GPU 云主机

用户使用指南

天翼云科技有限公司

1 产品简介

1.1产品定义

GPU 云主机是基于 GPU 的应用于视频解码、图形渲染、深度学习、科学计算等多种场景的计算服务。天翼云 GPU 云主机采用业界先进的 GPU 硬件，让客户得到极致性能体验的同时，获得最佳性价比。目前提供基于 NVIDIA GRID 虚拟化技术的图形加速基础型 (G 系列) 和基于硬件直通技术的计算加速型 (P 系列) 两类 GPU 云主机。

为什么选择 GPU 云主机

GPU 云主机规格族相比于其他弹性云主机规格族，增加了 GPU 计算力，与 CPU 相比，GPU 的特点如下：

维度	GPU	CPU
核心数量	数千个	几十个
运算单元	拥有大量擅长处理大规模并发计算的算术运算单元	拥有数量较少但强大的算术运算单元
逻辑控制单元	相对简单	复杂
缓存	少量缓存	大量缓存
适用场景	计算密集，相似度高，且多线程并行	逻辑复杂，串行运算

GPU 云主机与自建 GPU 服务器对比 ：

优势	GPU 云主机	自建 GPU 服务器
弹性	分钟级快速创建 。 同规格族内灵活升降配。 云盘、带宽等产品可按需扩容。	建设周期长，且建设完成后硬件配置无法灵活变更。
易用	提供和标准云主机一致的使用方式和管理功能。 与多种云产品无缝接入。 清晰的 GPU 驱动的安装、部署指引。	运营维护成本高、难度大。
安全	通过隧道技术实现 100%二层网络隔离，实现安全隔离。	很难阻止 MAC 欺骗和 ARP 攻击，需额外购买

优势	GPU 云主机	自建 GPU 服务器
	配置安全组和网络 ACL，实现云主机和子网层面的访问控制，满足不同行业客户的安全隔离需要。	基础安全防护服务。
成本	提供包周期和按需两种计费方式，用户可根据自身业务情况灵活选择	无法按需购买，一次投入成本巨大。

1.2产品优势

性能卓越可靠

- GPU 云主机具有超强的计算性能。
- 采用主流的 GPU 和 CPU。
- 提供了强大的单双精度浮点运算能力，单卡最高提供 312TFLOPS 单精度计算，同时支持单机多卡，实现性能翻倍。

功能丰富强大

- 支持 Tensorflow、Caffe、PyTorch、MXNet 等多种 AI 框架。
- 支持 DirectX 、OpenGL、Vulkan 等多种专业级图形加速接口。

覆盖范围广阔

- GPU 云主机目前已在近 30 个省份实现规模化部署上线，能够更好的满足客户的各种业务部署需求。

服务稳定安全

- GPU 云主机提供安全可靠的网络环境和完善的防护服务。
- 位于高速网络环境中，内网时延低，提供优秀的计算能力。
- 与云安全无缝对接，享有与弹性云主机同等的云安全基础防护和高防服务。

使用方便快捷

- 提供和弹性云主机一致的使用方式和管理功能，GPU 云主机可以做到分钟级快速发放。
- 入门简单，用户可以迅速搭建一个 GPU 云主机，无需跳板机登录，简单易用。
- 与负载均衡、云硬盘等多种云产品无缝接入。

1.3功能特性

GPU 云主机作为弹性云主机的一类实例规格，保持了部分与弹性云主机实例相同的功能特性，同时也新增了部分独有特性。

全面监控及告警机制，保障云主机正常运行

云资源池提供包括 CPU、内存、磁盘和网络使用情况的几十项云主机性能监控指标，并支持查看一段时间内的云主机监控信息，帮助用户了解云主机实例的历史运行情况，还可根据用户预设规则提供及时的告警通知。

通过虚拟私有云（VPC）实现网络的灵活规划

VPC 可为云服务器、云容器、云数据库等云上资源构建隔离、 私密的虚拟网络环境。通过 VPC 可灵活管理云上网络， 包括创建子网、设置安全组和网络 ACL、管理路由表、申请弹性公网 IP 和带宽等。

多种存储类型随意选择，满足不同 I/O 性能要求

提供普通 IO、高 IO、通用型 SSD 和超高 IO 等多个类型的云硬盘，满足不同业务对 IO 性能的不同需求。购买方式分为两种，在购买云主机同时购买云硬盘或后期根据需要单独购买云硬盘挂载至云主机上。

可视化管理平台，操作便捷

通过天翼云控制中心对云主机进行管理，可对云主机进行开机、关机、重置密码、重装系统等操作。

规格丰富，满足不同应用场景需求

提供多种实例规格，满足视频解码、图形渲染、深度学习、科学计算等多种场景下不同用户对于 GPU 的性能需求。配备业界超强算力的 GPU 显卡，结合高性能 CPU 平台，单卡最高提供 31.2TFLOPS 单精度和 9.7TFLOPS 双精度计算，同时支持单机多卡，性能翻倍。

提供多种登录方式，安全高效

通过 VNC 方式、SSH 方式（仅适用于 Linux 云主机）和 MSTSC 方式（仅适用于 Windows 云主机）可登录云主机。其中，VNC 方式适用于未绑定弹性 IP 的云主机登录查看。

多种镜像，实现业务的快速部署

支持常用的 Linux、Windows 镜像，如 CentOS、Ubuntu 等。支持将云主机导出私有镜像，并可基于私有镜像创建云主机，实现业务的批量、快速部署。支持将私有镜像共享给其它用户，方便多用户统一部署。

支持多网卡

通过为云主机配置多块网卡，将不同的内网 IP 地址与不同网卡进行绑定，实现同一云主机划分至不同的虚拟云子网以及配置不同的安全组策略。

提供云主机备份、云硬盘备份

支持对云主机的系统盘或数据盘进行备份。基于备份数据，用户可创建新的云主机或恢复磁盘数据。

支持按需重装操作系统

支持主流的 Windows Server2012、2016、2019、centos7.x、8.x 等操作系统，满足用户对不同业务的部署需求。如在安装应用过程中出现问题，还可通过控制台重装当前操作系统。

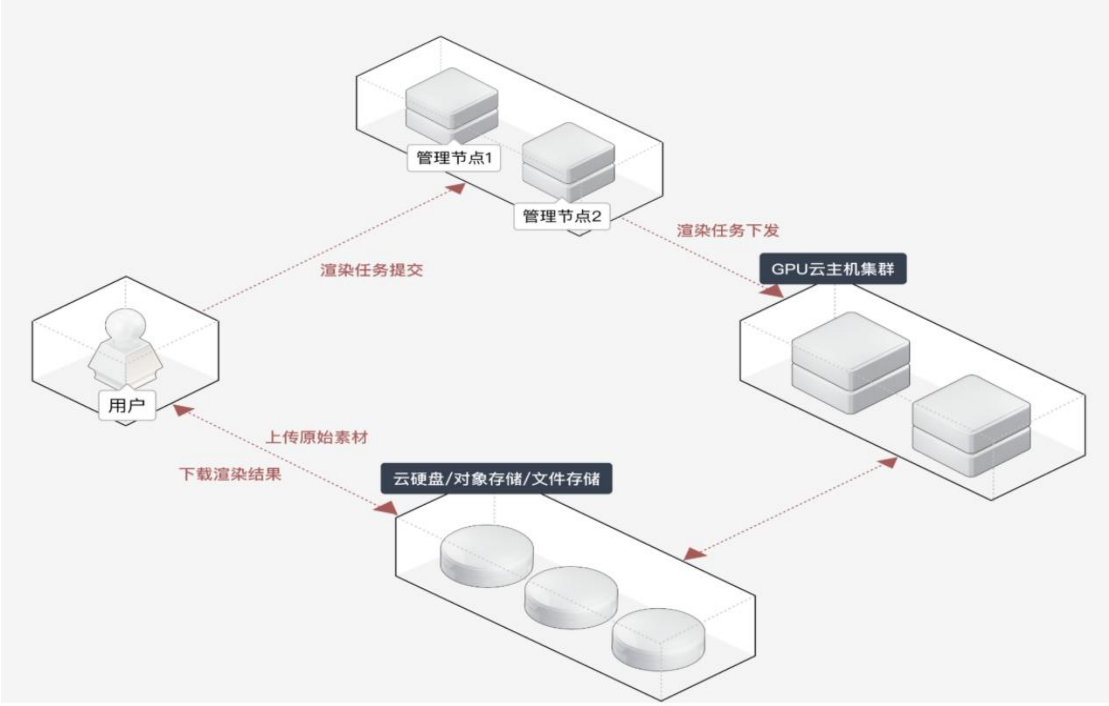
支持多种鉴权方式

在创建云主机时，可选择通过密码和密钥方式登录。如选择密码方式登录，可在创建时设置登录密码，或在创建后通过“重置密码”操作设置密码。

1.4产品应用场景

图形图像渲染

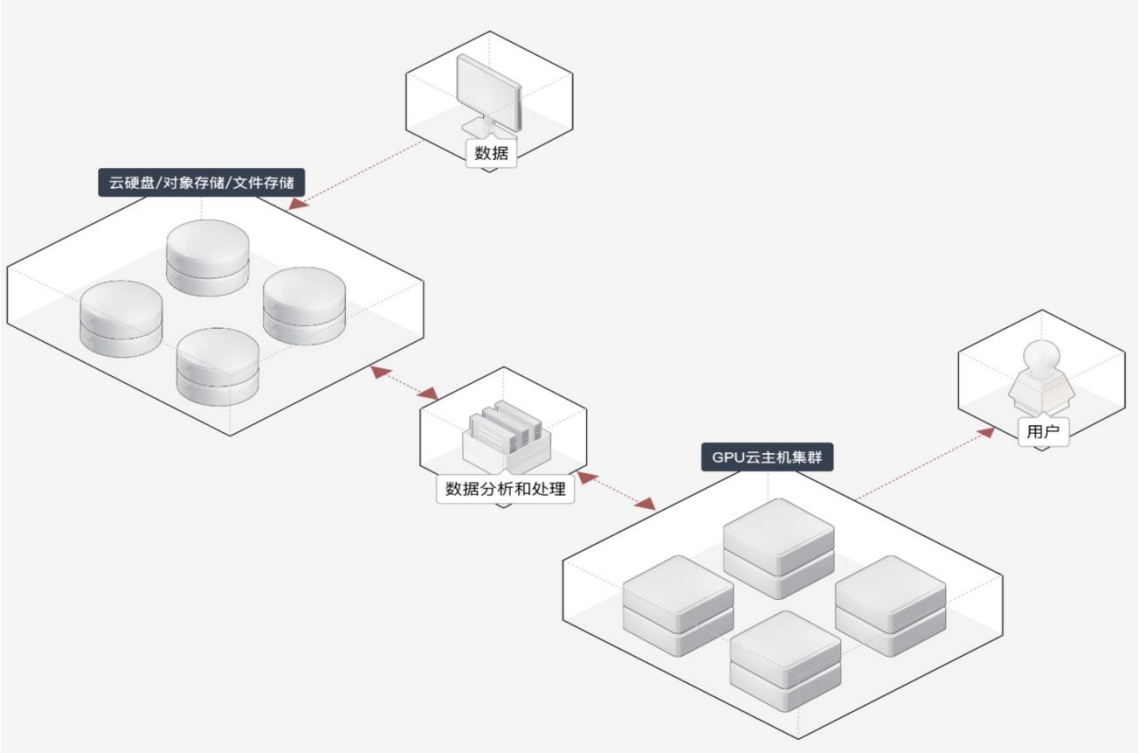
图形图像渲染场景下天翼云 GPU 云主机最高采用业界领先的 GPU A10 显卡，提供 24G 的大显存容量和强大的图形填充速率，支持多种图形加速接口，如 DirectX 12、OpenGL 4.5、Vulkan 1.0 等，配合英伟达官方 vWS Licence 授权，为专业级 CAD、视频渲染、图形处理提供所需的强大计算能力，为虚拟化工作站、桌面和应用程序提供行业内超高的用户性能，结合天翼云的对象存储、弹性云主机以及专线，可以快速构建自己的图像渲染以及分析计算中心。



科学计算

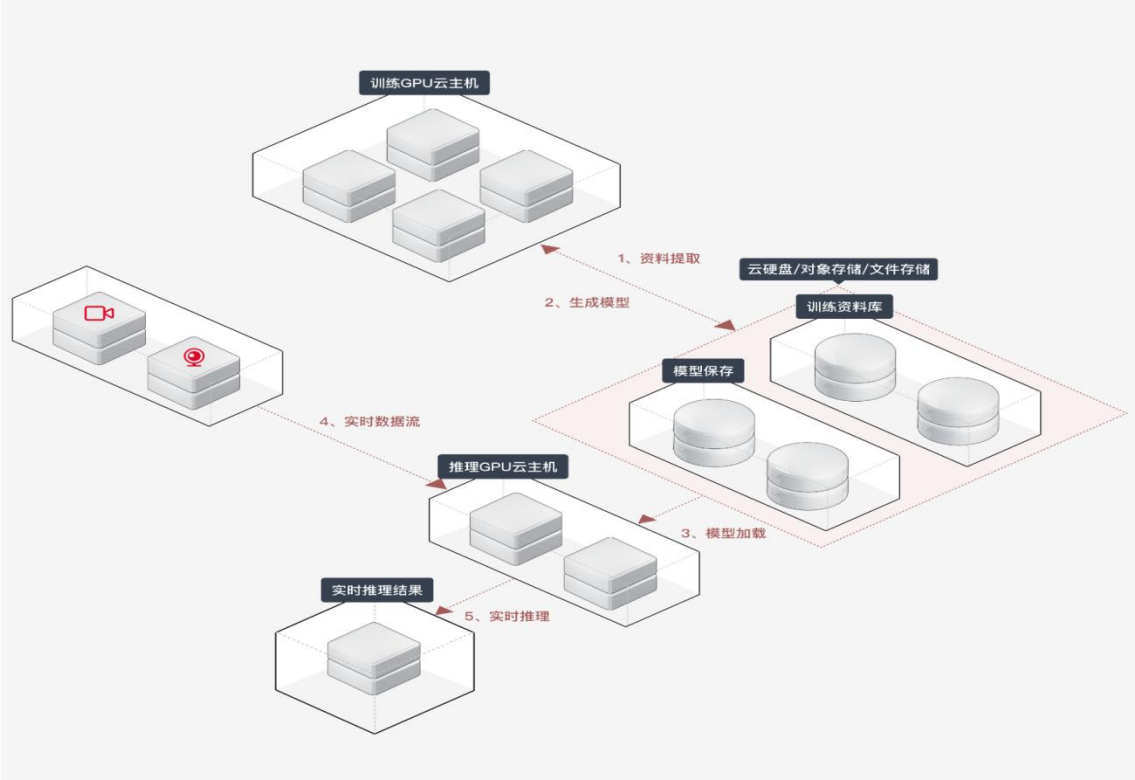
在科学计算领域，模拟仿真过程中，消耗大量计算资源的同时，会产生大量临时数据，对存储带宽与时延也有极高的要求。P 系列计算加速型 GPU 云主机，最高采用业界领先的 GPU 显卡 A100，提供 40GB 的

显存容量和 9.7TFLOPS 双精度计算能力以及大吞吐的带宽。同时支持一机多卡模式，让用户可以在一台云主机上体验多卡的计算能力，达到计算性能翻倍。



AI 深度学习

AI 深度学习有大批量的数据需要不断更新、迭代神经网络中的参数以满足业务对预测精度的要求，对运行稳定性要求更高，对服务器响应延时也有了更高要求。P 系列计算加速型 GPU 云主机，采用业界领先的 GPU 显卡，提供大容量显存和高双精度计算能力以及大吞吐的带宽，其深度学习 TF32 运算能力可到 156 TFLOPS，支持常见的深度学习框架 Tensorflow、Caffe、PyTorch、MXNet 等。配合天翼云弹性云主机、负载均衡、对象存储、关系型数据库 RDS、云监控等服务，可以搭建一个功能完备的深度学习平台，能够快速、高效、低成本的完成训练、推理任务。



1.5产品规格

1.5.1NVIDIA GPU 云主机

使用 Nvidia 显卡的 GPU 云主机分为图形加速基础型（G5、G5s、G6、G7）和计算加速型（P2V、P2Vs、PI2、PI7、P8A）。建议您先了解产品规格、性能和使用限制等信息，然后按照实际需要进行选择。

GPU 计算加速型

GPU 计算加速型 GPU 云主机采用 **GPU 硬件直通技术**，主要适用于 AI 深度学习训练、推理、科学计算、视频转码、图像渲染等场景。

在售：P8A、PI7、P2V、P2Vs、PI2

计算加速型 GPU 云主机特点

规格名称	显卡型号	显卡数量	单卡理论 GPU 性能	磁盘类型

规格名称	显卡型号	显卡数量	单卡理论 GPU 性能	磁盘类型
P8A	Nvidia Tesla A100（40G PCIE）	1、2、4	312 TFLOPS 半精度浮点计算 19.5 TFLOPS 单精度浮点计算 9.7 TFLOPS 双精度浮点计算 156 TFLOPS TF32AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO
PI7	Nvidia Tesla A10	1、2、4	125 TFLOPS 半精度浮点计算 31.2 TFLOPS 单精度浮点计算 62.5 TFLOPS TF32AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO
P2V	Nvidia Tesla V100	1、2、4	14 TFLOPS 单精度浮点计算 7 TFLOPS 双精度浮点计算 112 TFLOPS TF32 AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO
P2Vs	Nvidia Tesla V100s	1、2、4	16.4 TFLOPS 单精度浮点计算 8.2 TFLOPS 双精度浮点计算 130 TFLOPS TF32 AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO

规格名称	显卡型号	显卡数量	单卡理论 GPU 性能	磁盘类型
PI2	Nvidia Tesla T4	1、2、4	65 TFLOPS 半精度浮点计算 8.1 TFLOPS 单精度浮点计算 130 TOPS INT8 计算 260 TOPS INT4 计算	普通 IO 高 IO 通用型 SSD 超高 IO

P8A 型云主机

P8A 型云主机采用 NVIDIA A100 40GB PCIE GPU，采用 GPU 直通技术，使用第三代*英特尔*® 至强® 可扩展处理器（主频 2.6GHz），独享宿主机的 CPU 资源，实例间无 CPU 争抢，没有进行资源超配，在提供云主机灵活性的同时，提供高性能计算能力和优秀的性价比，单卡能够提供最大 312TFLOPS 的半精度浮点运算能力和 9.746 TFLOPS 的双精度浮点运算能力。P8A 型云主机能够提供超高的通用计算能力，适用于 AI 深度学习、科学计算，在深度学习训练、科学计算、计算流体动力学、计算金融、地震分析、分子建模、基因组学等领域都能表现出巨大的计算优势。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型	最大带宽（Gbps）/ 基准带宽（Gbps）	网卡多队列数	最大收发包能力（万 PPS）
p8a.6xlarge.4	24	96	1×A100	1×40GB	KVM	30/11	8	300

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型	最大带宽（Gbps）/基准带宽（Gbps）	网卡多队列数	最大收发包能力（万PPS）
p8a.12xlarge.4	48	192	2×A100	2×40GB	KVM	36/23	16	600
p8a.24xlarge.4	96	384	4×A100	4×40GB	KVM	47/45	32	1000

常规软件支持列表

P8A 型云主机主要用于计算加速场景，例如深度学习训练、推理、科学计算、分子建模、地震分析等场景。

应用软件如果使用到 GPU 的 CUDA 并行计算能力，可以使用 P8A 型云主机。常用的软件支持列表如下：

- Tensorflow、Caffe、PyTorch、MXNet 等常用深度学习框架。

使用须知

P8A 型云主机当前支持如下类型的操作系统：

- Windows Server
- CentOS
- Ubuntu
- Ctyunos

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

PI7 型云主机

PI7 型云主机采用专为 AI 推理打造的 NVIDIA A10 Tensor Core GPU，采用 GPU 直通技术，使用第三代*英特尔*® 至强® 可扩展处理器（主频 2.6GHz）， 独享宿主机的 CPU 资源，实例间无 CPU 争抢，没有

进行资源超配，能够提供超强的实时推理能力，同时也具备图像渲染能力。PI7 型弹性云主机借助 A10，单卡能够提供最大 31.2 TFLOPS 的 FP32 算力。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型	最大带宽（Gbps）/基准带宽(Gbps)	网卡多队列数	最大收发包能力（万PPS）
pi7.4xlarge.4	16	64	1×A10	1×24GB	KVM	17/7.5	8	200
pi7.8xlarge.4	32	128	2×A10	2×24GB	KVM	25/15	16	400
pi7.16xlarge.4	64	256	4×A10	4×24GB	KVM	47/45	32	800

常规支持软件列表

PI7 型主要用于 GPU 推理计算场景，例如图片识别、语音识别、自然语言处理等场景。也可以支持轻量级训练场景和视频编解码场景。

常用的软件支持列表如下：

- Tensorflow、Caffe、PyTorch、MXNet 等深度学习框架。
- RedShift for Autodesk 3dsMax、V-Ray for 3ds Max 等支持 CUDA 的 GPU 渲染。

使用须知

PI7 型云主机当前支持如下类型的操作系统：

- Windows Server
- CentOS
- Ubuntu
- Ctyunos

P2V 型云主机

P2V 型云主机采用 NVIDIA Tesla V100 PCIE GPU，采用 GPU 直通技术，在提供云主机灵活性的同时，提供高性能计算能力和优秀的性价比。P2V 型云主机能够提供超高的通用计算能力，适用于 AI 深度学习、科学计算，在深度学习训练、科学计算、计算流体动力学、计算金融、地震分析、分子建模、基因组学等领域都能表现出巨大的计算优势。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
p2v.4xlarge.8	16	128	1*V100	1*32GB	KVM
p2v.8xlarge.8	32	256	2*V100	2*32GB	KVM
p2v.2xlarge.4	8	32	1*V100	1*32GB	KVM
p2v.4xlarge.4	16	64	2*V100	2*32GB	KVM
p2v.8xlarge.4	32	128	4*V100	4*32GB	KVM

常规软件支持列表

P2V 型云主机主要用于计算加速场景，例如深度学习训练、推理、科学计算、分子建模、地震分析等场景。应用软件如果使用到 GPU 的 CUDA 并行计算能力，可以使用 P2V 型云主机。常用的软件支持列表如下：

- Tensorflow、Caffe、PyTorch、MXNet 等常用深度学习框架。
- RedShift for Autodesk 3dsMax、V-Ray for 3ds Max 等支持 CUDA 的 GPU 渲染。

使用须知

P2V 型云主机当前支持如下类型的操作系统：

- Windows Server

- CentOS
- Ubuntu
- Ctyunos

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

P2Vs 型云主机

P2Vs 型云主机采用 NVIDIA Tesla V100s PCIE GPU，采用 GPU 直通技术，在提供云主机灵活性的同时，提供高性能计算能力和优秀的性价比。P2Vs 型云主机能够提供超高的通用计算能力，适用于 AI 深度学习、科学计算，在深度学习训练、科学计算、计算流体动力学、计算金融、地震分析、分子建模、基因组学等领域都能表现出巨大的计算优势。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
p2vs.4xlarge.8	16	128	1*V100s	1*32GB	KVM
p2vs.8xlarge.8	32	256	2*V100s	2*32GB	KVM
p2vs.2xlarge.4	8	32	1*V100s	1*32GB	KVM
p2vs.4xlarge.4	16	64	2*V100s	2*32GB	KVM
p2vs.8xlarge.4	32	128	4*V100s	4*32GB	KVM

常规软件支持列表

P2Vs 型云主机主要用于计算加速场景，例如深度学习训练、推理、科学计算、分子建模、地震分析等场景。应用软件如果使用到 GPU 的 CUDA 并行计算能力，可以使用 P2Vs 型云主机。常用的软件支持列表如下：

- Tensorflow、Caffe、PyTorch、MXNet 等常用深度学习框架。

- RedShift for Autodesk 3dsMax、V-Ray for 3ds Max 等支持 CUDA 的 GPU 渲染。

使用须知

P2Vs 型云主机当前支持如下类型的操作系统：

- Windows Server
- CentOS
- Ubuntu
- Ctyunos

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

PI2 型云主机

PI2 型云主机采用专为 AI 推理打造的 NVIDIA Tesla T4 GPU，采用 GPU 直通技术，能够提供超强的实时推理能力。PI2 型弹性云主机借助 T4 的 INT8 运算器，能够提供最大 130 TOPS 的 INT8 算力。PI2 也可以支持轻量级训练场景。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
pi2.2xlarge.4	8	32	1×T4	1×16GB	KVM
pi2.4xlarge.4	16	64	2×T4	2×16GB	KVM
pi2.8xlarge.4	32	128	4×T4	4×16GB	KVM

常规支持软件列表

PI2 型云主机主要用于 GPU 推理计算场景，例如图片识别、语音识别、自然语言处理等场景。也可以支持轻量级训练场景。

常用的软件支持列表如下：

- Tensorflow、Caffe、PyTorch、MXNet 等常用深度学习框架。
- RedShift for Autodesk 3dsMax、V-Ray for 3ds Max 等支持 CUDA 的 GPU 渲染。

使用须知

PI2 型云主机当前支持如下类型的操作系统：

- Windows Server
- CentOS
- Ubuntu
- Ctyunos

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

GPU 图像加速基础型

图像加速基础型 GPU 云主机基于 **NVIDIA GRID 虚拟化** GPU 技术，公共镜像集成 GRID 驱动，并包含 NVIDIA GRID vWS 的软件 License，能够有效降低小规模需求的使用成本，同时主机共享宿主机的 CPU 资源，适用于图像渲染和小规模 AI 推理等场景。

在售：G7、G6、G5、G5s

图像加速基础型 GPU 云主机特点

规格名称	显卡型号	显卡数量	单卡 GPU 性能	磁盘类型
G7	Nvidia Tesla A10	1/4、1/2、1	31.2 TFLOPS 单精度浮点计算 62.5 TFLOPS TF32AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO
G6	Nvidia Tesla T4	1/4、1/2	31.2 TFLOPS 单精度浮点计算 62.5 TFLOPS TF32AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO

规格名称	显卡型号	显卡数量	单卡 GPU 性能	磁盘类型
G5	Nvidia Tesla V100	1/16、1/8、1/4、1/2	14 TFLOPS 单精度浮点计算 7 TFLOPS 双精度浮点计算 112 TFLOPS TF32 AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO
G5s	Nvidia Tesla V100s	1/16、1/8、1/4、1/2	16.4 TFLOPS 单精度浮点计算 8.2 TFLOPS 双精度浮点计算 130 TFLOPS TF32 AI 加速	普通 IO 高 IO 通用型 SSD 超高 IO

G7 型云主机

G7 型云主机基于 NVIDIA GRID 虚拟化 GPU 技术，采用第三代*英特尔*® 至强® 可扩展处理器（主频 3.0GHz）能够提供全面的专业级的图形加速能力。G7 型云主机使用 NVIDIA A10 Tensor Core GPU 显卡，能够支持 DirectX、OpenGL、Vulkan 接口，提供 6/12/24 GB 三种显存规格，理论性能 Pixe Rate：162.7Pixel/s，Texture Rate：488.2GTexel/s，满足从入门级到专业级的图形处理需求。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型	最大带宽（Gbps）/基准带宽（Gbps）	网卡多队列数	最大收发包能力（万 PPS）

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型	最大带宽（Gbps）/基准带宽(Gbps)	网卡多队列数	最大收发包能力（万PPS）
g7.2xlarge.4	8	32	A10-6Q	6	KVM	8/2.5	4	110
g7.4xlarge.4	16	64	A10-12Q	12	KVM	15/4.5	8	220
g7.8xlarge.4	32	128	A10-24Q	24	KVM	20/9	16	440

G7 型云主机功能如下：

- 处理器与内存配比为 1:4。
- 支持图形加速接口：
  - DirectX 12, Direct2D, DirectX Video Acceleration (DXVA)
  - OpenGL 4.5
  - Vulkan 1.0
- 支持 CUDA 和 OpenCL。
- 支持 Quadro vDWS 特性，为专业级图形应用提供加速。
- 支持 NVIDIA A10 GPU 卡。
- 支持图形加速应用。
- 提供 GPU 硬件虚拟化（vGPU）。
- 提供和弹性云主机相同的申请流程。
- 自动化的调度 G7 型弹性云主机到装有 NVIDIA A10 GPU 卡的可用区。

- 可以提供最大显存 24GB，分辨率为 7680*4320 的图形图像处理能力。

常规支持软件列表

G7 型云主机主要用于图形加速场景，例如图像渲染、云桌面、3D 可视化。应用软件如果依赖 GPU 的 DirectX、OpenGL 硬件加速能力可以使用 G7 型云主机。常用的图形处理软件支持列表如下：

- AutoCAD
- 3DS MAX
- MAYA
- Agisoft PhotoScan
- ContextCapture

使用须知

G7 型云主机当前支持如下版本的操作系统：

- Windows Server 2019 DataCenter 64bit
- Windows Server 2016 DataCenter 64bit
- Windows Server 2012 DataCenter 64bit
- CentOS 8.1 64bit（目前仅多 AZ 资源池提供）
- CentOS 8.2 64bit（目前仅多 AZ 资源池提供）
- Ubuntu Server 20.04 64bit（目前仅多 AZ 资源池提供）

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

G5 型云主机

G5 型云主机基于 NVIDIA GRID 虚拟化 GPU 技术，能够提供全面的、专业级的图形加速能力。G5 型云主机使用 NVIDIA Tesla V100 PCIE GPU 显卡，能够支持 DirectX、OpenGL、Vulkan 接口，提供 2/4/8/16 GB 四种显存规格，理论性能 Pixe Rate：176.6GPixel/s，Texture Rate：441.6GTexel/s，满足从入门级到专业级的图形处理需求。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
g5.2xlarge.2.1	8	16	V100-2Q	2	KVM

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
g5.2xlarge.2	8	32	V100-4Q	4	KVM
g5.2xlarge.8	8	64	V100-16Q	16	KVM
g5.4xlarge.4	16	64	V100-8Q	8	KVM
g5.8xlarge.4	32	128	V100-16Q	16	KVM

G5 型云主机功能如下：

- 处理器与内存配比为 1:4/1:2/1:8。
- 支持图形加速接口：
  - DirectX 12, Direct2D, DirectX Video Acceleration (DXVA)
  - OpenGL 4.5
  - Vulkan 1.0
- 支持 CUDA 和 OpenCL。
- 支持 Quadro vDWS 特性，为专业级图形应用提供加速。
- 支持 NVIDIA V100 GPU 卡。
- 支持图形加速应用。
- 提供 GPU 硬件虚拟化（vGPU）。
- 提供和弹性云主机相同的申请流程。
- 自动化的调度 G5 型弹性云主机到装有 NVIDIA V100 GPU 卡的可用区。
- 可以提供最大显存 16GB，分辨率为 4096×2160 的图形图像处理能力。

常规支持软件列表

G5 型云主机主要用于图形加速场景，例如图像渲染、云桌面、3D 可视化。应用软件如果依赖 GPU 的 DirectX、OpenGL 硬件加速能力可以使用 G5 型云主机。常用的图形处理软件支持列表如下：

- AutoCAD
- 3DS MAX
- MAYA
- Agisoft PhotoScan
- ContextCapture

使用须知

G5 型云主机当前支持如下版本的操作系统：

- Windows Server 2016 Standard 64bit
- Windows Server 2012 Standard 64bit
- CentOS 7.5 64bit
- CentOS 7.6 64bit
- Ubuntu Server 16.04 64bit

G5s 型云主机

G5s 型云主机基于 NVIDIA GRID 虚拟化 GPU 技术，能够提供全面的专业级的图形加速能力。G5s 型云主机使用 NVIDIA Tesla V100s PCIE GPU 显卡，能够支持 DirectX、OpenGL、Vulkan 接口，提供 2/4/8/16 GB 四种显存规格，理论性能 Pixe Rate：204.4GPixel/s，Texture Rate：511.0GTexel/s，满足从入门级到专业级的图形处理需求。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
g5s.2xlarge.2.1	8	16	V100s-2Q	2	KVM
g5s.2xlarge.2	8	32	V100s-4Q	4	KVM
g5s.2xlarge.8	8	64	V100s-16Q	16	KVM
g5s.4xlarge.4	16	64	V100s-8Q	8	KVM

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
g5s.8xlarge.4	32	128	V100s-16Q	16	KVM

G5s 型云主机功能如下：

- 处理器与内存配比为 1:4/1:2/1:8。
- 支持图形加速接口：
  - DirectX 12, Direct2D, DirectX Video Acceleration (DXVA)
  - OpenGL 4.5
  - Vulkan 1.0
- 支持 CUDA 和 OpenCL。
- 支持 Quadro vDWS 特性，为专业级图形应用提供加速。
- 支持 NVIDIA V100s GPU 卡。
- 支持图形加速应用。
- 提供 GPU 硬件虚拟化（vGPU）。
- 提供和弹性云主机相同的申请流程。
- 自动化的调度 G5s 型弹性云主机到装有 NVIDIA V100s GPU 卡的可用区。
- 可以提供最大显存 16GB，分辨率为 4096×2160 的图形图像处理能力。

常规支持软件列表

G5s 型云主机主要用于图形加速场景，例如图像渲染、云桌面、3D 可视化。应用软件如果依赖 GPU 的 DirectX、OpenGL 硬件加速能力可以使用 G5s 型云主机。常用的图形处理软件支持列表如下：

- AutoCAD
- 3DS MAX
- MAYA
- Agisoft PhotoScan
- ContextCapture

使用须知

G5s 型云主机当前支持如下版本的操作系统：

- Windows Server 2016 Standard 64bit

- Windows Server 2012 Standard 64bit
- CentOS 7.5 64bit
- CentOS 7.6 64bit
- Ubuntu Server 16.04 64bit

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

G6 型云主机

G6 型云主机基于 NVIDIA GRID 虚拟化 GPU 技术，使用 NVIDIA Tesla T4 GPU 显卡，能够支持 DirectX、OpenGL、Vulkan 接口，提供 4/8 GB 两种显存，理论性能 Pixe Rate：101.8GPixel/s，Texture Rate：254.4GTexel/s，满足从入门级到专业级的图形处理需求。

规格名称	vCPU	内存（GB）	GPU	显存（GB）	虚拟化类型
g6.xlarge.4	4	16	T4-4Q	4	KVM
g6.2xlarge.4	8	32	T4-8Q	8	KVM

常规支持软件列表

G6 型云主机主要用于图形加速场景，例如图像渲染、云桌面、3D 可视化。应用软件如果依赖 GPU 的 DirectX、OpenGL 硬件加速能力可以使用 G6 型云主机。常用的图形处理软件支持列表如下：

- AutoCAD
- 3DS MAX
- MAYA
- Agisoft PhotoScan
- ContextCapture

使用须知

G6 型云主机当前支持如下版本的操作系统：

- Windows Server 2016 Standard 64bit
- Windows Server 2012 Standard 64bit
- CentOS 7.5 64bit
- CentOS 7.6 64bit
- Ubuntu Server 16.04 64bit

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

1.5.2 国产计算加速型云主机

1.5.2.1 昇腾计算加速型云主机

PAK1 型云主机

PAK1 型昇腾计算加速型云主机采用专为 AI 推理打造的 Atlas 300I pro 加速卡，国产 ARM 架构鲲鹏 CPU，独享宿主机的 CPU 资源，实例间无 CPU 争抢，没有进行资源超配，属于计算加速型（直通）规格，云主机的 CPU/内存配比为 1：4，主要适用于搜索推荐、内容审核和 OCR 系统等推理场景。

规格特点

规格名称	CPU 型号	显卡型号	显卡数量	单卡 GPU 性能	磁盘类型
PAK1	Kunpeng 920 5250(主频 2.6GHz)	Atlas 300I pro	1、2、4	70 TFLOPS 半精度浮点计算 140 TOPS INT8 计算	普通 IO 高 IO 通用型 SSD 超高 IO

规格

系列	CPU	内存	GPU 显卡类型	显存	最大带宽/基准带宽（Gbit/s）	网络收发包（万 PPS）	多队列
pak1.4xlarge.4	18	72	Huawei Atlas 300I pro	1*24G	18/11.5	240	8

系列	CPU	内存	GPU 显卡类型	显存	最大带宽/基准带宽（Gbit/s）	网络收发包（万 PPS）	多队列
pak1.9xlarge.4	36	144	Huawei Atlas 300I pro	2*24G	30/22.5	480	16
pak1.18xlarge.4	72	288	Huawei Atlas 300I pro	4*24G	47/45	960	32

支持的镜像

- CTyunOS 2.0.1 64 位 ARM 版
- 银河麒麟高级服务器操作系统 V10 SP1 64 位 ARM 版
- 统信服务器操作系统 V20 64 位 ARM 版

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

1.5.2.2 寒武纪计算加速型云主机

PCH1 型云主机

PCH1 型寒武纪计算加速型云主机采用专为 AI 推理打造的 MLU370-S4 加速卡，国产 X86 架构海光 CPU，独享宿主机的 CPU 资源，实例间无 CPU 争抢，没有进行资源超配，属于计算加速型（直通）规格，云主机的 CPU/内存配比为 1：4，可广泛支持视觉、语音、自然语言处理等高度多样化的人工智能应用，帮助 AI 推理平台实现超高密度。

规格特点

规格名称	CPU 型号	显卡型号	显卡数量	单卡 GPU 性能	磁盘类型
PCH1	海光 7285(主	MLU370-S4	1、2、	72 TFLOPS 半精度浮点	普通 IO 高 IO

规格名称	CPU 型号	显卡型号	显卡数量	单卡 GPU 性能	磁盘类型
	频 2.0GHz)		3、4	计算 192 TOPS INT8 计算	通用型 SSD 超高 IO

规格

系列	CPU	内存	GPU 显卡类型	显存	最大带宽/基准带宽 (Gbit/s)	网络收发包（万 PPS)	多队列
pch1.4xlarge.4	16	64	Cambricon MLU370 s4	1*24G	18/11.5	200	8
pch1.6xlarge.4	24	96	Cambricon MLU370 s4	1*24G	18/11.5	200	8
pch1.9xlarge.4	36	144	Cambricon MLU370 s4	2*24G	30/22.5	400	16
pch1.12xlarge.4	48	192	Cambricon MLU370 s4	3*24G	30/22.5	400	16
pch1.21xlarge.3	84	252	Cambricon MLU370 s4	4*24G	47/45	800	32

支持的镜像

- CTyunOS 2.0.1 64 位 ARM 版
- CentOS 7.8 64 位

- KylinOS V10 SP1 64 位

备注：各资源池支持的具体版本可能略有出入，请以控制台实际支持的版本为准

1.6使用限制

GPU 云主机作为弹性云主机的一类实例规格，保持了与弹性云主机实例相同的使用限制。

GPU 云主机在产品功能和服务性能上的不同限制，以及如何申请更高配额，请参考弹性云主机 >产品概述 >产品使用限制。

1.7产品地域和可用区

GPU 云主机的产品地域和可用区请参考弹性云主机 >产品概述 >产品地域和可用区。

1.8基本概念

概念	介绍
GPU	图形处理器（Graphics Processing Unit）。相比 CPU 具有众多计算单元和更多的流水线，适合用于大规模并行计算等场景。
CUDA	NVIDIA 推出的通用并行计算架构，帮助您使用 NVIDIA GPU 解决复杂的计算问题。
cuDNN	NVIDIA 推出的用于深度神经网络的 GPU 加速库。
镜像	提供了运行实例所需的信息，包括操作系统、初始化应用数据等。

概念	介绍
地域和可用区	实例和其他资源的部署物理位置。
SSH 密钥对	一种安全便捷的登录认证方式，由公钥和私钥组成，仅支持 Linux 实例。
安全组	一种虚拟防火墙，您可以基于安全组控制实例的入流量和出流量。
弹性网卡	一种独立的虚拟网卡，可以绑定到弹性云主机或从弹性云主机解绑，实现业务的灵活扩展和迁移。
云硬盘	可弹性扩展的块存储设备，可以用作实例的系统盘或可扩展数据盘使用。
快照	某一时间点云盘数据状态的备份文件，用于备份或者恢复整个云盘。
VPC	虚拟私有云（Virtual Private Cloud）。为云服务器、云容器、云数据库等云上资源构建隔离、私密的虚拟网络环境。VPC 丰富的功能帮助用户灵活管理云上网络，包括创建子网、设置安全组和网络 ACL、管理路由表、申请弹性公网 IP 和带宽等。

2 计费说明

2.1包周期计费模式

收费方式

包年包月付费指按订单的购买周期计费，是一种预付费模式，即先付费再使用。

收费项

GPU 云主机收费项：CPU、内存、GPU 类型和显存。

云硬盘收费项：存储容量、存储类型。

公网带宽收费项：带宽大小。

续订规则

参照弹性云主机-购买指南-续费说明-规则说明。

退订规则

参照弹性云主机-购买指南-退费说明-规则说明。

2.2按量计费模式

收费方式

一种后付费模式，即先使用再付费。

注：自 2021 年 4 月 1 日 4:00 起，按量 GPU 云主机、按量云硬盘、按量独享带宽将计费单元由小时调整为秒。

例如您在 13:30:30 创建了一台按量付费 GPU 云主机，相关资源包括计算资源（vCPU、内存、显存）、镜像和云盘（系统盘），然后在 13:55:30 释放实例，则：结算周期为 13:00:00～14:00:00，在 13:30:30～13:55:30 间产生计费，该结算周期内的计费时长为 1500 秒。期间费用=实例小时价格/3600*1500。

收费项

GPU 云主机收费项：CPU、内存、显卡类型和显存

云硬盘收费项：存储容量、存储类型

公网带宽收费项：带宽大小

关机规则

在 GPU 云主机开通期间，如果用户主动执行了关机操作，则在云主机关机状态下不会收取 CPU、内存、显存的费用，其他关联资源继续计费。

删除规则

删除 GPU 云主机时，云主机的 CPU、内存、显存的费用停止计费，其他被保留的关联资源继续计费。

GPU 云主机删除后，数据不会保留，会立即释放资源。

账户欠费

如用户账户出现欠费，账户一旦充值，系统将会自动优先扣除欠费金额。

2.3价格总览

图形加速基础型 G5

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格(元/月)
g5.2xlarge.2	8	32	4	1/8	V100	5.48	2632
g5.4xlarge.4	16	64	8	1/4	V100	10.97	5263
g5.2xlarge.2.1	8	16	2	1/16	V100	2.74	1316
g5.2xlarge.8	8	64	16	1/2	V100	16.44	7890
g5.8xlarge.4	32	128	16	1/2	V100	21.93	10527

图形加速基础型 G5s

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
g5s.2xlarge.2	8	32	4	1/8	V100S	5.48	2632
g5s.4xlarge.4	16	64	8	1/4	V100S	10.97	5263
g5s.2xlarge.2.1	8	16	2	1/16	V100S	2.74	1316
g5s.2xlarge.8	8	64	16	1/2	V100S	16.44	7890
g5s.8xlarge.4	32	128	16	1/2	V100S	21.93	10527

图形加速基础型 G6

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
g6.xlarge.4	4	16	4	1/4	T4	2.445	1173.65

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
g6.2xlarge.4	8	32	8	1/2	T4	5.053	2425.56

图形加速基础型 G7

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需(元/小时)	价格（元/月）
g7.2xlarge.4	8	32	6	1/4	A10	4.24	2033.34
g7.4xlarge.4	16	64	12	1/2	A10	8.48	4066.68
g7.8xlarge.4	32	128	24	1	A10	16.96	8133.36

计算加速型 P2V

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
p2v.4xlarge.8	16	128	32	1	V100	32.87	15780

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
p2v.8xlarge.8	32	256	64	2	V100	65.75	31559
p2v.2xlarge.4	8	32	32	1	V100	15.37	7377.8
p2v.4xlarge.4	16	64	64	2	V100	30.74	14755.6
p2v.8xlarge.4	32	128	128	4	V100	61.48	29511.2

计算加速型 P2Vs

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
p2vs.4xlarge.8	16	128	32	1	V100S	32.87	15780
p2vs.8xlarge.8	32	256	64	2	V100S	65.75	31559
p2vs.2xlarge.4	8	32	32	1	V100S	15.37	7377.8
p2vs.4xlarge.4	16	64	64	2	V100S	30.74	14755.6

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
p2vs.8xlarge.4	32	128	128	4	V100S	61.48	29511.2

计算加速型 PI2

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
pi2.2xlarge.4	8	32	16	1	T4	7.32	3515
pi2.4xlarge.4	16	64	32	2	T4	14.65	7030
pi2.8xlarge.4	32	128	64	4	T4	29.30	14060

计算加速型 PI7

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格（元/月）
pi7.4xlarge.4	16	64	24	1	A10	9.27	4447.43
pi7.8xlarge.4	32	128	48	2	A10	18.53	8894.85
pi7.16xlarge.4	64	256	96	4	A10	37.05	17789.69

计算加速型 P8A

规格名称	vCPU	内存(GB)	显存(GB)	显卡数	显卡类型	按需（元/小时）	价格(元/月)
p8a.6xlarge.4	24	96	40	1	A100	21.3	10229.09
p8a.12xlarge.4	48	192	80	2	A100	42.6	20458.17
p8a.24xlarge.4	96	384	160	4	A100	85.2	40916.34

计算加速型 PAK1

规格名称	vCPU	内存（GB）	显存（GB）	显卡数	显卡类型	按需（元/小时）	价格（元/月）
pak1.4xlarge.4	18	72	24	1	Atlas 300i pro	10.69	5133.49
pak1.9xlarge.4	36	144	48	2	Atlas 300i pro	21.39	10266.97
pak1.18xlarge.4	72	288	96	4	Atlas 300i pro	42.78	20533.95

计算加速型 PCH1

规格名称	vCPU	内存（GB）	显存（GB）	显卡数	显卡类型	按需（元/小时）	价格（元/月）
pch1.4xlarge.4	16	64	24	1	Cambricon MLU370 s4	13.00	6241.02
pch1.6xlarge.4	24	96	24	1	Cambricon MLU370 s4	14.45	6934.46
pch1.9xlarge.4	36	144	48	2	Cambricon MLU370 s4	26.00	12482.04

规格名称	vCPU	内存（GB）	显存（GB）	显卡数	显卡类型	按需（元/小时）	价格（元/月）
pch1.12xlarge.4	48	192	72	3	Cambricon MLU370 s4	39.01	18723.05
pch1.21xlarge.3	84	252	96	4	Cambricon MLU370 s4	52.01	24964.07

3 用户指南

3.1 常用操作导航

使用限制

- 使用 GPU 云主机的注意事项，请参见使用须知。
- 使用 GPU 云主机的资源规则限制，请参见使用限制。

创建并管理 GPU 云主机

- 您可以按以下操作来管理 GPU 云主机的生命周期：
  - 创建 GPU 云主机
  - 连接 GPU 云主机
  - 停止 GPU 云主机
  - 释放 GPU 云主机
- 如果当前的 GPU 云主机规格或网络配置无法满足业务需求，您可以对 GPU 云主机进行变配。

管理计费

- 包年包月云主机可以以下方式续费：
  - 自动续费
  - 手动续费
- 转换云主机计费方式：
  - 包周期按量互转

创建并管理云硬盘

当云硬盘作数据盘用时，您可以按以下步骤使用云硬盘：

1.创建云硬盘。

2.挂载云硬盘。

3.初始化数据盘。

4.创建快照备份数据。具体操作，请参见创建云盘快照。

5.如果云硬盘容量无法满足需求，您可以扩容云硬盘。

6.如果云硬盘数据出错，您可以使用某个时刻的云硬盘快照回滚云硬盘。请参见快照回滚。

7.卸载数据盘。

创建和管理云硬盘快照

您可以按以下步骤使用云硬盘快照：

1.创建快照，支持手动创建快照和自动创建快照：

- 创建云硬盘快照。
- 使用自动快照策略，定期自动创建快照。具体操作，请参见启用或停用自动快照策略。

2.查看快照使用容量。

快照的常见应用场景如下：

- 您可以使用快照回滚云盘恢复数据，请参见快照回滚。
- 您可以通过快照创建多个具有相同数据的云硬盘，用于快速部署业务，请参见从快照创建云硬盘。

3.2注册账号

操作步骤

请参见账号中心 >操作指南 >注册天翼云账号。

3.3创建 GPU 云主机

3.3.1 创建未配备驱动的 GPU 云主机

准备工作

创建账号，以及完善账号信息。本教程创建的是按量付费实例。开通按量付费 ECS 资源时，您的天翼云账户余额（即现金余额）不得小于 100.00 元人民币。充值方式请参见费用中心 >账户充值。

可选：在创建弹性云主机时，如果您的账号在本地域没有创建 VPC，天翼云会提供一个默认的 VPC，如果您不想使用默认 vpc，可以在本地域创建 vpc。具体操作，请参见虚拟私有云 >创建 VPC、子网搭建私有网络。

操作步骤

步骤 1：进入创建云主机页面

1.点击天翼云门户首页的“控制中心”，输入登录的用户名和密码，进入控制中心页面。

2.单击“服务列表>弹性云主机”。

3.单击“创建云主机”，系统进入创建页。

步骤 2：基础配置

1.选择“计费模式”。

- 包年包月：一种预付费模式，即先付费再使用。一般适用于固定业务应用，例如网站服务。需要先支付包年包月资源账单，才能开始使用包年包月资源。
- 按量付费：一种后付费模式，即先使用再付费。一般适用于有业务变化的应用，例如临时扩展、临时测试。可以先开通并使用按量付费资源，系统在每个结算周期生成账单并从账户中扣除相应费用。

2.选择“地域和可用区”，此处我们默认华东-华东 1。

3.设置“实例名称”，长度为 2~63 个字符。

4.设置“主机名称”，长度为 2～15 个字符，允许使用大小写字母、数字或连字符（-）。不能以连字符（-）开头或结尾，不能连续使用连字符（-），也不能仅使用数字。

5.选择“CPU 架构”。

6.设置“规格”，选择“分类”为“GPU 计算加速型”。

7.镜像类型选择“公共镜像”，选择所需的操作系统及版本，不勾选“安装 GPU 驱动”的复选框

8.设置“存储”。 磁盘包括系统盘和数据盘。您可以为云主机添加多块数据盘，系统盘大小目前默认为 40GB。

步骤 3：网络配置

设置网络，包括“虚拟私有云”、“安全组”、“网卡”等信息。

参数	说明
虚拟私有云	云主机网络使用虚拟私有云（VPC）提供的网络，包括子网、安全组等。您可以选择使用已有的虚拟私有云网络，或者单击“前往控制台创建”来创建新的虚拟私有云。
安全组	安全组用来实现安全组内和安全组间云主机的访问控制，加强云主机的安全保护。用户可以在安全组中定义各种访问规则，当云主机加入该安全组后，即受到这些访问规则的保护。创建云主机时，可支持选择多个安全组。此时，云主机的访问规则遵循几个安全组规则的并集。 注意：如需通过远程桌面连接到 Windows 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP 端口范围：3389 如需通过 ssh 连接到 Linux 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP 端口范围：22 如需 Ping 云主机地址，请在安全组中添加如下规则 方向：入方向 协议：ICMP 类型：Any
网卡	添加一张主网卡，可使用已有内网 IP 地址，或自动分配。
弹性公网 IP	将云主机与弹性 IP 绑定，使云主机通过固定的公网 IP 地址与互联网互通。 您可以根据实际情况选择以下三种方式： 不使用：云主机不能直接与互联网互通，仅可作为私有网络中部署业务或者集群所需云主机进行使

参数	说明
	用。 自动分配：自动为每台云主机分配独享带宽的弹性 IP，带宽值可以由您设定。 使用已有：为云主机分配已有弹性 IP。使用已有弹性 IP 时，不能批量创建云主机。

步骤 4：高级配置

1.设置“登录方式”，并创建对应的密码或密钥对。

- 密钥对：指使用密钥对作为云主机的鉴权方式。您可以选择使用已有的密钥，或者单击“查看密钥对”创建新的密钥。注：如果选择使用已有的密钥，请确保您已在本地获取该文件，否则，将影响您正常登录云主机。
- 密码：指使用设置初始密码方式作为云主机的鉴权方式。此时，您可以通过用户名密码方式登录云主机，Linux 操作系统时为 root 用户的初始密码，Windows 操作系统时为 Administrator 用户的初始密码。密码复杂度需满足：

参数	规则
密码	8～30 个字符，必须同时包含三项（大写字母、小写字母、数字、()`~!@#$%^&*_-+={}[]:;'<>,.?/ 中的特殊符号），且不能以斜线号（/）开头

2.选择“云主机组”。

3.选择“用户数据”配置方式（目前仅部分资源池支持）。

4.单击“下一步，确认配置”。

步骤 5：确认配置

1.在“确认配置”页面，查看云主机配置详情。

2.企业项目：企业项目是对多个资源实例进行归类管理的单位，不同云服务区域的资源和项目可以归到一个企业项目中。企业可以根据不同的部门或项目组，将相关的资源放置在相同的企业项目内进行管理，支持资源在企业项目之间迁移。

3.设置购买量。

- 购买时长：“包年/包月”方式需要设置购买时长，最短为 1 个月，最长为 3 年。
- 自动续订：“包年/包月”方式可选是否开启自动续订。按月购买的自动续订周期为 1 个月，按年购买的自动续订周期为 1 年。
- 购买数量：设置购买弹性云主机的数量。为了保证所有资源的合理分配，如果您需要的弹性云主机数量超过当前您可以购买的最大数值，您要提交工单申请扩大配额。申请通过后，您可以购买到满足您需要的弹性云主机数量。

4.协议：阅读并勾选同意协议。

5.如果您确认配置无误，单击“立即购买”。

6.点击“立即支付”进行付款，付款成功即可创建弹性云主机。弹性云主机创建成功后，您可以在弹性云主机信息页面看到您新创建的弹性云主机。

3.3.2 创建配备 GPU 驱动的 GPU 云主机（Linux）

准备工作

创建账号，以及完善账号信息。本教程创建的是按量付费实例。开通按量付费 GPU 云主机资源时，您的天翼云账户余额（即现金余额）不得小于 100.00 元人民币。充值方式请参见费用中心 >账户充值。

可选：在创建弹性云主机时，如果您的账号在本地域没有创建 VPC，天翼云会提供一个默认的 VPC，如果您不想使用默认 vpc，可以在本地域创建 vpc。具体操作，请参见虚拟私有云 >创建 VPC、子网搭建私有网络。

操作步骤

步骤 1：进入创建云主机页面

1. 点击天翼云门户首页的“控制中心”，输入登录的用户名和密码，进入控制中心页面。

2. 单击“服务列表>弹性云主机”。

3. 单击“创建云主机”，系统进入创建页。

步骤 2：基础配置

1.选择“计费模式”。

- 包年包月：一种预付费模式，即先付费再使用。一般适用于固定业务应用，例如网站服务。需要先支付包年包月资源账单，才能开始使用包年包月资源。
- 按量付费：一种后付费模式，即先使用再付费。一般适用于有业务变化的应用，例如临时扩展、临时测试。可以先开通并使用按量付费资源，系统在每个结算周期生成账单并从账户中扣除相应费用。

2.选择“地域和可用区”。

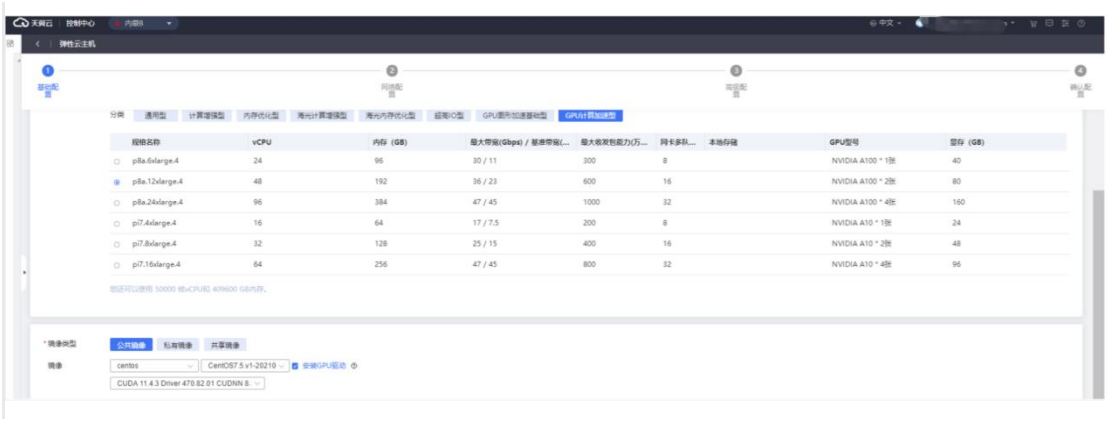
3.设置“实例名称”，长度为 2~63 个字符。

4.设置“主机名称”，长度为 2～15 个字符，允许使用大小写字母、数字或连字符（-）。不能以连字符（-）开头或结尾，不能连续使用连字符（-），也不能仅使用数字。

5.选择“CPU 架构”。

6.设置“规格”，选择“分类”为“GPU 计算加速型”。

7.选择目标的 Linux 的公共镜像，勾选“安装 GPU 驱动”，选择对应的 CUDA、Driver、CUDNN 版本。



8.设置“存储”。 磁盘包括系统盘和数据盘。您可以为云主机添加多块数据盘，系统盘大小目前默认为40GB。

步骤 3：网络配置

设置网络，包括“虚拟私有云”、“安全组”、“网卡”等信息。

参数	说明
虚拟私有云	云主机网络使用虚拟私有云（VPC）提供的网络，包括子网、安全组等。您可以选择使用已有的虚拟私有云网络，或者单击“前往控制台创建”来创建新的虚拟私有云。
安全组	安全组用来实现安全组内和安全组间云主机的访问控制，加强云主机的安全保护。用户可以在安全组中定义各种访问规则，当云主机加入该安全组后，即受到这些访问规则的保护。创建云主机时，可支持选择多个安全组。此时，云主机的访问规则遵循几个安全组规则的并集。 注意：如需通过远程桌面连接到 Windows 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP 端口范围：3389 如需通过 ssh 连接到 Linux 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP

参数	说明
	端口范围：22 如需 Ping 云主机地址，请在安全组中添加如下规则 方向：入方向 协议：ICMP 类型：Any
网卡	添加一张主网卡，可使用已有内网 IP 地址，或自动分配。
弹性 IP	将云主机与弹性 IP 绑定，使云主机通过固定的公网 IP 地址与互联网互通。 您可以根据实际情况选择以下三种方式： 不使用：云主机不能直接与互联网互通，仅可作为私有网络中部署业务或者集群所需云主机进行使用。 自动分配：自动为每台云主机分配独享带宽的弹性 IP，带宽值可以由您设定。 使用已有：为云主机分配已有弹性 IP。使用已有弹性 IP 时，不能批量创建云主机。

步骤 4：高级配置

1.设置“登录方式”，并创建对应的密码或密钥对。

- 密钥对：指使用密钥对作为云主机的鉴权方式。您可以选择使用已有的密钥，或者单击“查看密钥对”创建新的密钥。注：如果选择使用已有的密钥，请确保您已在本地获取该文件，否则，将影响您正常登录云主机。
- 密码：指使用设置初始密码方式作为云主机的鉴权方式。此时，您可以通过用户名密码方式登录云主机，Linux 操作系统时为 root 用户的初始密码，Windows 操作系统时为 Administrator 用户的初始密码。密码复杂度需满足：

参数	规则
密码	8～30 个字符，必须同时包含三项（大写字母、小写字母、数字、()`~!@#$%^&*_-+={}[]:;'<>,.?/中的特殊

2.选择“云主机组”。

3.选择“用户数据”配置方式（目前仅部分资源池支持）。

4.单击“下一步，确认配置”。

步骤 5：确认配置

1.在“确认配置”页面，查看云主机配置详情。

2.企业项目：企业项目是对多个资源实例进行归类管理的单位，不同云服务区域的资源和项目可以归到一个企业项目中。企业可以根据不同的部门或项目组，将相关的资源放置在相同的企业项目内进行管理，支持资源在企业项目之间迁移。

3.设置购买量。

- 购买时长：“包年/包月”方式需要设置购买时长，最短为 1 个月，最长为 3 年。
- 自动续订：“包年/包月”方式可选是否开启自动续订。按月购买的自动续订周期为 1 个月，按年购买的自动续订周期为 1 年。
- 购买数量：设置购买弹性云主机的数量。为了保证所有资源的合理分配，如果您需要的弹性云主机数量超过当前您可以购买的最大数值，您要提交工单申请扩大配额。申请通过后，您可以购买到满足您需要的弹性云主机数量。

4.协议：阅读并勾选同意协议。

5.如果您确认配置无误，单击“立即购买”。

6.点击“立即支付”进行付款，付款成功即可创建弹性云主机。弹性云主机创建成功后，您可以在弹性云主机信息页面看到您新创建的弹性云主机。

3.3.3 创建配备 GRID 驱动的 GPU 云主机（Windows）

创建账号，以及完善账号信息。本教程创建的是按量付费实例。开通按量付费 ECS 资源时，您的天翼云账户余额（即现金余额）不得小于 100.00 元人民币。充值方式请参见充值方式请参见费用中心 > 账户充值。

可选：在创建弹性云主机时，如果您的账号在本地域没有创建 VPC，天翼云会提供一个默认的 VPC，如果您不想使用默认 vpc，可以在本地域创建 vpc。具体操作，请参见虚拟私有云 >创建 VPC、子网搭建私有网络。

操作步骤

步骤 1：进入创建云主机页面

1. 点击天翼云门户首页的“控制中心”，输入登录的用户名和密码，进入控制中心页面。

2. 单击“服务列表>弹性云主机”。

3. 单击“创建云主机”，系统进入创建页。

步骤 2：基础配置

1.选择“计费模式”。

包年包月：一种预付费模式，即先付费再使用。一般适用于固定业务应用，例如网站服务。需要先支付包年包月资源账单，才能开始使用包年包月资源。

按量付费：一种后付费模式，即先使用再付费。一般适用于有业务变化的应用，例如临时扩展、临时测试。可以先开通并使用按量付费资源，系统在每个结算周期生成账单并从账户中扣除相应费用。

2.选择“地域和可用区”，此处我们默认华东-华东 1。

3.设置“实例名称”，长度为 2~63 个字符。

4.设置“主机名称”，长度为 2～15 个字符，允许使用大小写字母、数字或连字符（-）。不能以连字符（-）开头或结尾，不能连续使用连字符（-），也不能仅使用数字。

5.选择“CPU 架构”。

6.设置“规格”和“镜像”

规格	镜像	备注
图形加速基础型 GPU 云主机	镜像类型选择“公共镜像”，选择所需的 Windows 操作系统及版本。公共镜像中默认安装 GRID 驱动及配套 license 授权，无需单独安装。	驱动版本信息请参见 NVIDIA 驱动安装指引-GPU 云主机-用户指南-安装 NVIDIA 驱动 - 天翼云(ctyun.cn)
计算加速型 GPU 云主机 PI7 规格族	镜像类型选择“公共镜像”，选择预装 GRID 驱动的计费镜像：Windows2019-DataCenter-GRID13.2	目前预装GRID驱动的计费镜像价格如下： 包月价格为 220 元/月 按需价格为 0.46 元/小时

7.设置“存储”。 磁盘包括系统盘和数据盘。您可以为云主机添加多块数据盘，系统盘大小目前默认为 40GB。

步骤 3：网络配置

设置网络，包括“虚拟私有云”、“安全组”、“网卡”等信息。

参数	**说明**
虚拟私有云	云主机网络使用虚拟私有云（VPC）提供的网络，包括子网、安全组等。您可以选择使用已有的虚拟私有云网络，或者单击“前往控制台创建”来创建新的虚拟私有云。

参数	说明
安全组	安全组用来实现安全组内和安全组间云主机的访问控制，加强云主机的安全保护。用户可以在安全组中定义各种访问规则，当云主机加入该安全组后，即受到这些访问规则的保护。创建云主机时，可支持选择多个安全组。此时，云主机的访问规则遵循几个安全组规则的并集。 注意：如需通过远程桌面连接到 Windows 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP 端口范围：3389 如需通过 ssh 连接到 Linux 云主机，请在安全组中添加如下规则 方向：入方向 协议：TCP 端口范围：22 如需 Ping 云主机地址，请在安全组中添加如下规则 方向：入方向 协议：ICMP 类型：Any
网卡	添加一张主网卡，可使用已有内网 IP 地址，或自动分配。
弹性公网 IP	将云主机与弹性 IP 绑定，使云主机通过固定的公网 IP 地址与互联网互通。 您可以根据实际情况选择以下三种方式： 不使用：云主机不能直接与互联网互通，仅可作为私有网络中部署业务或者集群所需云主机进行使用。 自动分配：自动为每台云主机分配独享带宽的弹性 IP，带宽值可以由您设定。 使用已有：为云主机分配已有弹性 IP。使用已有弹性 IP 时，不能批量创建云主机。

步骤 4：高级配置

1.设置“登录方式”，并创建对应的密码或密钥对。

- 密钥对：指使用密钥对作为云主机的鉴权方式。您可以选择使用已有的密钥，或者单击“查看密钥对”创建新的密钥。注：如果选择使用已有的密钥，请确保您已在本地获取该文件，否则，将影响您正常登录云主机。
- 密码：指使用设置初始密码方式作为云主机的鉴权方式。此时，您可以通过用户名密码方式登录云主机，Linux 操作系统时为 root 用户的初始密码，Windows 操作系统时为 Administrator 用户的初始密码。密码复杂度需满足：

参数	规则
密码	8～30 个字符，必须同时包含三项（大写字母、小写字母、数字、()`~!@#$%^&*_-+={}[]:;'<>,.?/ 中的特殊符号），且不能以斜线号（/）开头

2.选择“云主机组”。

3.选择“用户数据”配置方式（目前仅部分资源池支持）。

4.单击“下一步，确认配置”。

步骤 5：确认配置

1.在“确认配置”页面，查看云主机配置详情。

2.企业项目：企业项目是对多个资源实例进行归类管理的单位，不同云服务区域的资源和项目可以归到一个企业项目中。企业可以根据不同的部门或项目组，将相关的资源放置在相同的企业项目内进行管理，支持资源在企业项目之间迁移。

3.设置购买量。

- 购买时长：“包年/包月”方式需要设置购买时长，最短为 1 个月，最长 3 年。
- 自动续订：“包年/包月”方式可选是否开启自动续订。按月购买的自动续订周期为 1 个月，按年购买的自动续订周期为 1 年。
- 购买数量：设置购买弹性云主机的数量。为了保证所有资源的合理分配，如果您需要的弹性云主机数量超过当前您可以购买的最大数值，您要提交工单申请扩大配额。申请通过后，您可以购买到满足您需要的弹性云主机数量。

4.协议：阅读并勾选同意协议。

5.如果您确认配置无误，单击“立即购买”。

6.点击“立即支付”进行付款，付款成功即可创建弹性云主机。弹性云主机创建成功后，您可以在弹性云主机信息页面看到您新创建的弹性云主机。

3.4连接 GPU 云主机

3.4.1 连接方式概述

约束与限制

- 只有运行中的 GPU 云主机才允许用户登录。
- Linux 操作系统用户名“root”， Windows 操作系统用户名“Administrator”。
- GPU 实例中，部分实例不支持云平台提供的远程登录功能，需要自行安装 VNC Server 进行登录。推荐使用 MSTSC 方式登录 GPU 云主机。
- 使用 MSTSC 方式访问 GPU 云主机时，使用 WDDM 驱动程序模型的 GPU 将被替换为一个非加速的远程桌面显示驱动程序，造成 GPU 加速能力无法实现。因此，如果需要使用 GPU 加速能力，您必须使用不同的远程访问工具。如果使用管理控制中心操作界面的“远程登录”功能无法满足您的访问需求，请自行在 GPU 云主机上安装符合要求的远程访问工具（如 Tight VNC）。

登录方式概述

请根据需要选择登录方式，登录 GPU 云主机。

表 1-Linux 操作系统的 GPU 云主机登录方式一览

GPU 云主机操作系统	**本地主机操作系统**	**连接方法**	**条件**
Linux	Windows	使用控制中心远程登录方式：参见 使用 VNC 方式登录 GPU 云主机(L	不依赖弹性 IP

GPU 云主机操作系统	本地主机操作系统	连接方法	条件
		inux）。	
		使用 PuTTY、Xshell 等远程登录工具： 参见 SSH 密码方式登录(本地使用 Windows 操作系统）。 参见 SSH 密钥方式登录(本地使用 Windows 操作系统）。	云主机绑定弹性 IP (通过内网登录 GPU 云主机时可以不绑定弹性 IP，例如 VPN、云专线等内网网络连通场景。)
	Linux	使用命令连接： 参见 SSH 密码方式登录(本地使用 linux 操作系统）。 参见 SSH 密钥方式登录(本地使用 Linux 操作系统）。	
	移动设备	使用 Termius、JuiceSSH 等 SSH 客户端工具登录云主机。参见在移动设备上登录 Linux 云主机。	
	Mac OS 系	使用系统自带的终端（Terminal）： 参见Mac OS系统登录Linux弹性	

GPU云主机操作系统	本地主机操作系统	连接方法	条件
	统	云主机。	

表 2- Windows 操作系统的 GPU 云主机登录方式一览

GPU云主机操作系统	本地主机操作系统	连接方法	条件
Windows	Windows	使用控制中心远程登录云主机。参见使用使用 VNC 方式登录 GPU 云主机（Windows）。	不依赖弹性 IP
	移动设备	安装远程连接工具，例如 Microsoft Remote Desktop 在移动设备上登录。参见在移动设备上登录 Windows 云主机。	云主机绑定弹性 IP（通过内网登录云主机时可以不绑定弹性 IP，例如 VPN、云专线等内网网络连通场景。）
	Windows	使用 mstsc 方式登录云 GPU 主机。参见远程桌面连接（MSTSC 方式）。	
	Linux	安装远程连接工具，例如 rdesktop，执行连接命	

GPU 云主机操作系统	本地主机操作系统	连接方法	条件
		令。参见在 Linux 主机上登录 Windows 云主机。	
	Mac OS 系统	安装远程连接工具，例如 Microsoft Remote Desktop for Mac 在 Mac OS 系统上登录。参见 Mac OS 系统登录 Windows 云主机。	

3.4.2 使用 VNC 方式登录 GPU 云主机（Linux）

约束与限制

- 远程登录功能使用系统配置的自定义端口进行访问。确保所需使用的端口未被防火墙屏蔽。例如，如果远程登录的链接是“xxx:8002”，请确保端口 8002 没有被防火墙屏蔽。
- 如果客户端操作系统使用了本地代理，且用户无法配置代理的防火墙端口，请在使用远程登录功能之前关闭代理模式。
- 对于采用了"密钥对"方式创建的 Linux 操作系统的 GPU 云主机，在使用控制中心操作界面的"远程登录"功能（VNC 方式）之前，您需要先通过"SSH 密钥方式"进行登录，并设置登录密码。只有完成了这一步骤，才能顺利使用 VNC 方式进行登录。

操作步骤

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 > 弹性云主机”。

3.在弹性云主机列表中，右上角的搜索框中输入 GPU 云主机的名称、ID 或 IP 地址进行搜索。

4.在搜索结果中找到目标 GPU 云主机，在其对应的“操作”列下，点击“远程登录”。

5.(可选)如果登录界面提示“按 Ctrl+Alt+DEL 解锁”,请点击远程登录操作面板右上方的“Send CtrlAltDel”按钮进行登录。

6.（可选）如果登录时发现远程登录界面上鼠标无法显示，请点击远程登录操作面板上方的“本地鼠标”按钮，恢复鼠标的正常显示。

7.根据界面提示，输入 GPU 云主机的密码。

后续处理

在成功登录 GPU 云主机后，您可以使用 VNC 方式提供的复制、粘贴功能，实现本地数据与 GPU 云主机之间的单向复制、粘贴（仅部分资源池支持）。以下是具体的操作步骤：

1.使用 VNC 方式成功登录 GPU 云主机。

2.单击页面右上角的“复制命令输入”。

3.在本地计算机上，使用快捷键 Ctrl+C 将要复制的数据选中并复制。

4.返回到 GPU 云主机的 VNC 窗口，使用快捷键 Ctrl+V 将复制的数据粘贴到命令行窗口中。

5.单击“发送”按钮，将复制的数据发送至命令行窗口。

3.4.3 使用 VNC 方式登录 GPU 云主机（Windows）

约束与限制

- 当前提供的远程登录功能是通过系统配置的自定义端口进行访问的，所以在使用远程登录功能时，请确保需要使用的端口未被防火墙屏蔽。例如：远程登录的链接为“xxx:8002”，则需要确保端口 8002 没有被防火墙屏蔽。
- 如果客户端操作系统使用了本地代理，且用户无法配置该本地代理的防火墙端口，请关闭代理模式后再使用远程登录功能。
- GPU 实例中，部分实例不支持控制中心操作界面的远程登录功能，需要自行安装 VNC Server 进行登录。推荐使用 MSTSC 方式登录 GPU 云主机。

登录 WindowsGPU 云主机

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 > 弹性云主机”。

3.获取 GPU 云主机密码。VNC 方式登录 GPU 云主机时，需已知其密码，然后再采用 VNC 方式登录。

- 当您的弹性云主机是采用密码方式鉴权时，请直接使用创建云主机时设置的密码进行登录。

- 当您的弹性云主机是采用密钥方式鉴权时，请提前准备好 Windows 登录密钥。

4.选择要登录的 GPU 云主机，单击“操作”列下的“远程登录”。

5.在弹出的“登录 Windows 弹性云主机”窗口中，选择“其他方式”下的 VNC 方式，单击“立即登录”。

6. (可选) 如果界面提示“按 Ctrl+Alt+DEL 解锁”，请单击远程登录操作面板右上方的“Send CtrlAltDel”按钮进行登录。

7.（可选）如果远程登录界面上无法显示鼠标，查看面板上方是否有“Local Cursor”按钮，单击“Local Cursor”按钮，鼠标就可以正常显示了。

8.根据界面提示，输入 GPU 云主机密码完成登录。

3.4.4SSH 密码方式登录 GPU 云主机（Linux）

客户端使用 Windows 系统

如果客户端使用的计算机系统为 Windows 操作系统，按照下面方式登录 GPU 云主机。下面操作步骤以 PuTTY 为例。

1.运行 PuTTY。

2.单击“Session”，在“Host Name (or IP address)”下的输入框中输入 GPU 云主机的弹性 IP。

3.单击“Window”，在“Translation”下的“Received data assumed to be in which character set:”选择“UTF-8”。

4.单击“Open”。

5.输入用户名和创建云主机时设置的密码登录 GPU 云主机。

客户端使用 Linux 系统

如果客户端使用的计算机系统为 Linux 操作系统，您可以在计算机的命令行中键入 ssh 云主机绑定的弹性 IP 登录云主机器。

1.输入 SSH 命令：ssh 用户名@弹性 IP。

2.输入用户名和创建云主机时设置的密码登录云主机。

3.4.5SSH 密钥方式登录 GPU 云主机（Linux）

前提条件

- 已获取该 GPU 云主机的密钥文件。

- GPU 云主机已经绑定弹性 IP。
- 已配置安全组入方向的访问规则。
- 使用的登录工具（如 PuTTY）与待登录的 GPU 云主机之间网络连通。例如，默认的 22 端口没有被防火墙屏蔽。

客户端使用 Windows 操作系统

如果您本地使用 Windows 操作系统登录 Linux GPU 云主机，可以按照下面方式登录 GPU 云主机。

方式一：使用 PuTTY 登录

我们以 PuTTY 为例介绍如何登录 GPU 云主机。使用 PuTTY 登录 GPU 云主机前，需要先将私钥文件转化为.ppk 格式。

1.下载 PuTTY 和 PuTTYgen。PuTTYgen 是密钥生成器，用于创建密钥对，生成一对公钥和私钥供 PuTTY 使用。

2.运行 PuTTYgen。

3.在“Actions”区域，单击“Load”，并导入创建 GPU 云主机时保存的私钥文件； 导入时注意确保导入的格式要求为“All files (.)”。

4.单击“Save private key”。

5.保存转化后的私钥到本地。例如：kp-123.ppk。

6.双击“PUTTY.EXE”，打开“PuTTY Configuration”。

7.选择“Connection > data”，在 Auto-login username 处输入镜像的用户名。

8.选择“Connection > SSH > Auth”，在最下面一个配置项“Private key file for authentication”中，单击“Browse”，选择步骤 5 转化的密钥。

9.单击“Session”，在“Host Name (or IP address)”下的输入框中输入 GPU 云主机的弹性 IP 地址。

10.单击“Open”，登录 GPU 云主机。

方式二：使用 Xshell 登录

1.打开 Xshell 工具。

2.通过弹性 IP，执行以下命令，SSH 远程连接 GPU 云主机。

```
ssh 用户名@弹性 IP
```

示例：ssh root@192.168.0.1

3.（可选）如果系统弹窗提示“SSH 安全告警”，此时需单击“接受并保存”。

4.选择“Public Key”，并单击“用户密钥(K)”栏的“浏览”。

5.在“用户密钥”窗口中，单击“导入”。

6.选择本地保存的密钥文件，并单击“打开”。

7.单击“确定”，登录 GPU 云主机。

客户端使用 Linux 操作系统

如果您本地使用 Linux 操作系统登录 Linux GPU 云主机，可以按照下面方式登录。下面以私钥文件是 kp-123.pem 为例进行介绍。

1.在您的 linux 计算机的命令行中执行如下命令，变更权限。下列命令的 path 为密钥文件的存放路径。

```
chmod 400 /path/kp-123
```

2.执行如下命令，登录 GPU 云主机。

```
ssh -i /path/kp-123 默认用户名@云主机
```

假设 Linux 云主机的默认用户名是 linux，则命令如下：

```
ssh -i /path/kp-123 linux@弹性 IP 地址
```

path 为密钥文件的存放路径。

弹性 IP 地址为 GPU 云主机绑定的弹性 IP 地址。

3.5管理 GPU 云主机

3.5.1 停止实例

操作场景

关机操作可以将弹性云主机从运行状态切换为关机状态，以进行维护等操作。

使用须知

- 按需付费的弹性云主机支持节省关机和普通关机两种计费模式（节省关机目前仅部分资源池支持）。
- 节省关机模式下，计算资源（vCPU、内存、GPU)不再收费，其余收费的资源正常计费，如系统盘、数据盘、带宽等，且再次开机后不会导致内网 IP 或弹性公网 IP 地址变更。
- 由于计算资源被回收，再次开机时可能因为资源不足导致启动失败，您可以稍后尝试再次开机或者尝试变配为其他规格，如一直没有可用资源，云主机有一直不能开机的风险。
- 现阶段为过渡阶段，节省关机和普通关机均不收费，预计 2024 年将针对普通关机进行收费。
- 如下场景不支持节省关机

- 所开通的云主机为包周期计费模式
- 所开通的云主机为本地盘云主机
- 已经加入主机组的云主机
- 已经加入云主机快照、云主机备份策略的云主机

操作步骤

1. 登录控制中心。

2. .单击控制中心顶部的  ，选择“地域”。

3.单击左侧导航栏“产品服务列表”，选择“计算 > 弹性云主机”。

4.根据实际需求对云主机进行关机。

（1）对单台云主机进行关机操作：在云主机列表中，选择需要开机的云主机，点击云主机列表左上角的开机操作按钮。

（2）对多台云主机进行关机操作：在云主机列表中，选择需要开机的多台云主机，点击云主机列表左上角的开机操作按钮。

5.在关机弹窗中，设置关机方式和模式，并确认操作是否正确，无误后点击确定。

设置	**说明**
关机方式	关机：正常关机流程。但为避免关机失败，当您选择关机时，普通关机超时会自动执行强制关机操作。 强制关机：等同于断电处理，可能丢失云主机操作系统中未写入磁盘的数据。
关机模式	普通关机模式：普通关机后保留计算、存储、网络资源。目前关机后计算资源不收费，存储、网络资源继续收费，预计 2024 年普通关机后计算、网络、存储资源均收费。 节省关机模式：节省关机后，计算资源（vCPU、内存、GPU)均被释放且不计费，其余网络、存储资源继续保留并收费。

3.5.2 启动实例

操作场景

启动 GPU 云主机则将其从关机状态切换为运行状态，以便正常运行应用程序和服务。

启动单台 GPU 云主机操作步骤

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 >弹性云主机”。

3.在云主机列表中，使用搜索功能输入 GPU 云主机的名称、ID 或 IP 地址以定位目标 GPU 云主机。

4.选择目标 GPU 云主机，点击云主机列表左上角的“开机”按钮。

5.在弹出的提示信息中，确认操作是否正确。请注意 GPU 云主机状态的说明。如果 GPU 云主机在中间状态停留超过 30 分钟，表示可能出现异常情况，请及时提交工单以寻求进一步处理。

启动多台 GPU 云主机操作步骤

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 >弹性云主机”。

3.在云主机列表中，选择需要停止的多台 GPU 云主机。

4.点击云主机列表左上角的“开机”按钮。

在弹出的提示信息中，确认操作是否正确。请注意云主机状态的说明：如果云主机在中间状态停留超过 30 分钟，表示可能出现异常情况，请及时提交工单以寻求进一步处理。

3.5.3 重启实例

操作场景

重启操作是维护云主机的一种常用方式，如系统更新、重启保存相关配置等。

操作步骤

1. 登录控制中心。

2. 单击控制中心顶部的  ，选择“地域”。

3.单击左侧导航栏“产品服务列表”，选择“计算 > 弹性云主机”。

4.根据实际需求对云主机进行重启。

（1）对单台云主机进行重启操作：在云主机列表中，选择需要重启的云主机，点击云主机列表左上角的重启操作按钮。

（2）对多台云主机进行重启操作：在云主机列表中，选择需要重启的多台云主机，点击云主机列表左上角的重启操作按钮。

5.在重启弹窗中，设置重启方式，并确认操作是否正确，无误后点击确定。

设置	说明
重启方式	重启：正常重启流程。 强制重启：等同于断电重启，可能丢失云主机操作系统中未写入磁盘的数据。

3.5.4 释放实例

操作场景

释放 GPU 云主机则将其从关机状态释放掉，您可以通过控制台释放按量付费实例。

释放单台 GPU 云主机操作步骤

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 >弹性云主机”。

3.在云主机列表中，使用搜索功能输入 GPU 云主机的名称、ID 或 IP 地址以定位目标 GPU 云主机。

4.选择目标 GPU 云主机，并单击“操作”列下的“更多 > 删除”。

5.在弹出的提示信息中，确认操作是否正确。请注意 GPU 云主机状态的说明。如果 GPU 云主机在中间状态停留超过 30 分钟，表示可能出现异常情况，请及时提交工单以寻求进一步处理。

释放多台 GPU 云主机操作步骤

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 >弹性云主机”。

3.在云主机列表中，选择需要停止的多台 GPU 云主机。

4.选择目标 GPU 云主机，并单击“操作”列下的“更多 > 删除”。

5.在弹出的提示信息中，确认操作是否正确。请注意云主机状态的说明：如果云主机在中间状态停留超过 30 分钟，表示可能出现异常情况，请及时提交工单以寻求进一步处理。

3.5.5 变配

操作场景

当您创建的 GPU 云主机规格无法满足业务需要时，可以升级或降级云主机的 vCPU、内存、显存。GPU 云主机仅支持同规格族内的规格变更。

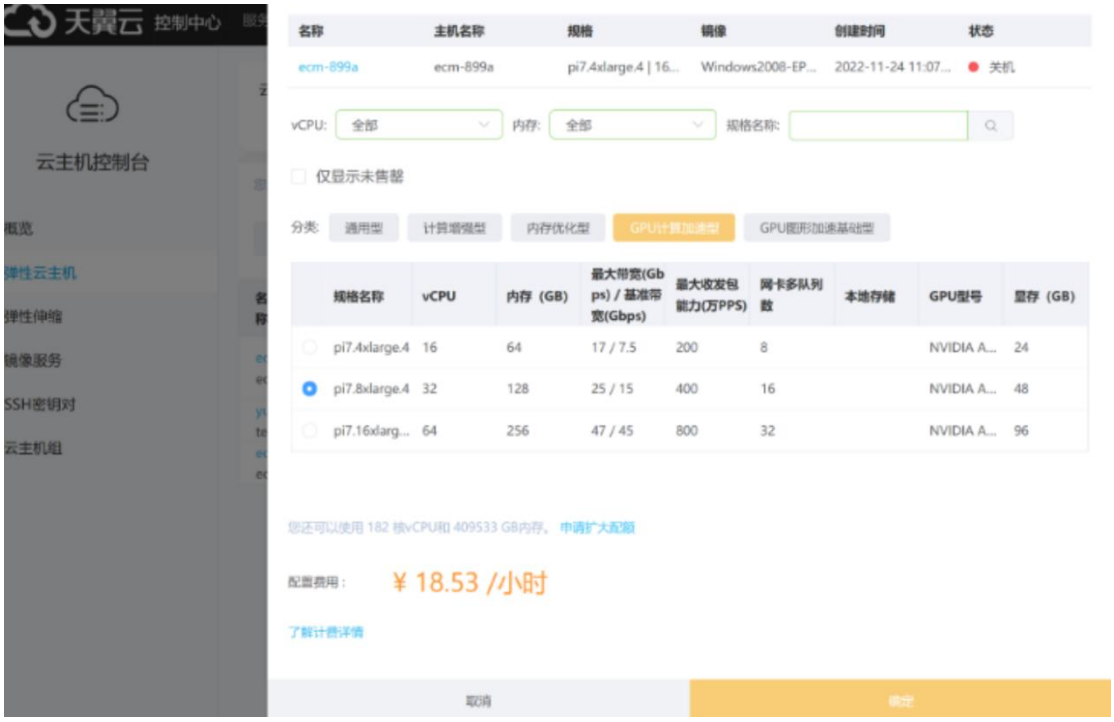
操作步骤

1.登录控制中心。

2.选择“计算 > 弹性云主机”。

3.在云主机列表，选择所要进行变配操作的 GPU 云主机，单击 GPU 云主机所在行的“操作”列下的“更多 > 变配”。

4.在弹出的变更规格页面，选择变更后的 GPU 云主机 vCPU、内存和显存。



5.单击“确定”。

6.支付成功后可完成规格变更。

3.5.6 重置密码

操作场景

- 首次连接 GPU 云主机后，建议您修改初始密码。
- 密码丢失，可以通过系统提供的重置密码功能找回。

前提条件

- GPU 云主机的状态为“运行中”。
- GPU 云主机网络正常通行。

操作步骤

1.登录控制中心。

2.选择“计算 >云主机”。

3.选中待重置密码的 GPU 云主机，并选择“操作”列下的“更多 > 重置密码”。

4.根据界面提示，设置 GPU 云主机的新密码，并确认新密码。

5.单击“确认”。

3.5.7 更改时区

请参见弹性云主机 >管理弹性云主机 >更改时区。

3.5.8 重装操作系统

操作场景

GPU 云主机操作系统无法正常启动时，或 GPU 云主机系统运行正常，但需要对系统进行优化，使其在最优状态下工作时，可以重装 GPU 云主机的操作系统。

前提条件

- 云硬盘的配额需大于 0。
- 如果是通过私有镜像创建的云主机，请确保原有镜像仍存在。
- 待重装操作系统的云主机处于“关机”状态或“重装失败”状态。
- 待重装操作系统的云主机挂载有系统盘。

- 重装操作系统会清除系统盘数据，包括系统盘上的系统分区和所有其它分区，请做好数据备份。

操作步骤

1.登录控制中心。

2.选择“计算 > 弹性云主机”。

3.在待重装操作系统的云主机的“操作”列下，单击“更多 > 一键重装”。

4.只有关机状态的云主机才能重装系统。如果云主机不是关机状态，请先关机。

5.在“一键重装”弹窗，选择想要重装的操作系统。

6.如果待重装操作系统的云主机是使用密码登录方式创建的，此时可以更换使用新密码。

7.单击“确定”，提交重装系统的申请。

8.提交重装系统的申请后，云主机的状态变为“重建中”，当该状态消失后，表示重装结束。

3.5.9 查看 GPU 云主机信息

在您申请了 GPU 云主机后，可以通过管理控制台查看您的 GPU 云主机。

操作步骤

1.登录控制中心。

2.选择“计算 > 弹性云主机”。

3.在弹性云主机列表中的右上角，输入 GPU 云主机名、IP 地址或 ID，并单击 进行搜索。

4.单击待查询 GPU 云主机的名称。

5.系统跳转至该 GPU 云主机详情页面。

6.查看 GPU 云主机的详细信息。

7.您可以选择“云硬盘/网卡/安全组/弹性 IP/监控”页签，更改 GPU 云主机安全组、为 GPU 云主机添加网卡、绑定弹性 IP 等。

3.5.10 修改 GPU 云主机名称

请参见弹性云主机 >管理弹性云主机 >修改云主机名称。

3.5.11GPU 监控

前提条件

- 确保 GPU 云主机已安装 GPU 驱动/GRID 驱动。驱动安装请参见 NVIDIA 驱动安装指引-GPU 云主机-用户指南-安装 NVIDIA 驱动 - 天翼云（ctyun.cn)。
- 确保您已在 GPU 云主机上安装云监控插件，关于如何安装云监控插件，请参见安装监控 Agent-弹性云主机-用户指南-监控 - 天翼云（ctyun.cn)。

3.6安装 NVIDIA 驱动

3.6.1NVIDIA 驱动安装指引

驱动类型选型概述

天翼云 GPU 云主机支持安装以下两种 NVIDIA 驱动：

- GPU 驱动：用于驱动物理 GPU，也称 Tesla 驱动。
- GRID 驱动：配套 NVIDIA GRID vGPU 方案使用、用于获得实时渲染能力。

实例类型	场景	驱动类型	驱动安装方式
GPU 计算加速型(windows)	通用计算	Tesla 驱动	NVIDIA 官网下载并安装 GPU 驱动
	图形渲染	GRID 驱动	● 在购买时选择已预装 GRID 驱动的计费镜像 ● 向 NVIDIA 或其代理商购买对应的 License 并自行安装 GRID 驱动（ 不推荐）
GPU 计算加速型(linux)	通用计算	Tesla 驱动	● 创建 GPU 实例时自动安装 GPU 驱动 ● NVIDIA 官网下载并安装 GPU 驱动
	图形渲染（离线渲染可使用 Tesla 驱动，部分实时渲染需使用 GRID 驱动）	Tesla 驱动/GRID 驱动	如安装 Telsa 驱动 ● 创建 GPU 实例时自动安装 GPU 驱动 ● NVIDIA 官网下载并安装 GPU 驱动 如安装 GRID 驱动 ● 在购买时选择已预装 GRID 驱动的计费镜像

			• 向 NVIDIA 或其代理商购买对应的 License 并自行安装 GRID 驱动（不推荐）
GPU 图形加速基础型（windows/linux）	通用计算	GRID 驱动	公共镜像中已预装 GRID 驱动，无需单独付费
	图形渲染	GRID 驱动	公共镜像中已预装 GRID 驱动，无需单独付费

注意目前仅部分资源池的 GPU 云主机支持自动安装 GPU 驱动/提供预装 GRID 驱动的计费镜像，其他资源池请您手动安装驱动。如您需手动安装 GPU 驱动，请参见 安装 Tesla 驱动 和 安装 GRID 驱动。预装 GRID 驱动的计费镜像的收费策略请参见计费说明-镜像服务-计费说明 - 天翼云 (ctyun.cn)。

各实例规格支持自动安装/预装的驱动版本

实例类型	支持驱动类型	自动安装支持的驱动版本	预装驱动的镜像	手动安装驱动版本支持
GPU 计算加速型 P8A/PI7/P2V/P2VS/PI2	Tesla 驱动	Driver 470.82.01CUDA 11.4.3 CUDNN 8.8.1.3	-	无特殊要求，NVIDIA 官方支持版本即可。
	GRID 驱动	-	Windows2019-DataCenter-GRID13.2 注：预装驱动版本 GRID13.2；部分资源池暂不支持。	-GRID 14（- Driver 514.08）NVIDIA Virtual GPU (vGPU) Software Do

				cumentation
GPU 图形加速基础型 G5/G6	GRID 驱动	-	• Windows Server 2016 Standard 64bit • Windows Server 2012 Standard 64bit • CentOS 7.5 64bit • CentOS 7.6 64bit • Ubuntu Server 16.04 64bit 注：预装驱动版本 GRID9.1	GRID9.0-9.4（Driver 430.30-432.44）

GPU 图形加速基础型 G7/G5S	GRID 驱动	-	非多 AZ 资源池 ● Windows Server 2019 DataCenter 64bit ● Windows Server 2016 DataCenter 64bit ● Windows Server 2012 DataCenter 64bit 多 AZ 资源池 ● Windows Server 2019 DataCenter 64bit ● Windows Server 2016 DataCenter 64bit ● Windows Server 2012 DataCenter 64bit ● CentOS 8.1 64bit ● CentOS 8.2 64bit ● Ubuntu Server 20.04 64bit 注：预装驱动版本 GRID13.2	GRID13.0-13.8(Driver 470.63.01-474..44)

3.6.2 安装 Tesla 驱动

您可根据如下操作步骤自行安装 Tesla 驱动，如要安装 CUDA 工具包请参见安装 CUDA。

如何选择驱动版本请参见如何选择驱动及相关库、软件版本。

前提条件

- GPU 云主机未安装驱动。
- GPU 云主机配备弹性 IP。

一 、Centos 操作系统驱动安装

1.下载对应驱动。访问 NVIDIA 驱动下载官网，选择对应 GPU 型号、操作系统和 CUDA Toolkit 版本后，进行下载，本文以 A100 为例，如下图所示。

2.点击搜索，选择要下载的驱动版本，点击下载。

3.将下载的驱动安装包上传到云主机中，执行以下命令，对安装包添加执行权限。

例如，对文件名为 NVIDIA-Linux-x86_64-470.199.02.run 添加执行权限。

```
chmod +x NVIDIA-Linux-x86_64-470.199.02.run
```

4.安装 kernel-devel、gcc 包，注意 kernel-devel 版本要和内核版本保持一致。

```
sudo yum install -y gcc kernel-devel
```

注意：**可通过公网或内网 yum 源下载，内网 yum 源链接如下：**

多 AZ 资源池内网 yum 下载依赖

http://169.254.169.253:10080/gpu/NVIDIAToolkit/depend/kernel-devel-$ (uname-r).rpm

http://169.254.169.253:10080/gpu/NVIDIAToolkit/depend/kernel-headers-$(uname -r).rpm

非多 AZ 资源池内网 yum 下载依赖

http://100.126.0.130:10080/gpu/NVIDIAToolkit/depend/kernel-devel-$(uname -r).rpm

http://100.126.0.130:10080/gpu/NVIDIAToolkit/depend/kernel-headers-$(uname:-r).rpm

5.执行以下命令，运行驱动安装程序，并按提示进行后续操作。

```
sudo sh NVIDIA-Linux-x86_64-418.126.02.run --disable-nouveau
--kernel-source-path=/usr/src/kernels/$(uname -r)
```

6.安装完成后，执行以下命令进行验证。

```
nvidia-smi
```

如返回信息类似下图中的 GPU 信息，则说明驱动安装成功。

```
[root@linux-c77-sz-200908 ~]# nvidia-smi
Wed Mar  2 09:36:25 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.82.01    Driver Version: 470.82.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| 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-PCI...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   32C    P0    39W / 250W |      0MiB / 40536MiB |      0%      Default |
|                               |                      |             Disabled |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

7.（可选）GPU 驱动开启持久化模式

Persistence-M(Persistence Mode)是一个用户可设置的驱动程序属性的术语。启用持久性模式后，即使没有活动的客户端，NVIDIA 驱动程序也会保持加载状态。这样可以最大程度地减少与运行依赖的应用程序(例如 CUDA 程序)相关的驱动程序加载延迟，同时减少 GPU 云主机掉卡问题的发生。

```
cd /usr/share/doc/NVIDIA_GLX-1.0/sample*
bunzip2 nvidia-persistenced-init.tar.bz2
```

```
tar xvf nvidia-persistenced-init.tarcd nvidia-persistenced-init && sh install.sh -u root
```

二 、Ubuntu 操作系统 驱动安装

1. 下载对应驱动。访问 NVIDIA 驱动下载官网，选择对应 GPU 型号、操作系统和 CUDA Toolkit 版本后，进行下载，本文以 A100 为例，如下图所示。

2.点击搜索，选择要下载的驱动版本，点击下载。

3.将下载的驱动安装包上传到云主机中，执行以下命令，对安装包添加执行权限。 例如，对文件名为 NVIDIA-Linux-x86_64-470.199.02.run 添加执行权限。

```
chmod +x NVIDIA-Linux-x86_64-470.199.02.run
```

4.安装 gcc 和 linux-kernel-headers。

```
sudo apt-get install gcc linux-kernel-headers
```

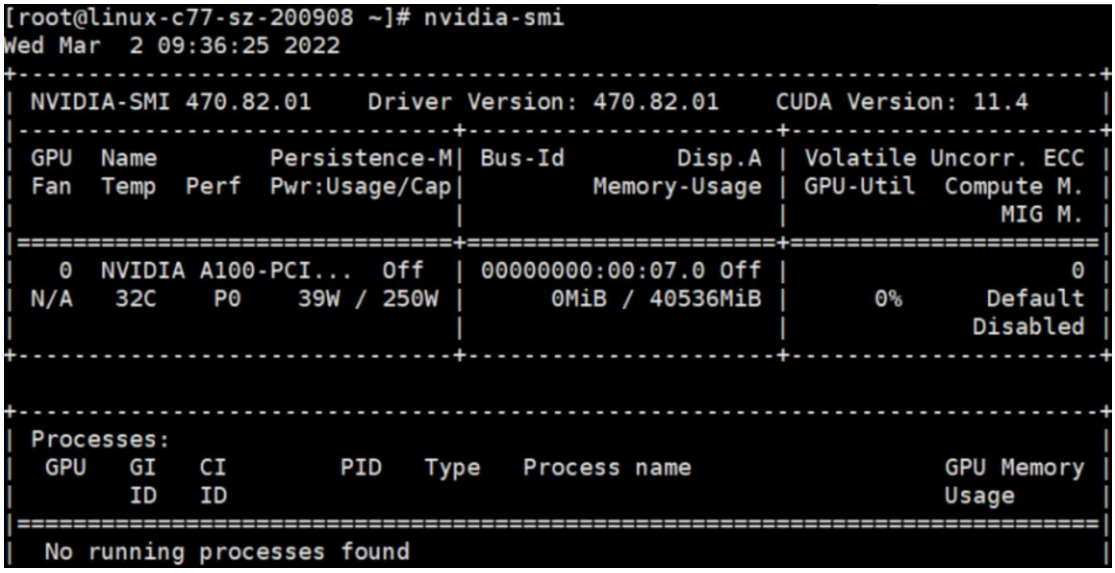
5.执行以下命令，运行驱动安装程序，并按提示进行后续操作。

```
sudo sh NVIDIA-Linux-x86_64-418.126.02.run --disable-nouveau
```

6.安装完成后，执行以下命令进行验证。

```
nvidia-smi
```

如返回信息类似下图中的 GPU 信息，则说明驱动安装成功。



7.（可选）GPU 驱动开启持久化模式

Persistence-M(Persistence Mode)是一个用户可设置的驱动程序属性的术语。启用持久性模式后，即使没有活动的客户端，NVIDIA 驱动程序也会保持加载状态。这样可以最大程度地减少与运行依赖的应用程序(例如 CUDA 程序)相关的驱动程序加载延迟，同时减少 GPU 云主机掉卡问题的发生。

```
cd /usr/share/doc/NVIDIA_GLX-1.0/sample*

bunzip2 nvidia-persistenced-init.tar.bz2

tar xvf nvidia-persistenced-init.tarcd nvidia-persistenced-init && sh install.sh -u root
```

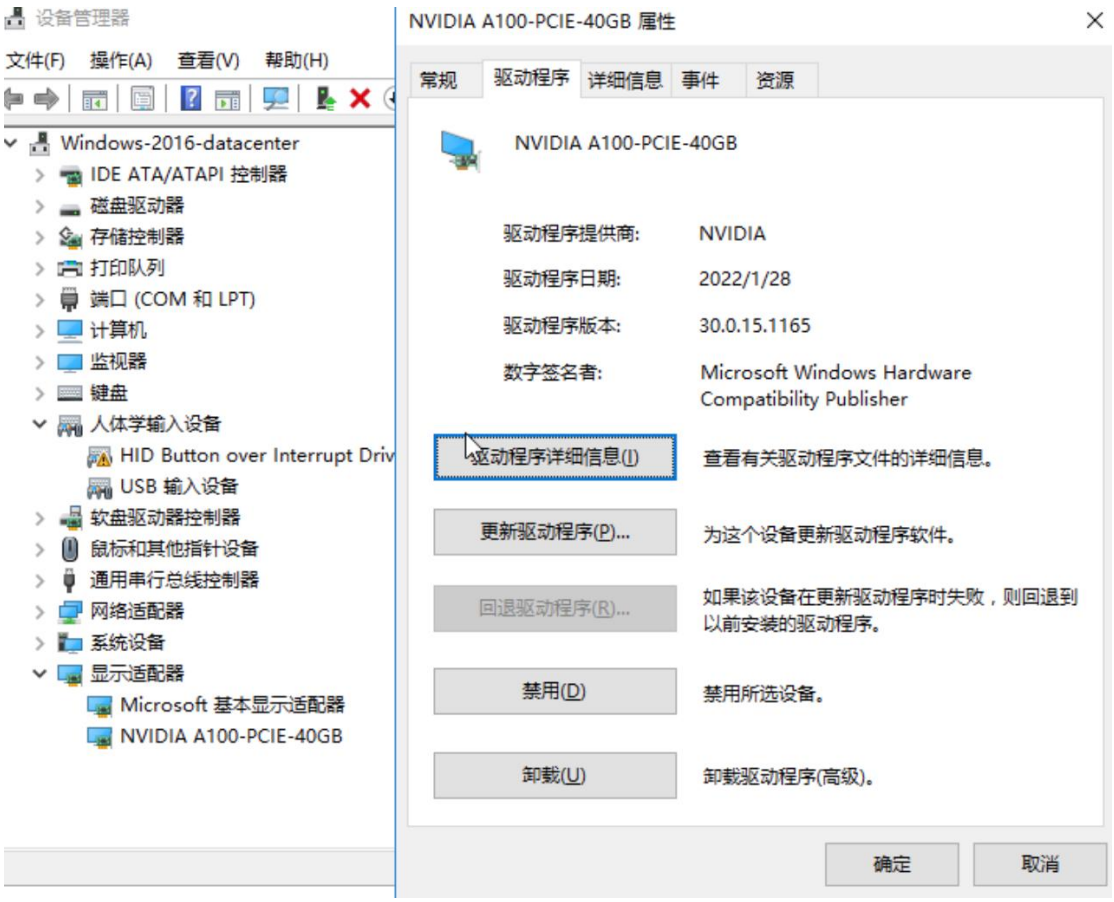
三 、Windows 操作系统驱动安装

1.下载对应驱动。在云主机内访问 NVIDIA 驱动下载官网，选择对应 GPU 型号、操作系统和 CUDA Toolkit 版本后，进行下载，本文以 A100 为例，如下图所示。

2.点击搜索，选择要下载的驱动版本，点击下载。

3.打开下载驱动程序所在的文件夹，双击安装文件开始安装，按照界面上的提示安装驱动程序并根据需要重启 GPU 云主机。

4.安装完成后查看设备管理器，显示如下则表示驱动安装成功。



3.6.3 安装 GRID 驱动

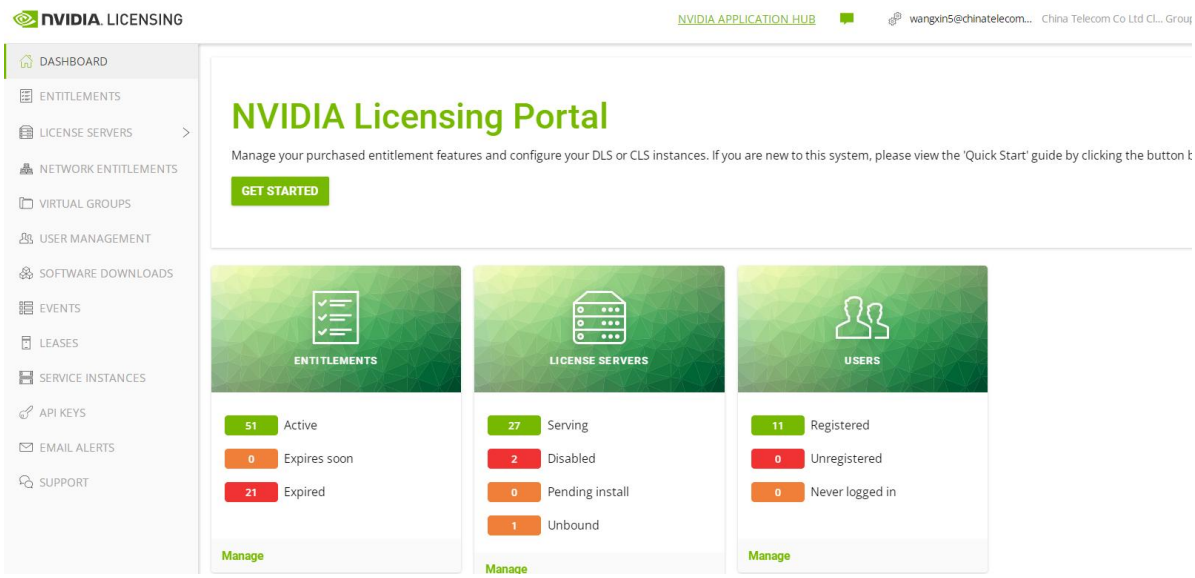
使用 PI7/P2V/P2VS/PI2 这几种计算加速型 GPU 云主机用作图形图像处理时，可以选择预装了 GRID 驱动的收费镜像，若已有镜像不能满足您的需求，您可自行向 NVIDIA 或其代理商的申请 License，并配置 License 服务器和安装 GRID driver。

一 、申请 license

1.打开 https://enterpriseproductregistration.nvidia.com/?LicType=EVAL &ProductFamily=vGPU，注册账号并申请 license。

2.审批通过后，会邮件通知到您，您可用该账号进行登录。

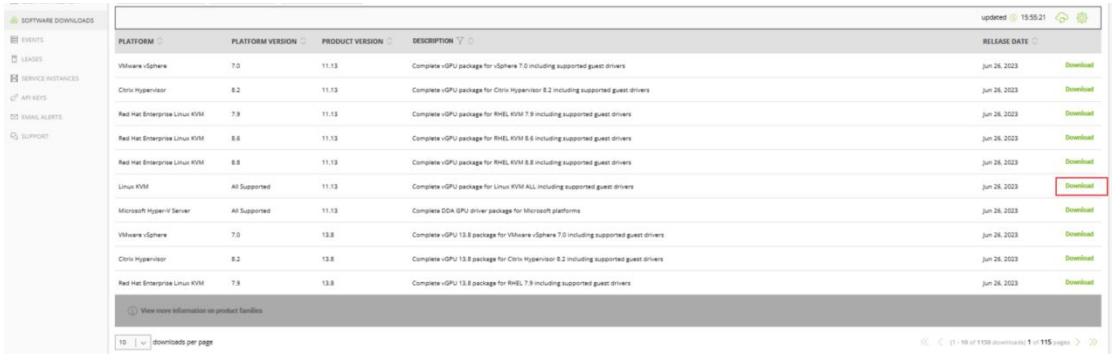
3.打开 https://nvid.nvidia.com，输入账号和密码，进入“Dashboard” 页面，如下图：



二 、软件下载

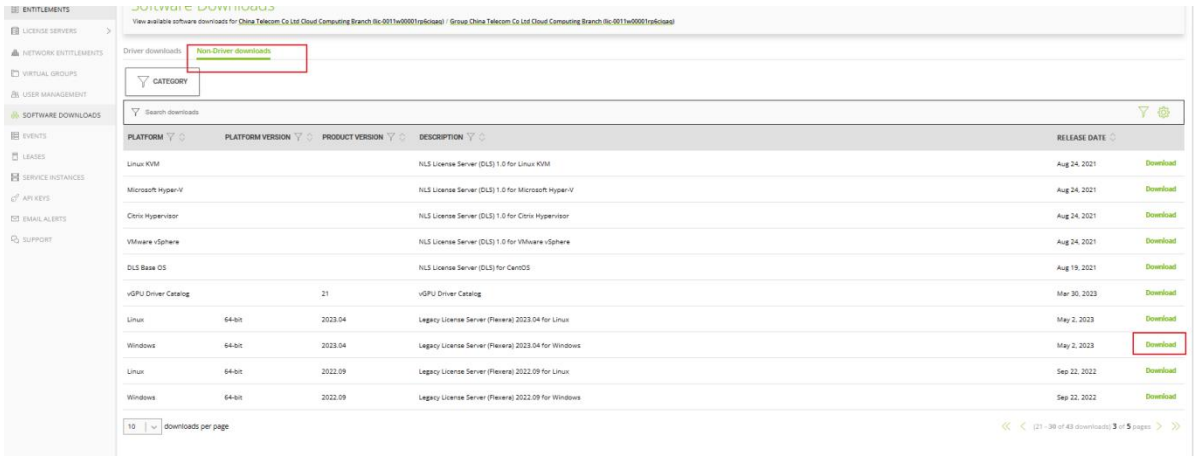
1.选择左侧导航栏中的 SOFTWARE DOWNLOADS，进入 “Software Downloads” 页面。

选择最新的 NVIDIA Virtual GPU SoftWare 版本，本文以 GRID 11.13 版本为例。如下图所示：



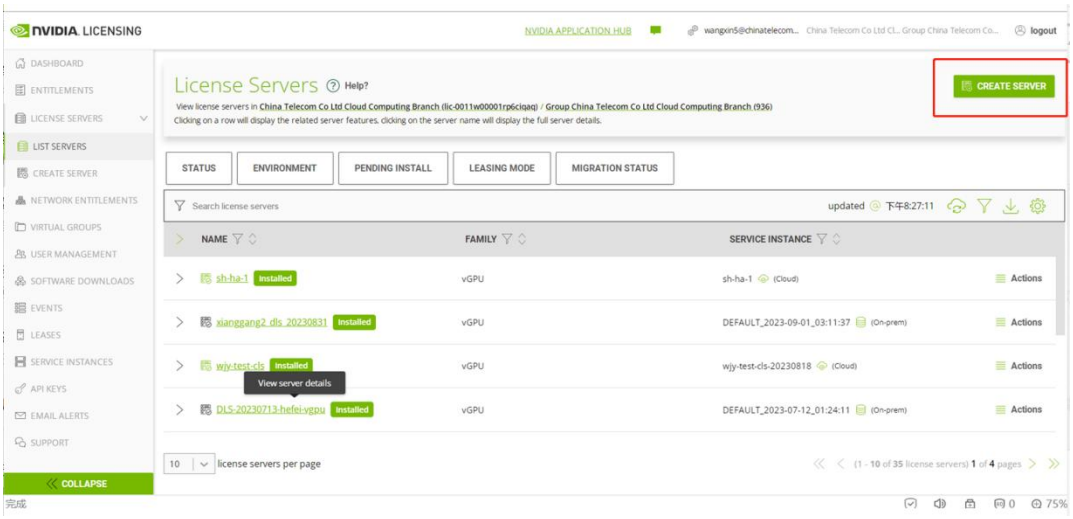
2.创建一台普通的云主机，作为 License 服务器，最小规格 4 核，8G 内存，50G 硬盘。

选择页面右侧 Non-Driver-downloads，单击所需下载的 License Server 软件。本文以下载 2023.04 Legacy License Server (Flexera) 2023.04 for Windows 为例，如下图所示：

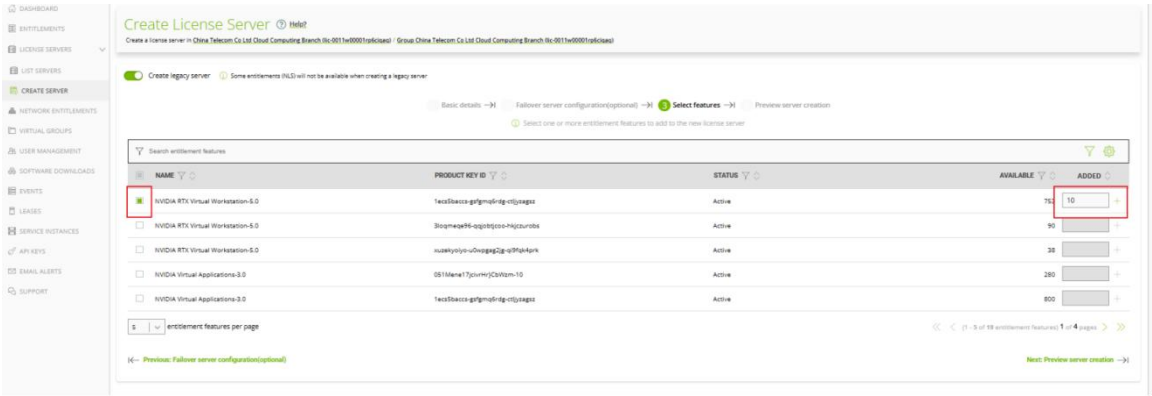


3.将 License Server 软件安装至步骤 2 创建的云主机，详情请参见 https://docs.nvidia.com/grid/ls/2023.04/grid-license-server-user-guide/index.html 。在完成安装后获取该 License 服务器的 MAC 地址。

4.选择左侧导航栏中的 LICENSE SERVERS，进入 “License Servers” 页面。在弹出的 “Create License Server” 窗口中填写相关信息注册新的 License Server。 其中 MAC Address 请填写 步骤 3 获取的 License 服务器 MAC 地址。

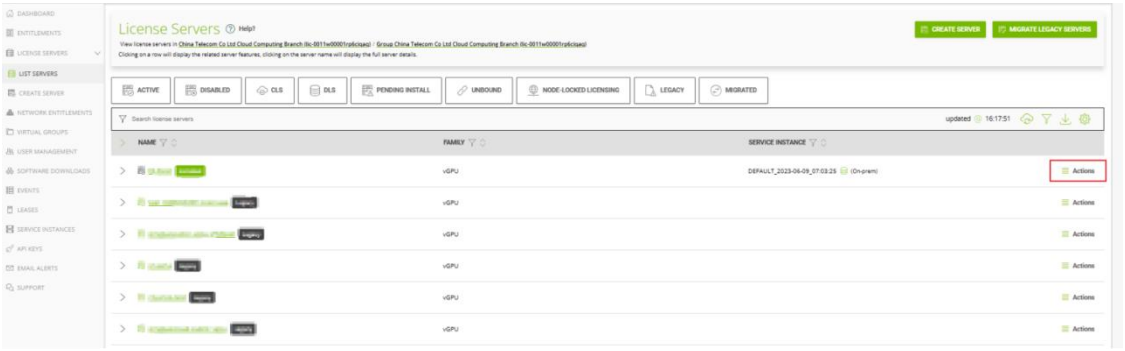


5.在 select feature 中选择对应的 license 和数量。



6.成功添加后，可在 “License Servers” 页面查看授权该 License 服务器的状态，包含数量和 License 过期时间。

7.选择 action->download 下载用于该 License Server 的 License 授权文件。

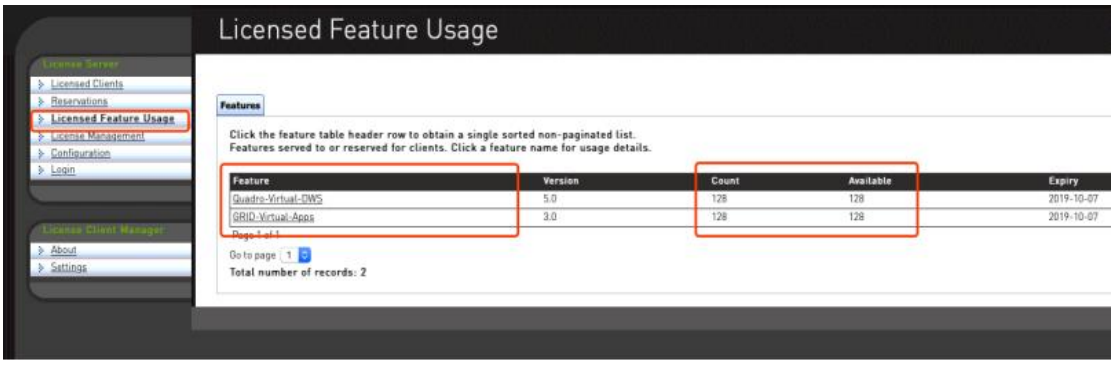


三、 配置 License 服务器

1.使用 License 服务器访问 License 管理控制台：http://localhost:8080/licserver。

2.选择左侧 License Server 栏中的 License Management，并导入 License 文件。

3.选择 Licensed Feature Usage，查看授权数量。如下图所示：



四、安装 GRID Driver

1.购买并创建一台计算加速型 GPU 云主机。

2.安装 GRID Driver 安装程序，即安装 NVIDIA vGPU for Windows 驱动程序。打开安装程序后按照界面提示完成安装，如下图所示：

3.重启 GPU 云主机。

4.重启后，在桌面右键打开 NVIDIA 控制面板，选择许可->管理许可证，填入对应的 server ip 和 port，配置 License 服务器的 IP 地址和端口号，要确保 License 服务器的 IP 地址可以被访问，以及端口号已设置为开放状态。填写完成后，点击应用，然后重启 GPU 云主机。

5.完成以上配置，GPU 云主机即可运行图形图像处理程序。

3.7卸载 NVIDIA 驱动

3.7.1 卸载 Tesla 驱动

背景信息

注意：GPU 云主机必须配备了相关驱动才可以正常使用。如果您因某种原因需要卸载当前驱动，请务必再安装与您实例规格及操作系统相匹配的正确驱动，否则会因 GPU 云主机与安装的驱动不匹配而造成业务无法正常进行的风险。

在 Windows 操作系统中卸载 Tesla 驱动

以下操作以操作系统为 Windows Server 2019 的 GPU 计算加速型云主机 PI7 为例。

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 > 弹性云主机”。

3.获取 GPU 云主机密码。VNC 方式登录 GPU 云主机时，需已知其密码，然后再采用 VNC 方式登录。

4.在云主机列表中，选择目标 GPU 云主机，其对应的“操作”列下，点击“远程登录”。

5.(可选)如果界面提示“Press CTRL+ALT+DELETE to log on”,请单击远程登录操作面板右上方的“Send CtrlAltDel”按钮进行登录。

6.根据界面提示，输入 GPU 云主机的密码登录。

7.单击 Windows 桌面左下角  图标，单击“控制面板”。

8.在控制面板中，选择“程序 > 卸载程序”。

9.右键单击待卸载的 GPU 驱动，然后单击“卸载/更改(U)”。

10.在弹出的卸载程序对话框中，单击“卸载(U)”。

11.卸载完成后，单击“马上重新启动(R)”。重启完成后，则 GPU 驱动已卸载成功。

在 Linux 操作系统中卸载 Tesla 驱动

如果您在创建 GPU 云主机时自动安装了 Tesla 驱动,则 Tesla 驱动的卸载需要选择通过 run 安装包的卸载方式。以 Driver 470.161.03、CUDA 11.4.1 为例，具体操作如下所示。

1.执行以下命令，卸载 GPU 驱动。

```
sudo /usr/bin/nvidia-uninstall
```

2.执行以下命令，卸载 CUDA。

```
sudo /usr/local/cuda/bin/cuda-uninstaller


sudo rm -rf /usr/local/cuda-11.4
```

说明

不同 CUDA 版本，卸载命令可能存在差别，如果未找到 cuda-uninstaller 文件，请到/usr/local/cuda/bin/目录下查看是否存在 uninstall_cuda 开头的文件。如果有，则将命令中的 cuda-uninstaller 替换为该文件名。3.执行以下命令，重启实例。

```
sudo  reboot
```

3.7.2 卸载 GRID 驱动

背景信息

注意：GPU 云主机必须配备了相关驱动才可以正常使用。如果您因某种原因需要卸载当前驱动，请务必再安装与您实例规格及操作系统相匹配的正确驱动，否则会因 GPU 云主机与安装的驱动不匹配而造成业务无法正常进行的风险。

在 Windows 操作系统中卸载 GRID 驱动

以下操作以操作系统为 Windows2019-DataCenter-GRID13.2 的 GPU 计算加速型云主机 PI7 为例。

1.登录控制中心。

2.单击“左侧导航栏>服务列表”，选择“计算 > 弹性云主机”。

3.获取 GPU 云主机密码。VNC 方式登录 GPU 云主机时，需已知其密码，然后再采用 VNC 方式登录。

4.在云主机列表中，选择目标 GPU 云主机，其对应的“操作”列下，点击“远程登录”。

5.(可选)如果界面提示“Press CTRL+ALT+DELETE to log on”，请单击远程登录操作面板右上方的“Send CtrlAltDel”按钮进行登录。

6.根据界面提示，输入 GPU 云主机的密码登录。

7.单击 Windows 桌面左下角  图标，单击“控制面板”。

8.在控制面板中，选择“程序 > 卸载程序”。

9.右键单击待卸载的 GPU 驱动，然后单击“卸载/更改(U)”。

10.在弹出的卸载程序对话框中，单击“卸载(U)”。

卸载完成后，单击“马上重新启动(R)”。重启完成后，则 GPU 驱动已卸载成功。

在 Linux 操作系统中卸载 GRID 驱动

执行如下命令卸载 GRID 驱动，并根据提示完成操作。

1.执行如下命令，卸载驱动。

```
sudo nvidia-uninstall
```

2.当系统显示如下信息，询问您是否需要备份 X screen 的配置文件时，建议您保持默认选项 NO，然后按回车键。

说明

不同操作系统的回显内容可能存在部分差别，您只需根据自身业务场景选择即可。

3.当系统显示如下信息时，表示卸载已完成。选择“OK”，然后按回车键。

3.8升级或降级 NVIDIA 驱动

如果驱动版本已不适用于当前业务场景，或者安装了错误的驱动版本导致 GPU 实例无法使用，您可以通过卸载当前驱动再安装所需驱动的方式，完成 NVIDIA 驱动的升级或降级。

步骤一：卸载 NVIDIA 驱动

根据待卸载的驱动类型，选择对应操作：

驱动类型	卸载方法
Tesla 驱动	卸载 Tesla 驱动
GRID 驱动	卸载 GRID 驱动

步骤二：安装 NVIDIA 驱动

根据需要安装的驱动类型和操作系统，选择对应操作：

驱动类型	操作系统	安装方法
Tesla 驱动	Windows 操作系统	参见安装 Tesla 驱动步骤 3 操作
	Linux 操作系统	参见安装 Tesla 驱动步骤 1 或步骤 2 操作

驱动类型	操作系统	安装方法
GRID 驱动	Windows 操作系统	安装 GRID 驱动 创建配备 GRID 驱动的 GPU 云主机

注意:为确保 GPU 实例的正常使用，在重新安装 GRID 驱动时，您必须安装与其规格族相匹配的 GRID 驱动版本。如果安装错误版本的 GRID 驱动版本，将严重影响实例的性能，甚至导致实例无法正常运行。

4 最佳实践

4.1 如何选择驱动及相关库、软件版本

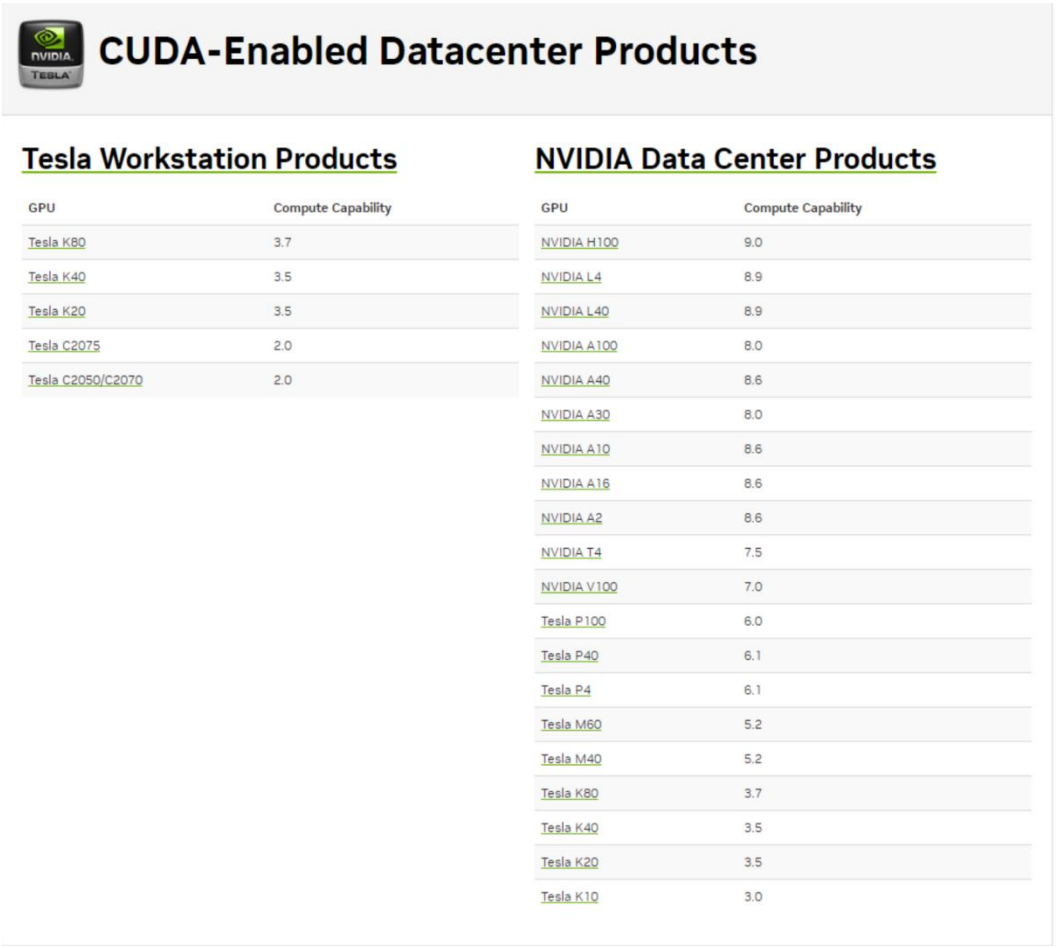
如何选择 CUDA 版本

CUDA（Compute Unified Device Architecture），是显卡厂商 NVIDIA 推出的运算平台。 CUDA™是一种由 NVIDIA 推出的通用并行计算架构，该架构使 GPU 能够解决复杂的计算问题。 它包含了 CUDA 指令集架构（ISA）以及 GPU 内部的并行计算引擎。 开发人员可以使用 C 语言来为 CUDA™架构编写程序，所编写出的程序可以在支持 CUDA™的处理器上以超高性能运行。

在选择 CUDA 版本前，需要先了解 GPU 云主机所挂载的显卡的算力，然后根据显卡算力来选择对应的 CUDA 版本。

具体步骤如下：

步骤一：通过英伟达官网查看显卡算力。以 NVIDIA T4 为例，可以看到其显卡计算能力为 7.5。



CUDA-Enabled Datacenter Products

Tesla Workstation Products

GPU	Compute Capability
Tesla K80	3.7
Tesla K40	3.5
Tesla K20	3.5
Tesla C2075	2.0
Tesla C2050/C2070	2.0

NVIDIA Data Center Products

GPU	Compute Capability
NVIDIA H100	9.0
NVIDIA L4	8.9
NVIDIA L40	8.9
NVIDIA A100	8.0
NVIDIA A40	8.6
NVIDIA A30	8.0
NVIDIA A10	8.6
NVIDIA A16	8.6
NVIDIA A2	8.6
NVIDIA T4	7.5
NVIDIA V100	7.0
Tesla P100	6.0
Tesla P40	6.1
Tesla P4	6.1
Tesla M60	5.2
Tesla M40	5.2
Tesla K80	3.7
Tesla K40	3.5
Tesla K20	3.5
Tesla K10	3.0

步骤二：根据显卡计算能力查看可支持 CUDA 版本，详情请参见 NVIDIA 数据中心。以 NVIDIA T4 为例，CUDA 10 以上的版本均能够支持,建议您选择最新版本的 CUDA。

Table 4. CUDA and Architecture Matrix

Architecture	CUDA Capabilities	First CUDA Toolkit Support	Last CUDA Toolkit Support	Last Driver Support
Fermi	2.0	CUDA 3.0	CUDA 8.0	R390
Kepler	3.0 3.2	CUDA 6.0	CUDA 10.2	R470
Kepler	3.5 3.7	CUDA 6.0	CUDA 11.x	R470
Maxwell	5.0 5.2 5.3	CUDA 6.5	Ongoing	Ongoing
Pascal	6.0 6.1	CUDA 8.0	Ongoing	Ongoing
Volta	7.0	CUDA 9.0	Ongoing	Ongoing
Turing	7.5	CUDA 10.0	Ongoing	Ongoing
Ampere	8.0 8.6	CUDA 11.0	Ongoing	Ongoing
Ada	8.9	CUDA 11.8	Ongoing	Ongoing
Hopper	9.0	CUDA 11.8 CUDA 12.0	Ongoing	Ongoing

如何选择显卡驱动版本

根据确定的 CUDA 版本来选择显卡的驱动版本，如下图所示。例如您选择的 CUDA 版本为 11.4.3，使用 linux 操作系统时，驱动版本应大于 450.80.02。详情请参见 https://docs.nvidia.com/datacenter/tesla/drivers/index.html#cuda-drivers。

CUDA Toolkit and Minimum Required Driver Version for CUDA Minor Version Compatibility

CUDA Toolkit	Minimum Required Driver Version for CUDA Minor Version Compatibility*	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 12.2.x	>=525.60.13	>=525.41
CUDA 12.1.x	>=525.60.13	>=527.41
CUDA 12.0.x	>=525.60.13	>=527.41
CUDA 11.8.x	>=450.80.02	>=452.39
CUDA 11.7.x	>=450.80.02	>=452.39
CUDA 11.6.x	>=450.80.02	>=452.39
CUDA 11.5.x	>=450.80.02	>=452.39
CUDA 11.4.x	>=450.80.02	>=452.39
CUDA 11.3.x	>=450.80.02	>=452.39
CUDA 11.2.x	>=450.80.02	>=452.39
CUDA 11.1 (11.1.0)	>=450.80.02	>=452.39
CUDA 11.0 (11.0.3)	>=450.36.06**	>=451.22**

如何选择 cuDNN 版本

NVIDIA CUDA 深度神经网络库（cuDNN）是一个 GPU 加速的深度神经网络基元库，能够以高度优化的方式实现标准例程（如前向和反向卷积、池化层、归一化和激活层）。借助 cuDNN，研究人员和开发者可以专注于训练神经网络及开发软件应用，而不必花时间进行低层级的 GPU 性能调整。cuDNN 可加速

广泛应用的深度学习框架，包括 Caffe2、Chainer、Keras、MATLAB、MxNet、PaddlePaddle、PyTorch 和 TensorFlow。根据选择的 CUDA 版本选择对应的 cuDNN 版本，版本对应关系及 cuDNN 下载地址可参考如下链接：cuDNN Archive | NVIDIA Developer。

如何选择 Pytorch 版本

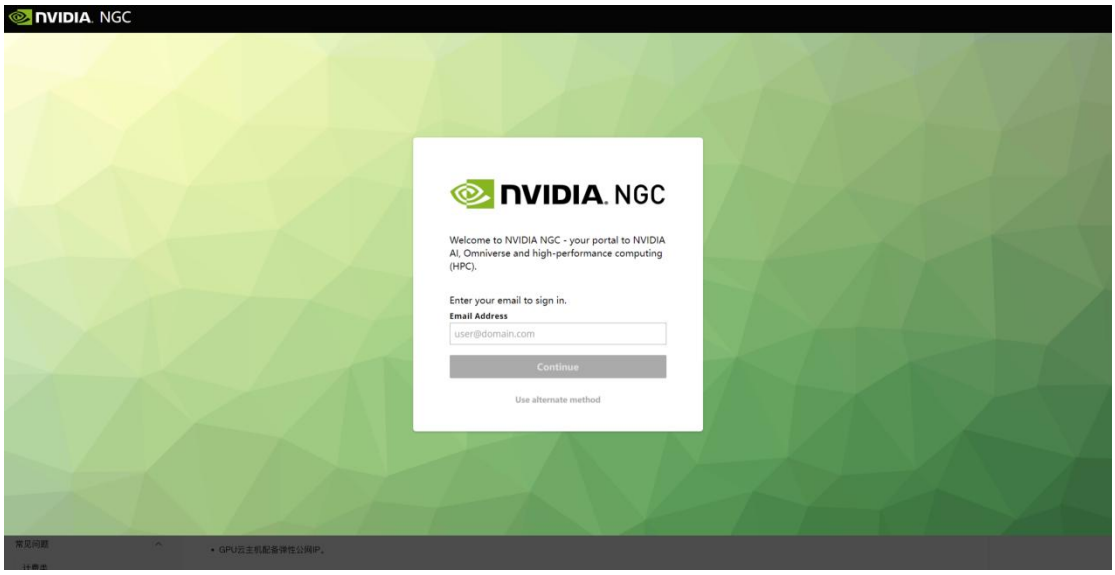
根据选择的 CUDA 版本选择对应的 Pytorch 版本，版本对应关系可参考如下链接：Previous PyTorch Versions | PyTorch。

4.2在 GPU 实例上部署 NGC 环境

NVIDIA NGC 是用于深度学习、机器学习和 HPC 的 GPU 优化软件的中心，可提供容器、模型、模型脚本和行业解决方案，以便数据科学家、开发人员和研究人员可以专注于更快地构建解决方案和收集见解。

前提条件

- 用户需要注册 NGC 的账号：https://ngc.nvidia.com/signin



- GPU 云主机配备弹性公网 IP。

安装步骤

1.创建一台 GPU 云主机，操作方法请参见创建未配备 GPU 驱动的 GPU 云主机。

2.安装 GPU 云主机驱动， 建议安装最新版本的操作系统驱动，操作方法请参见 NVIDIA 驱动安装指引。

3.安装 Docker 和针对 NVIDIA GPU 的 Docker Utility Engine，即 nvidia-docker。

Docker 的安装方法可以参考 Ubuntu、CentOS。这里我们以 CentOS 为例进行操作步骤说明。

a. 在安装 Docker 新版本之前，请卸载所有的旧版本以及关联的依赖项。

```
sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
```

```
[root@ecm-aeba ~]# sudo yum remove docker \
>                   docker-client \
>                   docker-client-latest \
>                   docker-common \
>                   docker-latest \
                  docker-engine
>                   docker-latest-logrotate \
>                   docker-logrotate \
>                   docker-engine
Modular dependency problems:

 Problem 1: conflicting requests
  - nothing provides module(perl:5.26) needed by module perl-IO-Socket-SSL:2.066:8040020200924212038:1aedcbfe-0.x86_64
 Problem 2: conflicting requests
  - nothing provides module(perl:5.26) needed by module perl-libwww-perl:6.34:8040020211102170116:bf75fe78-0.x86_64
No match for argument: docker
No match for argument: docker-client
No match for argument: docker-client-latest
No match for argument: docker-common
No match for argument: docker-latest
No match for argument: docker-latest-logrotate
No match for argument: docker-logrotate
No match for argument: docker-engine
No packages marked for removal.
Dependencies resolved.
Nothing to do.
Complete!
[root@ecm-aeba ~]#
```

b. 设置 Docker 存储库。

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo

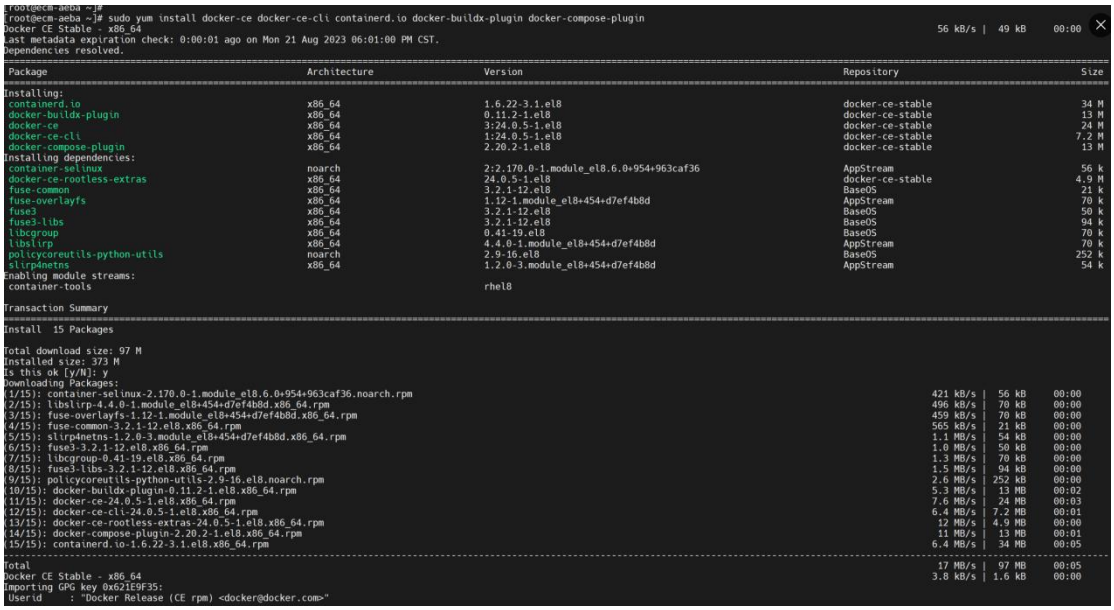
https://download.docker.com/linux/centos/docker-ce.repo

```
[root@ecm-aeba ~]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[root@ecm-aeba ~]#
[root@ecm-aeba ~]#
```

c. 安装 Docker 引擎。

sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin

docker-compose-plugin



d. 启动 docker。

sudo systemctl start docker

e. 安装 nvidia-docker。

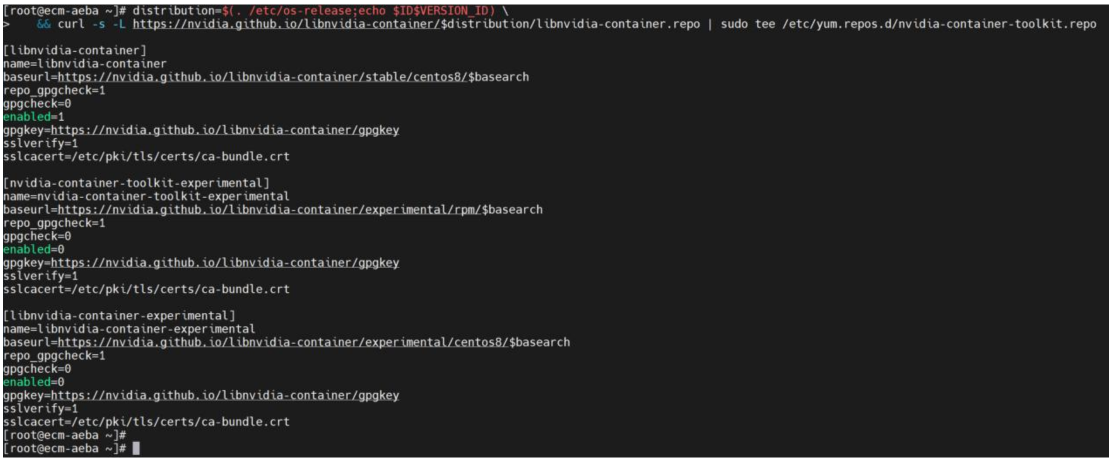
设置存储库和 GPG 密钥。

distribution=$(. /etc/os-release;echo IDVERSION_ID) \

&& curl -s -L

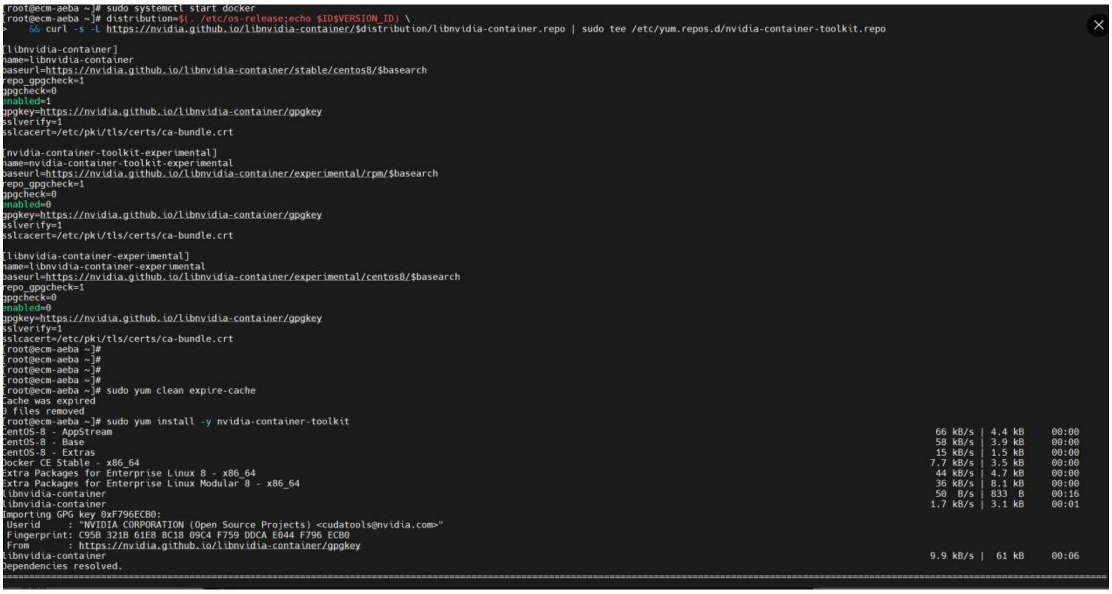
https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.repo | sudo tee

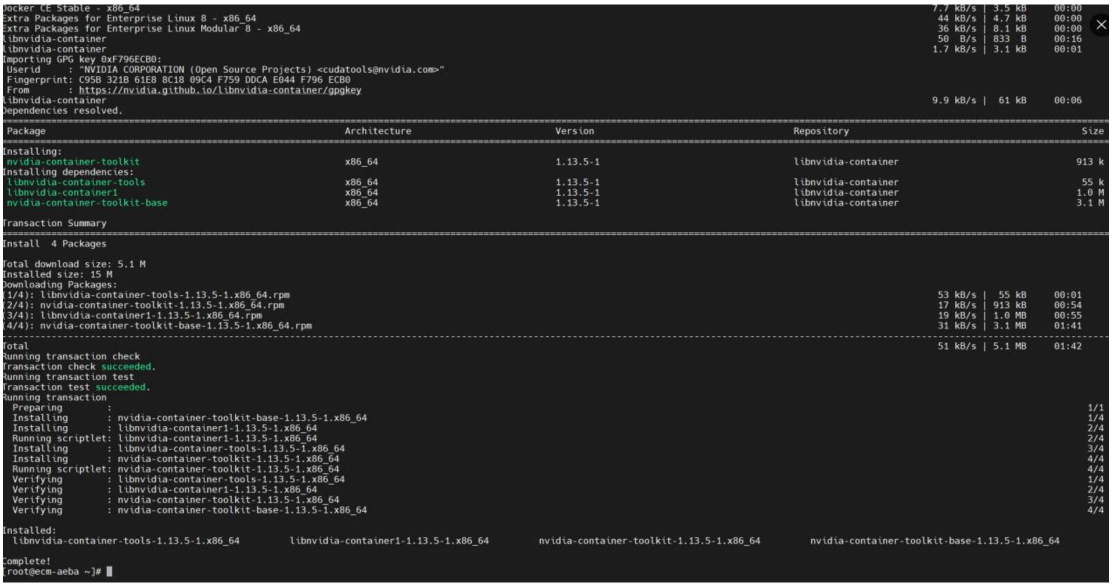
/etc/yum.repos.d/nvidia-container-toolkit.repo



更新包列表后安装 nvidia-container-toolkit 包（和依赖项）。

```
sudo yum clean expire-cachesudo yum install -y nvidia-container-toolkit
```





配置 Docker 守护程序以识别 NVIDIA 容器运行时。

```
sudo nvidia-ctk runtime configure --runtime=docker

sudo systemctl restart docker
```

```
[root@ecm-aeba ~]# sudo nvidia-ctk runtime configure --runtime=docker
INFO[0000] Loading docker config from /etc/docker/daemon.json
INFO[0000] Config file does not exist, creating new one
INFO[0000] Wrote updated config to /etc/docker/daemon.json
INFO[0000] It is recommended that the docker daemon be restarted.
[root@ecm-aeba ~]# sudo systemctl restart docker
[root@ecm-aeba ~]#
```

通过运行基本 CUDA 容器来测试工作设置。

```
sudo docker run --rm --runtime=nvidia --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04

nvidia-smi
```

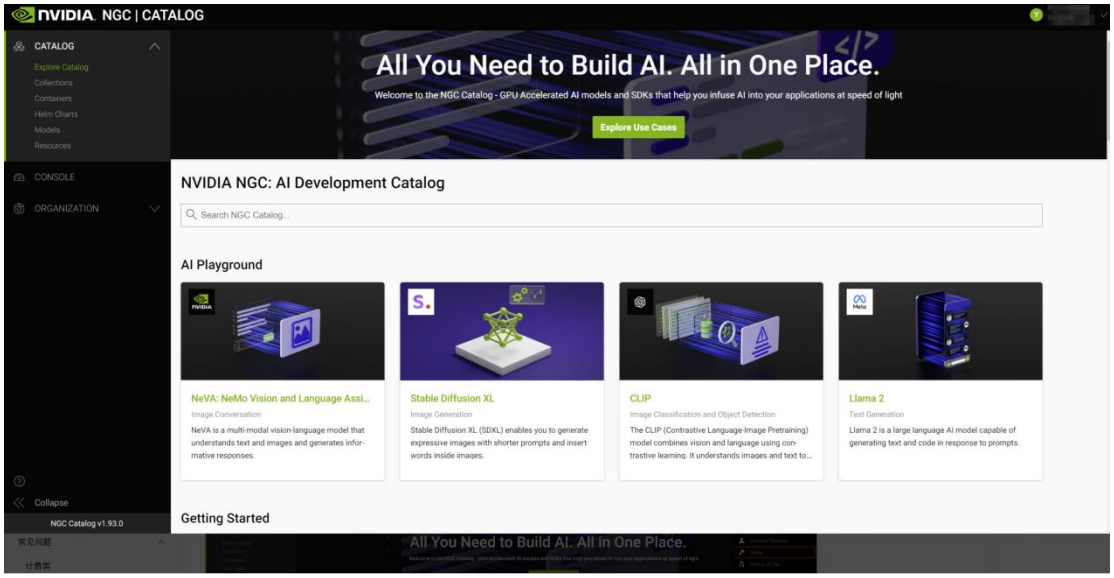
```
[root@ecm-aeba ~]#
[root@ecm-aeba ~]# sudo docker run --rm --runtime=nvidia --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
Unable to find image 'nvidia/cuda:11.6.2-base-ubuntu20.04' locally
11.6.2-base-ubuntu20.04: Pulling from nvidia/cuda
56e0351b9876: Pull complete
0e353182dfa4: Pull complete
63add13c711b: Pull complete
1210b79751b0: Pull complete
eb1e2ff09225: Pull complete
Digest: sha256:4b0c83c0f2e66dc97b52f28c7acf94c1461bfa746d56a6f63c0fef5035590429
Status: Downloaded newer image for nvidia/cuda:11.6.2-base-ubuntu20.04
Mon Aug 21 10:24:38 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.199.02   Driver Version: 470.199.02   CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| 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  Tesla T4            Off  | 00000000:00:08.0 Off |                    0 |
| N/A   38C    P0    26W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[root@ecm-aeba ~]#
```

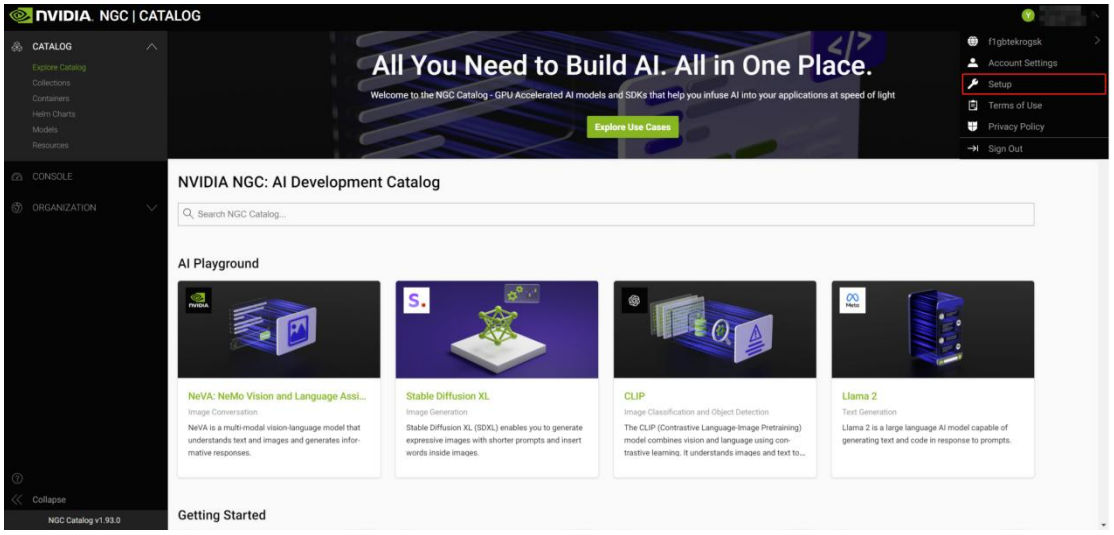
使用 NVIDIA NGC

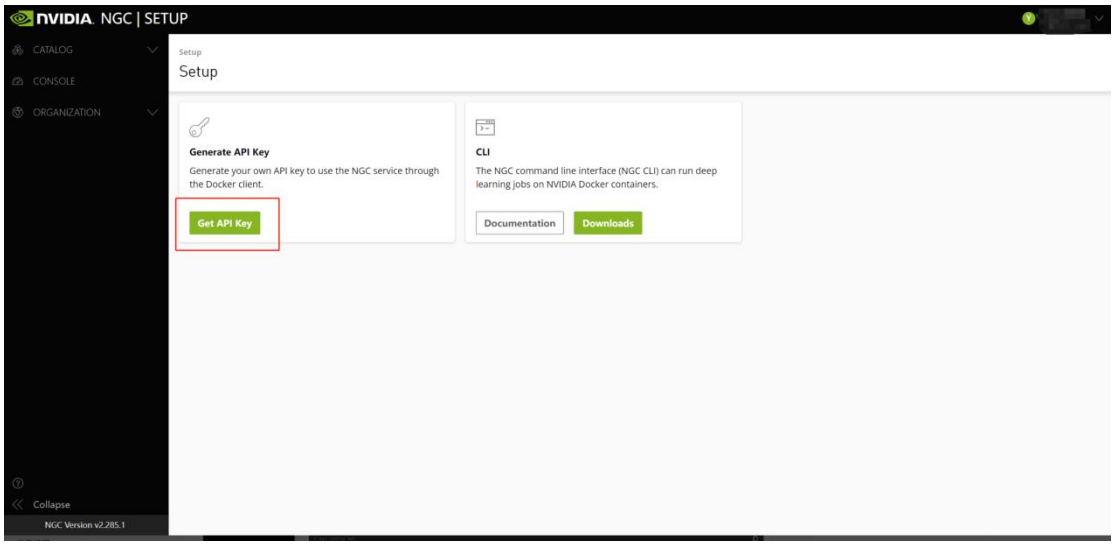
1.生成 NGC 的 API key 。

a. 在 https://ngc.nvidia.com/signin 成功注册完 NGC 账号之后，需要生成账户的 API key。

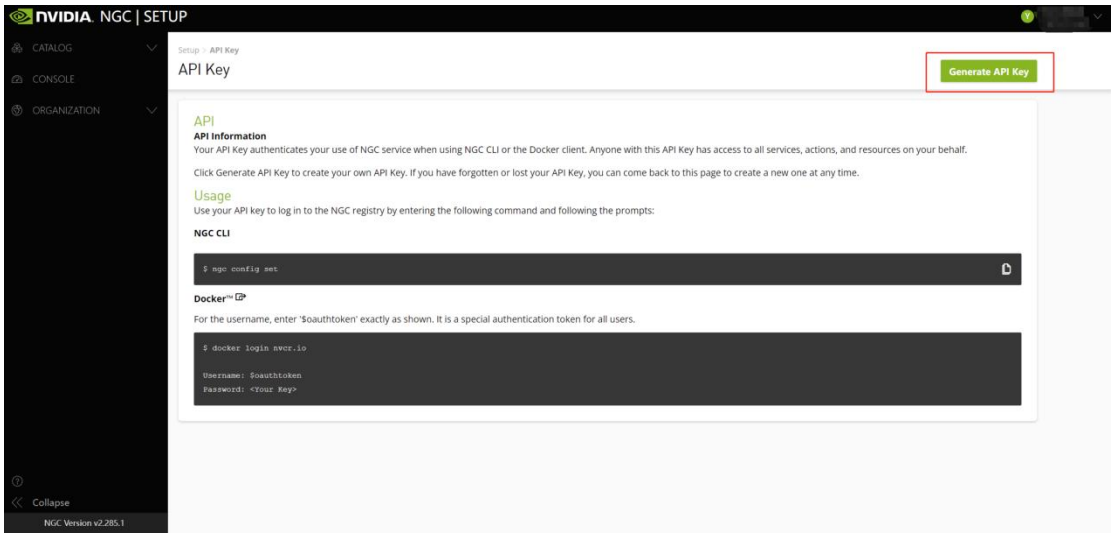
登录 NGC 页面，单击“账户名”，选择“Setup”，会进入 Setup 页面，然后单击“Get API Key”，进入生成 API Key 的页面。



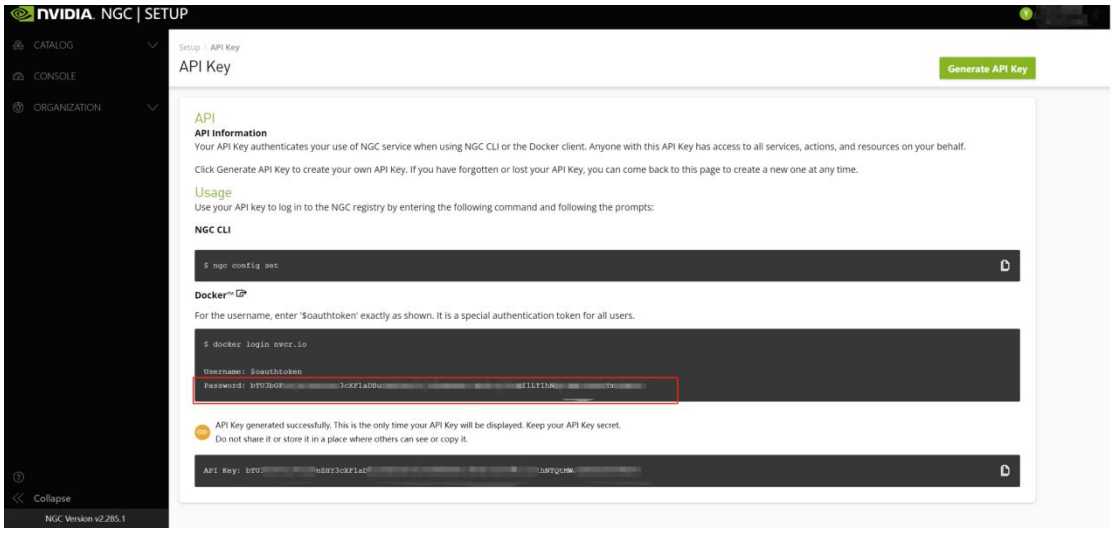




b. 在 API Key 的页面，单击“Generate API Key”，进入确认对话框。



c. 在确认对话框，单击“Confirm”，页面会变为类似于下图所示的页面。

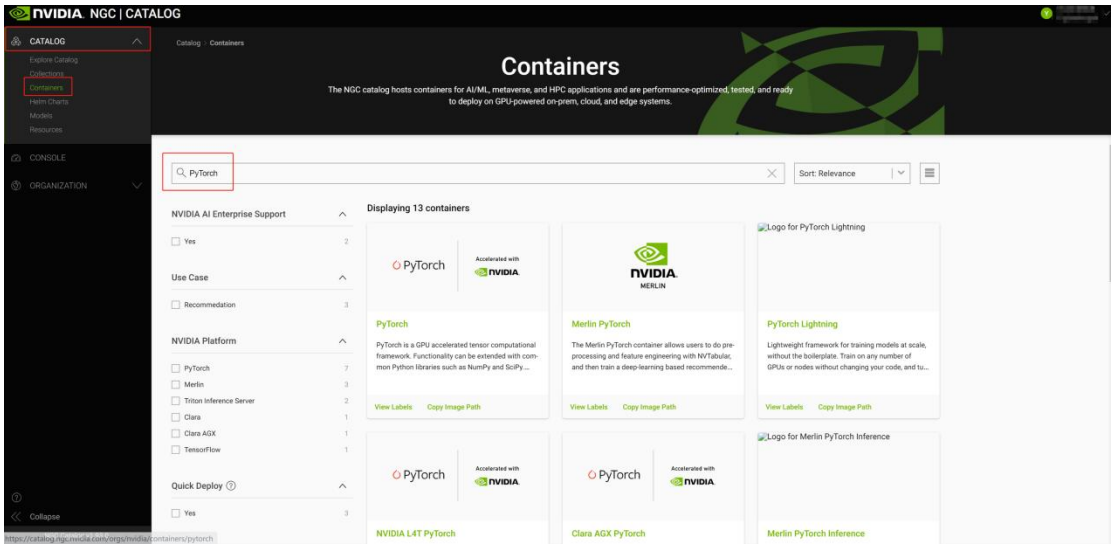


d. 在 Password 处会显示一连串密码，用户返回 GPU 实例的 shell 界面按照图中的操作即可。

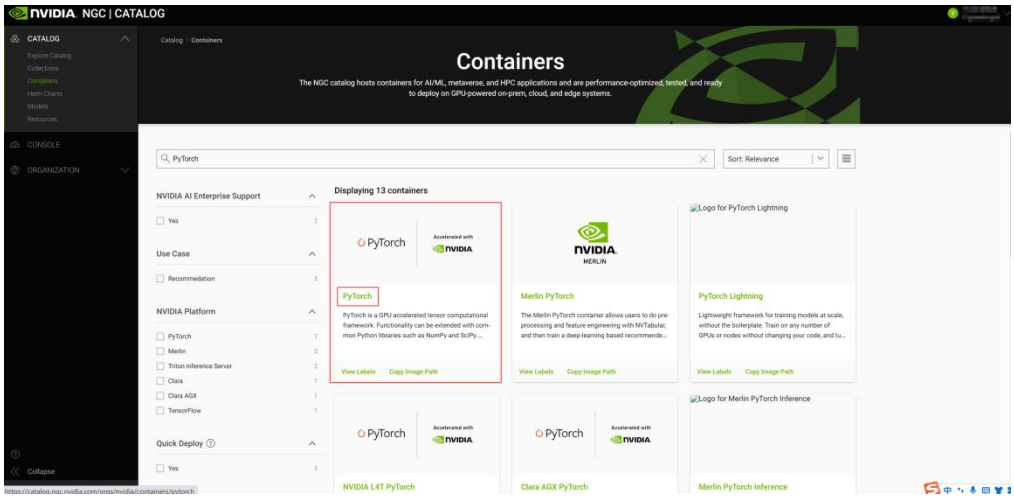
$ docker login nvcr.ioUsername: $oauthtokenPassword: 【输入生成的秘钥】

```
root at yinhao in ~
$ docker login nvcr.io
Username: $oauthtoken
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

root at yinhao in ~
$
```

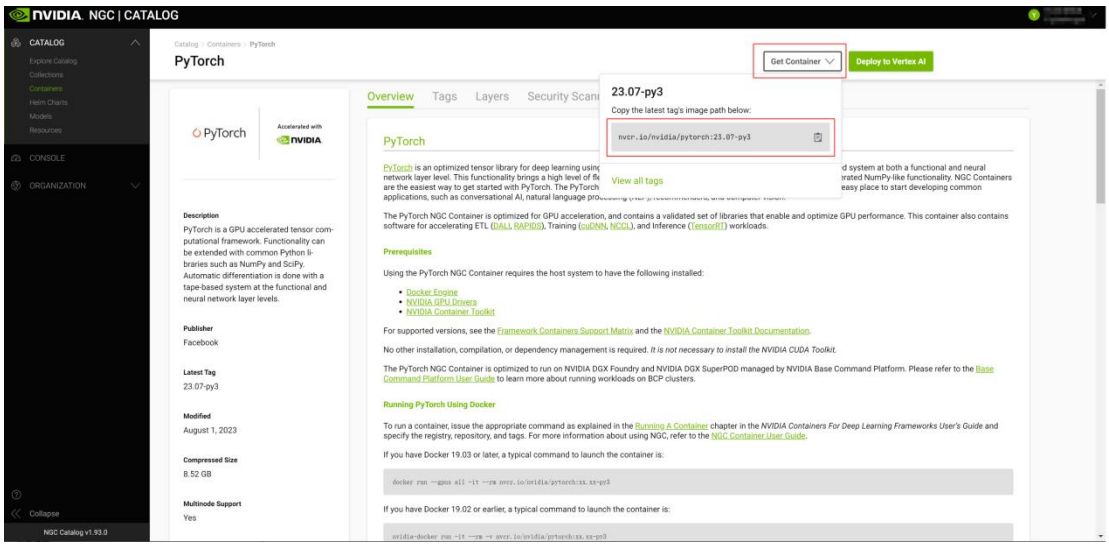
1.使用 NGC 中的镜像（以 PyTorch 为例）。



a. 进入 NGC 的 CATALOG 的目录部分，选择 CONTAINERS 分支，在 Query 查询中输入 PyTorch，并单击“PyTorch”。

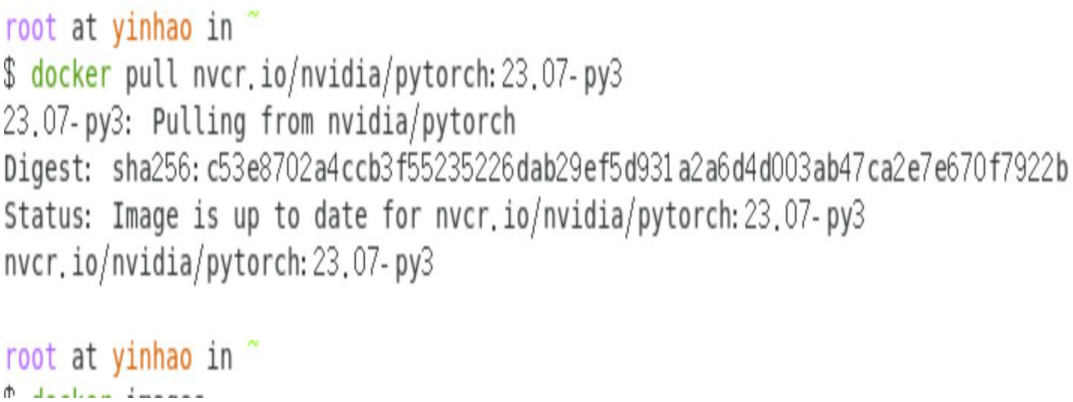


b. 单击“Get Container”，关于容器的拉取镜像的方法则会展示出来。



c. 按照上图中红色方框中的命令，可以获得最新版本的容器镜像，继续在 GPU 实例的命令行中输入以下命令。

```
$ docker pull nvcr.io/nvidia/pytorch:23.07-py3
```



这样，我们就可以用 docker 容器的方式去使用框架或软件产品了。

4.3使用 Docker 安装 TensorFlow 并设置 GPU/CPU 支持

您可通过 Docker 快速在 GPU 实例上运行 TensorFlow，该方式仅需实例已安装 NVIDIA 驱动程序，无需安装 NVIDIA CUDA 工具包。

前提条件

- GPU 云主机已安装 GPU 驱动。
- 本文操作步骤以 CentOS 操作系统为例。

操作步骤

1.安装 Docker

a. 在安装 Docker 新版本之前，请卸载任何此类旧版本以及关联的依赖项。

```
sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
```

```
                docker-logrotate \
                docker-engine
```

```
[root@ecm-aeba ~]# sudo yum remove docker \
>                   docker-client \
>                   docker-client-latest \
>                   docker-common \
>                   docker-latest \
                  docker-engine
>                   docker-latest-logrotate \
>                   docker-logrotate \
>                   docker-engine
Modular dependency problems:

 Problem 1: conflicting requests
  - nothing provides module(perl:5.26) needed by module perl-IO-Socket-SSL:2.066:8040020200924212038:1aedcbfe-0.x86_64
 Problem 2: conflicting requests
  - nothing provides module(perl:5.26) needed by module perl-libwww-perl:6.34:8040020211102170116:bf75fe78-0.x86_64
No match for argument: docker
No match for argument: docker-client
No match for argument: docker-client-latest
No match for argument: docker-common
No match for argument: docker-latest
No match for argument: docker-latest-logrotate
No match for argument: docker-logrotate
No match for argument: docker-engine
No packages marked for removal.
Dependencies resolved.
Nothing to do.
Complete!
[root@ecm-aeba ~]#
```

b. 设置 Docker 存储库。

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo

https://download.docker.com/linux/centos/docker-ce.repo

```
[root@ecm-aeba ~]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[root@ecm-aeba ~]#
[root@ecm-aeba ~]#
```

c. 安装 Docker 引擎。

sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin

docker-compose-plugin

```
[root@ecm-aeba ~]#
[root@ecm-aeba ~]# sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Docker CE Stable - x86_64                                                              56 kB/s |  49 kB     00:00
Last metadata expiration check: 0:00:01 ago on Mon 21 Aug 2023 06:01:00 PM CST.
Dependencies resolved.
========================================================================================================================
 Package                        Architecture   Version                                      Repository            Size
========================================================================================================================
Installing:
 containerd.io                  x86_64         1.6.22-3.1.el8                               docker-ce-stable      34 M
 docker-buildx-plugin           x86_64         0.11.2-1.el8                                 docker-ce-stable      13 M
 docker-ce                      x86_64         3:24.0.5-1.el8                               docker-ce-stable      24 M
 docker-ce-cli                  x86_64         1:24.0.5-1.el8                               docker-ce-stable     7.2 M
 docker-compose-plugin          x86_64         2.20.2-1.el8                                 docker-ce-stable      13 M
Installing dependencies:
 container-selinux              noarch         2:2.170.0-1.module_el8.6.0+954+963caf36      AppStream             56 k
 docker-ce-rootless-extras      x86_64         24.0.5-1.el8                                 docker-ce-stable     4.9 M
 fuse-common                    x86_64         3.2.1-12.el8                                 BaseOS                21 k
 fuse-overlayfs                 x86_64         1.12-1.module_el8+454+d7ef4b8d               AppStream             70 k
 fuse3                          x86_64         3.2.1-12.el8                                 BaseOS                50 k
 fuse3-libs                     x86_64         3.2.1-12.el8                                 BaseOS                94 k
 libcgroup                      x86_64         0.41-19.el8                                  BaseOS                70 k
 libslirp                       x86_64         4.4.0-1.module_el8+454+d7ef4b8d              AppStream             70 k
 policycoreutils-python-utils   noarch         2.9-16.el8                                   BaseOS               252 k
 slirp4netns                    x86_64         1.2.0-3.module_el8+454+d7ef4b8d              AppStream             54 k
Enabling module streams:
 container-tools                               rhel8

Transaction Summary
========================================================================================================================
Install  15 Packages

Total download size: 97 M
Installed size: 373 M
Is this ok [y/N]: y
Downloading Packages:
(1/15): container-selinux-2.170.0-1.module_el8.6.0+954+963caf36.noarch.rpm            421 kB/s |  56 kB     00:00
(2/15): libslirp-4.4.0-1.module_el8+454+d7ef4b8d.x86_64.rpm                           496 kB/s |  70 kB     00:00
(3/15): fuse-overlayfs-1.12-1.module_el8+454+d7ef4b8d.x86_64.rpm                      459 kB/s |  70 kB     00:00
(4/15): fuse-common-3.2.1-12.el8.x86_64.rpm                                           565 kB/s |  21 kB     00:00
(5/15): slirp4netns-1.2.0-3.module_el8+454+d7ef4b8d.x86_64.rpm                        1.1 MB/s |  54 kB     00:00
(6/15): fuse3-3.2.1-12.el8.x86_64.rpm                                                 1.0 MB/s |  50 kB     00:00
(7/15): libcgroup-0.41-19.el8.x86_64.rpm                                              1.3 MB/s |  70 kB     00:00
(8/15): fuse3-libs-3.2.1-12.el8.x86_64.rpm                                            1.5 MB/s |  94 kB     00:00
(9/15): policycoreutils-python-utils-2.9-16.el8.noarch.rpm                            2.6 MB/s | 252 kB     00:00
(10/15): docker-buildx-plugin-0.11.2-1.el8.x86_64.rpm                                 5.3 MB/s |  13 MB     00:02
(11/15): docker-ce-24.0.5-1.el8.x86_64.rpm                                            7.6 MB/s |  24 MB     00:03
(12/15): docker-ce-cli-24.0.5-1.el8.x86_64.rpm                                        6.4 MB/s | 7.2 MB     00:01
(13/15): docker-ce-rootless-extras-24.0.5-1.el8.x86_64.rpm                             12 MB/s | 4.9 MB     00:00
(14/15): docker-compose-plugin-2.20.2-1.el8.x86_64.rpm                                 11 MB/s |  13 MB     00:01
(15/15): containerd.io-1.6.22-3.1.el8.x86_64.rpm                                      6.4 MB/s |  34 MB     00:05
------------------------------------------------------------------------------------------------------------------------
Total                                                                                  17 MB/s |  97 MB     00:05
Docker CE Stable - x86_64                                                             3.8 kB/s | 1.6 kB     00:00
Importing GPG key 0x621E9F35:
 Userid     : "Docker Release (CE rpm) <docker@docker.com>"
```

d. 启动 docker。

sudo systemctl start docker

2.安装 TensorFlow

a. 设置 NVIDIA 容器工具包

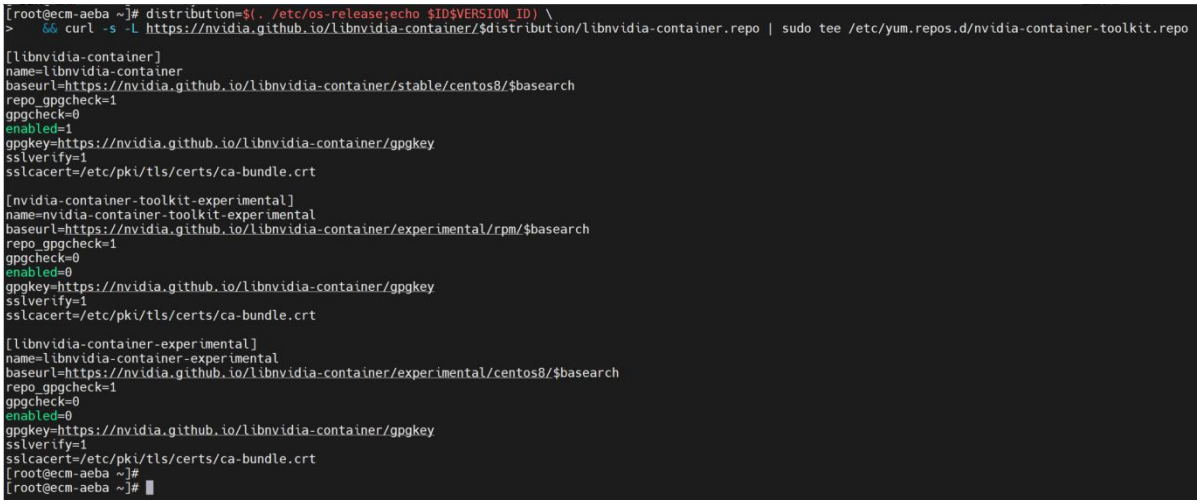
设置存储库和 GPG 密钥。

distribution=$(. /etc/os-release;echo IDVERSION_ID) \

&& curl -s -L

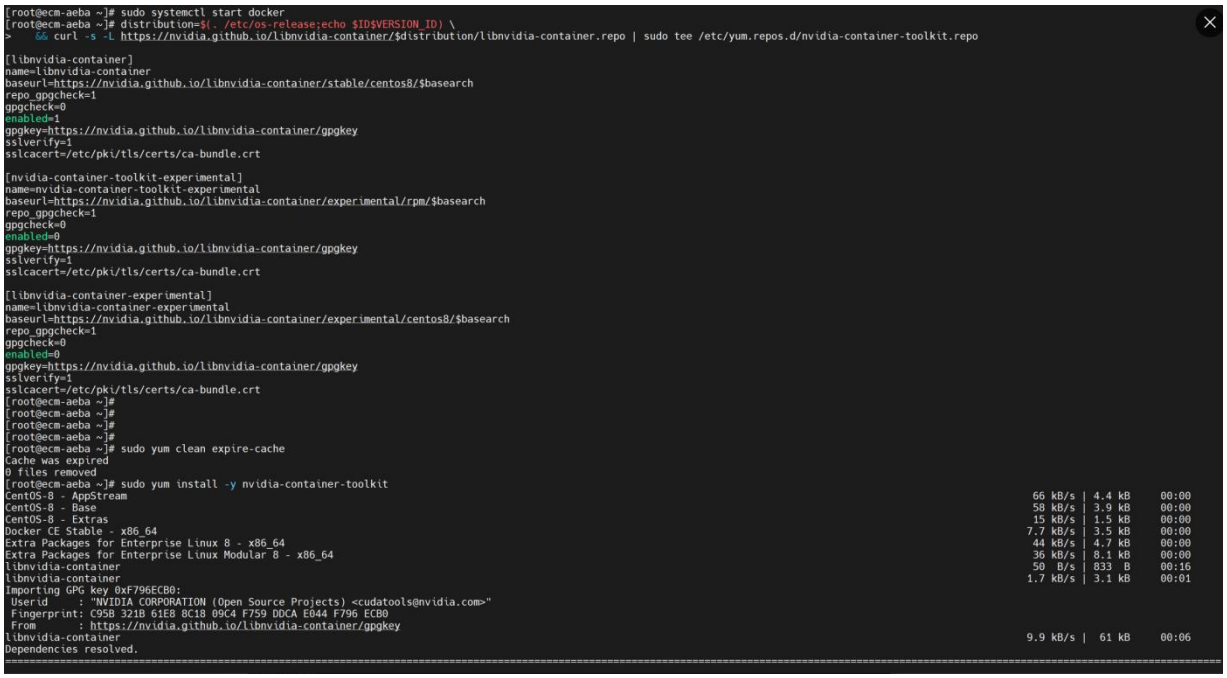
https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.repo | sudo tee

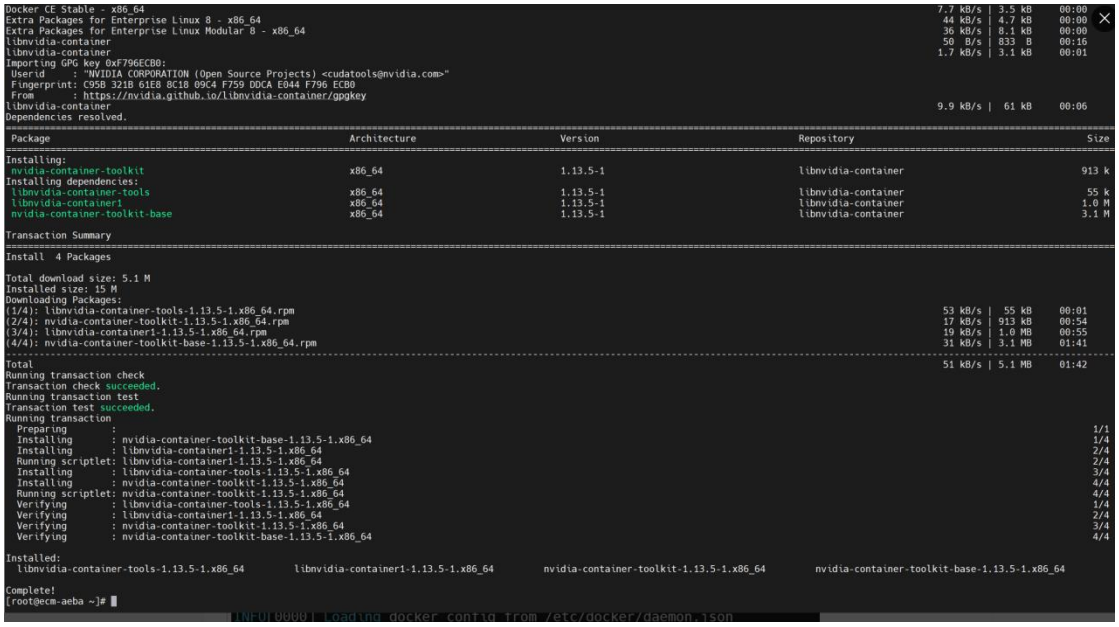
/etc/yum.repos.d/nvidia-container-toolkit.repo



更新包列表后安装 nvidia-container-toolkit 包（和依赖项）。

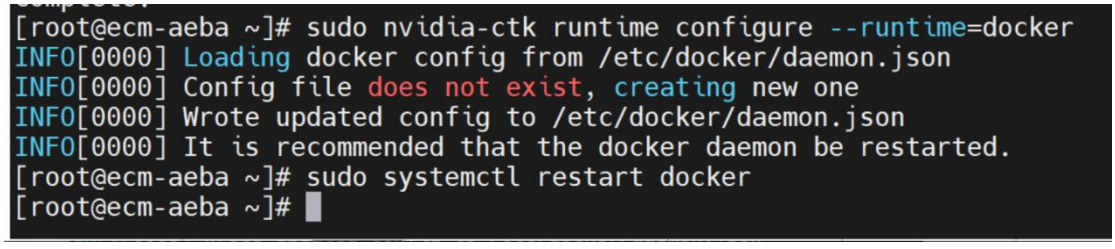
```
sudo yum clean expire-cachesudo yum install -y nvidia-container-toolkit
```





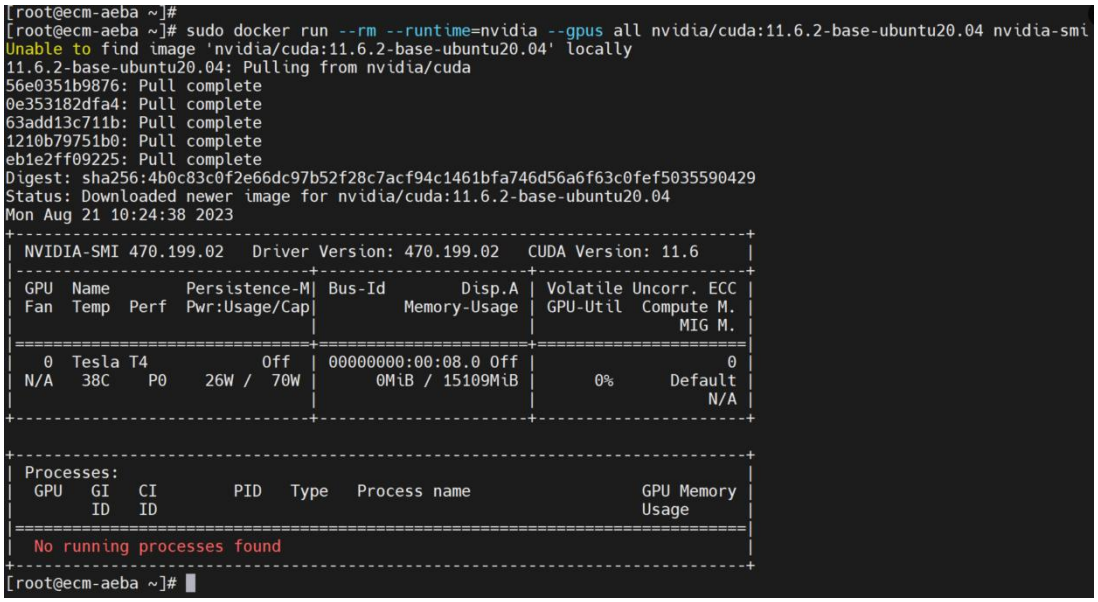
配置 Docker 守护程序以识别 NVIDIA 容器运行时。

```
sudo nvidia-ctk runtime configure --runtime=docker

sudo systemctl restart docker
```



通过运行基本 CUDA 容器来测试工作设置。

```
sudo docker run --rm --runtime=nvidia --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04
nvidia-smi
```



b.下载 TensorFlow Docker 镜像

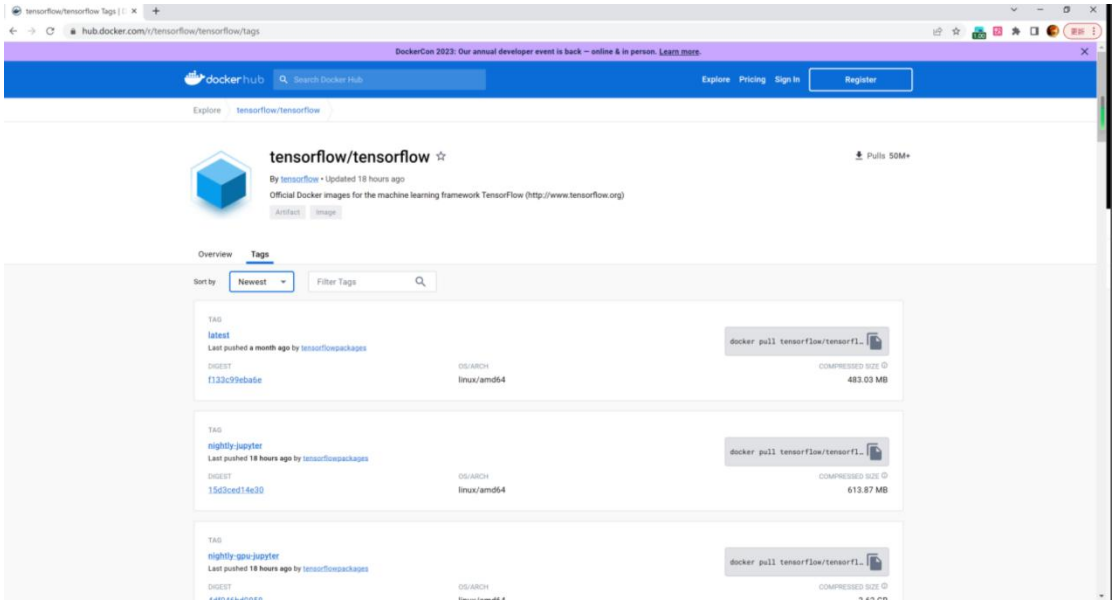
官方 TensorFlow Docker 镜像位于 tensorflow/tensorflow Docker Hub 代码库中。镜像版本按照以下格式进行 标记：

标记	说明
latest	标签包含最新版本（不包括候选版本、alpha 和 beta 等预发布版本）
nightly	附带最新的 TensorFlow nightly Python 包
version	指定 TensorFlow 二进制镜像的版本，例如 2.13.0

标记	说明
devel	标签不再支持，请改用 TensorFlow SIG Build Dockerfiles
custom-op	标签不再支持，请改用 TensorFlow SIG Build Dockerfiles

每个基本标记都有会添加或更改功能的变体：

标记变体	说明
tag-gpu	标签基于 Nvidia CUDA。您需要 nvidia-docker 来运行它们
tag-jupyter	标签包括 Jupyter 和一些 TensorFlow 教程笔记本



您可以一次使用多个变体。例如，以下命令会将 TensorFlow 版本镜像下载到计算机上。

```
docker pull tensorflow/tensorflow                    # latest stable release
docker pull tensorflow/tensorflow:nightly-gpu        # nightly release w/ GPU support
```

docker pull tensorflow/tensorflow:latest-gpu-jupyter　　# latest release w/ GPU support and Jupyter

```
[root@ecm-aeba ~]# docker pull tensorflow/tensorflow
Using default tag: latest
latest: Pulling from tensorflow/tensorflow
01085d60b3a6: Pull complete
de96f27d9487: Pull complete
0d0dce5452b7: Pull complete
3b190c0764b5: Pull complete
9e55d77b5a31: Pull complete
eb5c0fde2e19: Pull complete
1eb5af93509e: Pull complete
4a60f8dff7fd: Pull complete
85ba1cd0f140: Pull complete
4983425daedd: Pull complete
d10bf76e378a: Pull complete
17f02e3f1db1: Pull complete
Digest: sha256:f133c99eba6e59b921ea7543c81417cd831c9983f5d6ce65dff7adb0ec79d830
Status: Downloaded newer image for tensorflow/tensorflow:latest
docker.io/tensorflow/tensorflow:latest
[root@ecm-aeba ~]#
```

```
[root@ecm-aeba ~]#
[root@ecm-aeba ~]# docker pull tensorflow/tensorflow:nightly-gpu
nightly-gpu: Pulling from tensorflow/tensorflow
6b851dcae6ca: Already exists
4586c00479c6: Already exists
4304fa233a80: Already exists
afa3f70b397f: Already exists
d963a42bc712: Already exists
550ae4bb69cb: Already exists
d2ba79382571: Already exists
493b2c98b564: Already exists
ea49ca3afa74: Already exists
c80204be0995: Pull complete
523faa97f655: Pull complete
a9b37bafb59e: Pull complete
829d7d3748ef: Pull complete
a85798bfdece: Pull complete
d9c084415a37: Pull complete
2407f7c0a2ed: Pull complete
b99a6f3de523: Pull complete
da75bcc1c93a: Pull complete
Digest: sha256:dbeda2e714a3ff0bb1f3bf82a173c016b60ce42c9431cd9c0ddb8bfd911a138e
Status: Downloaded newer image for tensorflow/tensorflow:nightly-gpu
docker.io/tensorflow/tensorflow:nightly-gpu
[root@ecm-aeba ~]#
```

c.启动 TensorFlow Docker 容器

启动配置 TensorFlow 的容器，请使用以下命令格式。如需了解更多信息，请参见 Docker run reference。

docker run [-it] [--rm] [-p hostPort:containerPort] tensorflow/tensorflow[:tag] [command]

示例

1.使用仅支持 CPU 的镜像的示例

如下所示，使用带 latest 标记的镜像验证 TensorFlow 安装效果。Docker 会在首次运行时下载新的 TensorFlow 镜像：

```
docker run -it --rm tensorflow/tensorflow python
```

```
[root@ecm-aeba ~]# docker run -it --rm tensorflow/tensorflow python
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

其他 TensorFlow Docker 方案示例如下：

在配置 TensorFlow 的容器中启动 bash shell 会话：

```
docker run -it tensorflow/tensorflow bash
```

```
[root@ecm-aeba ~]# docker run -it --rm tensorflow/tensorflow python
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
KeyboardInterrupt
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>>
[root@ecm-aeba ~]#
[root@ecm-aeba ~]#
[root@ecm-aeba ~]#
[root@ecm-aeba ~]# docker run -it tensorflow/tensorflow bash

________                               _______________
___  __/__________________________________  ____/__  /________      __
__  /  _  _ \_  __ \_  ___/  __ \_  ___/_  /_   __  /_  __ \_ | /| / /
_  /   /  __/  / / /(__  )/ /_/ /  /   _  __/   _  / / /_/ /_ |/ |/ /
/_/    \___//_/ /_//____/ \____//_/    /_/      /_/  \____/____/|__/


WARNING: You are running this container as root, which can cause new files in
mounted volumes to be created as the root user on your host machine.

To avoid this, run the container by specifying your user's userid:

$ docker run -u $(id -u):$(id -g) args...

root@1fecdbf355cd:/#
```

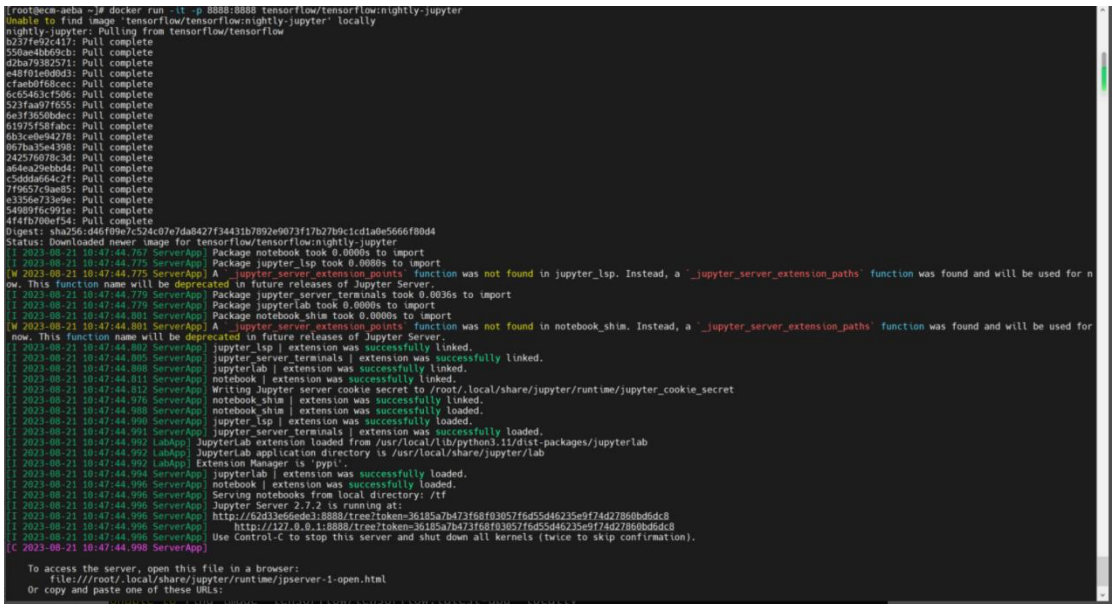
如需在容器内运行在主机上开发的 TensorFlow 程序，请通过 -v hostDir:containerDir -w workDir 参数，装载主机目录并更改容器的工作目录。示例如下：

```
docker run -it --rm -v $PWD:/tmp -w /tmp tensorflow/tensorflow python ./script.py
```

使用 nightly 版 TensorFlow 启动 Jupyter 笔记本服务器：

```
docker run -it -p 8888:8888 tensorflow/tensorflow:nightly-jupyter
```

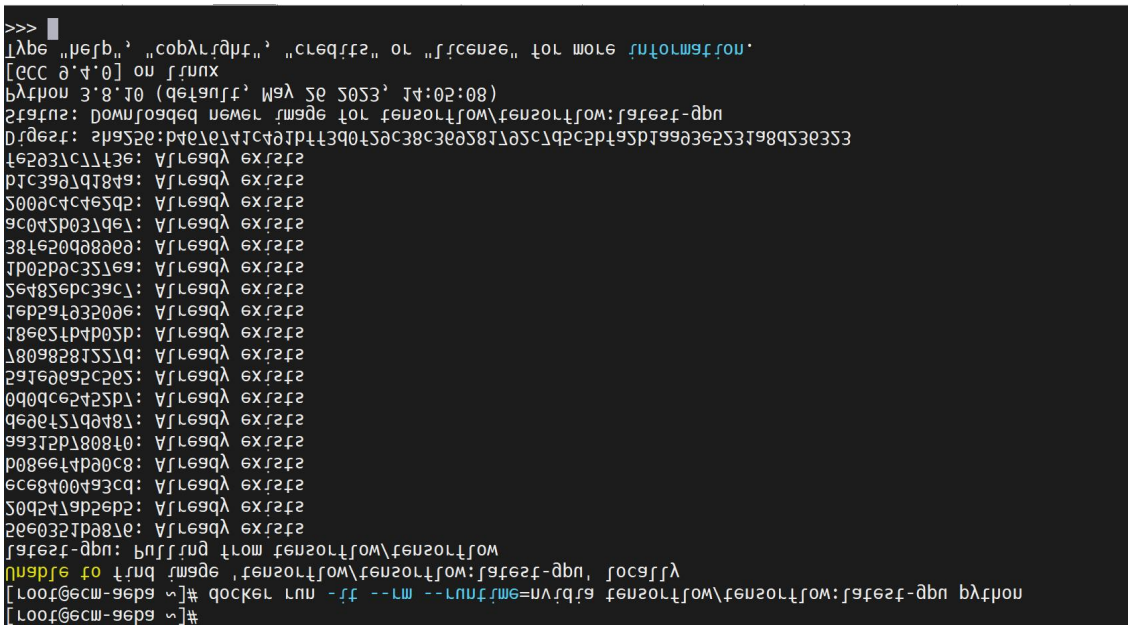
请参考 Jupyter 官网 相关说明，使用浏览器访问 http://127.0.0.1:8888/?token=...。



2.使用支持 GPU 的镜像的示例

a. 执行以下命令，下载并运行支持 GPU 的 TensorFlow 镜像，并使用 Python 解释器启动。

```
docker run -it --rm --runtime=nvidia tensorflow/tensorflow:latest-gpu python
```



b. 设置支持 GPU 镜像可能需要一段时间。如果重复运行基于 GPU 的脚本，您可以使用 **docker exec** 重复使用容器。执行以下命令，使用最新的 TensorFlow GPU 镜像在容器中启动 **bash** shell 会话：

```
docker run --gpus all -it tensorflow/tensorflow:latest-gpu bash
```

```
[root@ecm-aeba ~]#
[root@ecm-aeba ~]# docker run --gpus all -it tensorflow/tensorflow:latest-gpu bash


WARNING: You are running this container as root, which can cause new files in
mounted volumes to be created as the root user on your host machine.

To avoid this, run the container by specifying your user's userid:

$ docker run -u $(id -u):$(id -g) args...

root@e81eb7313f74:/#
```

4.4安装 CUDA

CUDA 是 NVIDIA 推出的通用并行计算架构，帮助您使用 NVIDIA GPU 解决复杂的计算问题。您可参考如下操作说明安装 CUDA 工具包。CUDA 版本的选择请参见如何选择驱动及相关库、软件版本。

前提条件

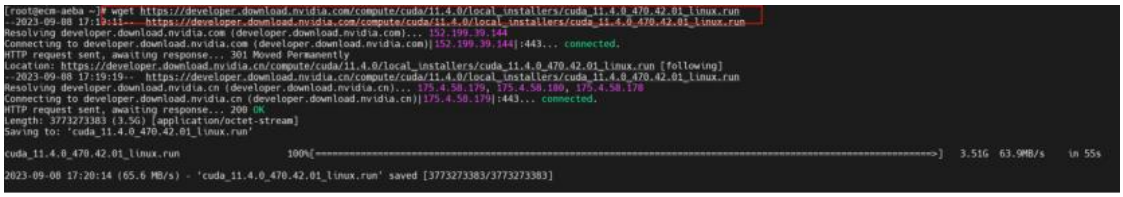
- GPU 云主机配备弹性 IP。

在 Linux 操作系统中安装 CUDA

1.获取 cuda 安装包下载链接。访问 CUDA 下载官网，选择对应的 CUDA 版本，依次选择操作系统和安装包，复制 cuda 工具包下载链接。本文以 centos 系统为例，cuda 版本为 11.4.0 为例。

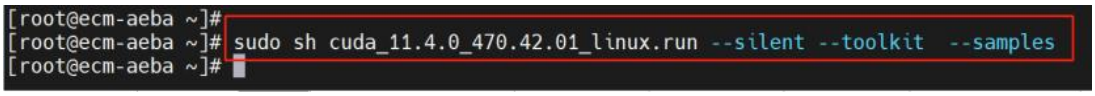
2.输入如下命令下载 CUDA 安装包。

```
wget
https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda_11.4.0_470.42.01_linux.run
```



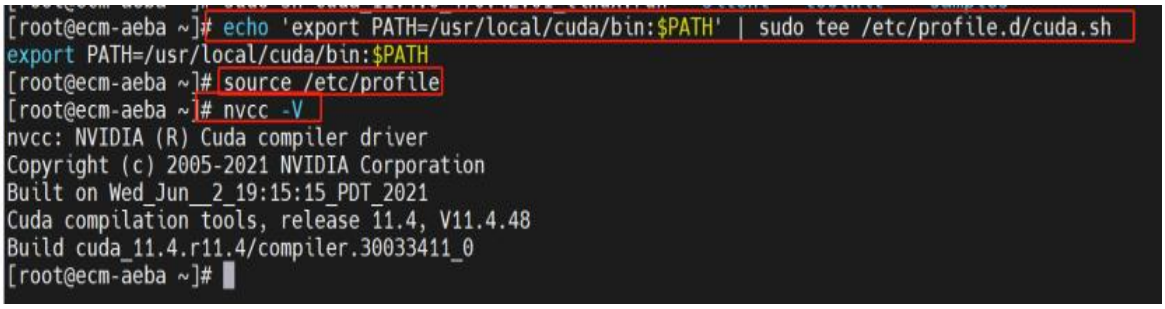
3.输入如下命令安装 CUDA 工具包。

```
sudo sh cuda_11.4.0_470.42.01_linux.run --silent --toolkit  --samples
```



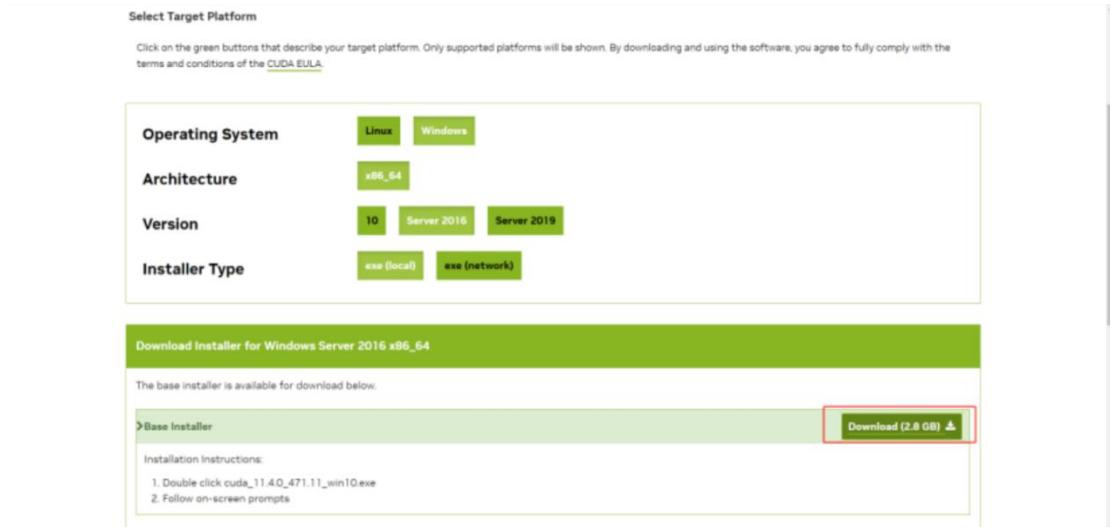
4.配置环境变量后查看 CUDA 版本，出现如下结果则说明 CUDA 安装成功。

```
echo 'export PATH=/usr/local/cuda/bin:$PATH' | sudo tee /etc/profile.d/cuda.shsource

/etc/profile

nvcc -V
```



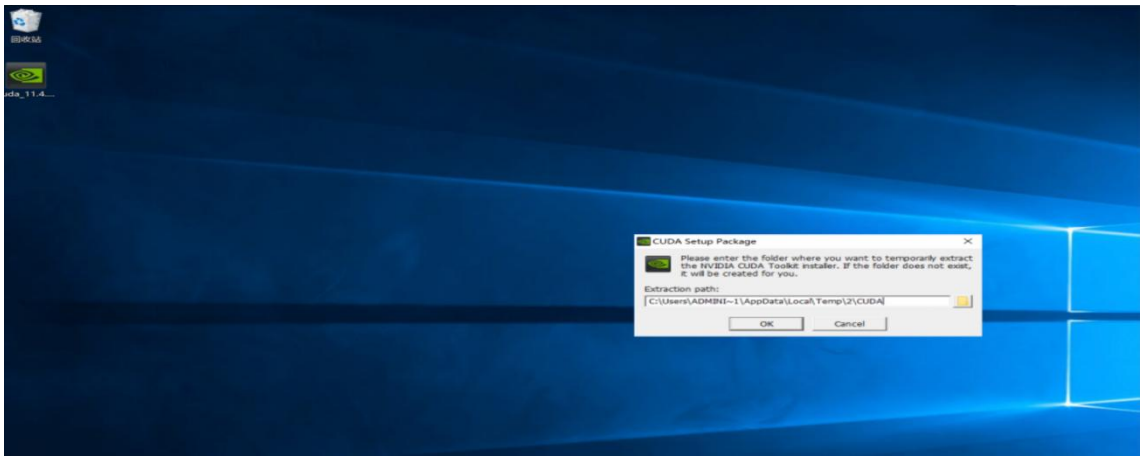
在 Windows 操作系统中安装 CUDA

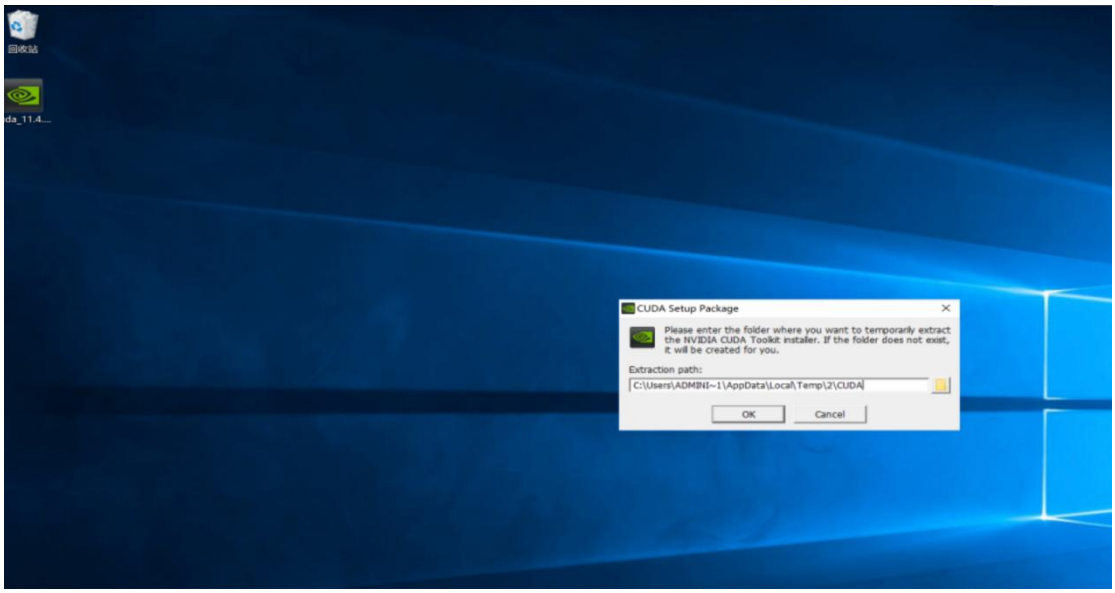
1. 下载对应CUDA安装包。访问CUDA 下载官网,选择对应的CUDA 版本。本文以Windows Server 2016 x86_64 版本为例进行安装。



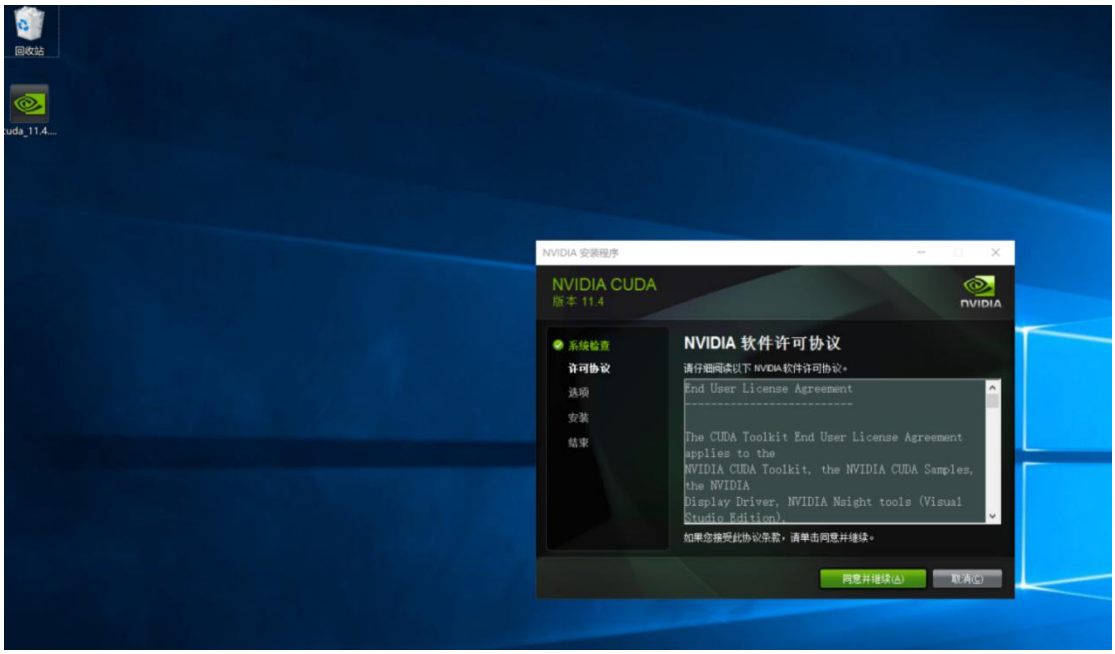
2.双击 cuda_11.4.0_471.11_win10 文件开始安装。

3.选择安装路径，单击“OK”。



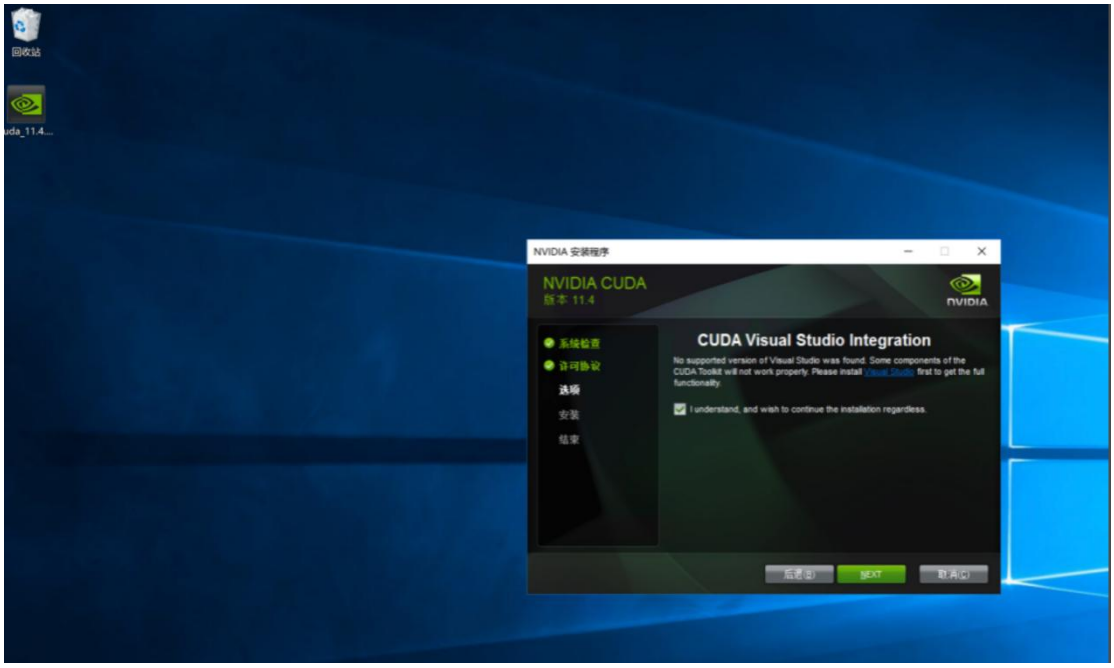


4.单击“同意并继续”。

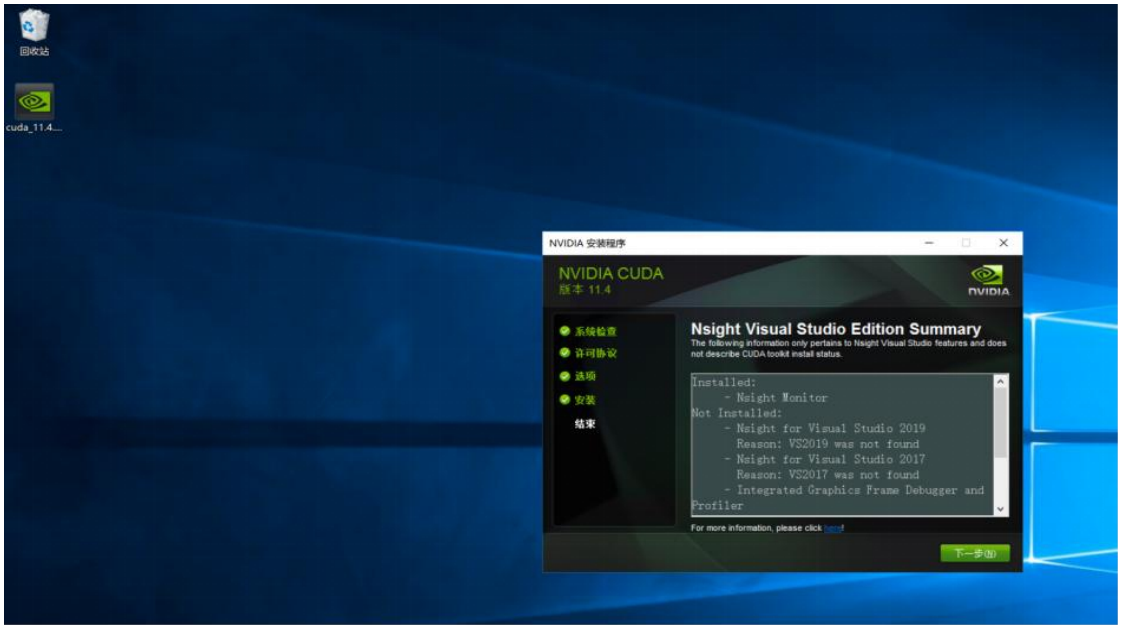


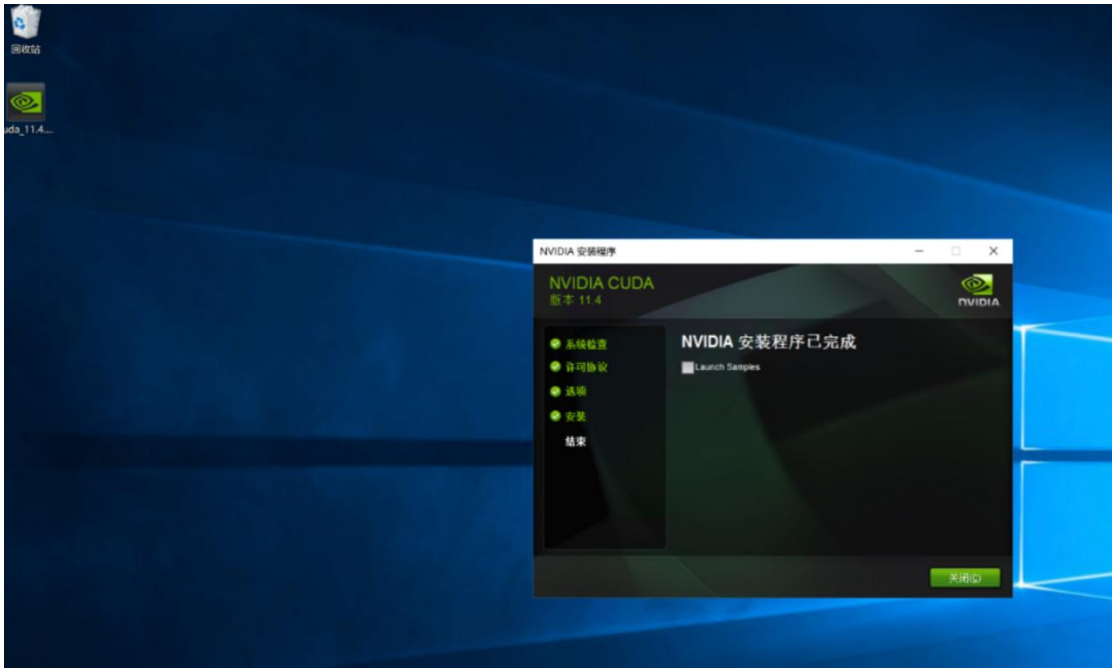
5.单击“下一步”。

6.勾选后单击“NEXT”。



7.单击“下一步”，出现下图表示安装成功。





4.5使用 Windows GPU 云主机搭建深度学习环境

背景信息

实例环境如下表所示。

实例类型	**pi2.2xlarge.4**
操作系统	Windows Server 2019 数据中心版 64 位 中文版
CPU	8vCPU
内存	32GB
GPU	NVIDIA T4 * 1 张
驱动及相关库、软件版本	CUDA11.3.0、Python 3.9、cuDNN8.2.1、Pytorch 1.11.0、Tensorflow_gpu_2.6.0

说明

如何选择对应版本请参见如何选择驱动及相关库、软件版本。

操作步骤

步骤一： 创建 GPU 实例

请参见用户指南- >创建 GPU 云主机 >创建未配备驱动的 GPU 云主机，创建 GPU 云主机实例。

步骤二：安装显卡驱动

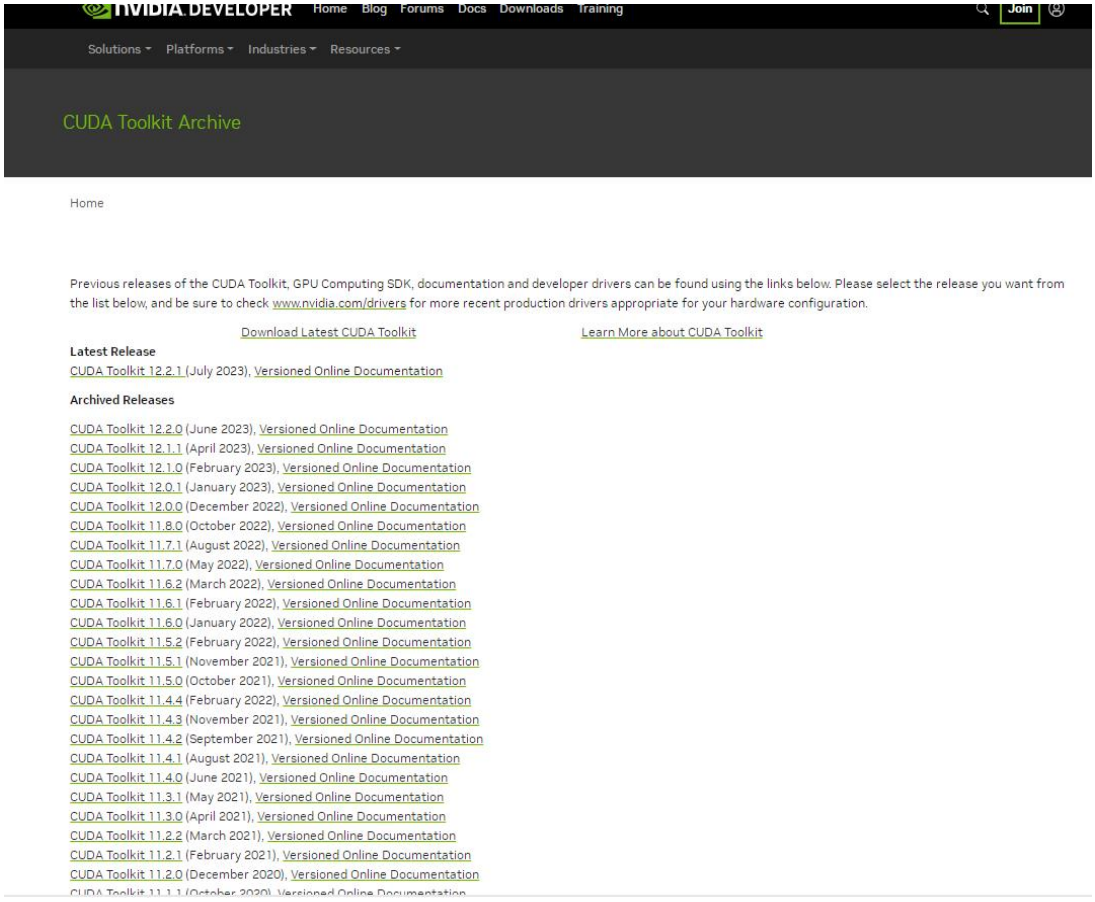
1.登录已创建的 GPU 云主机，操作参见 Windows 弹性云主机登录方式概述。

2.访问 NVIDIA 官网，选择显卡的驱动版本。单击“SEARCH”进入下载页面，单击进行下载。

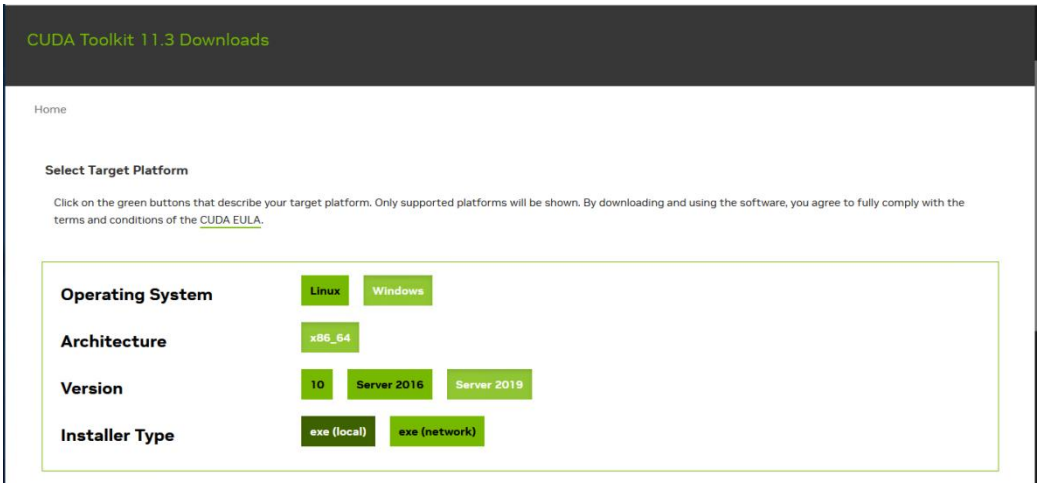
3.完成下载后，根据提示完成安装。

步骤三：安装 CUDA

1.访问英伟达官网 CUDA Toolkit Archive，选择对应版本。

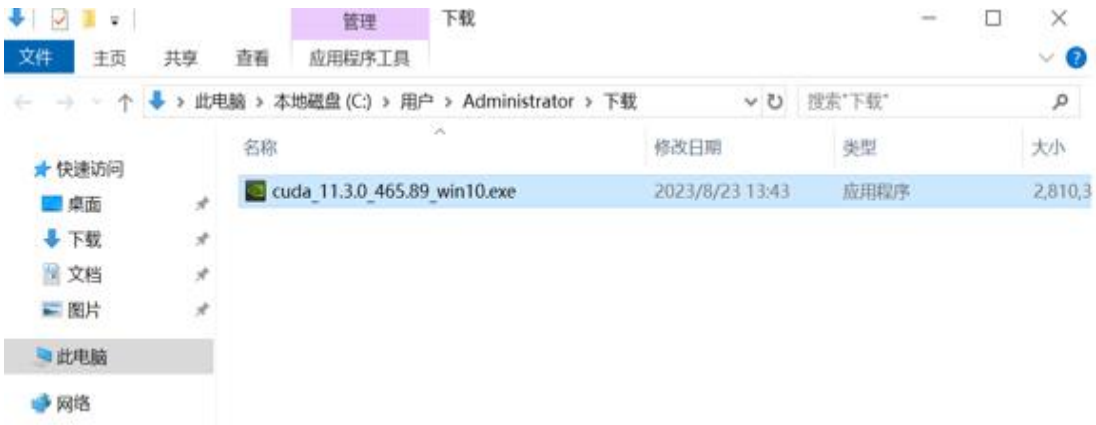


2.进入 CUDA Toolkit 11.3.0 Download 页面，选择对应系统配置。



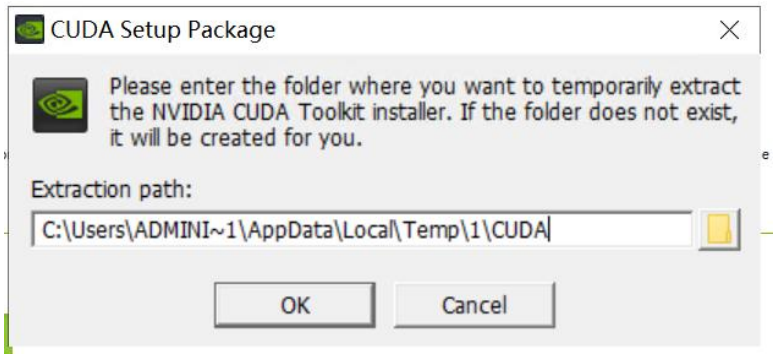
3.单击 “Download”，开始下载。

4.下载完成后，请双击安装包，并根据提示进行安装。



请注意以下步骤：

在弹出的 CUDA Setup Package 窗口中，Extraction path 为暂时存放地址，无需修改，保持默认并单击 OK。



在许可协议步骤中，选择“自定义”并单击“下一步”。

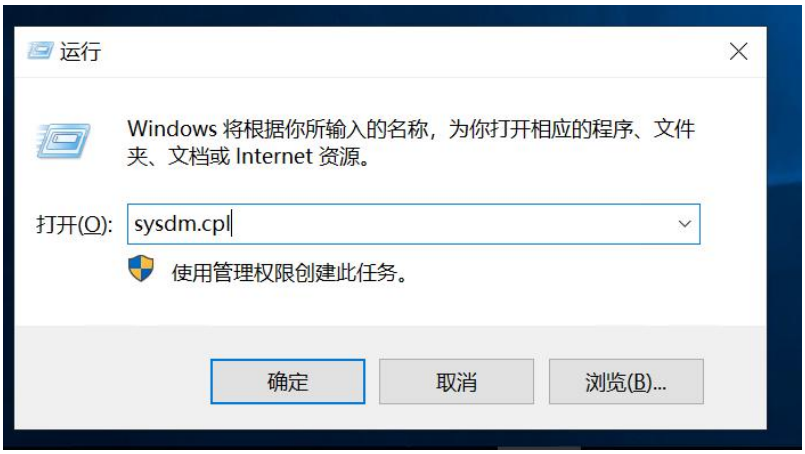
根据实际需求选择安装组件，并单击“下一步”。

完成安装，根据提示重启云主机。

步骤四：配置环境变量

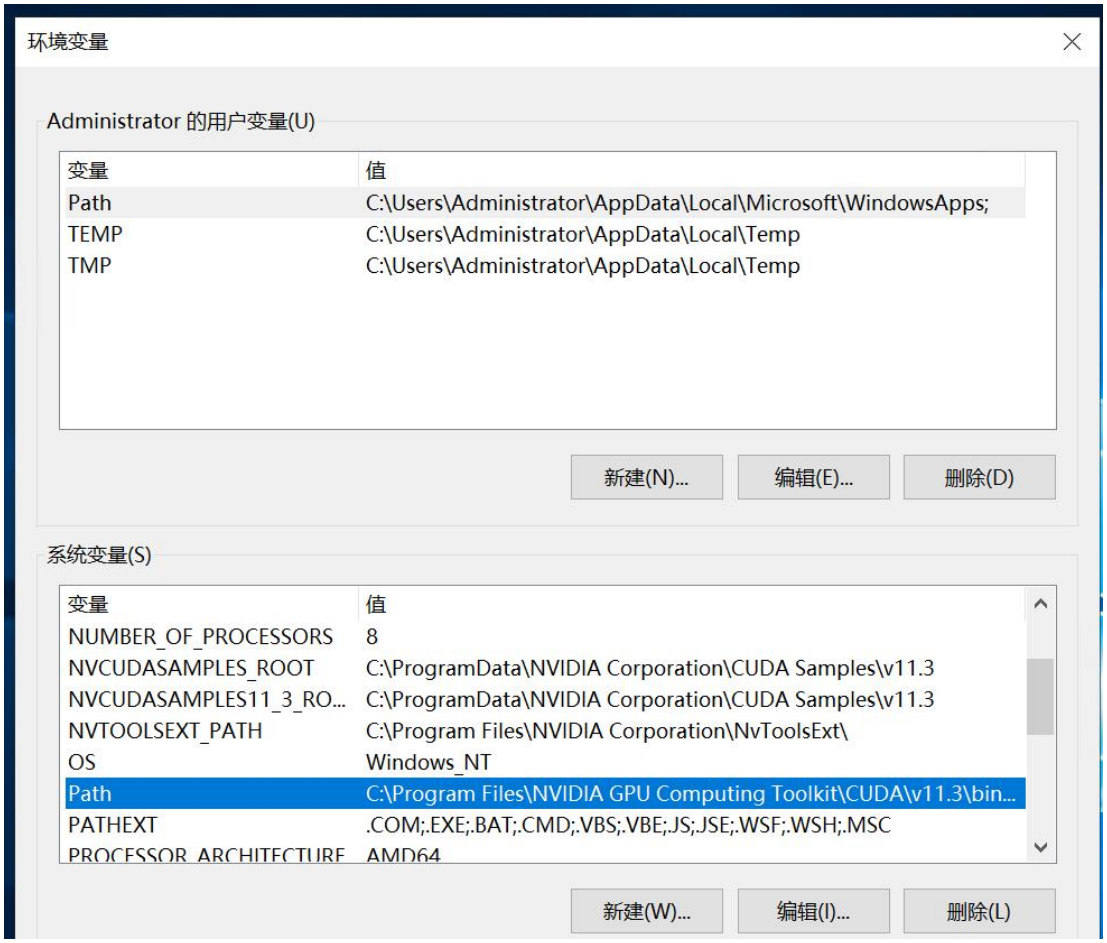
1.在操作系统界面　　使用“win +R”快捷键打开运行。

2.在运行窗口中输入 sysdm.cpl，并单击“确定”。



3.在打开的系统属性窗口中，选择“高级”页签，并单击“环境变量”。

4.选择系统变量中的“Path”，单击“编辑”。



5.在弹出的编辑环境变量窗口中，新建并输入如下环境变量配置（部分已有的无需再次新建）。

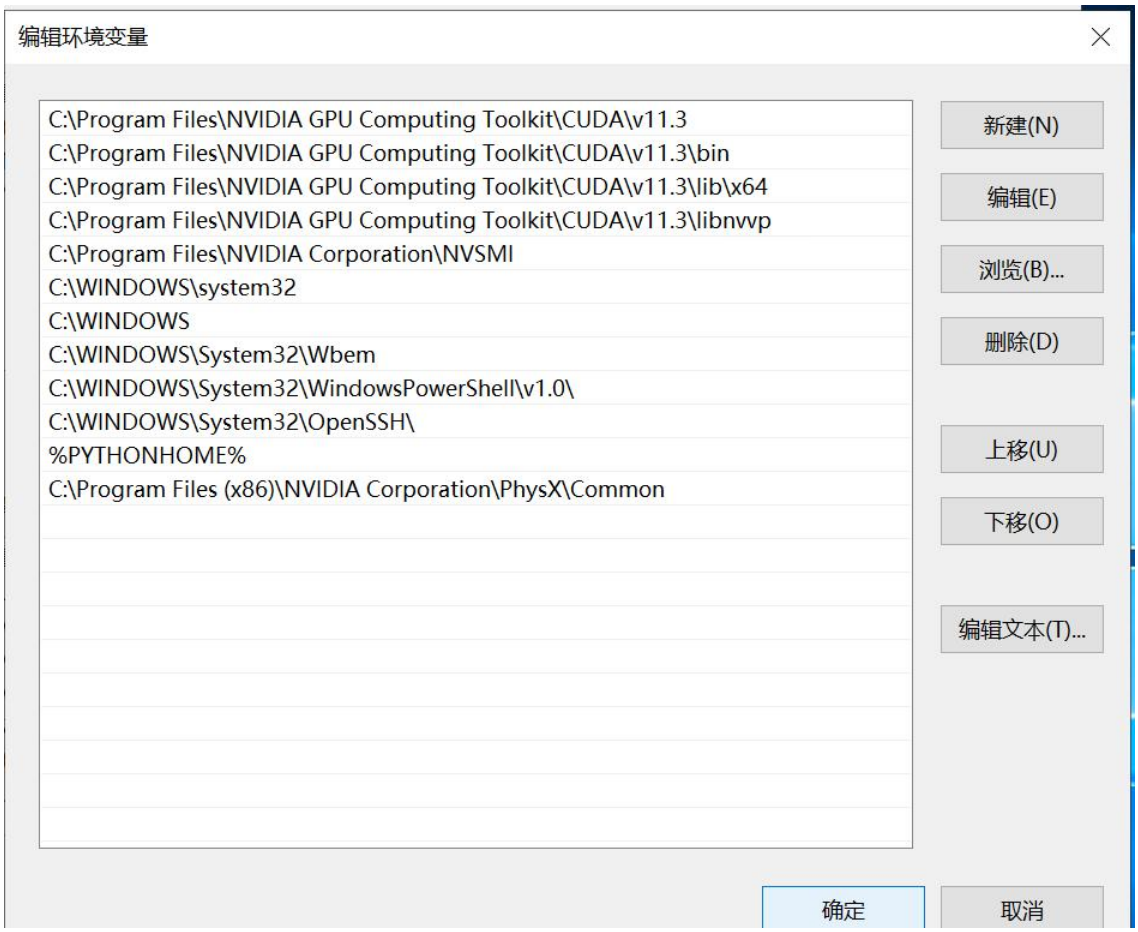
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.3

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.3\bin

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.3\libnvvp

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.3\lib\x64

C:\Program Files\NVIDIA Corporation\NVSMI

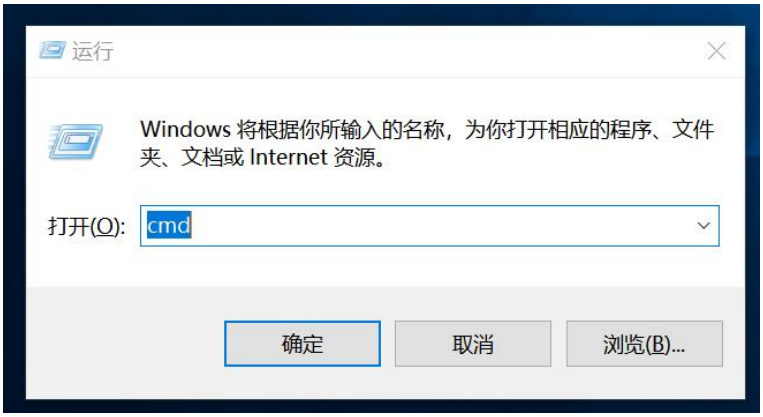


6.连续单击 3 次“确定”，保存设置。

步骤五：检查显卡驱动及 CUDA

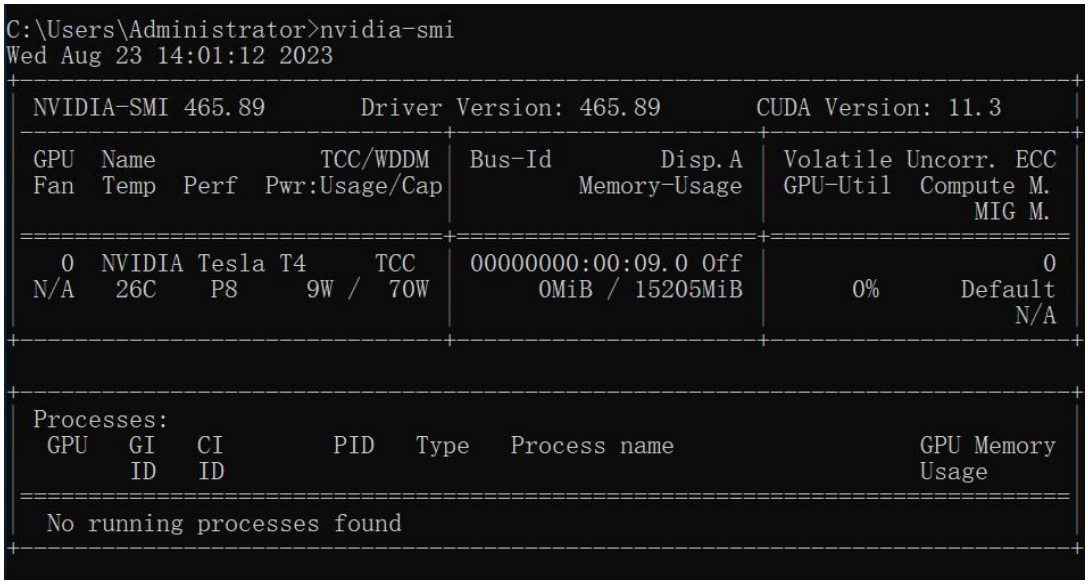
1.在操作系统界面使用“win +R”快捷键打开运行。

2.在运行窗口中输入 cmd，并单击“确定”。



3.在 cmd 窗口中，执行以下命令，检查显卡驱动是否安装成功。

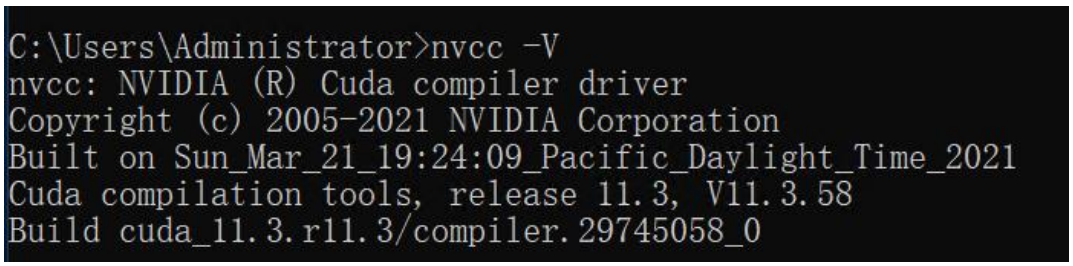
```
nvidia-smi
```



执行以下命令，检查 CUDA 是否安装成功。

```
nvcc -V
```

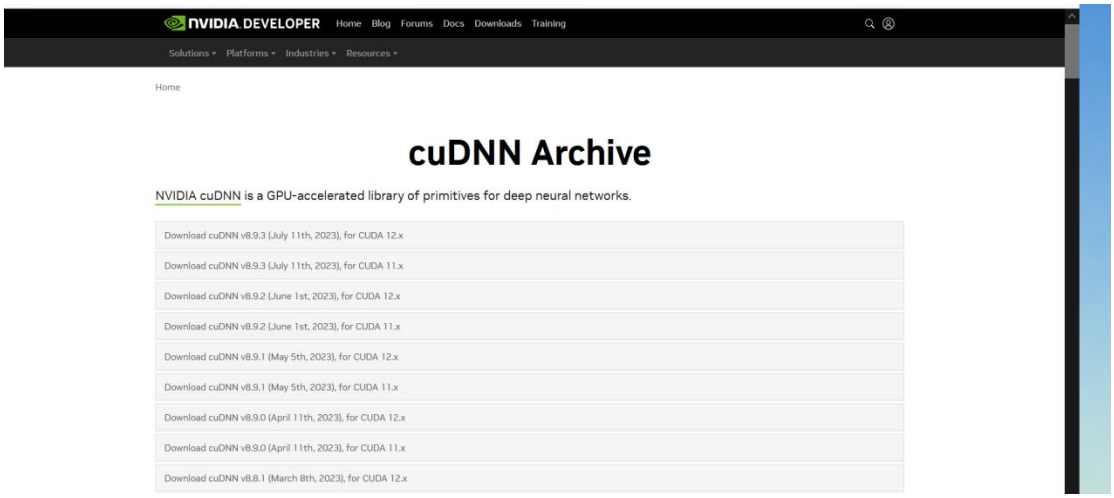
返回如下图所示界面表示 CUDA 安装成功。



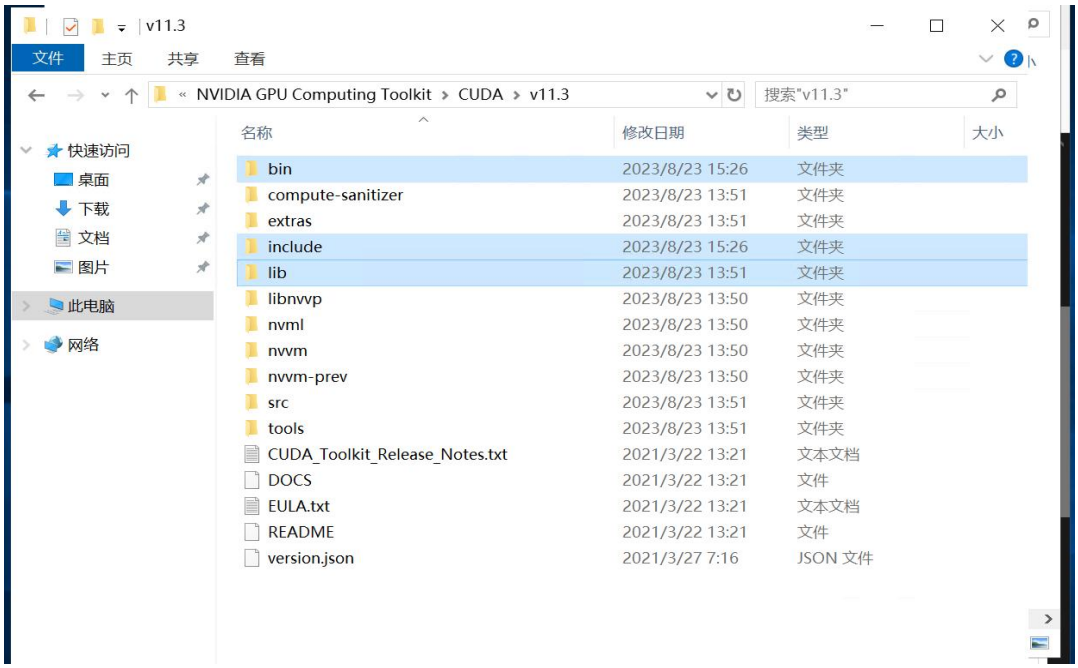
步骤六：安装 cuDNN

1.前往 cuDNN Download 页面，单击“Archived cuDNN Releases”，查看更多版本。

2.找到所需 cuDNN 版本，并下载。



3.解压 cuDNN 压缩包，并将 bin、include 及 lib 文件夹拷贝至 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.3 目录下。



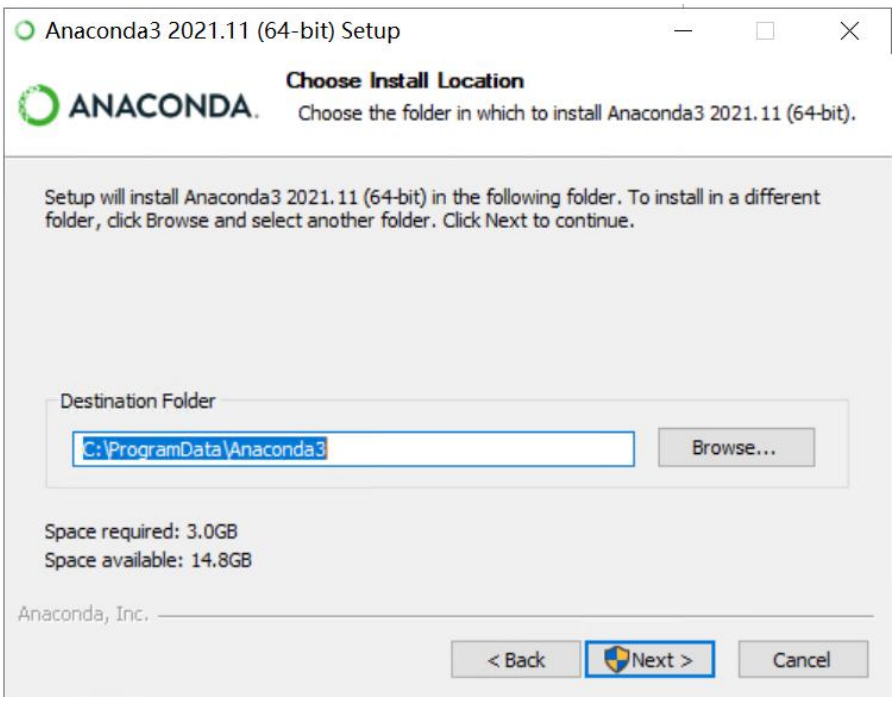
至此已完成 cuDNN 安装。

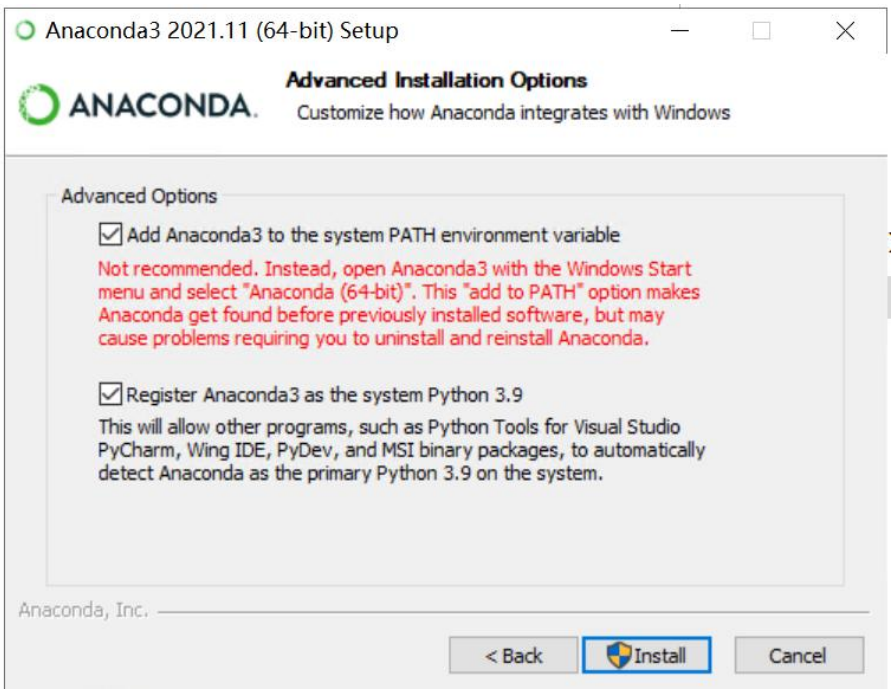
步骤七：安装 Anaconda 深度学习库

建议通过 Anaconda 创建的虚拟环境安装 Pytorch 和 Tensorflow。通过 Anaconda，可便捷获取包并对包进行管理，同时可统一管理环境。Anaconda 包含了 conda、Python 在内的超过 180 个科学包及其依赖项，安装过程简单，能高性能使用 Python 和 R 语言，且有免费的社区支持。

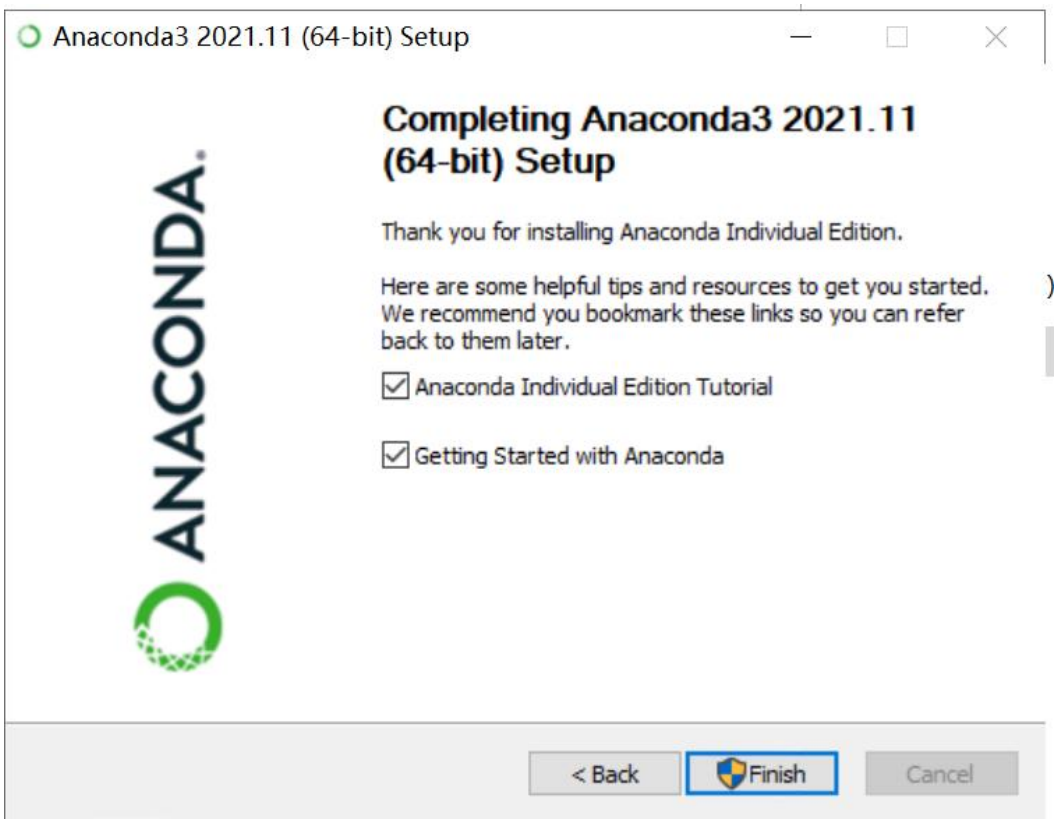
1.前往 Anaconda 官网，在页面中下载所需版本，以 Anaconda3-2021.11-Windows-x86_64 为例。

2.请双击安装包，并根据页面提示进行安装。请注意在 Choose Install Location 步骤中，更改默认安装路径。因默认安装路径 C 盘中的 ProgramData 文件夹为隐藏文件夹，为了方便管理，建议安装在其他文件夹。





3.单击“Install”，根据提示完成安装。



步骤八：配置 Anaconda 深度学习库。

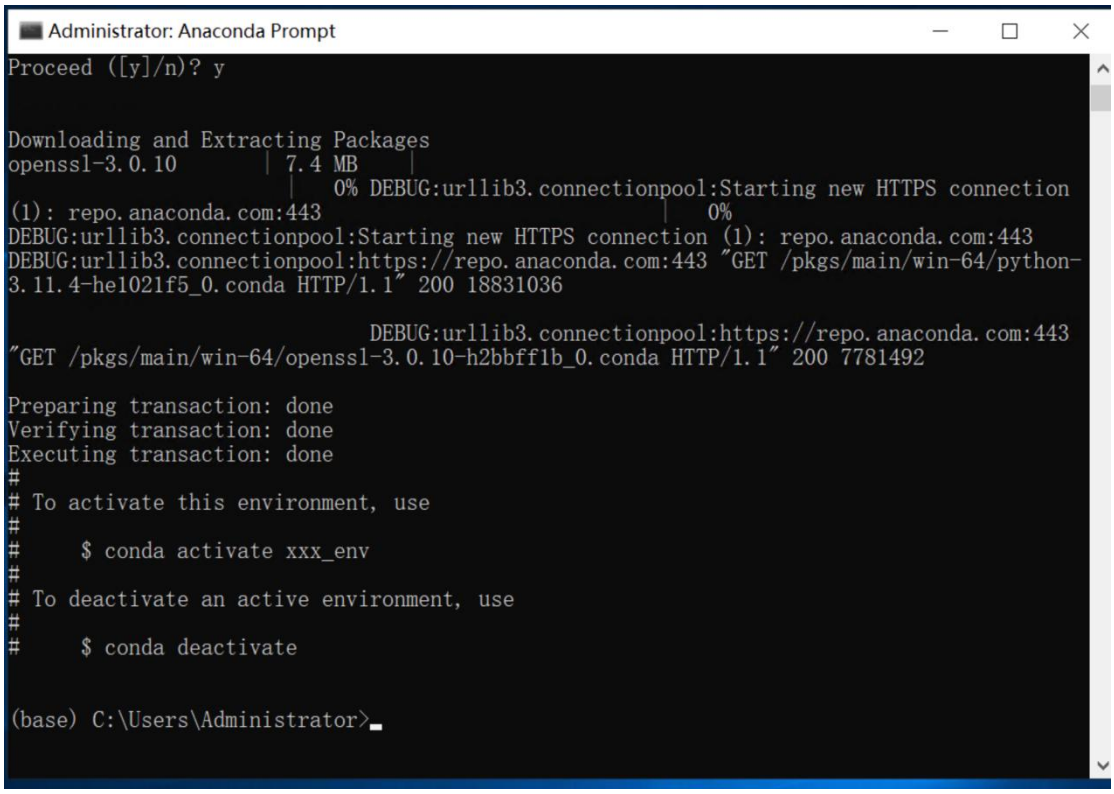
1.在操作系统界面，单击左下角的  ，在弹出菜单中选择 Anaconda Prompt。

2.在打开的 Anaconda Prompt 命令行窗口中，执行以下命令，创建虚拟环境。

```
conda create -n xxx_env python=3.9
```

说明 xxx_env 为环境名，python=3.11 为 Python 版本，您可根据实际需求进行修改。

如下所示即为安装成功。



您可使用以下命令进入或退出已创建的虚拟环境。进入虚拟环境后，即可按照实际需求安装包。

#激活命令

```
conda activate xxx_env
```

#退出命令

```
conda deactivate
```

步骤九：安装 Pytorch。

前往 Pytorch 官网，使用官网推荐的安装代码。本文已安装 CUDA 版本为 11.3，在已创建的 xxx_env 虚拟环境中执行如下命令进行安装：

```
# CUDA 11.3conda install pytorch==1.10.1 torchvision==0.11.2 torchaudio==0.10.1 cudatoolkit=11.3
```

步骤十：安装 Tensorflow。

1.执行以下命令，安装 Tensorflow_gpu_2.6.0。

```
pip install tensorflow-gpu==2.6.0 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2.执行以下命令，安装 keras。

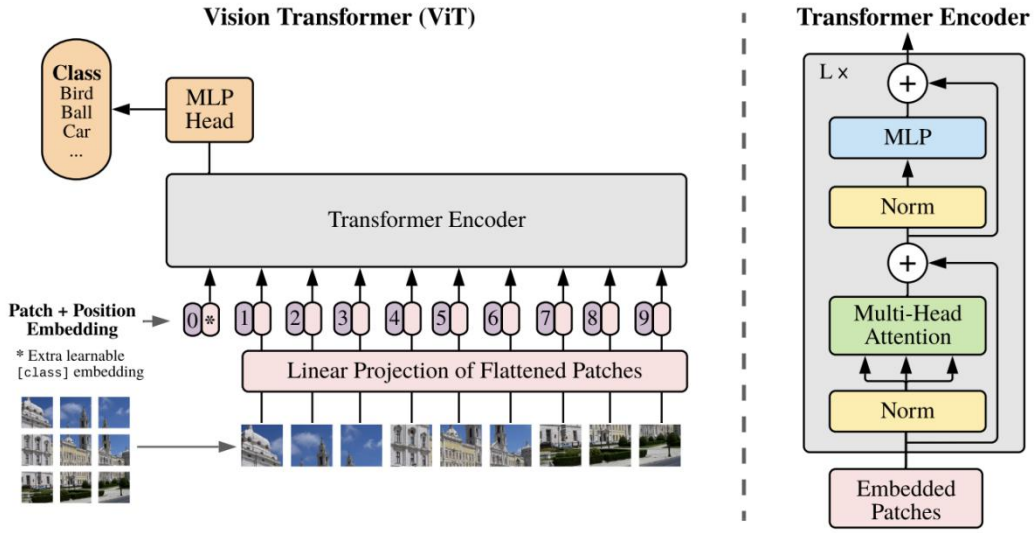
```
pip install keras -i https://pypi.tuna.tsinghua.edu.cn/simple
```

深度学习库的安装已基本完成。您可参考本文方法安装更多所需要的包，并利用 Anaconda 自带的 jupyter notebook、Spyder 工具或者安装 PyCharm 等工具开始代码学习。

4.6使用 GPU 弹性云主机训练 ViT 模型

背景信息

ViT 全称 Vision Transformer，该模型是在 2020 年由 Alexey Dosovitskiy 等人提出，将 Transformer 应用在图像分类的模型，虽然不是第一次将 Transformer 应用在视觉任务，但模型结构效果好，可扩展性强，成为了 Transformer 在 CV 领域应用的里程碑。模型示意图如下：



实例环境如下表所示。

实例类型	**pi2.2xlarge.4**
所在地域	上海 7
系统盘	40GB
数据盘	10GB

实例类型	**pi2.2xlarge.4**
操作系统	Ubuntu 18.04.5 LTS
公网弹性 IP 带宽	5Mbps

操作步骤

1.配置 PyTorch 开发环境。

a. 安装 NVIDIA GPU 驱动、CUDA 和 CUDNN 组件。

执行以下命令，安装 NVIDIA 显卡驱动。

```
apt install tar gcc g++ make build-essential

chmod +x NVIDIA-Linux-x86_64-515.65.01.run

./NVIDIA-Linux-x86_64-515.65.01.run --no-opengl-files
```

安装完成后执行 nvidia-smi 命令，查看是否安装成功。

```
(base) root@ecm-4e33:~# nvidia-smi
Fri Aug 25 14:56:27 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    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  Tesla T4            Off  | 00000000:00:09.0 Off |                    0 |
| N/A   47C    P0    28W /  70W |      2MiB / 15360MiB |      5%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
./cuda_11.7.0_515.43.04_linux.run

tar xJvf  cudnn-linux-x86_64-8.5.0.96_cuda11-archive.tar.xz
```

```
cd cudnn-linux-x86_64-8.5.0.96_cuda11-archive
sudo cp include/* /usr/local/cuda-11.7/include/
sudo cp lib/* /usr/local/cuda-11.7/lib64/
sudo chmod a+r /usr/local/cuda-11.7/include/cudnn*
sudo chmod a+r /usr/local/cuda-11.7/lib64/libcudnn*
```

b. 配置 conda 环境。

依次执行以下命令，配置 conda 环境。

```
wget -c
https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/Miniconda3-py39_4.12.0-Linux-x86_64.sh
chmod +x Miniconda3-py39_4.12.0-Linux-x86_64.sh
./Miniconda3-py39_4.12.0-Linux-x86_64.sh
```

c. 编辑~/.condarc 文件，加入下图配置信息，将 conda 的软件源替换为清华源。

```
channels:
  - defaults
show_channel_urls: true
default_channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2
custom_channels:
  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  msys2: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  bioconda: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  menpo: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  pytorch-lts: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  simpleitk: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
  deepmodeling: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud
```

详情请参见：https://mirror.tuna.tsinghua.edu.cn/help/anaconda/

执行 conda info，确认软件源已替换。

```
(base) root@ecm-4e33:~# conda info

     active environment : base
    active env location : /root/miniconda3
            shell level : 1
       user config file : /root/.condarc
 populated config files : /root/.condarc
          conda version : 4.12.0
    conda-build version : not installed
         python version : 3.9.12.final.0
       virtual packages : __cuda=11.4=0
                          __linux=4.15.0=0
                          __glibc=2.27=0
                          __unix=0=0
                          __archspec=1=x86_64
       base environment : /root/miniconda3  (writable)
      conda av data dir : /root/miniconda3/etc/conda
  conda av metadata url : None
          channel URLs : https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/linux-64
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/noarch
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/linux-64
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/noarch
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2/linux-64
                         https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2/noarch
         package cache : /root/miniconda3/pkgs
                         /root/.conda/pkgs
      envs directories : /root/miniconda3/envs
                         /root/.conda/envs
              platform : linux-64
            user-agent : conda/4.12.0 requests/2.27.1 CPython/3.9.12 Linux/4.15.0-137-generic ubuntu/18.04.5 glibc/2.27
               UID:GID : 0:0
            netrc file : None
          offline mode : False
```

d. 执行以下命令替换 pip 源为清华源。

```
pip config set global.index-url
https://pypi.tuna.tsinghua.edu.cn/simple/
```

e. 安装 Pytorch 组件。

执行以下命令，安装 PyTorch。

```
pip install torch==1.13.1+cu117 torchvision==0.14.1+cu117 torchaudio==0.13.1
--extra-index-url https://download.pytorch.org/whl/cu117
```

依次执行以下命令，查看 PyTorch 是否安装成功。

```
]
>>> print(torch.cuda.get_device_name())
Tesla
>>> print(torch.cuda.is_available())
>>> import torch
Type "help", "copyright", "credits" or "license" for more information.
[GCC 11.3.0] :: Anaconda, Inc. on linux
Python 3.9.13 (main, Jun  2 2023, 20:43:20)
(ai_demo) root@ecm-4e33:~# python
```

2.实验数据。

CIFAR-10（Canadian Institute for Advanced Research-10）是一个常用的计算机视觉数据集，用于图像分类任务。它由 60000 个 32x32 彩色图像组成，这些图像均来自于 10 个不同的类别，每个类别包含 6000 个图像。数据集被分为两个部分：训练集和测试集，其中训练集包含 50000 个图像，测试集包含 10000 个图像。CIFAR-10 数据集中的图像涵盖了广泛的对象类别，包括飞机、汽车、鸟类、猫、鹿、狗、青蛙、

马、船和卡车。每个图像都有一个标签，表示它所属的类别。这个数据集被广泛用于计算机视觉领域的算法开发、模型训练和性能评估。

3.使用 ColossalAI-Examples 模型训练。

本文在分布式训练框架 Colossal-AI 的基础上进行模型训练和开发。Colossal-AI 提供了一组便捷的接口，通过这组接口能方便地实现数据并行、模型并行、流水线并行或者混合并行。

a. 安装 Colossal-AI 和其他组件。

```
pip install colossalai timm titans
```

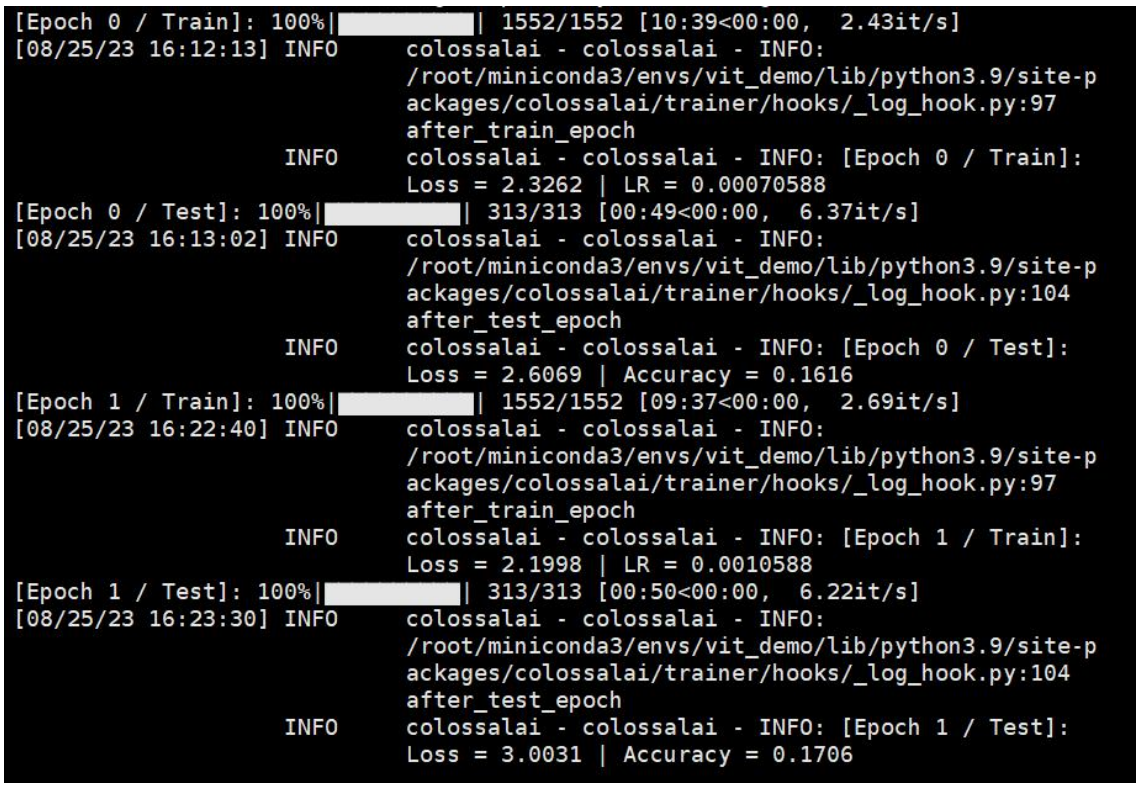
b. ViT 示例模型训练。

```
git clone https://github.com/hpcaitech/ColossalAI-Examples.git

cd ColossalAI-Examples/image/vision_transformer/data_parallel
```

由于单卡 T4 显存有限，修改 config.py 文件，将 BATCH_SIZE 设置为 32。执行以下命令启动训练：

```
colossalai run --nproc_per_node 1  train_with_cifar10.py --config config.py
```

模型运行过程如下图所示：



4.7如何使用天翼云 GPU 云主机构建 Blender 云端渲染服务

背景信息

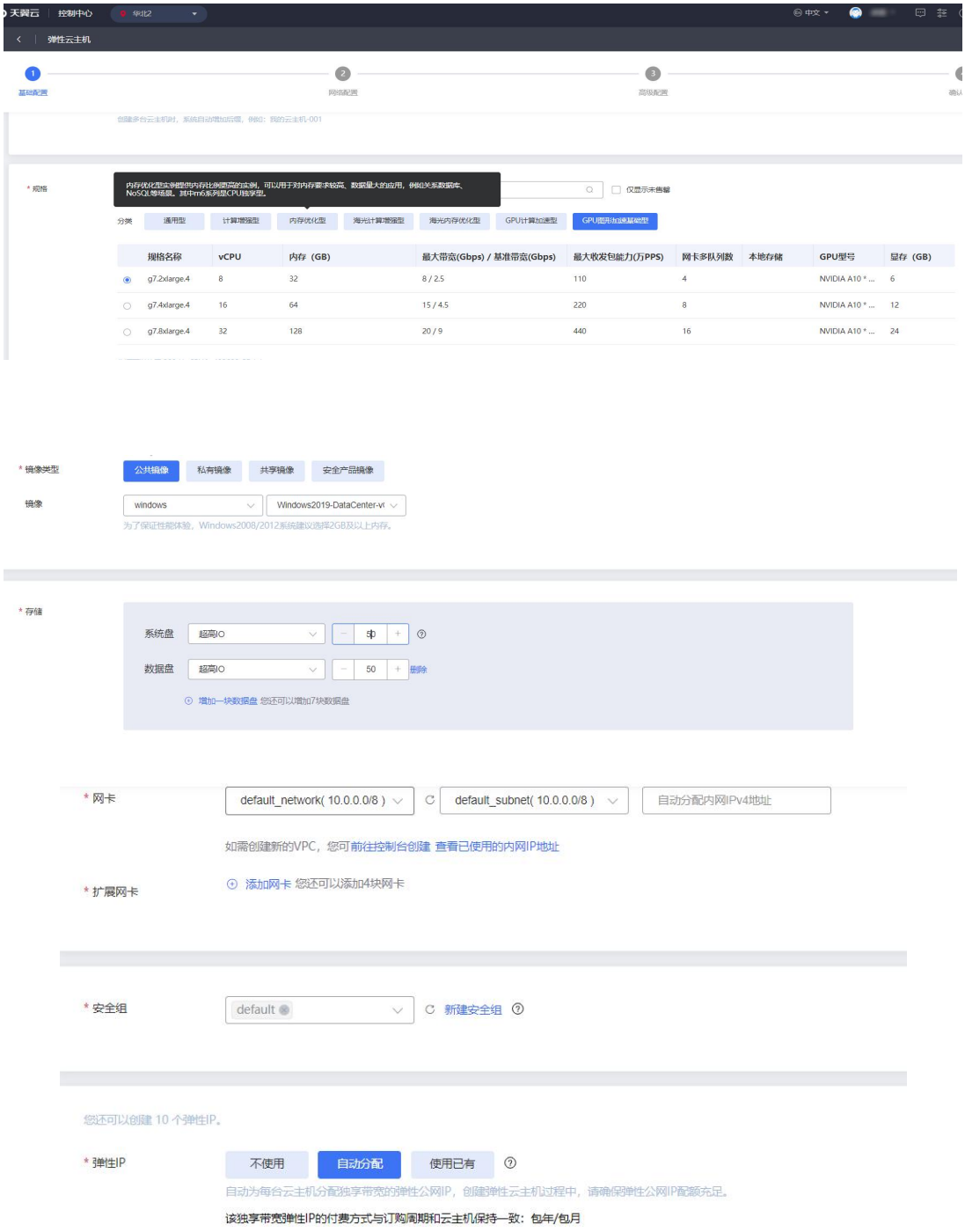
Blender 是一款永久开源免费的 3D 创作软件，支持整个 3D 创作流程：建模、雕刻、骨骼装配、动画、模拟、实时渲染、合成和运动跟踪，甚至可用作视频编辑及游戏创建。

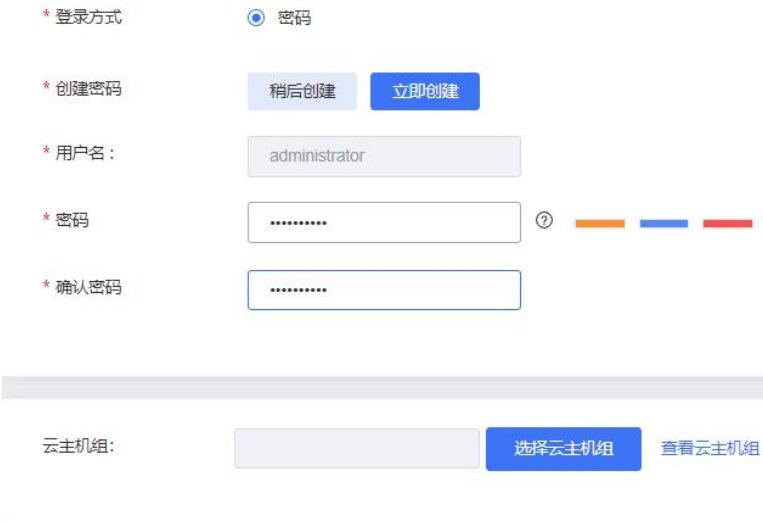
实例环境如下表所示。

实例类型	**g7.2xlarge.4**
所在地域	华北 2
系统盘	50GB
数据盘	50GB
操作系统	Windows2019-DataCenter-vGPU
公网弹性 IP 带宽	5Mbps

操作步骤

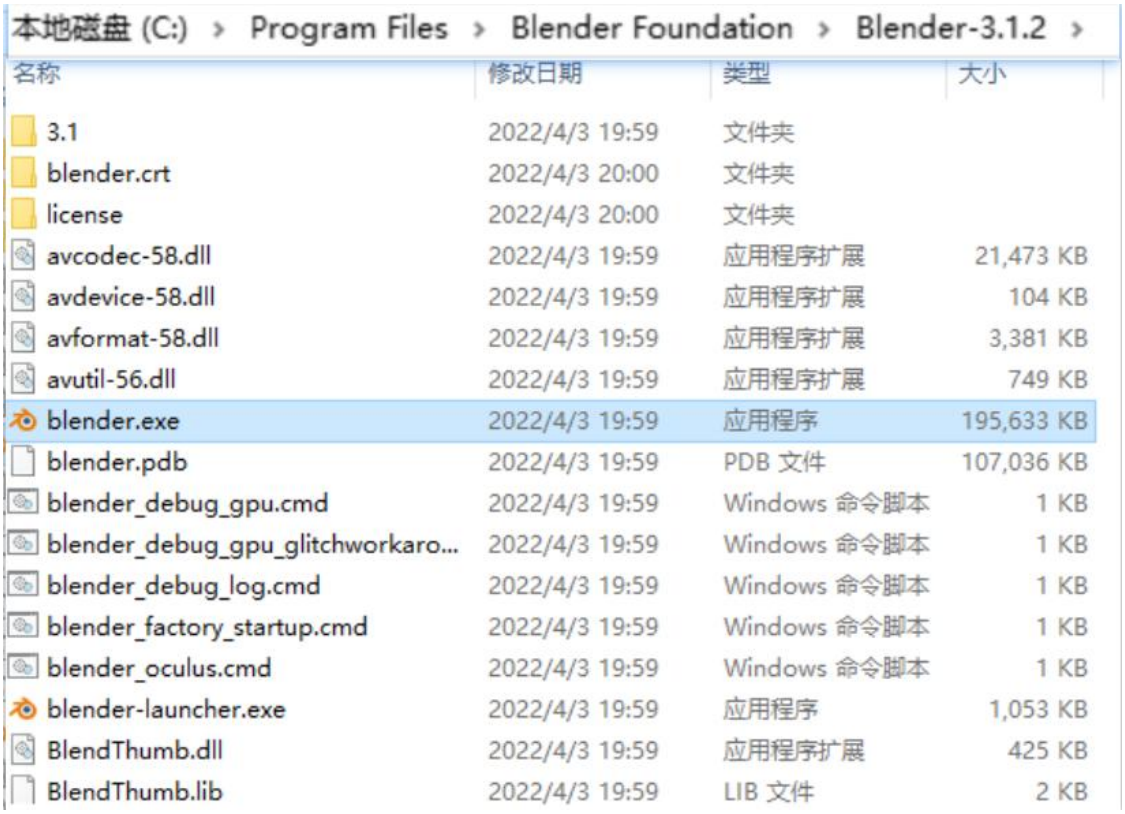
1.在天翼云申请 GPU 云主机实例。本文创建了一台规格为 g7.2xlarge.4 的图形加速基础型 GPU 云主机，选择 Windows2019-DataCenter-vGPU，配置超高 IO 系统盘及数据盘各 50GB，添加网卡，选择 VPC 及 Subnet，添加默认安全组，购买 EIP，创建用户名、密码，购买成功后开机使用。


天翼云 | 控制中心 华北2
弹性云主机
1 基础配置
2 网络配置
3 高级配置
* 规格
内存优化型实例提供内存比例更高的实例，可以用于对内存要求较高、数据量大的应用，例如关系数据库、NoSQL等场景。其中m6系列是CPU独享型。
仅显示未售罄
分类 通用型 计算增强型 内存优化型 海光计算增强型 海光内存优化型 GPU计算加速型 GPU图形加速基础型
规格名称 vCPU 内存（GB） 最大带宽(Gbps) / 基准带宽(Gbps) 最大收发包能力(万PPS) 网卡多队列数 本地存储 GPU型号 显存（GB）
g7.2xlarge.4 8 32 8 / 2.5 110 4 NVIDIA A10 * ... 6
g7.4xlarge.4 16 64 15 / 4.5 220 8 NVIDIA A10 * ... 12
g7.8xlarge.4 32 128 20 / 9 440 16 NVIDIA A10 * ... 24
* 镜像类型
公共镜像 私有镜像 共享镜像 安全产品镜像
镜像
windows
Windows2019-DataCenter-vt
为了保证性能体验，Windows2008/2012系统建议选择2GB及以上内存。
* 存储
系统盘 超高IO 40
数据盘 超高IO 50 删除
增加一块数据盘 您还可以增加7块数据盘
* 网卡
default_network(10.0.0.0/8)
default_subnet(10.0.0.0/8)
自动分配内网IPv4地址
如需创建新的VPC，您可前往控制台创建 查看已使用的内网IP地址
* 扩展网卡
添加网卡 您还可以添加4块网卡
* 安全组
default
新建安全组
您还可以创建 10 个弹性IP。
* 弹性IP
不使用 自动分配 使用已有
自动为每台云主机分配独享带宽的弹性公网IP，创建弹性云主机过程中，请确保弹性公网IP配额充足。
该独享带宽弹性IP的付费方式与订购周期和云主机保持一致：包年/包月

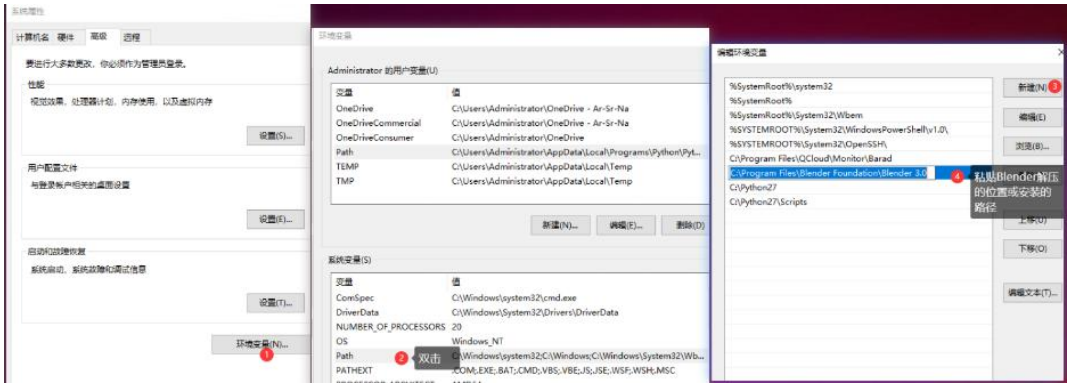


注意如果使用自己的镜像没有 GRID 图形驱动，将无法使用渲染 OpenGL 功能，请安装驱动，详情请参见安装 GRID 驱动。

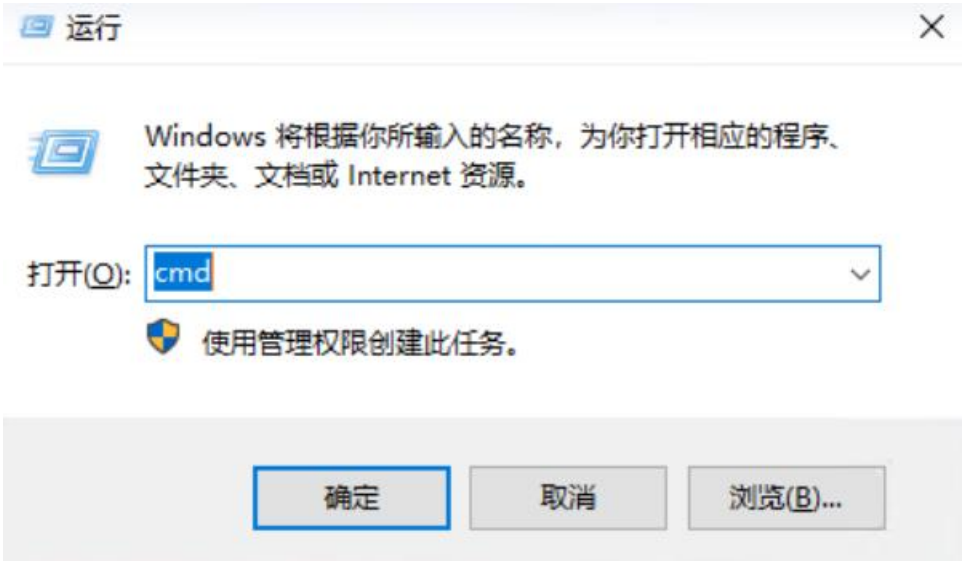
2.安装 Blender，在官网下载并安装，下载地址：https://www.blender.org/download/，选择 Blender -3.1.2 版本，并解压缩到指定目录下。



3.配置环境变量，右击此电脑，单击属性，在弹出的弹窗中选择高级，点击环境变量进行配置。

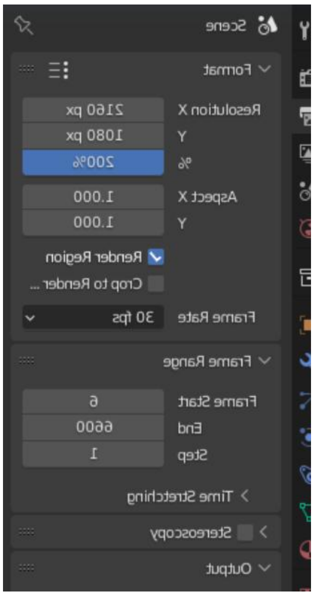


4.重启云主机，开机后运行 Windows+R 键，输入 cmd。

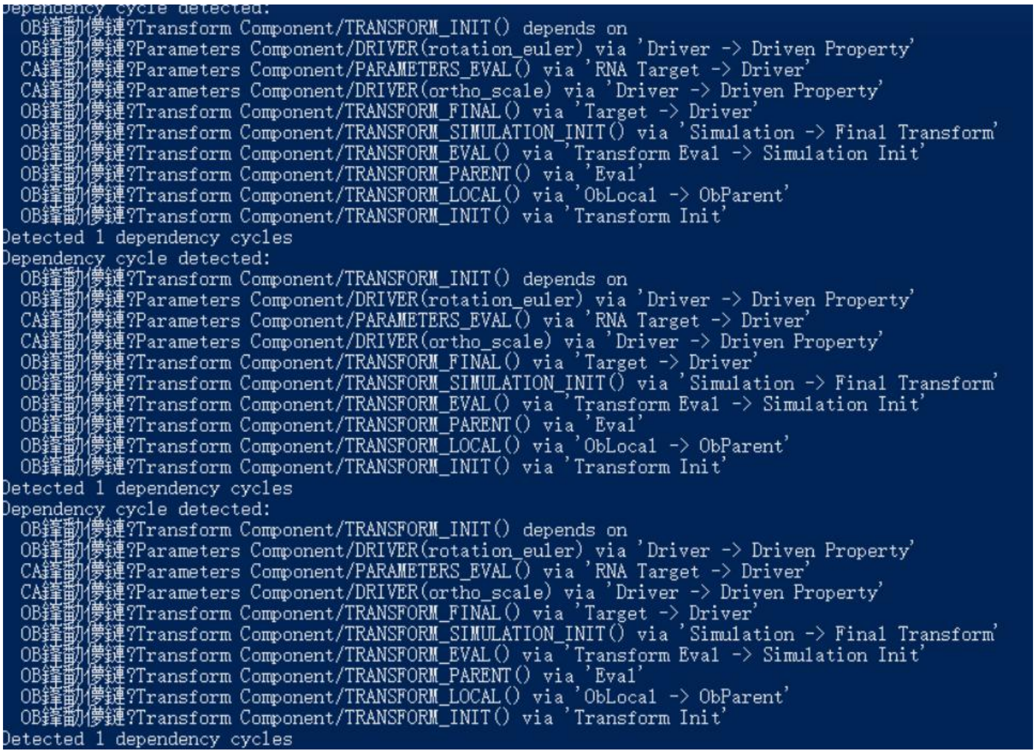


5.在命令行输入 blender，如果能够启动 blender 页面，证明已经成功。

6.渲染参数设定，建议直接在 blender 里面设定好所有的参数，命令行只是确定渲染的帧数。



7.建议将工程文件（blend）保存在容易记的位置，这里以 C:\test.blend 为例，输入 blender -b "C:\test.blend" -o frame_##### -f 2128，运行上述代码后，执行一段时间后，可以在工程目录下看到输出的内容了。



上述代码的作用

参数	**内容**

参数	内容
-b	静默运行（不运行 GUI 界面），后跟工程目录地址，如果带有空格的，要加双引号
-o	输出目录及文件名，#代表帧号，一个#代表一位数，不足的会补 0
-f	渲染的帧号，要保证这个参数在最后面

8.执行以下动画图像命令行，将会渲染 2128 到 3000 帧，并输出到 工程目录/out/ 目录下。

```
blender -b "C:\test.blend" -o "/out/frame_#####" -s 2128 -e 3000
```

注意命令行没有指定的参数，都需要通过工程文件来设置，否则将按照工程文件的设置进行输入，命令行更多参数请查阅：https://docs.blender.org/manual/zh-hans/dev/advanced/command_line/render.html

4.8 本地文件如何上传到 Linux 云主机

Windows 系统通过 WinSCP 方式上传到 Linux 云主机

WinSCP 是一个 Windows 平台上的免费开源的 SFTP 客户端软件，WinSCP 提供图形化界面，通过 WinSCP，用户可以连接到远程服务器，并且可以在本地计算机和远程服务器之间进行文件的上传、下载、复制、移动和删除等操作。WinSCP 支持 SSH 协议，可以确保数据传输的安全性，并且支持使用公钥和密码进行身份验证。

如您的本地电脑使用 Windows 操作系统，您购买的云主机使用 Linux 操作系统，您可通过 WinSCP 方式将本地文件上传至云主机。

前提条件：

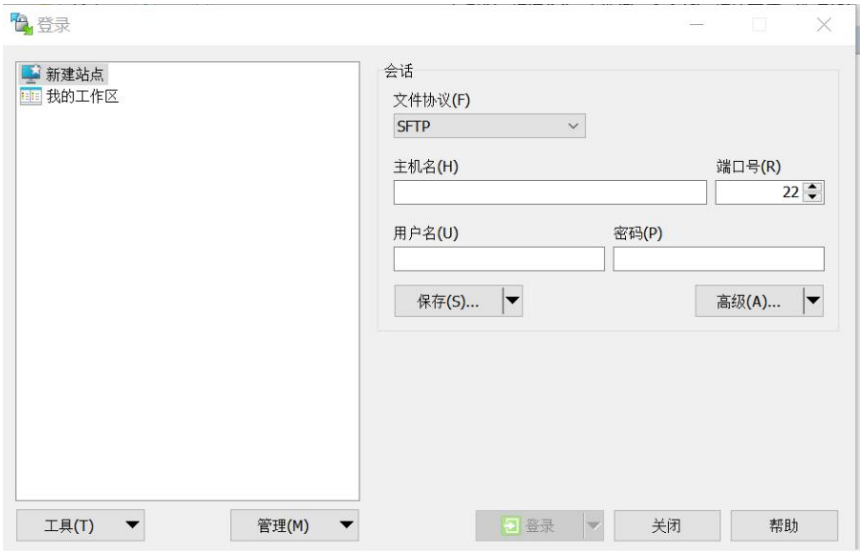
- 本地操作系统类型为 Windows。
- 云主机操作系统类型为 Linux。

- 云主机配备弹性 IP。
- 云主机所在的安全组放行了 22 端口（SSH 服务）。

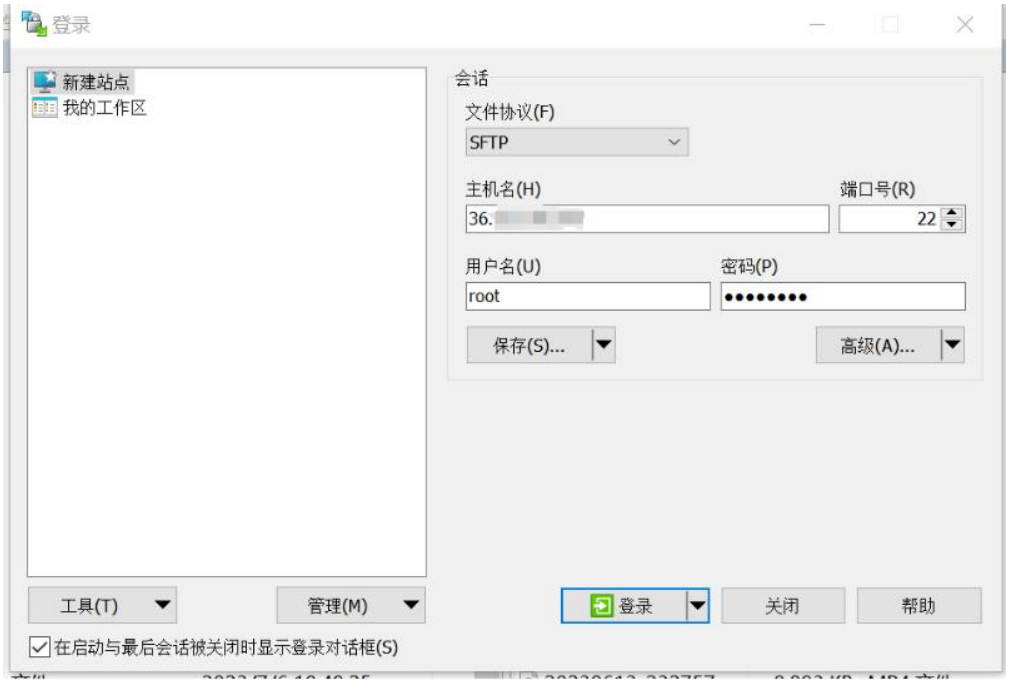
操作步骤：

1.下载 WinSCP 客户端并安装。

2.启动 WinSCP，启动后界面如下：



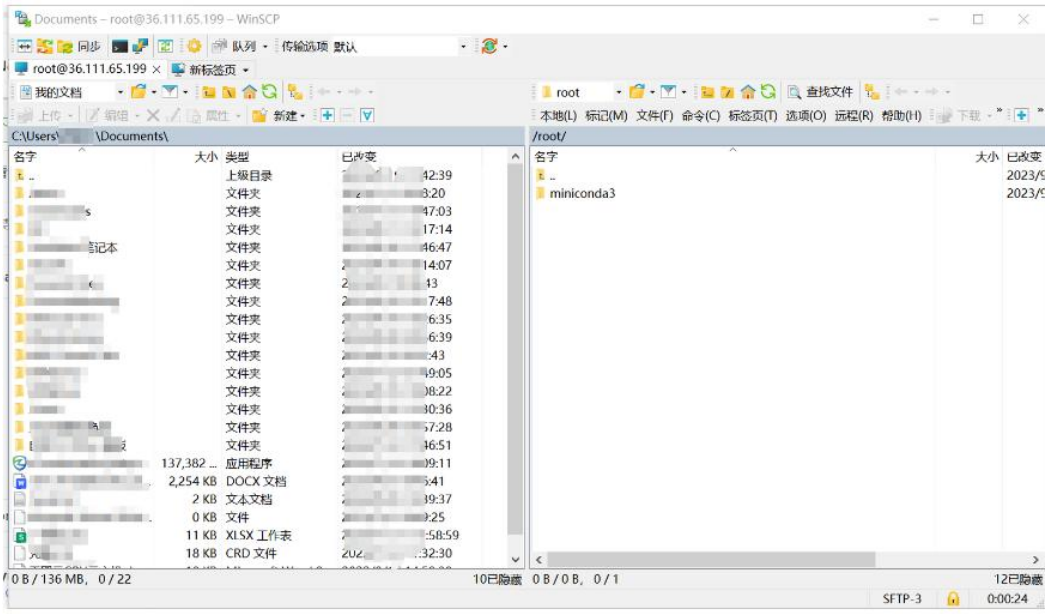
3.在 WinSCP 登录界面填写登录参数。



- 协议：选填 SFTP 或者 SCP 。
- 主机名：云主机的弹性 IP。

- 端口：默认为 22。
- 用户名：登录云主机的用户名。
- 密码：用户名对应的密码。

4.单击登录，进入 “WinSCP” 文件传输界面。



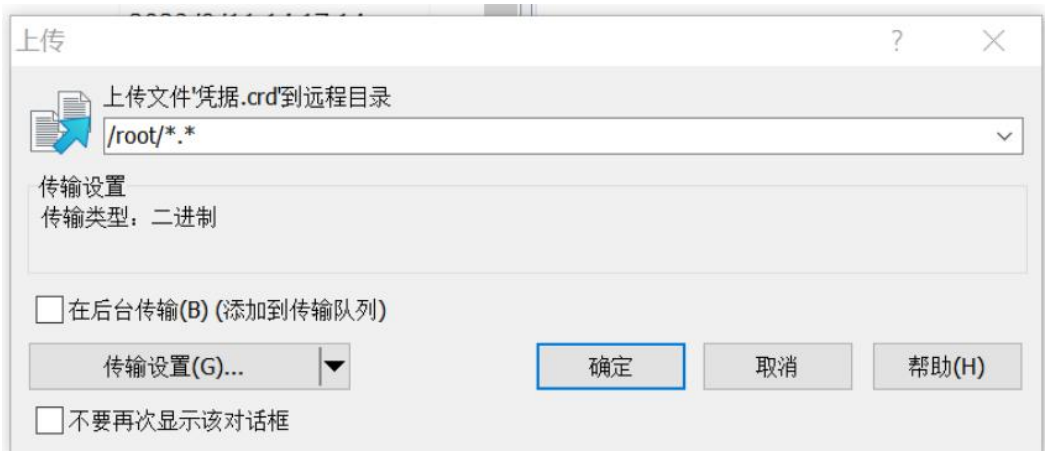
5.上传文件。

a. 在 “WinSCP” 文件传输界面的右侧窗格中，选择文件在云主机中待存放的目录。

b. 在 “WinSCP” 文件传输界面的左侧窗格中，选中待传输的文件。

c. 在 “WinSCP” 文件传输界面的左侧菜单栏中，单击上传。

d. 在上传确认弹窗中确认文件信息，无误后点击确认。



e. 查看界面，核实是否上传成功。

Linux 系统通过 SCP 方式上传到 Linux 云主机

SCP 是一种安全的文件传输协议，可以方便地在本地计算机和远程服务器之间进行文件的复制操作，并通过 SSH 协议提供了数据加密和身份验证的安全保障。

如您的本地电脑使用和购买的云主机均使用 Linux 操作系统,您可通过 SCP 方式将本地文件上传至云主机。

前提条件:

● 本地操作系统类型为 Linux。

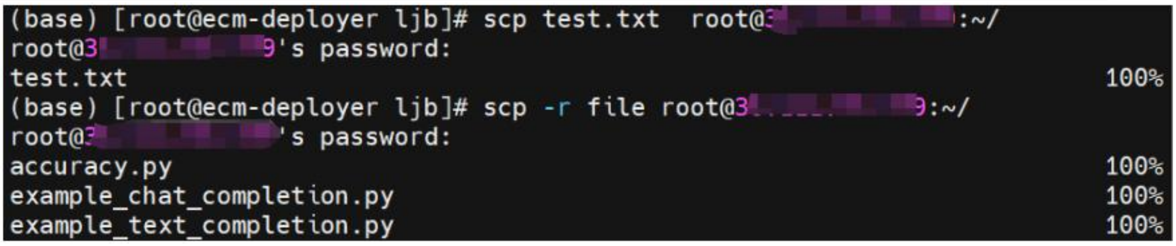
● 云主机操作系统类型为 Linux。

● 云主机配备弹性 IP。

● 云主机所在的安全组放行了 22 端口（SSH 服务）。

操作步骤:

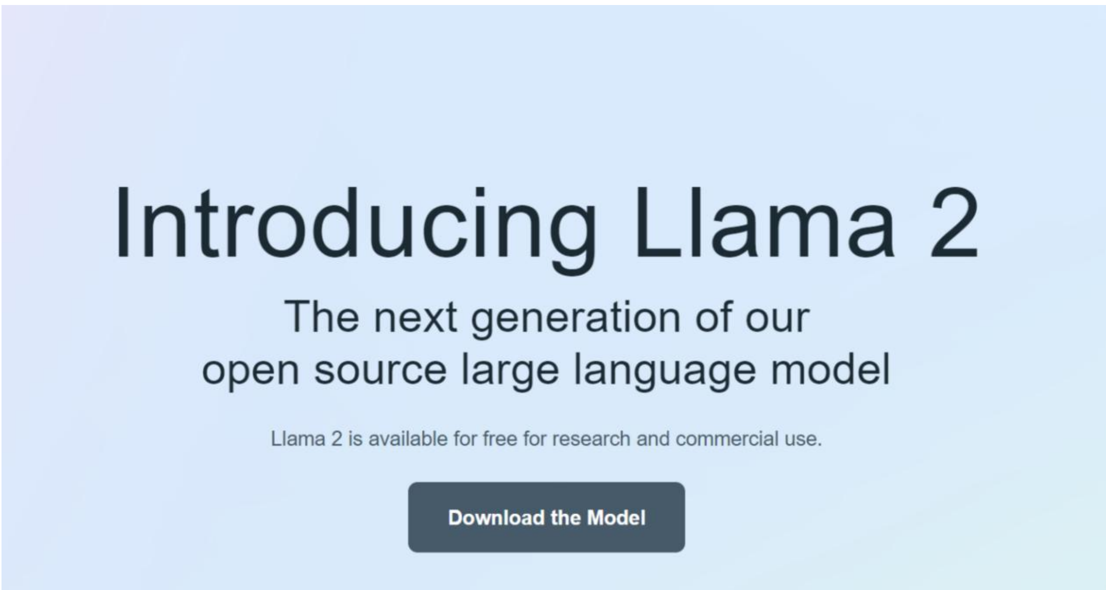
```
scp source_file user@target_ip:destination_file# 将本地文件通过 scp 方式上传到云主机上
scp -r source_dir user@target_ip:destination_path/# 将本地目录通过 scp -r 方式上传到云主机上
```



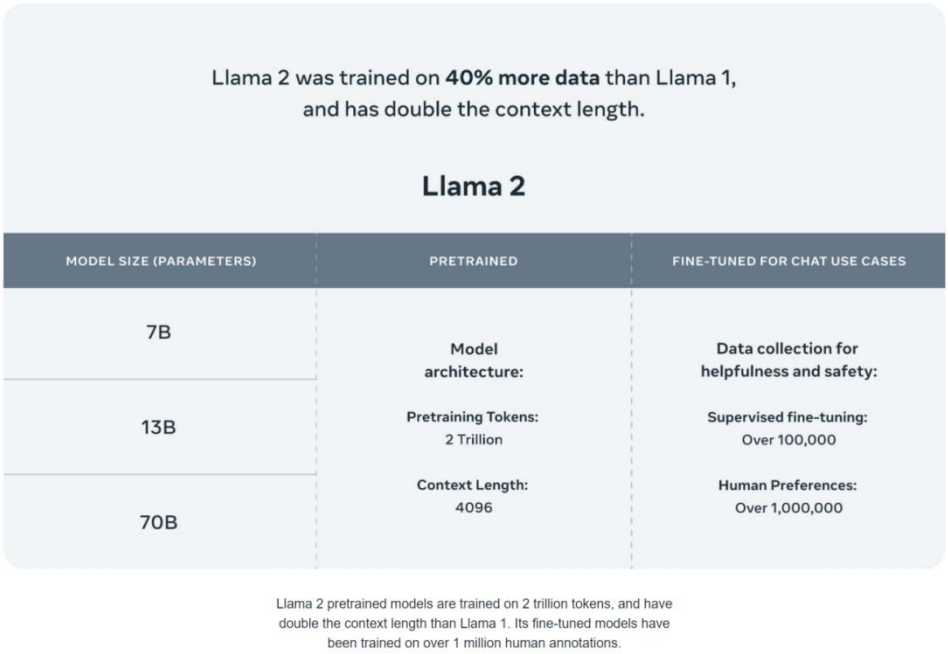
4.9以 Llama 2 为例进行大模型推理实践

1. 什么是 Llama2

Meta 在 7 月 18 日发布了可以免费用于学术研究或商业用途的 Llama2 开源大语言模型。



Llama 的训练方法是先进行无监督预训练，再进行有监督微调，训练奖励模型，根据人类反馈进行强化学习。 Llama 2 的训练数据比 Llama 1 多 40%，用了 2 万亿个 tokens 进行训练，并且上下文长度是 Llama 1 的两倍。 目前提供 7B 、13B、70B 三种参数量的版本。



根据 Meta 公布的官方数据，Llama 2 在许多基准测试上都优于其他开源语言模型，包括推理、编程、对话能力和知识测试，在帮助性、安全性方面甚至比部分闭源模型要好。

Benchmarks

Llama 2 outperforms other open source language models on many external benchmarks, including reasoning, coding, proficiency, and knowledge tests.

Benchmark (Higher is better)	MPT (7B)	Falcon (7B)	Llama-2 (7B)	Llama-2 (13B)	MPT (30B)	Falcon (40B)	Llama-1 (65B)	Llama-2 (70B)
MMLU	26.8	26.2	45.3	54.8	46.9	55.4	63.4	68.9
TriviaQA	59.6	56.8	68.9	77.2	71.3	78.6	84.5	85.0
Natural Questions	17.8	18.1	22.7	28.0	23.0	29.5	31.0	33.0
GSM8K	6.8	6.8	14.6	28.7	15.2	19.6	50.9	56.8
HumanEval	18.3	N/A	12.8	18.3	25.0	N/A	23.7	29.9
AGIEval (English tasks only)	23.5	21.2	29.3	39.1	33.8	37.0	47.6	54.2
BoolQ	75.0	67.5	77.4	81.7	79.0	83.1	85.3	85.0
HellaSwag	76.4	74.1	77.2	80.7	79.9	83.6	84.2	85.3
OpenBookQA	51.4	51.6	58.6	57.0	52.0	56.6	60.2	60.2
QuAC	37.7	18.8	39.7	44.8	41.1	43.3	39.8	49.3
Winogrande	68.3	66.3	69.2	72.8	71.0	76.9	77.0	80.2

Llama 2-Chat 在 Llama 2 的基础上针对聊天对话场景进行了微调和安全改进，使用 SFT (监督微调) 和 RLHF (人类反馈强化学习)进行迭代优化，以便更好的和人类偏好保持一致，提高安全性。

Llama 2-Chat 更专注于聊天机器人领域，主要应用于以下几个方面：

客户服务：Llama 2-Chat 可以用于在线客户服务，回答关于产品、服务的常见问题，并向用户提供帮助和支持。

社交娱乐：Llama 2-Chat 可以作为一个有趣的聊天伙伴，与用户进行随意、轻松的对话，提供笑话、谜语、故事等娱乐内容，增加用户的娱乐体验。

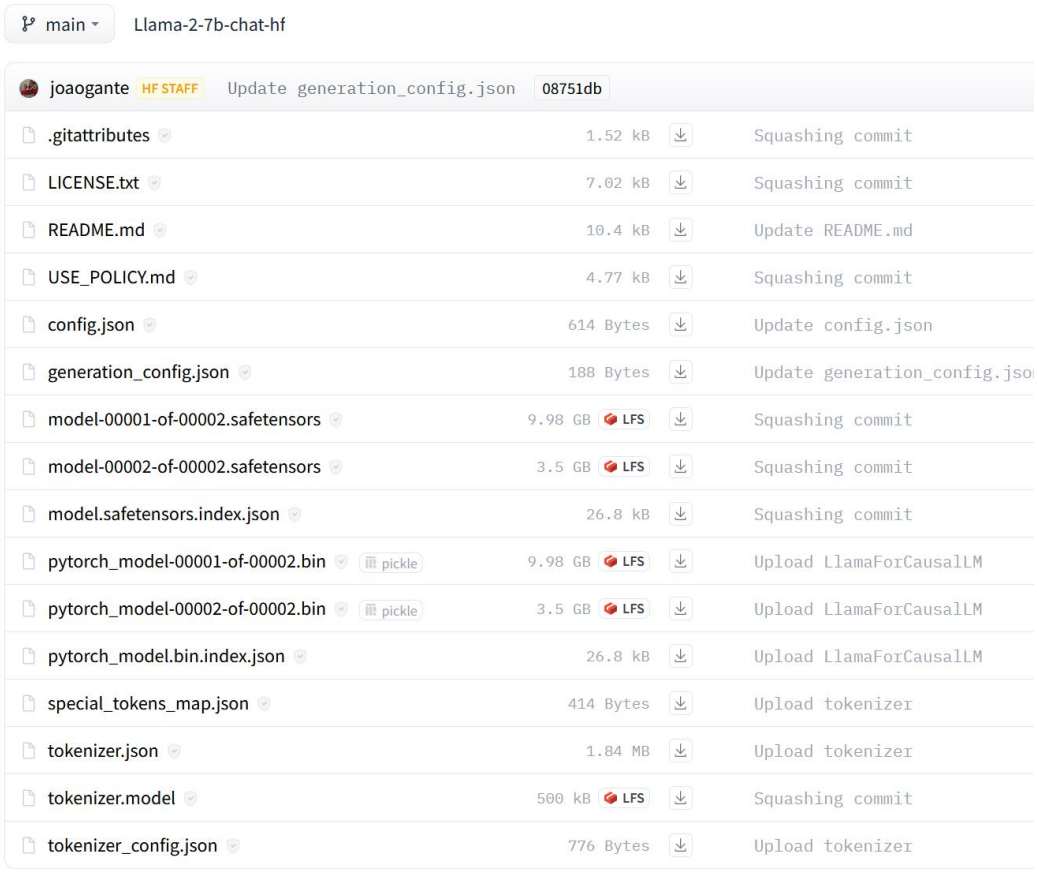
个人助理：Llama 2-Chat 可以回答一些日常生活中的问题，如天气查询、时间设置、提醒事项等，帮助用户解决简单的任务和提供一些实用的功能。

心理健康：Llama 2-Chat 可以作为一个简单的心理健康支持工具，可以与用户进行交流，提供情绪调节、压力缓解的建议和技巧，为用户提供安慰和支持。

2. **在 GPU 云主机上搭建模型运行环境**

步骤一：下载模型并上传

从 huggingface 官网下载 Llama-2-7b-chat-hf 模型，如下图所示。下载完成后上传至 GPU 云主机。



说明

如何将本地文件上传到 Linux 云主机请参考本地文件如何上传到 Linux 云主机。

步骤二：环境搭建

1.上传并安装 GPU 驱动

从 Nvidia 官网下载 GPU 驱动并上传至 GPU 云主机，按照如下步骤安装驱动。

```
# 对安装包添加执行权限 chmod +x NVIDIA-Linux-x86_64-515.105.01.run# 安装 gcc 和 linux-kernel-headerssudo apt-get install gcc linux-kernel-headers# 运行驱动安装程序 sudo sh NVIDIA-Linux-x86_64-515.105.01.run --disable-nouveau# 查看驱动是否安装成功 nvidia-smi
```

```
[redacted]~# nvidia-smi
Tue Sep 12 20:16:14 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  Tesla V100-PCIE...  Off  | 00000000:00:09.0 Off |                    0 |
| N/A   39C    P0    37W / 250W |      0MiB / 32768MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

说明

如何选择驱动及相关库、软件版本请参见如何选择驱动及相关库、软件版本。

2.安装 Nvidia CUDA Toolkit 组件

```
wget
http://developer.download.nvidia.com/compute/cuda/11.7.0/local_installers/cuda_11.7.0_515.43.
04_linux.run# 安装 CUDA
bash cuda_11.7.0_515.43.04_linux.run# 编辑环境变量文件
vi ~/.bashrc# 增加环境变量
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH# 使环境变量生效
source ~/.bashrc# 查看是否安装成功
nvcc -V
```

3.安装 Miniconda

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh# 安装
Miniconda3
bash Miniconda3-latest-Linux-x86_64.sh# 配置 conda 环境变量
vim /etc/profile# 添加环境变量 export ANACONDA_PATH=~/miniconda3export
PATH=$PATH:$ANACONDA_PATH/bin# 使环境变量生效 source /etc/profile# 查看是否安装成功
which anaconda
```

```
conda --version

conda info -e

python# 查看虚拟环境

conda env list
```

```
conda env list
# conda environments:
#
base                  *  /root/miniconda3
```

4.安装 cuDNN

从 cudnn-download 下载 cuDNN 压缩包并上传至 GPU 云主机，按照如下步骤进行安装。

```
# 解压

tar -xf cudnn-linux-x86_64-8.9.4.25_cuda11-archive.tar.xz# 进目录

cd cudnn cudnn-linux-x86_64-8.9.4.25_cuda11-archive# 复制

cp ./include/*  /usr/local/cuda-11.7/include/

cp ./lib/libcudnn*  /usr/local/cuda-11.7/lib64/ # 授权

chmod a+r /usr/local/cuda-11.7/include/* /usr/local/cuda-11.7/lib64/libcudnn*# 查看是否安装成功

cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

```
cat /usr/local/cuda/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 9
#define CUDNN_PATCHLEVEL 4
--
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

/* cannot use constexpr here since this is a C-only file */
```

5.安装依赖

a. 下载 Llama 模型代码

```
git clone https://github.com/facebookresearch/llama.git
```

b. 在线安装依赖

```
python -m pip install --upgrade pip -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host
mirrors.aliyun.com# 下载依赖
pip install -e . -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com
pip install transformer   -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host
mirrors.aliyun.com# 下载 peft
git clone https://github.com/huggingface/peft.git# 传到离线服务器上切换分支，安装特定版本 peft
git checkout 13e53fc# 安装 peft
pip install . -i https://pypi.tuna.tsinghua.edu.cn/simple --trusted-host pypi.tuna.tsinghua.edu.cn
```

步骤三：镜像打包

为了使您能更快的搭建模型运行环境，在完成步骤一和步骤二的操作后，我们对 GPU 云主机的系统盘进行了打包，生成了标准的 GPU 云主机镜像。目前已经上传至天翼云成都 4、海口 2 资源池，您可直接对该镜像进行使用。

镜像打包步骤如下：

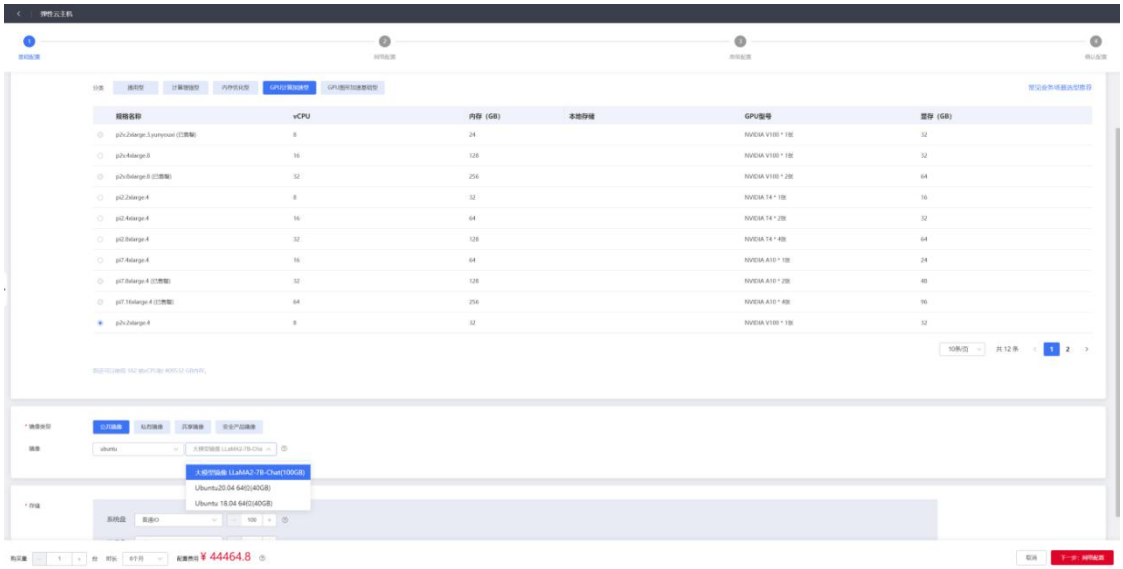
```
echo "nameserver 114.114.114.114" > /etc/resolv.conf
echo "localhost" > /etc/hostname


# 清除 machine-id。
yes | cp -f /dev/null /etc/machine-id
# 若有 /var/lib/dbus/machine-id，则：
# rm -f /var/lib/dbus/machine-id
# ln -s /etc/machine-id /var/lib/dbus/machine-id


cloud-init clean -l   # 清理 cloud-init。若此命令不可用，则可尝试：rm -rf /var/lib/cloud
rm -f /tmp/*.log   # 清除镜像脚本日志。


# 清理 /var/log 日志。read -r -d '' script <<-"EOF"import osdef clear_logs(base_path="/var/log"):
    files = os.listdir(base_path)
    for file in files:
        file_path = os.path.join(base_path, file)
```

```
            if os.path.isfile(file_path):
                with open(file_path, "w") as f:
                    f.truncate()
            elif os.path.isdir(file_path):
                clear_logs(base_path=file_path)
if __name__ == "__main__":
    clear_logs()
EOFif [ -e /usr/bin/python ]; then
    python -c "$script"
elif [ -e /usr/bin/python2 ]; then
    python2 -c "$script"
elif [ -e /usr/bin/python3 ]; then
    python3 -c "$script"else
    echo "### no python env in /usr/bin. clear_logs failed ! ###"
fi


# 清空历史记录。
rm -f /root/.python_history
rm -f /root/.bash_history
rm -f /root/.wget-hsts
```

3. **使用大模型镜像进行模型快速部署**

步骤一：创建 GPU 云主机

登录天翼云控制台，进入弹性云主机主机订购页，选择计算加速型 GPU 云主机，在公共镜像中选择大模型镜像 LLaMA2-7B-Chat。



大模型镜像 LLaMA2-7B-Chat 最低规格推荐：p2v.2xlarge.4 8vCPU 32GB 内存 单张 v100 GPU。

步骤二：在线推理

登录 GPU 云主机，根据如下步骤执行推理任务。

#进入/opt/llama 目录并执行 sh run.sh 命令 cd /opt/llama && sh run.sh

#根据提示在 “ please input your question :” 后输入推理问题

```
(base) root@ecm-bc44:~# cd /opt/llama/ && sh run.sh
please input your question: why do we should protect environment?
Loading checkpoint shards: 100%|                                                | 2/2 [00:11<00:00,  5.54s/it]
Vocab of the base model: 32000
Vocab of the tokenizer: 32000
Start inference.
Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.
=======0=======
Input: why do we should protect environment?

Output: Great question! Protecting the environment is crucial for the well-being of our planet and its inhabitants. Here are s
ome reasons why we should prioritize environmental protection:
1. Preserve Biodiversity: The Earth is home to an incredible array of life forms, from towering trees to tiny microorganisms.
Environmental protection helps maintain this biodiversity, which is essential for the health of ecosystems and the survival of
 countless species.
2. Ensure Clean Air and Water: A clean environment is vital for human health. Pollution from industrial activities, transporta
tion, and other sources can contaminate the air we breathe and the water we drink, leading to serious health problems. Protect
ing the environment means preserving these essential resources.
3. Mitigate Climate Change: Human activities like burning fossil fuels and deforestation contribute to climate change, which h
as far-reaching consequences, including rising sea levels, more frequent natural disasters, and disruptions to food production
. Environmental protection measures can help reduce greenhouse gas emissions and slow the pace of climate change.
4. Support Ecological Services: Ecosystems provide essential services like pollination, pest control, and nutrient cycling, wh
ich are critical for agriculture and food security. Protecting the environment ensures that these services continue to functio
n properly.
5. Promote Sustainable Development: Environmental protection is essential for sustainable development. When natural resources
are depleted or polluted, it can hinder economic growth and social progress. By protecting the environment, we can ensure that
 future generations have access to the resources they need to thrive.
6. Enhance Human Health: Exposure to pollutants in the environment can lead to respiratory problems, cancer, and other health
issues

(base) root@ecm-bc44:/opt/llama#
```

5 API 参考

GPU 云主机作为弹性云主机的一类实例规格，适用的 API 与弹性云主机一致，详见弹性云主机 >API 参考 >API 概览。

6 常见问题

6.1 计费类

是否支持余额不足提醒?

用户可在费用中心总览页面自助设置可用额度预警，当您的余额低于预警阈值时，系统将发送短信提醒。

GPU 云主机快过期了，我还想继续用，该怎么办?

您可以在 GPU 云主机列表页，选中一个或多个 GPU 云主机，单击列表上方“更多 > 续订”进行续订。详细请参见规则说明。

GPU 云主机到期后忘记续费了，会出现什么后果?

如果您忘记续费，到期当天您的 GPU 云主机将无法操作。建议您尽快进行手动续费，否则到期 15 天后数据就会清除。

关于账户提现的说明

若客户账号下没有开通按量付费业务，允许在账户余额范围内任意提取。若客户账号下开通了按量付费业务，须保留 100 元不能提取，允许提现的金额为账户余额与 100 的差值。详细请参见余额提现。

同一台 GPU 云主机是否同时支持两种计费方式?

不可以。同一台 GPU 云主机只能选择一种计费方式，无法同时选择。

但是按量计费的 GPU 云主机可以转为包周期计费，包周期计费的 GPU 云主机到期后可以转为按量计费。

- 按量计费转换为包周期计费：

按量计费是后付费模式，按 GPU 云主机的实际使用时长计费，可以随时开通/删除 GPU 云主机。

如果您需要长期使用当前 GPU 云主机，可以将按量购买的 GPU 云主机转为包周期计费模式，节省开支。

- 包周期计费转换为按量计费：

包周期是预付费模式，按订单的购买周期计费，适用于可预估资源使用周期的场景。

如果您需要更灵活的计费方式，按照 GPU 云主机的实际使用时长计费，您可以将实例的计费方式转为按量计费。但是请注意包周期转换为按量计费，需包周期资费模式到期后，按量的资费模式才会生效。

GPU 云主机关机后还会继续计费吗?

GPU 云主机关机后是否计费根据计费方式不同，情况有所不同：

- 包周期计费方式：包周期资源一次性付费，到期自动停止使用。

- 按量计费方式：按量计费的 GPU 云主机关机后，基础资源（包括 vCPU、内存）不计费，但系统盘仍会收取容量对应的费用。 如有其它绑定的产品，如云硬盘、弹性 IP、带宽等，按各自产品计费方法收费。

什么情况下 GPU 云主机会被冻结，冻结后怎么办?

客户在天翼云购买产品后，如果没有及时的进行续费或充值，欠费后将冻结您账号下所有的按量资源，并会发送短信及邮件提醒您。建议您尽快进行手动续费，否则到期 15 天后，存储在 GPU 云主机中的数据将被删除、GPU 云主机资源将被释放。

当 GPU 云主机资源被冻结后，用户可通过续费或充值来解冻资源，恢复 GPU 云主机正常使用。

欠费冻结的 GPU 云主机允许续费、释放或删除；已经到期的包周期 GPU 云主机不能发起退订，未到期的包周期 GPU 云主机可以退订。

- 资源冻结时：资源将被限制访问和使用，会导致您的业务中断。
- 资源解冻时：资源将被解除限制，但是需要您自行检查并恢复业务。
- 资源释放时：资源将被释放，存储在资源中的数据将被删除，数据无法找回。

如何退订 GPU 云主机?

如果您需要退订 GPU 云主机，需要符合天翼云 7 天无理由退款条件。

- 订购时间超过 7 天，以及订购 7 天内但执行过升级、续订操作的主机均不能执行退订操作。
- 由于退订操作会导致资源回收和清理，GPU 云主机上的数据将无法恢复，因此，在退订前请您做好数据备份工作。
- 按量订购的 GPU 云主机不支持退订操作。详细请参见规则说明。

6.2操作类

重装操作系统失败如何处理?

如果重装操作系统失败，页面会提示重装操作系统失败，运维人员会在后台进行人工恢复，如果您有紧急业务需要立即恢复，请通过在官网提交工单来联系运维人员进行紧急恢复。

无法导入密钥对，怎么办?

当您的浏览器是 IE9 时，可能无法导入密钥对或无法使用文件注入功能，请参考如下步骤修改浏览器默认属性后重试。

1.在浏览器主界面，单击  。

2.选择“Internet 选项”。

3.单击选择“安全”页签。

4.单击“Internet”。

5.如果安全级别显示为“自定义”，单击“默认级别”按钮，把设置还原为默认级别。

6.滑动安全级别滑块，把安全级别调到“中”级别，单击“应用”按钮。

7.选择“自定义级别”。

8.将“对未标记为可安全执行脚本的 ActiveX 控件初始化并执行脚本”设置为“提示”。

9.单击“确定”。

若仍不能解决请在官网提交工单来联系运维人员进行解决。

我创建的 GPU 云主机是否在同一子网?

由于您可以自定义网络，所以 GPU 云主机是否在一个子网，完全由您来控制。

您可以在虚拟私有云内创建一个或多个子网，子网划分可以帮助您合理规划 IP 地址资源，但是子网的网段必须在虚拟私有云网段范围内。同子网内网络默认互通，同 VPC 下不同子网之间默认互通。子网网段创建后无法修改，如何规划子网网段、数量请参见如何规划子网网段、数量；如何进行子网管理请参见子网管理。

我能否自己安装或者升级操作系统?

可以，GPU 云主机支持用户重装操作系统。您可以重装至其他公有镜像，也可上传私有镜像，重装至目标私有镜像。详细请参见重装操作系统。

为什么 Windows 操作系统不支持 DirectX 等功能?

由于 Windows 自带的远程连接（RDP）协议本身并不支持 DirectX、OpenGL 等相关应用。因此，您需要自行安装 TightVNC 服务和客户端，或其它支持 PCOIP、XenDesktop HDX 3D 等协议的远程连接客户端。

如何在 GPU 云主机和普通弹性云主机间传输数据?

GPU 云主机除 GPU 加速能力外，与普通弹性云主机使用体验一致。同一安全组内的 GPU 云主机和弹性云主机之间默认内网互通，无需特别设置。

普通弹性云主机实例规格族是否支持升级或变更为 GPU 云主机实例规格族?

目前不支持该功能。如果您需要使用 GPU 则请购买 GPU 云主机或 GPU 物理机。

如何查询 GPU 显卡的详细信息?

如果您的 GPU 云主机安装了 Linux 操作系统，您可以执行命令 nvidia-smi，查询 GPU 显卡的详细信息。

如果您的 GPU 云主机安装了 Windows 操作系统，您可以在设备管理器中查看 GPU 显卡的详细信息。

为什么购买 GPU 云主机后，执行命令 nvidia-smi 找不到 GPU 显卡?

当您执行命令 **nvidia-smi** 无法找到 GPU 显卡时，通常是由于您的未成功安装 NVIDIA 驱动。请根据您所购买的 GPU 云主机的规格选择对应的操作指引来安装驱动，请参见 NVIDIA 驱动安装指引。

GPU 图形加速基础型云主机需要安装什么驱动?

图形加速基础型云主机需使用 GRID 驱动，创建云主机时 GRID 驱动已预装在镜像中，无需用户自己安装。

为什么创建 GPU 云主机时选择的 CUDA 版本与安装完成后查看到的 CUDA 版本不一致?

您执行命令 nvidia-smi 查询到的 CUDA 版本代表您的 GPU 云主机能够支持的最高 CUDA 版本，并不代表您创建 GPU 云主机时选择的 CUDA 版本。

如何获取 GRID License?

如果您使用的是图形加速基础型 GPU 云主机，创建实例的时候 GRID 驱动已预装在公共镜像中，并配备了 license 授权，无需您自行安装。

如果您使用的是计算加速型 GPU 云主机，需要购买预装了 GRID 驱动的收费镜像。

说明

目前仅部分资源池支持预装 GRID 驱动的收费镜像，其他资源池需要您手动安装驱动并搭建 license server，请参见安装 GRID 驱动。

6.3管理类

我能变更 GPU 云主机的配置吗?

可以，详细的 GPU 云主机变配操作请参见变配。

Windows GPU 云主机中的 cloudbase-init 帐户是什么?

Windows GPU 云主机中的 cloudbase-init 帐户为 Cloudbase-Init 代理程序的内置帐户，用于弹性云主机启动的时候获取元数据并执行相关配置。如果删除此帐户，会影响云管理平台的相关功能，建议您不要修改、删除此帐户。 如果自行修改、删除此帐户或者卸载 Cloudbase-Init 代理程序，会导致由此 GPU 云主机创建的 Windows 私有镜像所生成的新云主机初始化的自定义信息注入失败。

GPU 云主机在什么时候进入开通状态?

当您支付完费用且系统扣款成功后，将自动为您开通 GPU 云主机。在创建 GPU 云主机时，由于系统盘的创建需要少许时间，所以等系统盘创建出来后才可看到创建中的 GPU 云主机。

如何处理支付订单后 GPU 云主机开通失败的问题?

一般 3 分钟内即可开通成功，如果长时间无法开通成功，请您提交工单，客服会协助您排除故障、开通 GPU 云主机。如果故障无法及时排除，您可以选择取消订单，客服会做退费处理，将订单费用退还至您的的账户中。

GPU 云主机重启后，主机名为什么被还原为安装时的主机名?

以 CentOS 7 操作系统的 GPU 云主机为例:

1.登录 Linux GPU 云主机，查看“cloud-init”的配置文件。

2.检查“/etc/cloud/cloud.cfg”文件中“update_hostname”是否被注释或者删除。 如果没有被注释或者删除，则需要注释或删除 “-update_hostname”语句。

说明

“update_hostname”表示每次重启时， “cloud-init”都会更新主机名。

如何删除、重启 GPU 云主机?

删除 GPU 云主机:

1.登录天翼云控制中心。

2.选择 GPU 云主机所在的区域。

3.选择“计算 > 弹性云主机”。

4.选中目标 GPU 云主机，并单击"操作"列下的"更多 > 删除"。

重启 GPU 云主机:

1.登录天翼云控制中心。

2.选择 GPU 云主机所在的区域。

3.选择“计算 > 弹性云主机”。

4.选中目标 GPU 云主机，并单击"操作"列下的"更多 > 重启"。

更多 GPU 云主机管理操作请参见管理 GPU 云主机。

如何在操作系统内部修改 GPU 云主机密码?

Windows GPU 云主机

1.登录 Windows GPU 云主机。

2.使用快捷键“Win+R”打开“运行”页面。

3.输入命令行“cmd”打开命令行窗口。

4.执行以下命令，修改密码。

```
net user Administrator 新密码
```

Linux GPU 云主机

1.以 root 用户登录 Linux GPU 云主机

2.执行以下命令，重置 root 的用户密码。

```
passwd
```

根据系统回显信息，输入新密码。

系统显示如下回显信息时，表示密码重置成功。

passwd: all authentication tokens updates successfully

注意

更多管理类问题请参见 弹性云主机>常见问题>云服务器管理 。

6.4登录类

支持 Cloud-init 特性的 GPU 云主机登录失败怎么办?

使用 Cloud-init 特性的 GPU 云主机时，如果登录失败，可以从以下几个原因进行排查：

1.判断登录 GPU 云主机时使用的密钥对是否正确。

2.GPU 云主机需绑定弹性 IP。

如果以上操作均正常，但仍无法启动或连接 GPU 云主机，请提交工单联系客服进行处理。

如何处理 VNC 方式登录 GPU 云主机后，查看数据失败，VNC 无法正常使用的问题?

当您通过 VNC 方式登录 GPU 云主机后，出现查看数据出现乱码、错误码时，主要原因有如下几点，您可依次对照排查：

- 查看大文件长时间未读取完成；
- 游戏像素过高、播放视频清晰度过高；
- 浏览器缺陷，占用内存大。

如存在以上场景，则您需要重新登录 GPU 云主机或者更换浏览器。更换浏览器请从如下版本中进行选择：

远程登录弹性云主机时，对浏览器版本的要求？ 。

通过控制台登录 GPU 云主机时提示 1006 或 1000 怎么办?

出现以上的现象，有可能是由于以下原因造成：

- 连接期间长时间未操作，链接已断开；
- GPU 云主机非正常可用状态；

- 存在其他子用户登录 VNC。

当出现该现状时，您可首先通过重新连接 VNC 远程登录界面进行排查，操作步骤如下：

1.退出当前 VNC 登录界面。

2.选中 GPU 云主机，选择远程登录，重新登录主机。若登录失败，则请用户检查当前 GPU 云主机运行状态，是否处于“运行中”，若此时云主机状态异常，请联系人工客服解决问题。

3.确认是否有其他子用户正在使用该台 GPU 云主机的 VNC 界面。

Windows 系统的图形加速基础型 GPU 云主机通过控制台的 VNC 远程连接实例出现黑屏怎么办?

当 Windows 系统的 GPU 云主机安装了 GRID 驱动后，VM 的显示输出将由 GRID 驱动管理，VNC 无法再获取到集成显卡的画面，因此，VNC 显示会变成黑屏状态，属于正常现象。建议您使用 MSTSC 方式连接 GPU 云主机，详细操作步骤请参见远程桌面连接（MSTSC 方式）。

6.5 显卡及驱动类

6.5.1 执行 nvidia-smi 命令时遇到报错：Failed to initialize NVML: Driver/library version mismatch

问题描述

用户执行 nvidia-smi 命令时遇到报错：Failed to initialize NVML: Driver/library version mismatch。

可能原因

用户在升级或者降级 NVIDIA 驱动时如果未按照升级或降级 NVIDIA 驱动进行操作，导致之前加载的旧的驱动未卸载干净，并与新的驱动冲突，引发驱动未正常加载。

解决方案

此问题原因是 NVIDIA 内核驱动版本与系统驱动不一致，可参考以下步骤解决。

1.查看内核加载了哪些涉及 GPU 的模块。

```
lsmod | grep nvidia
```

2.查看下有哪些进程使用了 nvidia*。

```
sudo lsof -n -w /dev/nvidia*
```

3.卸载对应的 GPU 在内核中的模块或者启动的进程。

```
rmmod nvidia_drm

rmmod nvidia_modeset

rmmod nvidia
```

```
kill -9 xxx
```

4.重启 nvidia-smi。

```
nvidia-smi
```

6.5.2 未禁用 nouveau 导致的驱动安装失败

问题描述

客户在安装驱动时报错，提示 ERROR: The Nouveau kernel driver is currently in use by your system. This driver is incompatible with the NVIDIA driver……”

可能原因

安装 nvidia 显卡驱动首先需要禁用 nouveau，不然会碰到冲突的问题，导致无法安装 nvidia 显卡驱动。

解决方法

1、 执行命令判断是否加载 Nouveau 驱动程序。

如果以下命令打印任何内容，则说明已加载 Nouveau 驱动程序。

```
lsmod |grep nouveau
```

```
[root@ecm-b7ef ~]# lsmod |grep nouveau
nouveau              1940454  0
mxm_wmi                13021  1 nouveau
wmi                    21636  2 mxm_wmi,nouveau
video                  24538  1 nouveau
i2c_algo_bit           13413  1 nouveau
ttm                   100769  2 cirrus,nouveau
drm_kms_helper        186531  2 cirrus,nouveau
drm                   468454  5 ttm,drm_kms_helper,cirrus,nouveau
```

2、执行如下命令将 Nouveau 驱动加入黑名单。

```
vim /etc/modprobe.d/blacklist-nouveau.conf

blacklist nouveauoptions nouveau modeset=0
```

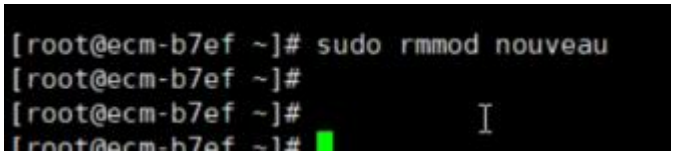
3、卸载 Nouveau 模块。

#CentOS 类系统执行

```
sudo dracut --force

sudo rmmod nouveau
```

#Ubuntu 类系统执行

```
sudo update-initramfs -u

sudo rmmod nouveau
```



4、执行命令确认是否禁用成功。

如果以下命令没有打印内容则为禁用成功

```
lsmod |grep nouveau
```

6.5.3GPU 云主机发生掉卡现象

问题描述

GPU 云主机发生掉卡现象，比如申请 4 卡的计算加速型 GPU 云主机，但是 nvidia-smi 显示的显卡数少于 4 张。

具体现象：

执行以下命令查看 dmesg 日志

```
dmesg |grep -i nvrm
```

查找相关字段发现如下错误：

```
NVRM:GPU 0000:00:07.0: RmInitAdapter failed!....NVRM:GPU 0000:00:07.0: rm init-adapter failed,device minor numb....
```

可能原因

宿主机硬件故障

解决方案

如遇到该问题请提工单联系运维处理。

6.5.4 缺少 libelf-dev, libelf-devel or elfutils-libelf-devel 导致 centos 8.x 的计算加速型 GPU 云主机安装驱动时报错

问题描述

centos 8.x 的计算加速型 GPU 云主机安装驱动不成功，通过查看 nvidia 驱动安装日志发现报错。

```
cat /var/log/nvidia-installer.log #查看 GPU 驱动安装日志
```

返回结果：

可能原因

```
-> Error.
ERROR: An error occurred while performing the step: "Checking t
o see whether the nvidia kernel module was successfully built".
 See /var/log/nvidia-installer.log for details.
-> The command `cd ./kernel; /usr/bin/make -k -j16  NV_EXCLUDE_
KERNEL_MODULES="" SYSSRC="/usr/src/kernels/4.18.0-193.el8.x86_6
4" SYSOUT="/usr/src/kernels/4.18.0-193.el8.x86_64" NV_KERNEL_MO
DULES="nvidia"` failed with the following output:

make[1]: Entering directory '/usr/src/kernels/4.18.0-193.el8.x8
6_64'
Makefile:975: *** "Cannot generate ORC metadata for CONFIG_UNWI
NDER_ORC=y, please install libelf-dev, libelf-devel or elfutils
-libelf-devel".  Stop.
make[1]: Leaving directory '/usr/src/kernels/4.18.0-193.el8.x86
_64'
make: *** [Makefile:82: modules] Error 2
ERROR: The nvidia kernel module was not created.
```

缺少 libelf-dev, libelf-devel or elfutils-libelf-devel 导致。

解决方法

1.通过 yum 来安装所缺少的依赖

```
yum install -y elfutils-libelf-devel
```

2.重新安装驱动，详情请参见安装 Tesla 驱动。

6.5.5PI7 规格云主机安装 510.47.03 版本 Tesla 驱动时，GPU 利用率过低

问题描述

客户使用 PI7 型 GPU 云主机，安装 TESLA 驱动版本为 510.47.03，运行部分多线程任务时，显卡利用率较低，耗时却较长。

可能原因

英伟达 510.47.03 版本的 TESLA 驱动默认开启了 GSP。GSP 全称为 GPU 系统处理器（GSP），可用于卸载 GPU 初始化和管理任务，但部分情况下开启 GSP 会导致掉卡或显卡利用率低，此时需要关闭 GSP。

解决方法

1.禁用 GSP-RM：

```
sudo su -c 'echo options nvidia NVreg_EnableGpuFirmware=0 >
/etc/modprobe.d/nvidia-gsp.conf'
```

2.启用内核

```
#if ubuntu
sudo update-initramfs -u#if centos
dracut -f
```

3.重新启动

```
reboot
```

4.检查是否有效。如果 EnableGpuFirmware: 0 表示 GSP-RM 被禁用。

```
cat /proc/driver/nvidia/params | grep EnableGpuFirmware
```

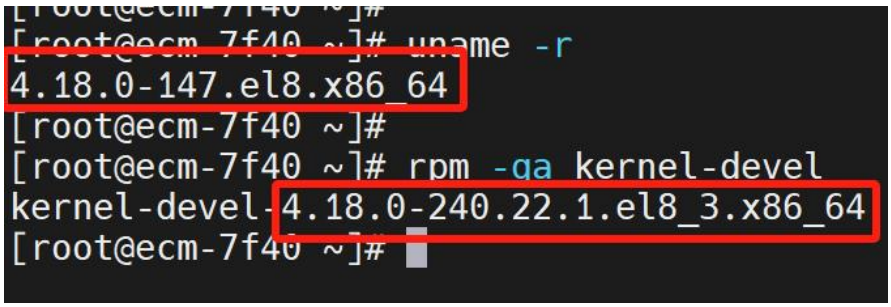
6.5.6 内核版本与 kernel-devel 版本不一致导致 centos 8.x 的计算加速型 GPU 云主机安装驱动时报错

问题描述

centos 8.x 的计算加速型 GPU 云主机安装驱动不成功，通过如下两个命令查看操作系统内核版本与 kernel-devel 版本不一致。

```
uname -r      #查看内核版本
rpm -qa kernel-devel     #查看 kernel-devel 版本
```

示例：



可能原因

操作系统内核版本与 kernel-devel 版本不一致导致驱动安装失败

解决方法

1.重新下载与操作系统内核版本一致的 rpm 包并上传至云主机。

方法一：kernel-devel 的 rpm 包下载地址（可以找到 80%版本的

包）： https://pkgs.org/download/kernel-devel

方法二：centos 的 kernel-devel 的 rpm 包下载地址： https://vault.centos.org/ （例如在 centos8.2 系统中缺少 4.18.0-193.el8.x86_64 可以在

https://vault.centos.org/8.2.2004/BaseOS/x86_64/os/Packages/ 中找到）

注意

方法二仅适用于 centos 操作系统。

2.安装 rpm 包

```
rpm -Uvh --replacefiles --force --nodeps *.rpm
```

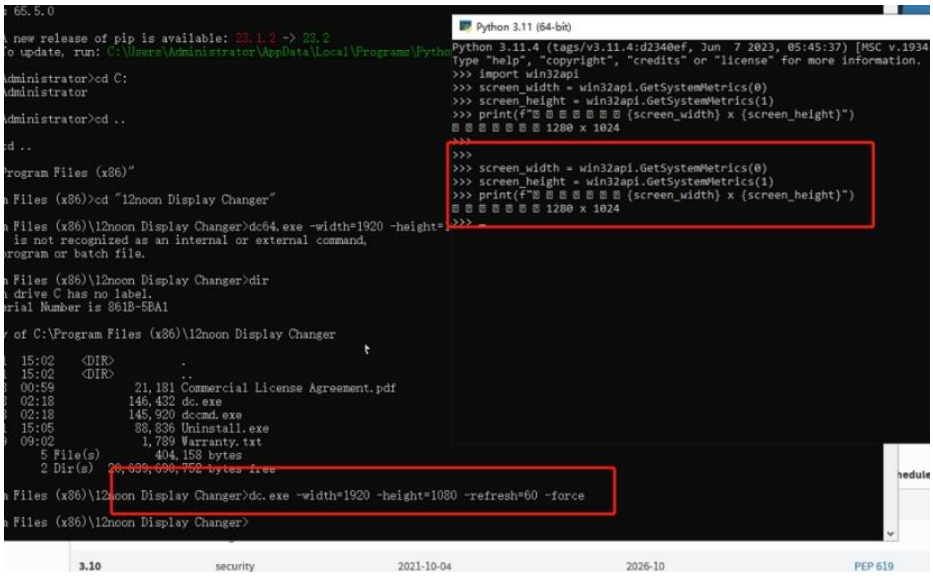
3.重新安装驱动，详情请参见安装 Tesla 驱动-GPU 云主机-用户指南-安装 NVIDIA 驱动 - 天翼云 (ctyun.cn)

6.6.7 通过 Display Changer 分辨率修改工具修改 PI7 规格云主机的分辨率不生效。

问题描述

使用 TightVNC 登陆 windows2019-vGPU 操作系统的 PI7 规格 GPU 云主机，运行 Display Changer，通过 dc64.exe -width=1920 -height=1080 -refresh=60 -force 命令修改分辨率为 1920*1080，但实际不生效，分辨率仍为 1280*1024。

如下图：



可能原因

通过 TightVNC 登录时默认选择的是显示器 1，需要在执行 Display Changer 命令修改分辨率时指定为显示器 2（在 Nvidia A10 上）。

解决方法

1.执行如下命令修改分辨率并指定 Monitor。

```
dccmd.exe -monitor="\\.\DISPLAY2" -width=1920 -height=1080
```

2.执行命令查看分辨率修改成功。

```
from win32api import GetSystemMetrics
```

注意 NIDIA Virtual Applications 许可证类型最大支持的分辨率仅为 1280*1024，若要修改分辨率为 1920*1080 需要确保许可证类型为 NVIDIA RTX Virtual Workstation。