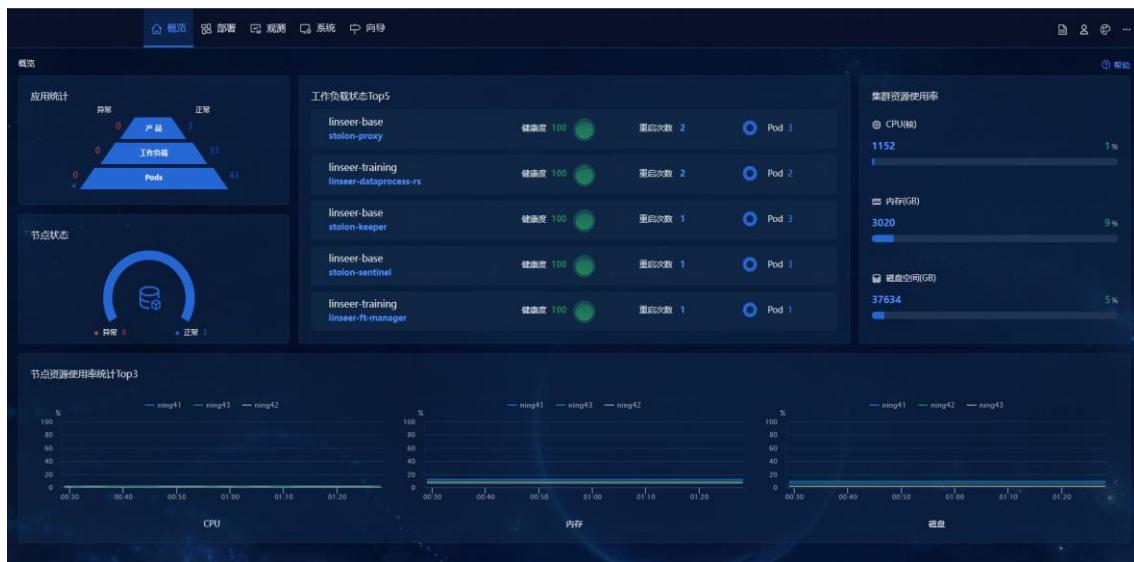
- (1) 在浏览器中输入 Matrix 的登录地址，登录地址格式为：[https://ip\\_address:8443/matrix/ui/](https://ip_address:8443/matrix/ui/)，其中，*ip\_address* 为 Master 节点 IP 地址，8443 为缺省端口号。输入用户名、密码和图形验证码后，单击<登录>按钮进入概览页面。如图 16-1 所示。



说明

登录用户名默认为 admin，初始密码为 Pwd@12345，若安装操作系统时重新设置过密码，请按填写新密码。

图16-1 Matrix 概览界面



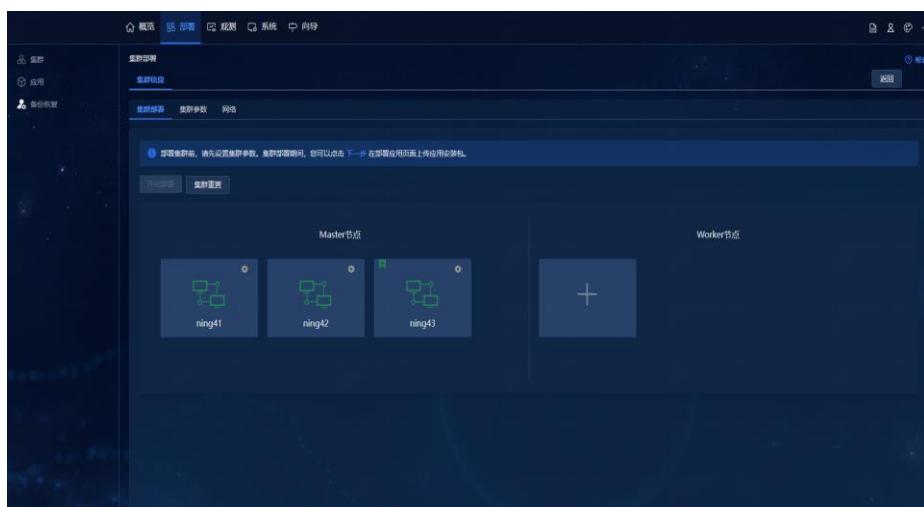
- (2) 点击[部署>集群]，进入部署集群页面，如图 16-2 所示

图16-2 部署集群页面



(3) 单击图 16-2 中操作栏中的详情按钮 ，进入集群部署页面，如图 16-3 所示。

图16-3 集群部署页面



(4) 单击 Worker 节点区域的图标 ，弹出增加节点窗口，如图 16-4 所示。

- 增加单个 Worker 节点，配置参数后，单击<应用>按钮，完成增加 Worker 节点操作。
  - 节点 IP 地址：规划的 Master 节点的 IP 地址，为安装 NingOS 系统时设置的 IP 地址。
  - 用户名：节点操作系统的用户名，为安装 NingOS 系统时设置的 root 用户名。
  - 密码：节点操作系统的用户密码，为安装 NingOS 系统时设置的 root 用户密码。

图16-4 添加单个 Worker 节点示意图



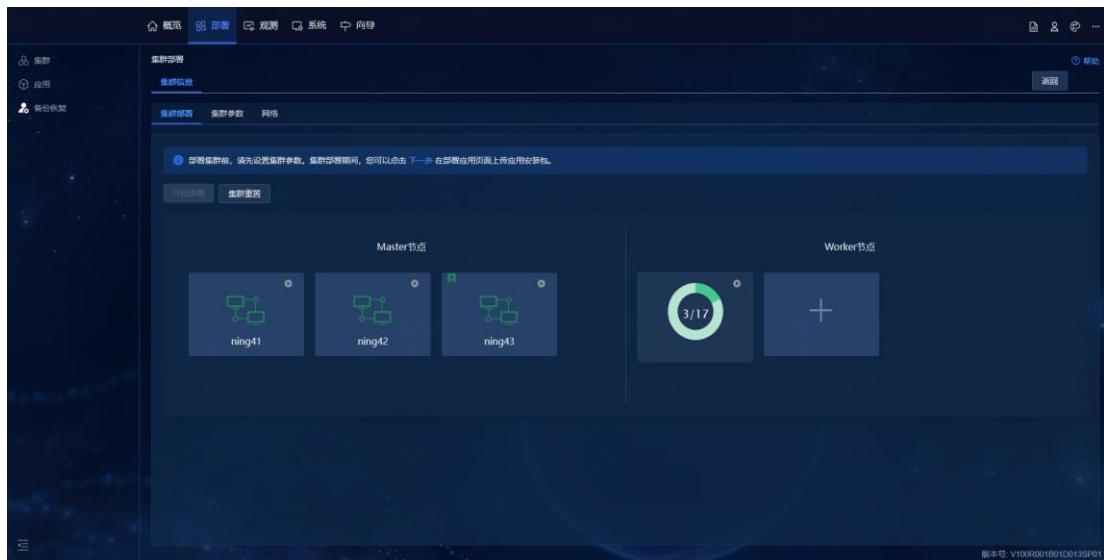
- 批量增加多个 Worker 节点，在增加 Worker 节点页面点击“批量增加”按钮，如图 16-5 所示，单击<下载模板>按钮，并根据模板提示信息填写节点信息，再上传模板完成增加 Worker 节点操作。

图16-5 批量增加多个 Worker 节点示意图



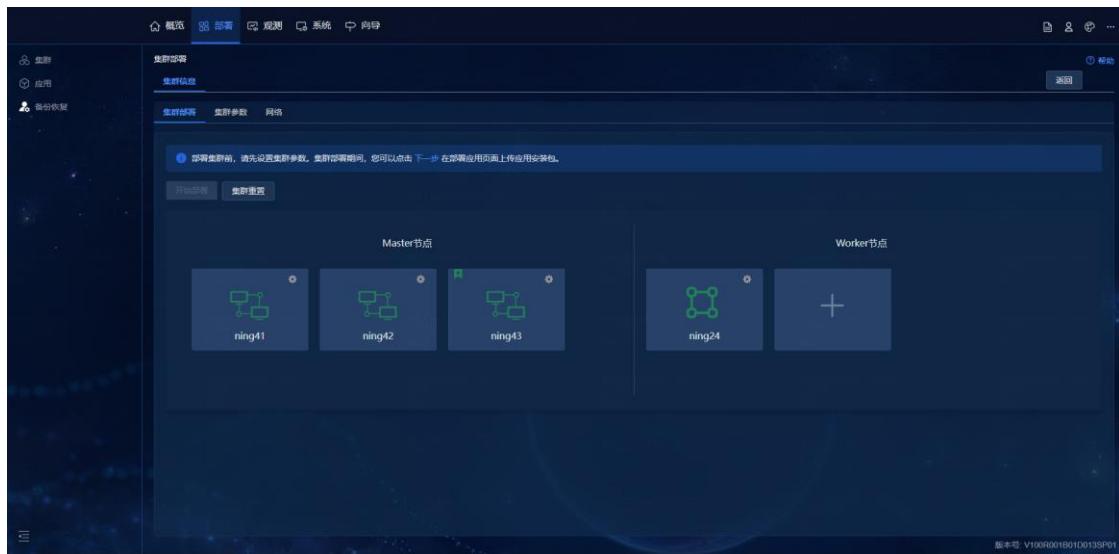
(5) 单击<开始部署>按钮部署集群，如图 16-6 所示。

图16-6 集群部署



(6) 部署成功后，集群部署界面显示该 Worker 节点。如图 16-7 所示。

图16-7 集群部署成功页面



(7) 对增加的 NVIDIA GPU 服务器 Worker 节点进行 device-plugin 配置，详细操作步骤请参见 [8.3.3 部署 NVIDIA 设备插件（离线部署方式）](#)。

#### 说明

- 本步骤仅扩容 NVIDIA GPU 服务器 Worker 节点时涉及。
- 增加的 Worker 节点无 `kubectl` 等命令的权限，相关内容请在 Master 节点上进行配置。

在 Master 节点上重启 Pod，重启服务使 GPU 可识别。

```
[root@localost ~]# kubectl get pod -A | grep k-gpu
linseer-service   k-gpu-exporter-8kps2  1/1  Running   0  44h
linseer-service   k-gpu-exporter-bgpkj  1/1  Running   0  21m
linseer-service   k-gpu-exporter-rpqjx  1/1  Running   0  44h
linseer-service   k-gpu-exporter-xx22n  1/1  Running   0  44h
[root@localost ~]# kubectl delete pod -n linseer-service k-gpu-exporter-8kps2
k-gpu-exporter-bgpkj k-gpu-exporter-rpqjx  k-gpu-exporter-xx22n
pod "k-gpu-exporter-8kps2" deleted
pod "k-gpu-exporter-bgpkj" deleted
pod "k-gpu-exporter-rpqjx" deleted
pod "k-gpu-exporter-xx22n" deleted
```

## 16.2 集群缩容Worker节点

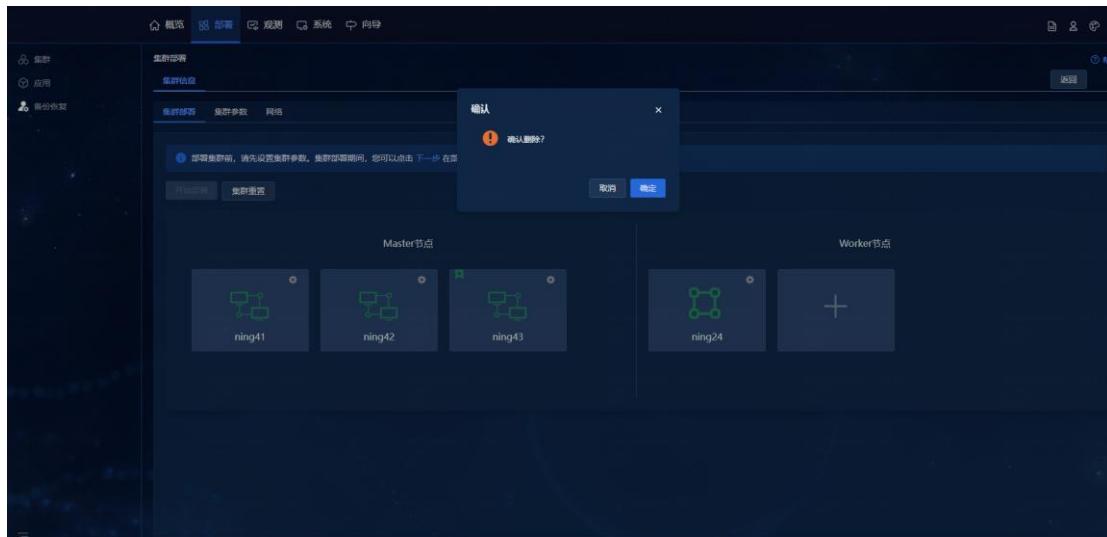
- (1) 单击 Worker 节点右上角的齿轮图标 ，在弹出的菜单项中选择“删除”，如图 16-8 所示。

图16-8 选择删除 Worker 节点



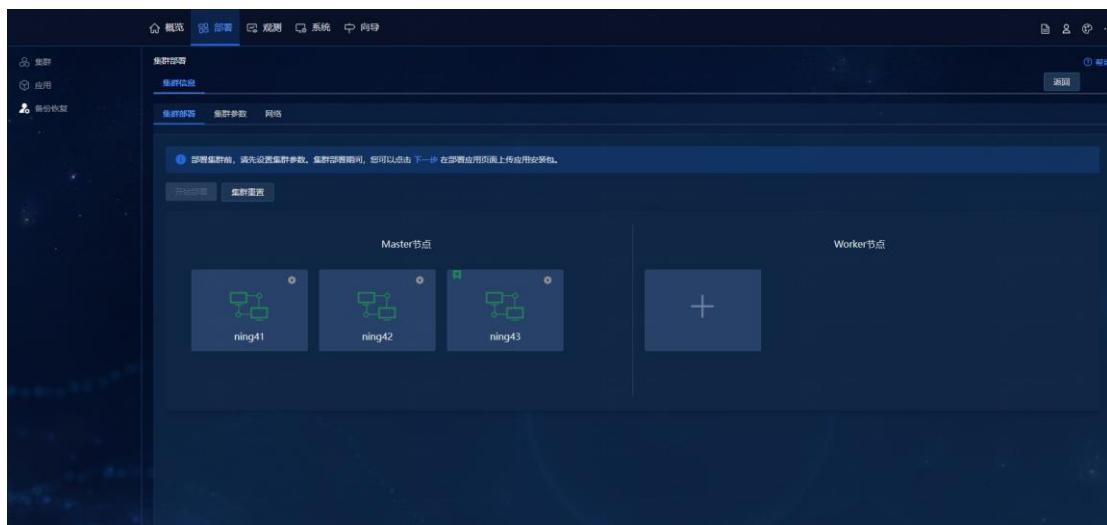
- (2) 在弹出的确认窗口中单击<确定>按钮，如图 16-9 所示。

图16-9 确认删除 Worker 接口



(3) 确认完成删除 Worker 节点后，集群部署界面上不再显示该节点信息，如图 16-10 所示。

图16-10 Worker 节点删除成功



## 17 卸载组件

在浏览器中输入 Matrix 的登录地址进入登录页面，输入用户名和密码，单击<登录>按钮登录 Matrix。

- (1) 如图 17-1 所示，选择[应用]菜单项打开应用列表，单击“LinSeer”下挂的各个组件操作行的删除按钮，在操作提示框中单击<确定>按钮完成卸载。
- (2) 请遵循如下顺序卸载各个组件安装包： linseer-rt-kunlunxin-inference 安装包>linseer-rt-tianshu-inference 安装包>linseer-rt-inference 安装包> linseer-rt-base 安装包。

图17-1 卸载组件



# 18 常见问题解答

## 18.1 安装操作系统时，各磁盘分区分别有什么作用？

安装 NingOS 操作系统时，各主要磁盘分区作用如下：

- **/:** Matrix 使用，包括 K8s、Harbor 和各组件上传的安装包，该分区的容量和各组件上传的镜像大小有关，需要确定各组件占用的磁盘容量大小，在此基础上扩缩容。
- **/var/lib/docker/:** 与 Docker 的运行相关，需要根据实际运行情况确定容量大小。
- **/var/lib/ssdata/:** 供 PXC、Kafka、ZooKeeper 使用。
- **/linseerFileShare:** 推荐预设容量为 1TB。
  - 单机部署模式下，服务器需要额外预留本地磁盘空间，本处以 /linseerFileShare 为例介绍。/linseerFileShare 供推理组件使用，作为模型仓库的共享目录。
  - 3 机集群部署模式、3+N 机集群部署模式下，使用外置的 NFS 存储空间作为服务器预留磁盘空间。

## 18.2 为什么要禁用 Nouveau？

部署百业灵犀推理引擎，需要使用 NVIDIA 显卡驱动。安装 NVIDIA 显卡驱动前必须禁用 Linux 自带的显卡驱动 Nouveau，否则会提示显卡驱动冲突。