高性能应用服务 HAI

应用指南

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

应用指南
部署 Stable Diffusion WebUI
快速创建
访问方式
提示词指南
导入外部模型
新增插件
其他常见操作
部署 ChatGLM3 6B
快速创建
访问方式
手动中断与重启 Gradio WebUI 服务
在 HAI 环境中部署自定义项目
部署 Pytorch 2.0.0
快速创建
访问方式
管理系统 CUDA
管理系统 cuDNN
管理 Python 虚拟环境依赖

版权所有：腾讯云计算（北京）有限责任公司

应用指南
部署 Stable Diffusion WebUI
快速创建

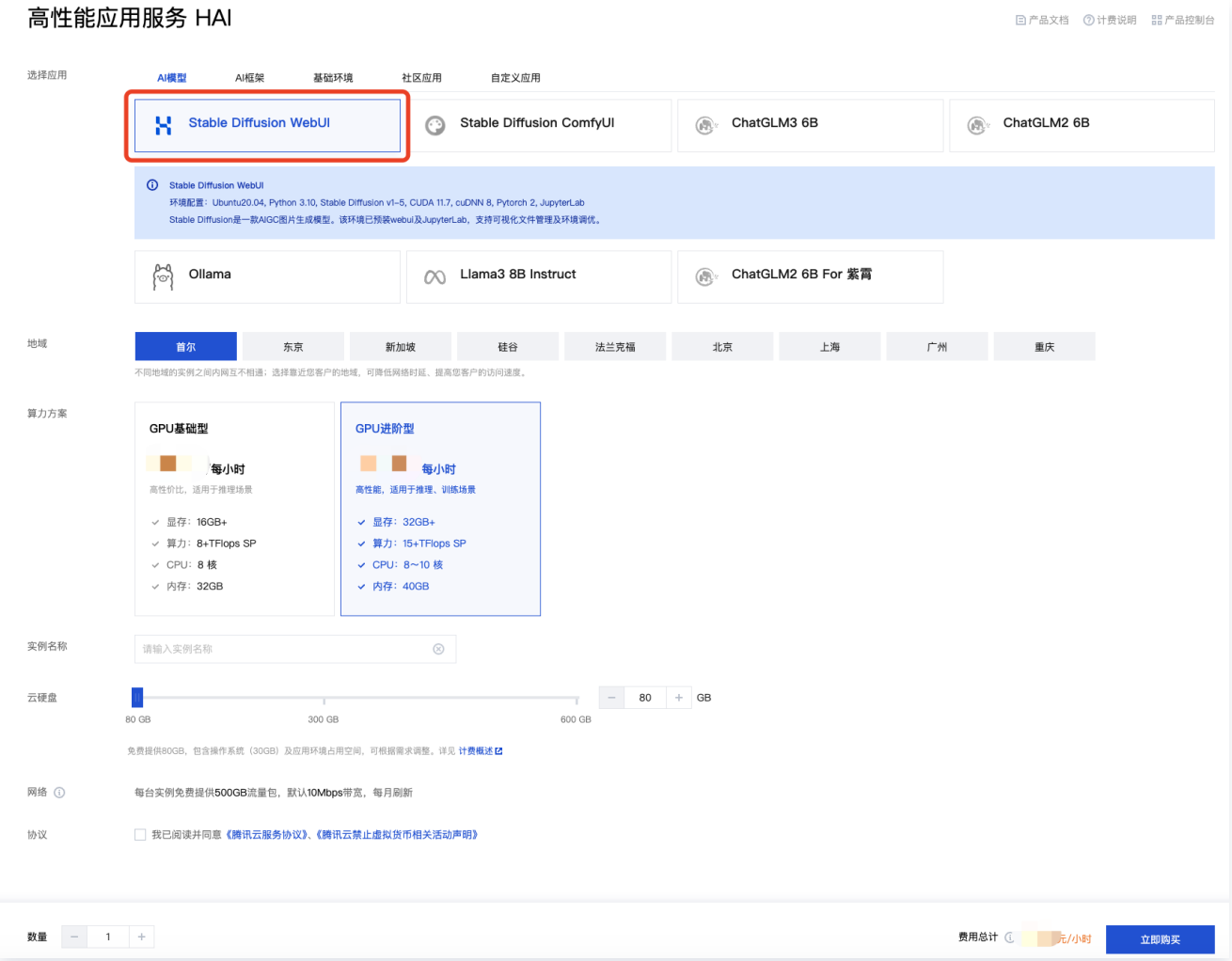
最近更新时间：2025-06-26 14:57:42

操作场景

本次介绍 腾讯云高性能应用服务 HAI Stable Diffusion WebUI 应用实例的购买流程。

操作步骤

1. 登录 高性能应用服务 HAI 控制台 。
2. 单击**新建**，进入 高性能应用服务 HAI 购买页面 。



- **选择应用：**选择AI模型，应用选择 Stable Diffusion WebUI。
- **地域：**建议选择靠近自己实际地理位置的地域，降低网络延迟、提高您的访问速度。

- **算力方案：**按需选择配置。
- **实例名称：**自定义实例名称，若不填则默认使用实例 ID 替代。
- **硬盘：**默认提供 80GB 免费空间，可根据实际使用需求进行调整。
- **网络：**每台实例每月免费提供 500GB 流量包，默认10Mbps 带宽，每月刷新。
- **购买数量：**默认1台。

3. 单击**立即购买**。
4. 核对配置信息后，单击**提交订单**，并根据页面提示完成支付。
5. 等待创建完成。单击实例**任意位置**并进入该实例的详情页。

6. 您可以在此页面查看 Stable Diffusion WebUI 详细的配置信息，到此为止，说明您的 Stable Diffusion WebUI 应用实例购买成功。

访问方式

最近更新时间：2024-11-22 14:57:32

操作场景

购买 Stable Diffusion WebUI 应用实例后，我们有Gradio WebUI、JupyterLab、SSH 3 种方式访问该应用实例，以下将逐一介绍。

前提条件

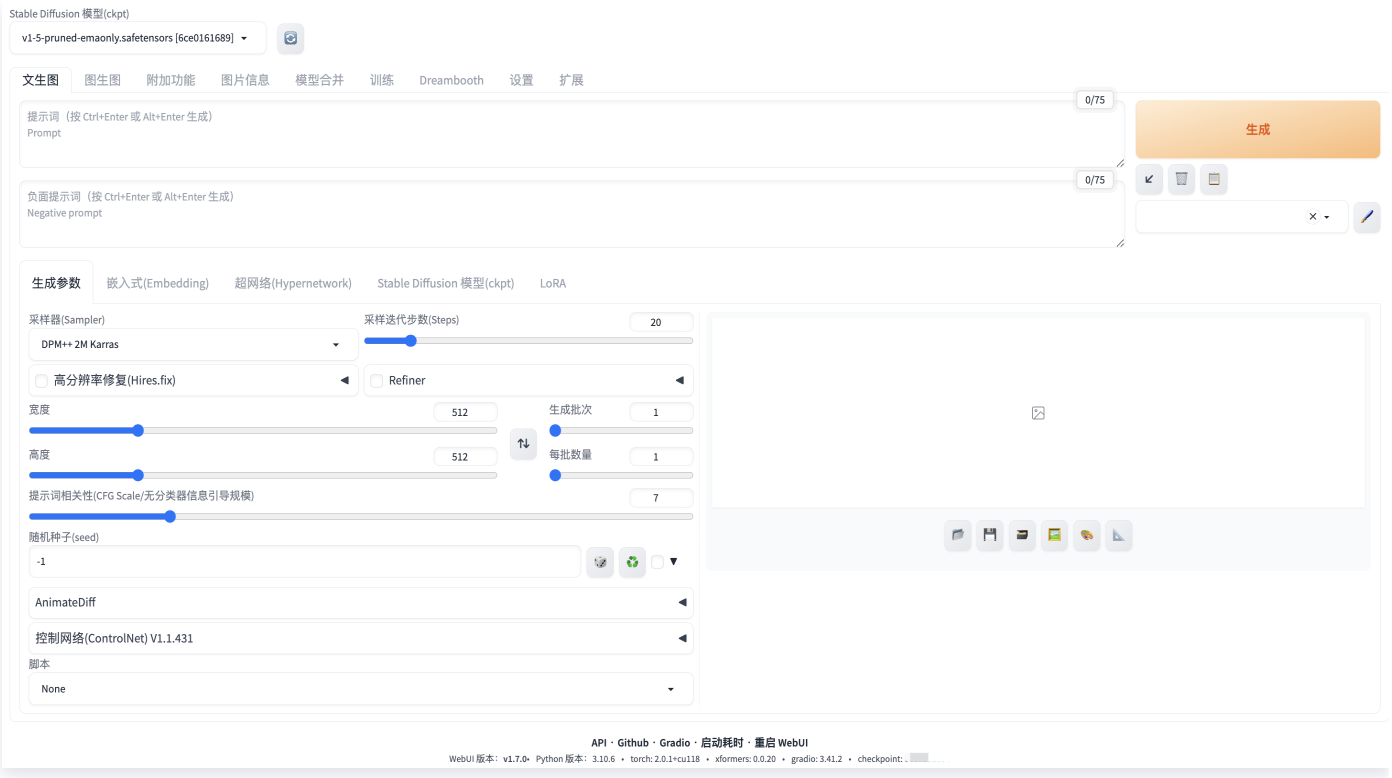
已经购买了 Stable Diffusion WebUI 应用实例（参见 Stable Diffusion WebUI 的快速创建），能够登录腾讯云高性能应用服务控制台。

操作步骤

方法1：通过 Gradio WebUI 访问

1. 登录 高性能应用服务控制台。
2. 在**算力管理**页中选择**算力连接** > Gradio WebUI 。

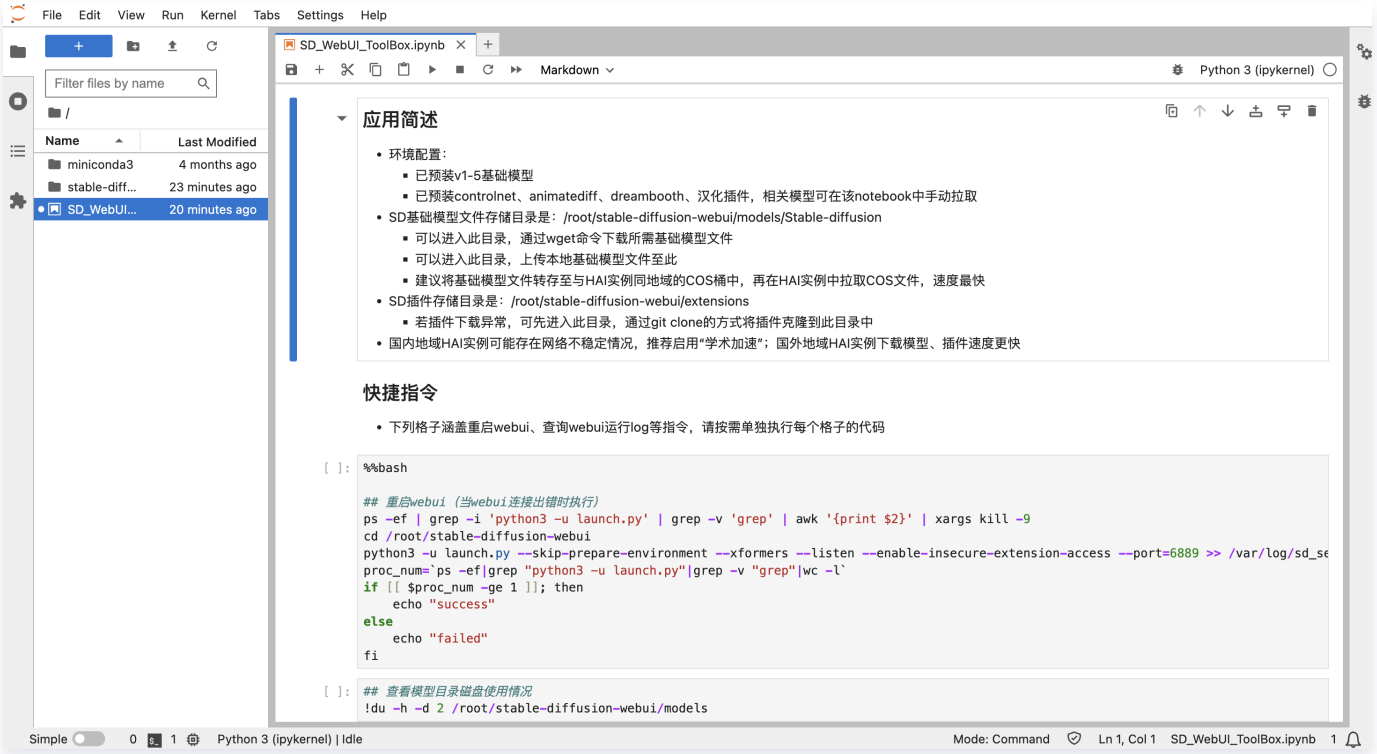
3. 弹出如下图所示的新的页面，说明通过 Gradio WebUI 的访问已建立，您可以在此使用 WebUI 进行图片生成。生成指南可以参见 Stable Diffusion WebUI的提示词指南。



方法2：通过 JupyterLab 访问

1. 在**算力管理**页中选择**算力连接** > **JupyterLab** 。

2. 弹出如下图所示的新的页面，说明通过 **JupyterLab** 的访问已建立，您可以参阅这里的应用简述和快捷指令，利用 **Stable Diffusion WebUI** 的成熟环境，快速创建您自己的代码项目或应用服务。



方法3：通过 SSH 访问

1. 在控制台的上边栏的右侧，找到并单击 站内信 ，找到应用实例创建时推送的消息，单击消息的任意位置进入该条消息。

2. 在该条消息中，找到**默认用户名**，**登录密码**，**IP地址（公）**这三个配置信息，并根据这些信息，进行 SSH 登录。

提示词指南

最近更新时间：2025-06-26 14:57:42

操作场景

使用高性能应用服务 HAI 部署的 **Stable Diffusion WebUI** 快速进行 AI 绘画。

前提条件

已经购买了 **Stable Diffusion WebUI** 应用实例（参见 Stable Diffusion WebUI 的快速创建），成功登录腾讯云高性能应用服务控制台并已经打开 **Gradio WebUI**（参见Stable Diffusion WebUI 的访问方式）。

操作步骤

> **注意：**
> 提示词（Prompt）越多，AI 绘图结果会更加精准。另外，目前中文提示词效果不佳，建议使用英文提示词。

1. 在实例列表中选择**算力连接** > **Gradio WebUI**。

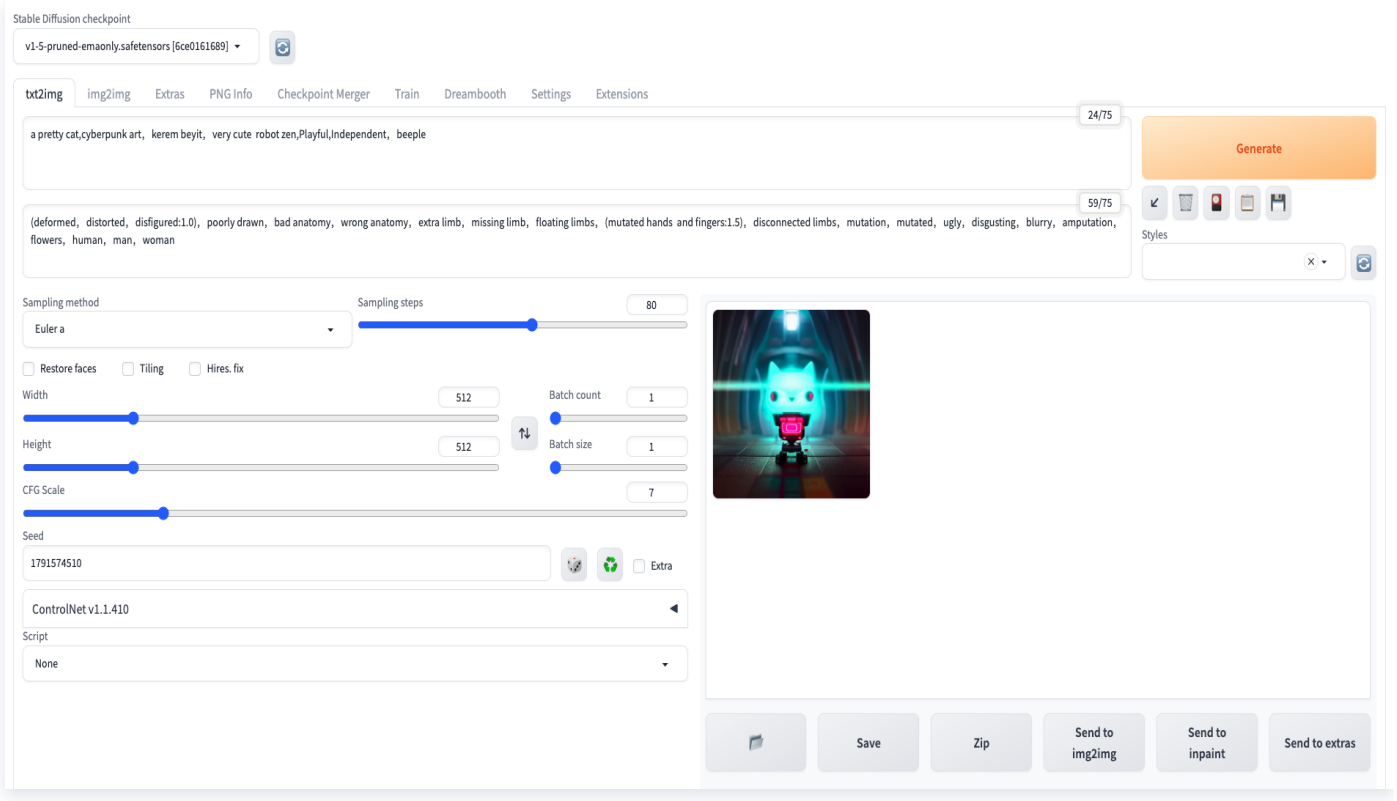
2. 使用高性能应用服务 HAI 部署的 Stable Diffusion WebUI 快速进行 AI 绘画。

   我们使用 **Stable Diffusion WebUI** 生成一张猫咪图片，配置以下参数后，单击 **Generate** 即可。

参数名	描述	值
Prompt	主要描述图像，包括内容风格等信息，原始的 WebUI会对此处有字数的限制，您可以通过安装一些插件来突破字数的限制。	a pretty cat,cyberpunk art，kerem beyit，very cute robot zen,Playful,Independent，beeple
Negative prompt	为了提供给模型，您不需要的风格。	(deformed，distorted，disfigured:1.0)，poorly drawn，bad anatomy，wrong

		anatomy，extra limb，missing limb，floating limbs，(mutated hands and fingers:1.5)，disconnected limbs，mutation，mutated，ugly，disgusting，blurry，amputation，flowers，human，man，woman
CFG scale	分类器自由引导尺度，图像与提示符的一致程度。越低的值产生的结果越有创意，数值越大成图越贴近描述文本。一般设置为7。	7
Sampling method	扩散算法的去噪声采样模式会影响最终效果，不同的采样模式的结果会有很大差异，一般是默认选择 euler。	Euler a
Sampling steps	在使用扩散模型生成图片时所进行的迭代步骤。需要注意的是，更高的迭代步数会消耗更多的计算时间和成本，但并不意味着一定会得到更好的结果。	80
Seed	随机数种子，生成每张图片时的随机种子。	1791574510

3. 配置并生成如下图：


Stable Diffusion checkpoint
v1-5-pruned-emaonly.safetensors [6ce0161689]
txt2img
img2img
Extras
PNG Info
Checkpoint Merger
Train
Dreambooth
Settings
Extensions
24/75
a pretty cat,cyberpunk art, kerem beyit, very cute robot zen,Playful,Independent, beeple
59/75
(deformed, distorted, disfigured:1.0), poorly drawn, bad anatomy, wrong anatomy, extra limb, missing limb, floating limbs, (mutated hands and fingers:1.5), disconnected limbs, mutation, mutated, ugly, disgusting, blurry, amputation, flowers, human, man, woman
Generate
Styles
Sampling method
Euler a
Sampling steps
80
Restore faces
Tiling
Hires. fix
Width
512
Height
512
Batch count
1
Batch size
1
CFG Scale
7
Seed
1791574510
Extra
ControlNet v1.1.410
Script
None
Save
Zip
Send to img2img
Send to inpaint
Send to extras

导入外部模型

最近更新时间：2025-06-26 14:57:42

操作场景

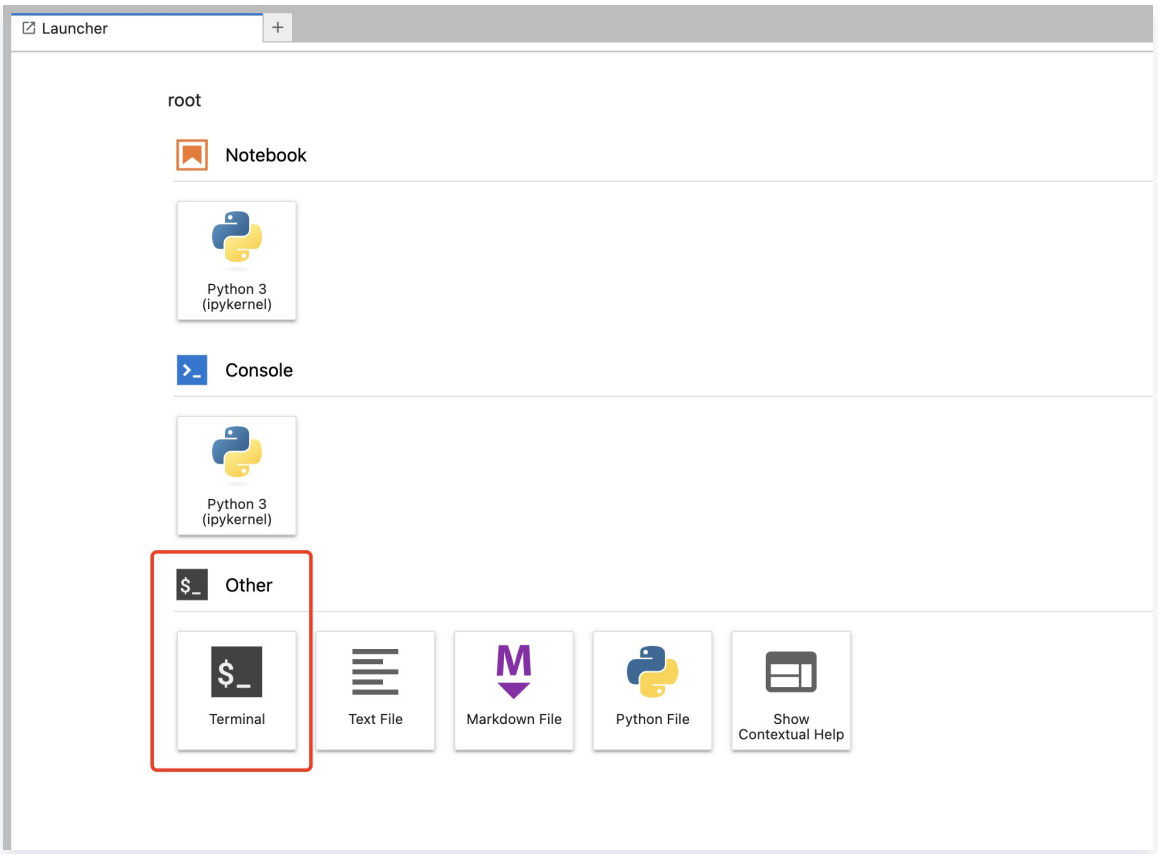
您可以在Stable Diffusion WebUI中导入外部模型，以自定义您的使用体验。导入外部模型一般有**直接下载**、**通过JupyterLab上传**、**通过COS上传**、**通过第三方客户端**几种方式。

操作方式

方法一：直接下载（推荐境外地域实例使用）

1. 登录 高性能应用服务 HAI 控制台。
2. 选择需要连接的算力，然后单击**算力连接**。在下拉菜单中，单击 **JupyterLab**。

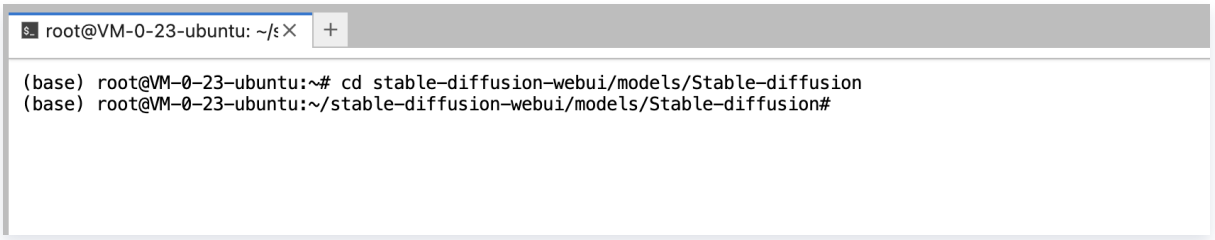
3. 进入 JupyterLab 后，选择 **Other** > **Terminal**，进入终端界面。



4. 进入后默认处于根目录下，输入如下指令进入 stable diffusion 的模型文件夹。

```
cd stable-diffusion-webui/models/Stable-diffusion
```

截图如下：



5. 获取预期使用的模型文件下载地址，在 `stable-diffusion-webui/models/Stable-diffusion` 目录下，将文件使用 `wget` 指令下载到目录中。

```
wget https://xyz.com/api/download/models/xxxx
```

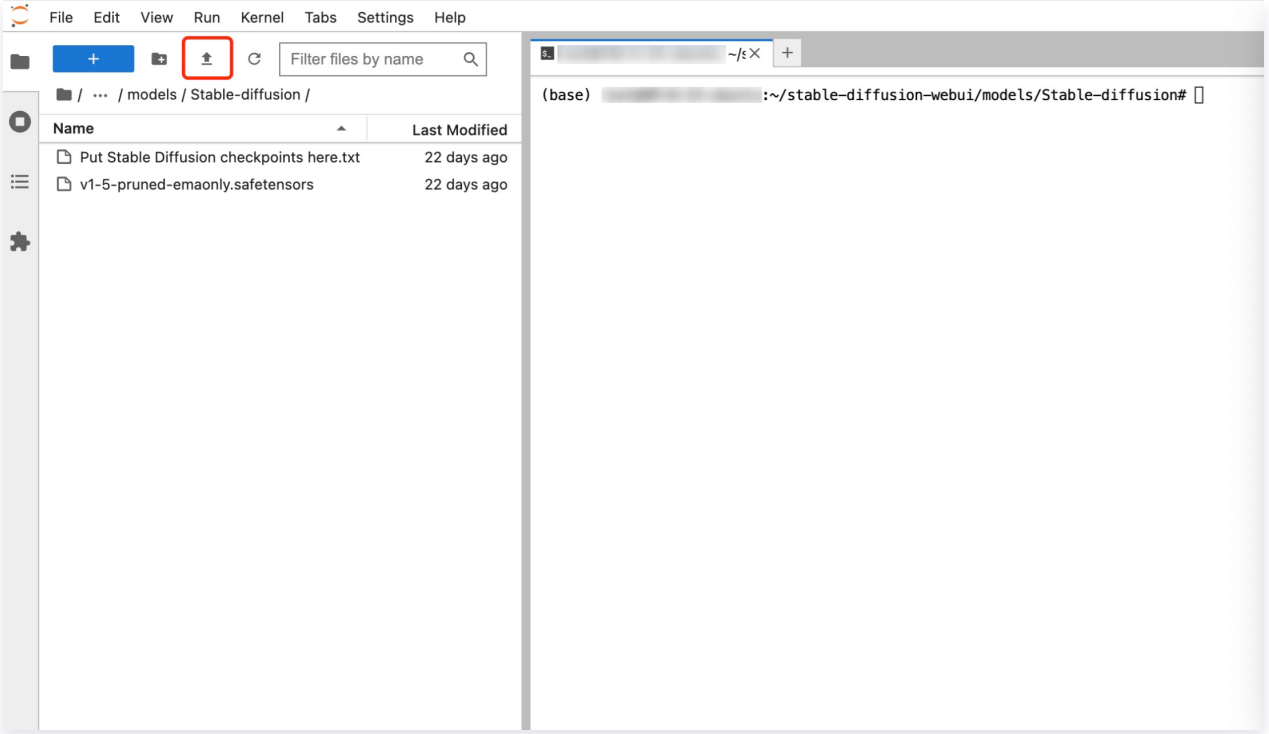
6. 刷新 WebUI，即可使用新模型。

方式二： 通过 JupyterLab 上传（适合1GB以内小文件）

1. 将预期使用的模型提前下载到电脑本地。
2. 选择 JupyterLab 进行算力连接，在左侧文件目录处，依次双击进入如下文件夹：

```
root/stable-diffusion-webui/models/Stable-diffusion
```

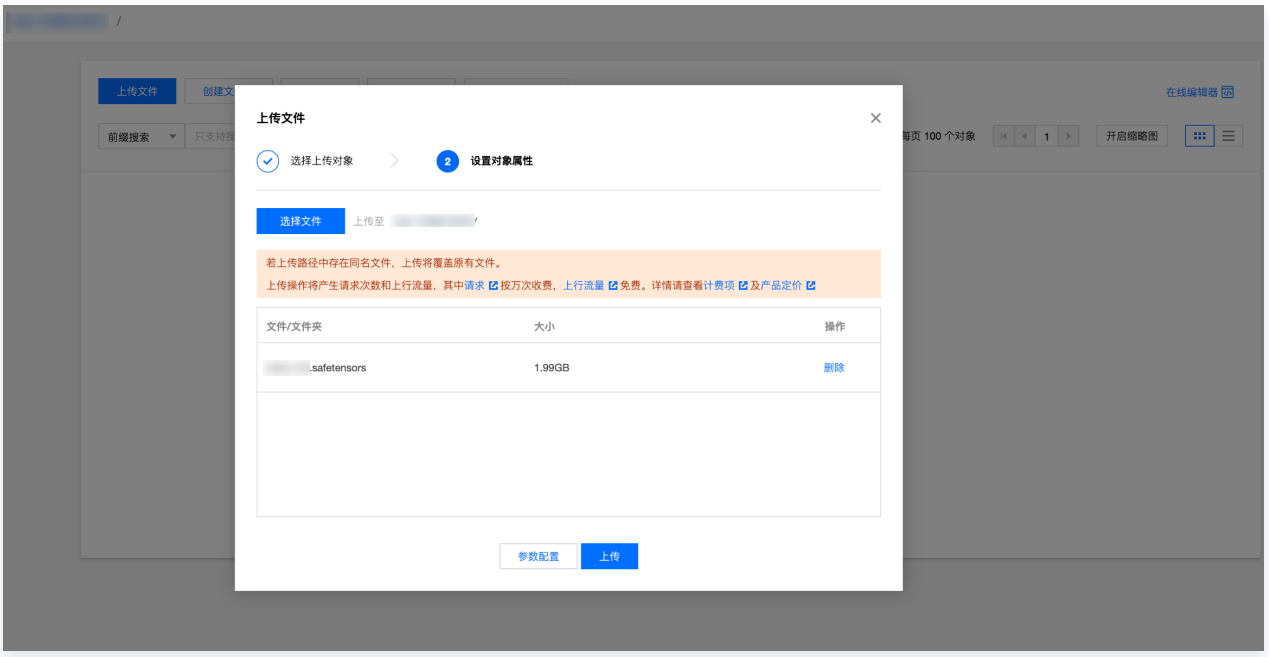
3. 单击左侧文件目录处的**文件上传**，选择已下载的模型文件，即可将模型上传至高性能应用服务实例。



方式三：通过 COS 上传（适合大文件）

1. 将预期使用的模型提前下载到电脑本地。
2. 进入腾讯云对象存储控制台（cos），创建新存储桶并将模型文件上传至该存储桶。

> **注意：**
> 创建 COS 存储桶时，需注意与应用实例相同地域。否则会在拉取文件时产生额外流量费用。



3. 选择 JupyterLab 进行算力连接，进入Terminal后，进入后默认处于根目录下，输入如下指令进入 stable diffusion 的模型文件夹。

```
cd stable-diffusion-webui/models/Stable-diffusion
```

4. 获取在存储桶中的模型文件下载地址，并在该目录下，将文件 wget 到高性能应用服务实例中。

```
wget https://test-xxx.cos.ap-guangzhou.myqcloud.com/lyriel_v16.safetensors
```

5. 刷新 WebUI，即可使用新模型。

方式四：通过第三方客户端

参见 通过 Filezilla 上传和下载数据 。

新增插件

最近更新时间：2025-06-26 14:57:42

操作场景

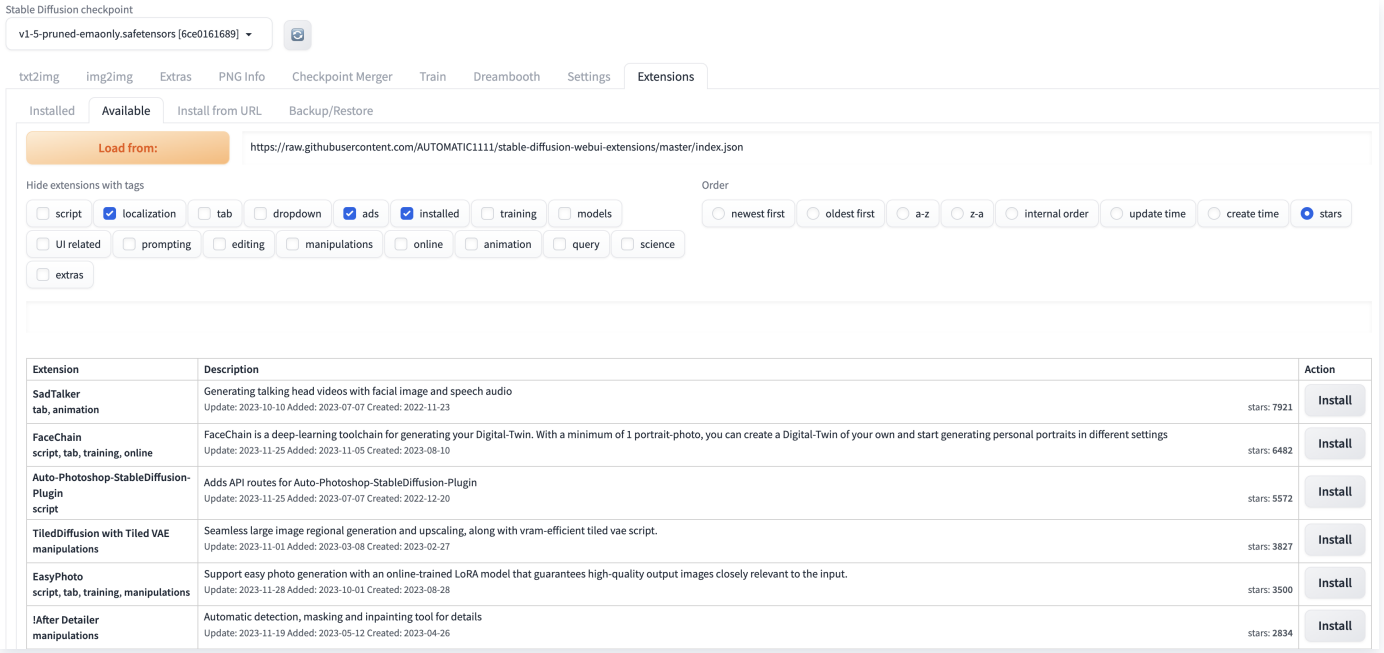
您可以在 Stable Diffusion WebUI 中导入新增插件，以拓展并自定义您的使用体验。

操作方法

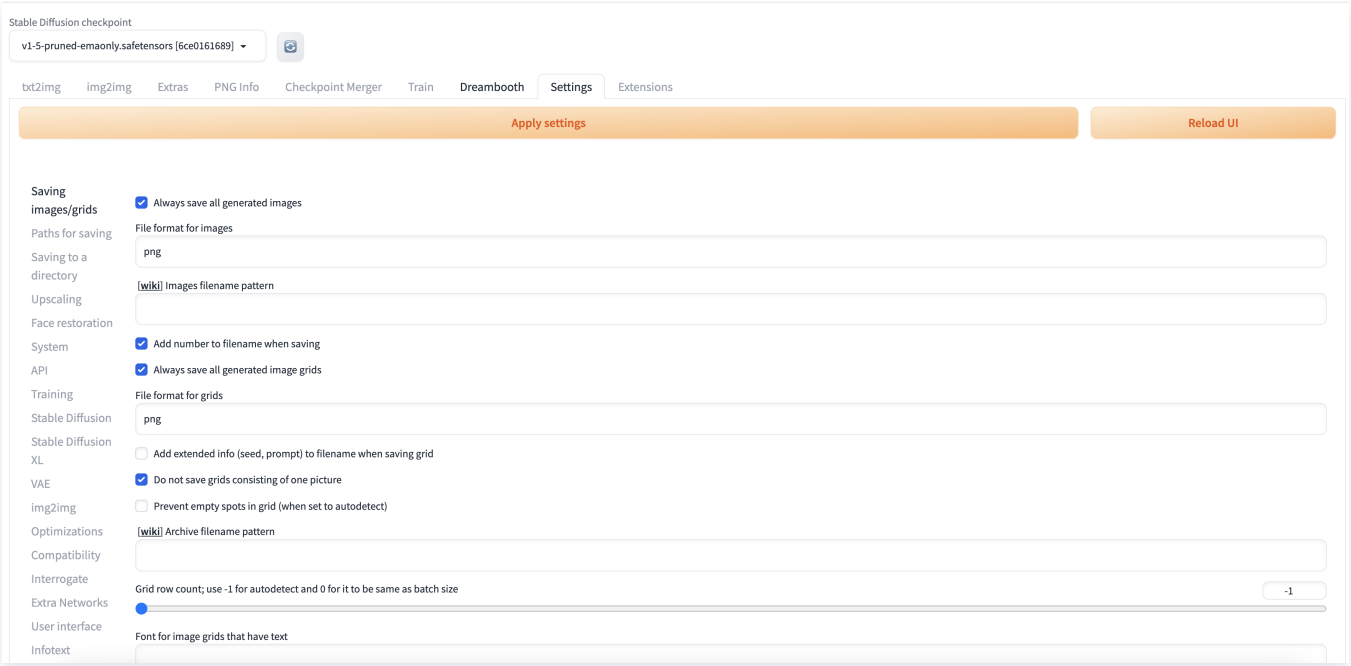
方法一：通过 Gradio WebUI 安装（推荐境外地域实例使用）

1. 登录 高性能应用服务 HAI 控制台。
2. 选择对应的算力，单击**算力连接**。在下拉菜单中，单击 Gradio WebUI。

3. 进入 WebUI，单击进入 extensions，单击 Load from 后，选择预期使用的插件，单击 Install。



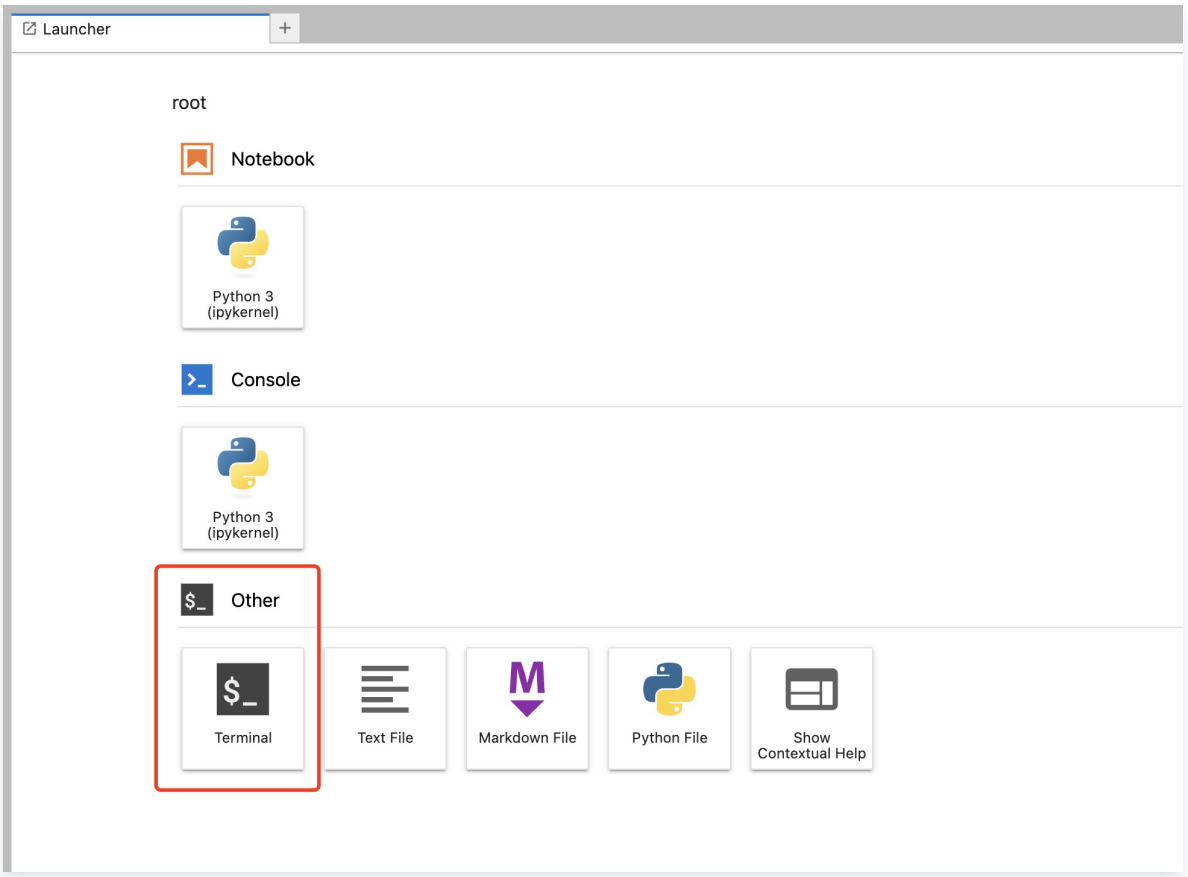
4. 安装完成后，单击进入 Settings，单击 Reload UI 后，插件即可生效。



方法二：通过终端安装

1. 登录 高性能应用服务 HAI 控制台。
2. 选择需要连接的算力，然后单击**连接算力**。在下拉菜单中，单击 **JupyterLab**。

3. 选择 JupyterLab 进行算力连接，进入 Terminal。



4. 进入后默认处于根目录下，输入如下指令进入 extensions 文件夹。

```
cd stable-diffusion-webui/extensions
```

5. 获取预期使用的插件下载地址，在 `stable-diffusion-webui/extensions` 目录下，将文件使用 `git clone` 指令下载到目录中。

```
git clone https://github.com/OpenTalker/SadTalker
```

6. 刷新 WebUI，即可使用新插件。

其他常见操作

最近更新时间：2025-06-26 14:57:42

本文将介绍常用使用 Stable Diffusion 的其他常见操作，包括重启 WebUI、查看运行日志、清空垃圾箱、拉取常见模型等 。

重启 WebUI

若您在使用过程中遇到 WebUI 界面无法正常打开，可进入 Terminal，通过如下指令重新启动 WebUI 服务。

```
cd stable-diffusion-webui
(python3 -u launch.py --skip-prepare-environment --xformers --listen --
enable-insecure-extension-access --port=6889 >> /var/log/sd_service.log
2>&1  &) && tail -f /var/log/sd_service.log
```

查看 WebUI 运行日志

若您希望查看 WebUI 运行状态，包括模型加载、图片生成、插件下载进度等内容，可进入 Terminal，通过如下指令查看 WebUI 运行日志。

```
cd stable-diffusion-webui
tail -f /var/log/sd_service.log
```

查看模型目录磁盘使用情况

这个命令可以查看某个目录的磁盘占用情况。

```
du -h -d 2 /root/stable-diffusion-webui/models
```

拉取模型

执行下列命令，可快速拉取预存模型。

1. 下载 ControlNet 及预处理器（适配 SD1.5及 SDXL，需 23GB 存储空间）。

```
wget -N http://mirrors.tencentyun.com/install/HAI/install_hai_tools.sh
-P /tmp && bash /tmp/install_hai_tools.sh && python3
/root/hai_application/qcloud_hai/hai_tools/download_models_main.py --
model-class controlnet Annotators
```

2. 下载常用基础模型（SDXL、anthingv5，需 8.5GB 存储空间）。

```
wget -N http://mirrors.tencentyun.com/install/HAI/install_hai_tools.sh
-P /tmp && bash /tmp/install_hai_tools.sh && python3
/root/hai_application/qcloud_hai/hai_tools/download_models_main.py --
model-class checkpoint
```

3. 下载常用组件（VAE、embeddings、lcm_lora，需 700MB 存储空间）。

```
wget -N http://mirrors.tencentyun.com/install/HAI/install_hai_tools.sh
-P /tmp && bash /tmp/install_hai_tools.sh && python3
/root/hai_application/qcloud_hai/hai_tools/download_models_main.py --
model-class vae embeddings lora
```

4. 下载 animatediff 模型（需 1.7GB 存储空间）。

```
wget -N http://mirrors.tencentyun.com/install/HAI/install_hai_tools.sh
-P /tmp && bash /tmp/install_hai_tools.sh && python3
/root/hai_application/qcloud_hai/hai_tools/download_models_main.py --
model-class animatediff_model
```

版权所有：腾讯云计算（北京）有限责任公司

部署 ChatGLM3 6B
快速创建

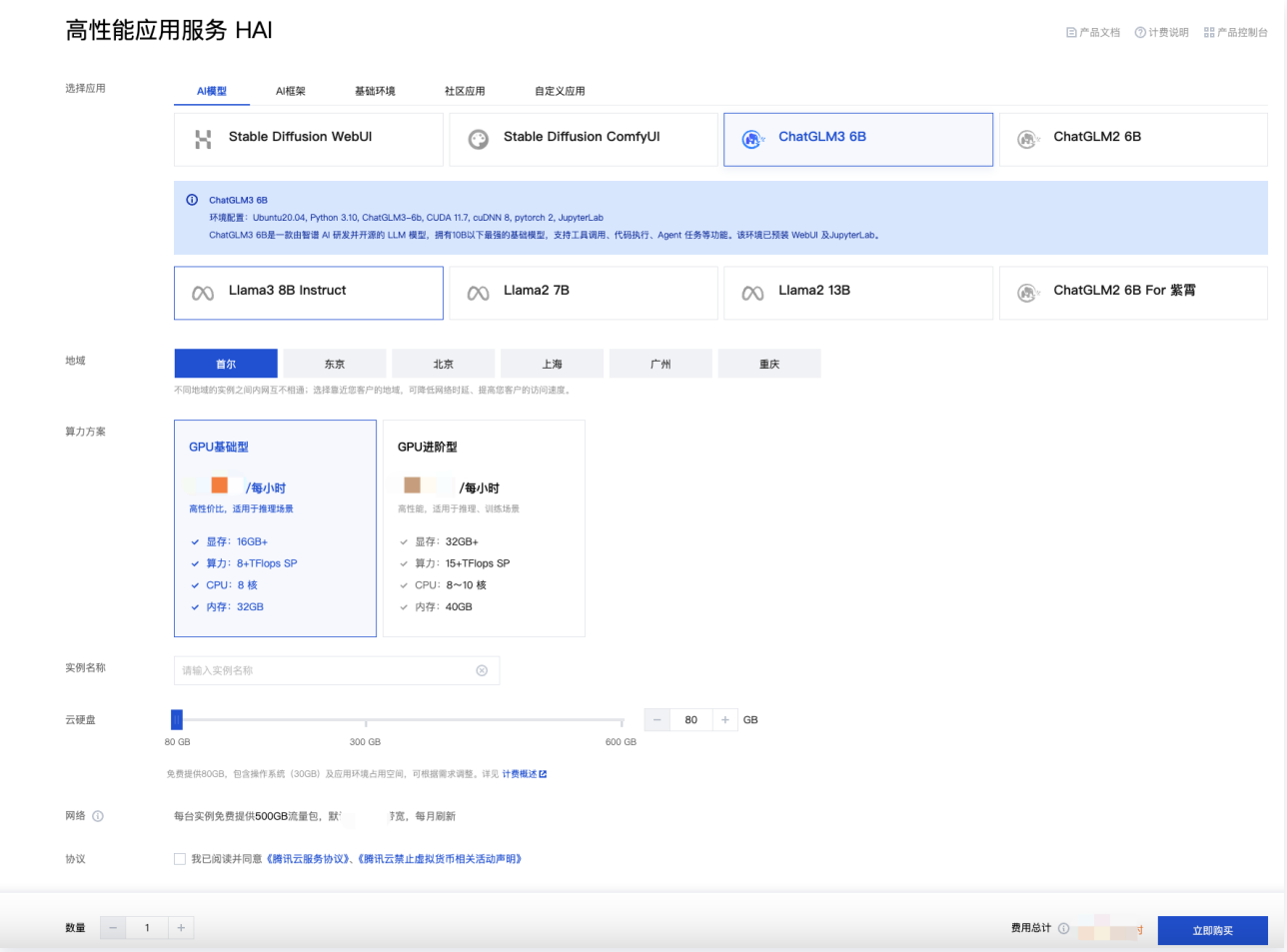
最近更新时间：2025-06-26 14:57:42

操作场景

本次介绍 腾讯云高性能应用服务 HAI ChatGLM3-6B 应用实例的购买流程。

操作步骤

1. 登录 高性能应用服务控制台 。
2. 单击**新建**，进入 高性能应用服务购买页面 。

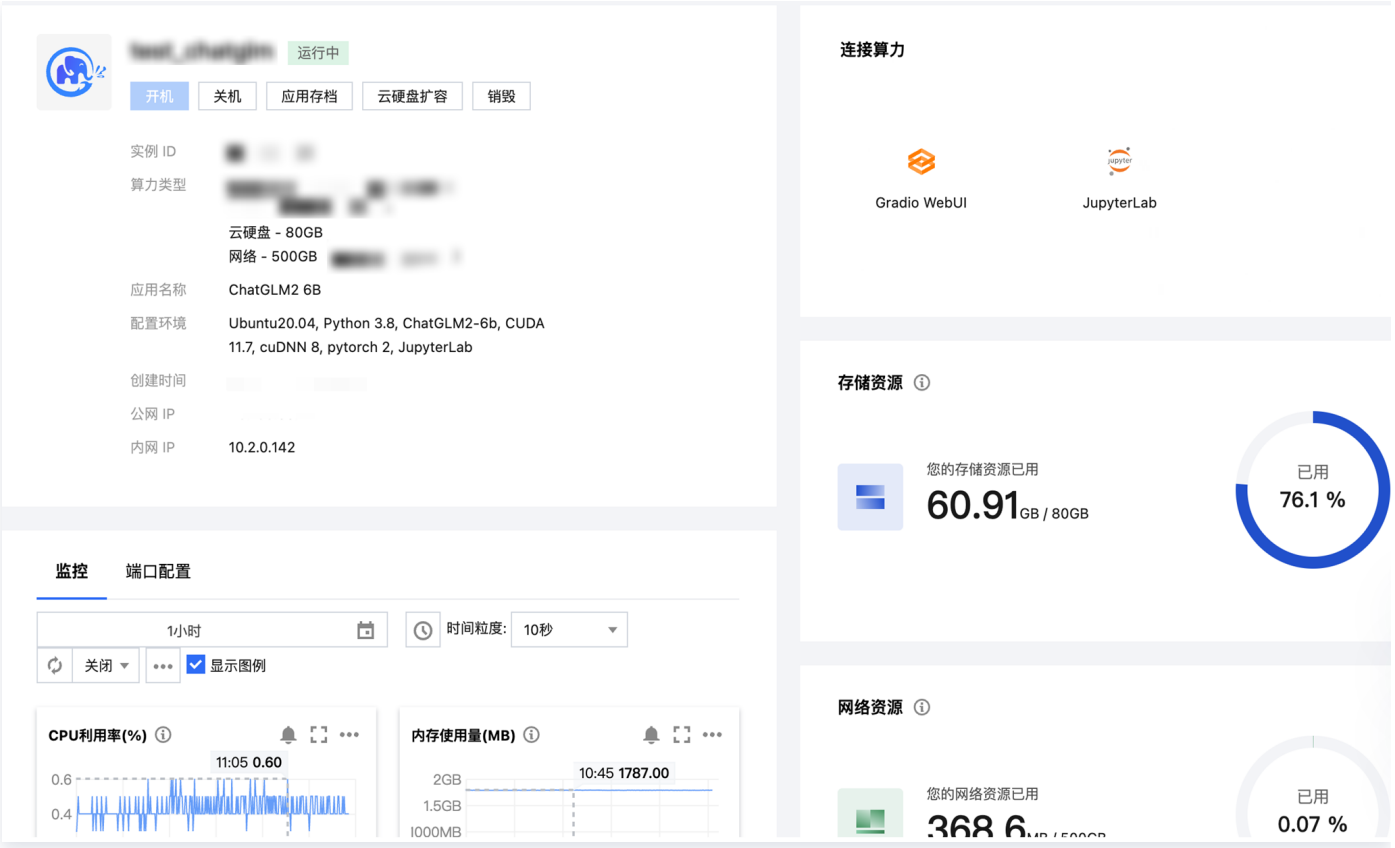


- **选择应用：**选择AI模型，应用选择 ChatGLM3-6B。
- **地域：**建议选择靠近自己实际地理位置的地域，降低网络延迟、提高您的访问速度。
- **算力方案：**本次算力方案选择进阶型，生成图片效率更高。
- **实例名称：**自定义实例名称，若不填则默认使用实例 ID 替代。
- **硬盘：**默认提供 80GB 免费空间，可根据实际使用需求进行调整。

- **网络：**每台实例每月免费提供 500GB 流量包，默认10Mbps 带宽，每月刷新。
- **购买数量：**默认1台。

3. 单击**立即购买**。
4. 核对配置信息后，单击**提交订单**，并根据页面提示完成支付。
5. 等待创建完成。单击实例**任意位置**并进入该实例的详情页。

6. 您可以在此页面查看 ChatGLM3-6B 详细的配置信息，到此为止，说明您的 ChatGLM3-6B 应用实例购买成功。


运行中
开机
关机
应用存档
云硬盘扩容
销毁
实例 ID
算力类型
云硬盘 - 80GB
网络 - 500GB
应用名称
ChatGLM2 6B
配置环境
Ubuntu20.04, Python 3.8, ChatGLM2-6b, CUDA 11.7, cuDNN 8, pytorch 2, JupyterLab
创建时间
公网 IP
内网 IP
10.2.0.142
连接算力
Gradio WebUI
JupyterLab
存储资源
您的存储资源已用
60.91GB / 80GB
已用
76.1 %
监控
端口配置
1小时
时间粒度:
10秒
关闭
显示图例
CPU利用率(%)
11:05 0.60
0.6
0.4
内存使用量(MB)
10:45 1787.00
2GB
1.5GB
1000MB
网络资源
您的网络资源已用
已用
0.07 %

访问方式

最近更新时间：2025-06-26 14:57:42

操作场景

购买 ChatGLM3-6B 应用实例后，我们有Gradio WebUI、JupyterLab、SSH 3 种方式访问该应用实例，以下将逐一介绍。

前提条件

已经购买了 ChatGLM3-6B 应用实例（参见 ChatGLM3 6B 的快速创建 ），能够登录腾讯云 高性能应用服务控制台 。

操作步骤

方法1：通过 Gradio WebUI 访问

1. 登录 高性能应用服务控制台 。
2. 在**算力管理**页中选择**算力连接** > **Gradio WebUI** 。

3. 弹出如下图所示的新的页面，说明通过 Gradio WebUI 的访问已建立，您可以在此和 ChatGLM3-6B 进行畅聊。

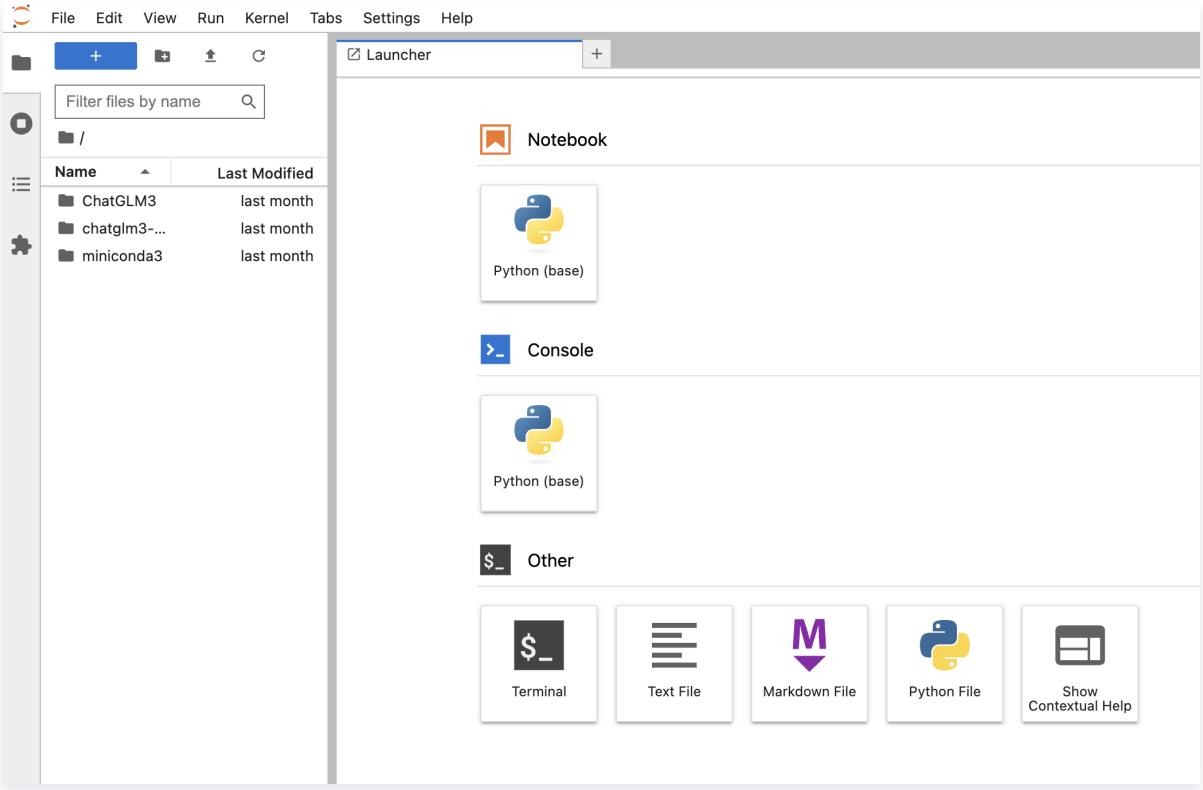
页面的右下角有三个参数配置，它们的含义如下：

- **Maximum length**：控制生成文本的长度。增加这个值会允许模型生成更长的文本，但可能会增加生成无关或重复内容的风险。减少这个值可以更快地得到结果，但可能会牺牲文本的完整性。
- **Top P**：影响生成文本的多样性和集中度。调高这个值会使得生成的文本更加多样化。调低这个值会使得生成的文本更加集中于最常见的选择，可能更符合预期但也可能显得比较单调。
- **Temperature**：控制生成文本的创造性和稳定性。调高这个值适合需要创意输出的场景，会增加生成文本的创造性和不可预测性。调低这个值适合需要精确和一致性结果的场景，会使得输出更加稳定和可预测。

方法2：通过 JupyterLab 访问

1. 在**算力管理**页中选择**算力连接** > **JupyterLab** 。

2. 弹出如下图所示的新的页面，说明通过 **JupyterLab** 的访问已建立，您可以利用 **ChatGLM3-6B** 的成熟环境，快速创建您自己的代码项目或应用服务。



方法3：通过 SSH 访问

1. 在控制台的上边栏的右侧，找到并点击 站内信 ，找到应用实例创建时推送的消息，单击消息的任意位置进入该条消息。

2. 在该条消息中，找到**默认用户名**，**登录密码**，**IP地址（公）**这三个配置信息，并根据这些信息，进行 SSH 登录。

手动中断与重启 Gradio WebUI 服务

最近更新时间：2024-11-22 14:57:33

操作场景

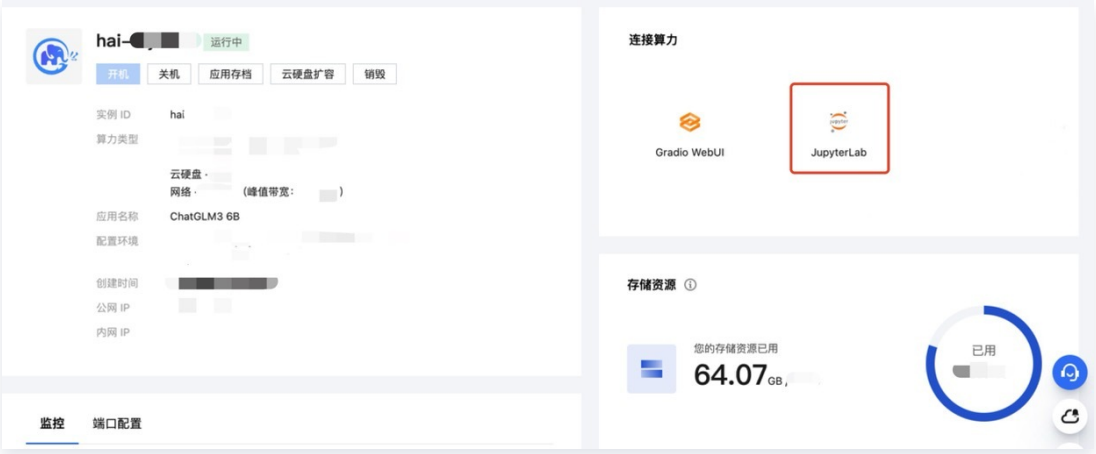
腾讯云高性能应用服务 HAI 是为开发者量身打造的澎湃算力平台。HAI 的 ChatGLM3-6B 应用，预装了支持 ChatGLM3-6B 模型运行的全部环境依赖。应用实例启动时，默认启动了 Gradio WebUI 服务，该服务对显存有一定的占用。
除了使用 HAI 官方的 Gradio WebUI 外，用户也可以手动中断 Gradio WebUI 服务对显存的占用，以利用 HAI 的预装环境依赖，部署自定义的代码项目。

中断 Gradio WebUI 服务

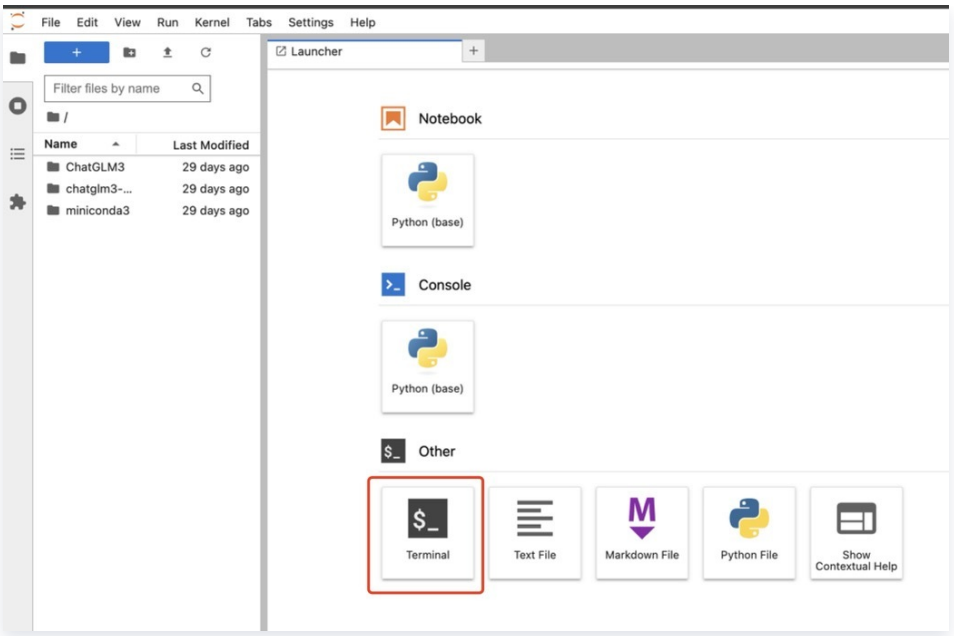
步骤1：打开终端

通过 JupyterLab 打开终端

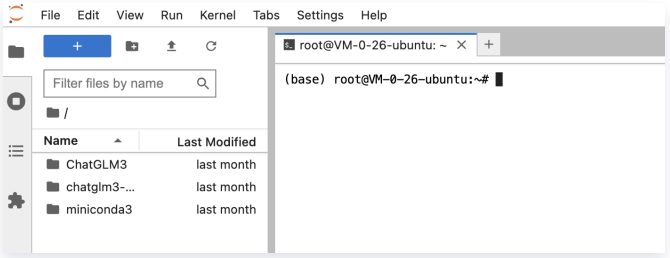
1. 在 腾讯云控制台 的页面，找到 ChatGLM3-6B 应用实例，进入实例管理页面，单击红框所示的 JupyterLab 连接算力。



2. 在弹出的 JupyterLab 页面中，选择 Terminal。



成功进入终端。



通过 SSH 客户端程序使用终端

或者，您可以在腾讯云的站内信中找 ChatGLM3-6B 应用实例对应的公网 IP、用户名和密码，通过 SSH 客户端程序进入终端。

步骤2：查找 Gradio WebUI 对应的进程

1. 在终端输入如下命令，查看活动进程：

```
ps aux
```

得到的活动进程信息展示如下：

```
(base) root@VM-0-2-ubuntu:~# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0    216     4 ?        Ss   06:28   0:00 /etc/.hai/dumb-init -- /usr/local/bin/application_init.sh
root           6  0.0  0.0   3976  2832 pts/0    Ss+  06:28   0:00 /bin/bash /usr/local/bin/application_init.sh
root           8  0.0  0.0  29368 24584 pts/0    S+   06:28   0:00 /etc/.hai/miniconda3/bin/python3 /etc/.hai/miniconda3/bin/supervisord -c /etc/su
root           9  0.0  0.0   4204  3072 pts/0    S    06:28   0:00 /bin/bash /usr/local/bin/launch_chatglm3_6b_webui.sh
root          10  0.0  0.0   4204  2996 pts/0    S    06:28   0:00 /bin/bash /usr/local/bin/launch_jupyter.sh
root          11  0.0  0.0  12184  6792 pts/0    S    06:28   0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
root          12  0.1  0.2 478872 77272 pts/0    Rl   06:28   0:01 /etc/.hai/miniconda3/bin/python3 /etc/.hai/miniconda3/bin/jupyter-lab --allow-ro
root          13  0.0  0.0   2756   516 pts/0    S    06:28   0:00 tee -a /var/log/jupyter_service.log
root          14  2.2  7.4 25491696 2391344 pts/0 Sl  06:28   0:26 python3 -u hai_web_demo.py --listen --port=6889
root          15  0.0  0.0   2756   524 pts/0    S    06:28   0:00 tee -a /var/log/chatglm3_gradio_service.log
root          58  1.3  0.0   4468  3540 pts/1    Ss+  06:47   0:00 /bin/bash -l
root          77 25.0  0.0   4468  3644 pts/2    Ss   06:47   0:00 /bin/bash -l
root          93  0.0  0.0   6124  3008 pts/2    R+   06:47   0:00 ps aux
```

2. 找到 COMMAND 列中，涉及 `hai_web_demo.py` 字样的进程，记录其进程 PID，本例中进程 PID 为 `14` 。

步骤3：终止 Gradio WebUI 对应的进程

在终端输入如下命令，终止 Gradio WebUI 对应的进程。请将命令中的 `xxx` 替换为上一步查找到的进程 PID。

```
kill -9 xxx
```

> **注意：**
> 终止错误的进程 PID 可能影响系统稳定性，请务必确认进程 PID 序号后再执行终止命令。

步骤4：查看显存占用情况

在终端输入如下命令，查看 GPU 使用情况。

```
nvidia-smi
```

得到的 GPU 使用情况展示如下，如果成功终止 Gradio WebUI 对应的进程，Memory-Usage 一项对应的显存占用会变得很小，本示例中仅为 2M。

```
(base) root@VM-0-2-ubuntu:~# nvidia-smi
Wed May 15 06:49:56 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| 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            On   | 00000000:00:09.0 Off |                    0 |
| N/A   43C    P0    26W /  70W |      2MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

如果没有终止 Gradio WebUI 对应的进程，Memory-Usage 一项对应的显存占用为下图所示，作为对比。

```
(base) root@VM-0-2-ubuntu:~# nvidia-smi
Wed May 15 06:49:43 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| 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            On   | 00000000:00:09.0 Off |                    0 |
| N/A   42C    P0    26W /  70W |  12463MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

重启 Gradio WebUI 服务

中断 Gradio WebUI 服务后，您将不能在 HAI 实例的算力管理页面使用 Gradio WebUI。遵循如下流程可以重启该服务。请确保重启服务前，该 HAI 实例拥有充足的空余显存容量。

步骤1：服务重启

1. 在终端输入如下命令：

```
export MODEL_PATH='/root/chatglm3-6b-model' && export
TOKENIZER_PATH='/root/chatglm3-6b-model' && cd
/root/ChatGLM3/basic_demo/ && (python3 -u hai_web_demo.py --listen --
port=6889 >> /var/log/chatglm3_gradio_service.log 2>&1 &) && tail -f
/var/log/chatglm3_gradio_service.log
```

2. 以上命令实现了模型部署、指定端口监听、将控制台输出重定向到指定 log 文档等步骤，最终实现了 Gradio WebUI 服务重启。执行后，如果终端输出类似如下的字样，说明服务重启成功。

```
(base) root@VM-0-2-ubuntu:~# export MODEL_PATH='/root/chatglm3-6b-model' && export TOKENIZER_PATH='/root/chatglm3-6b-model' && cd /root/ChatGLM3/ba
sic_demo/ && (python3 -u hai_web_demo.py --listen --port=6889 >> /var/log/chatglm3_gradio_service.log 2>&1 &) && tail -f /var/log/chatglm3_gradio_s
ervice.log
Loading checkpoint shards: 100%|██████████| 7/7 [01:23<00:00, 11.86s/it]
Setting eos_token is not supported, use the default one.
Setting pad_token is not supported, use the default one.
Setting unk_token is not supported, use the default one.
Running on local URL:  http://0.0.0.0:6889

To create a public link, set `share=True` in `launch()`.
IMPORTANT: You are using gradio version 4.26.0, however version 4.29.0 is available, please upgrade.
--------
```

步骤2：验证重启

通过终端命令验证

> **说明：**

> 上一步启动服务的操作可能占据终端的输入输出流。可以同时按下 `control` 键和 `C` 键，取得终端的控制权。

在终端输入如下命令，查看显存占用情况。

```
nvidia-smi
```

得到如下终端输出，显示显存占用量有 12G 以上，说明 Gradio WebUI 服务已经重启。

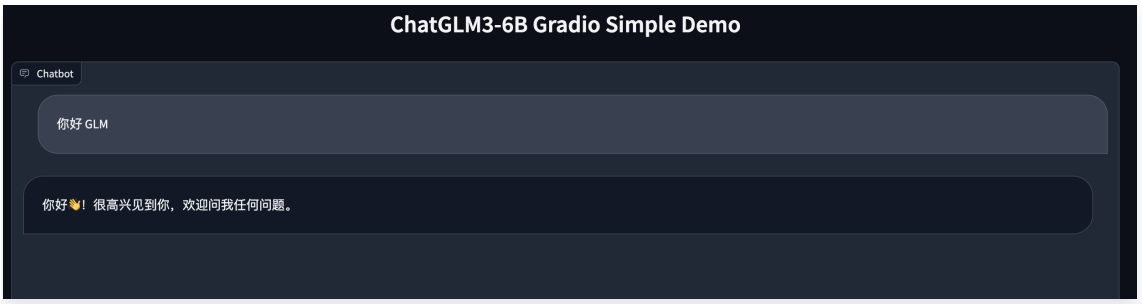
```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:00:09.0 Off |                    0 |
| N/A   39C    P0    26W /  70W |  12463MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

通过访问腾讯云控制台该实例的 Gradio WubUI 功能验证

进入腾讯云控制台，找到该 HAI 实例的管理页面，单击红框所示的 Gradio WebUI 。

JupyterLab 页面正常弹出，可以正常对话，证明 Gradio WebUI 服务重启成功。


ChatGLM3-6B Gradio Simple Demo
Chatbot
你好 GLM
你好👋！很高兴见到你，欢迎问我任何问题。

在 HAI 环境中部署自定义项目

最近更新时间：2024-07-18 11:54:51

操作场景

腾讯云高性能应用服务 HAI 是为开发者量身打造的澎湃算力平台。无需复杂配置，便可享受即开即用的 GPU 云服务体验。HAI 的 ChatGLM3-6B 应用，预装了支持 ChatGLM3-6B 模型运行的全部环境依赖。
如果您想部署自定义的代码项目，您可参考如下的部署实例，利用 腾讯云高性能应用服务 HAI 的预装环境，无需进行复杂环境配置，实现即开即用的灵活部署。

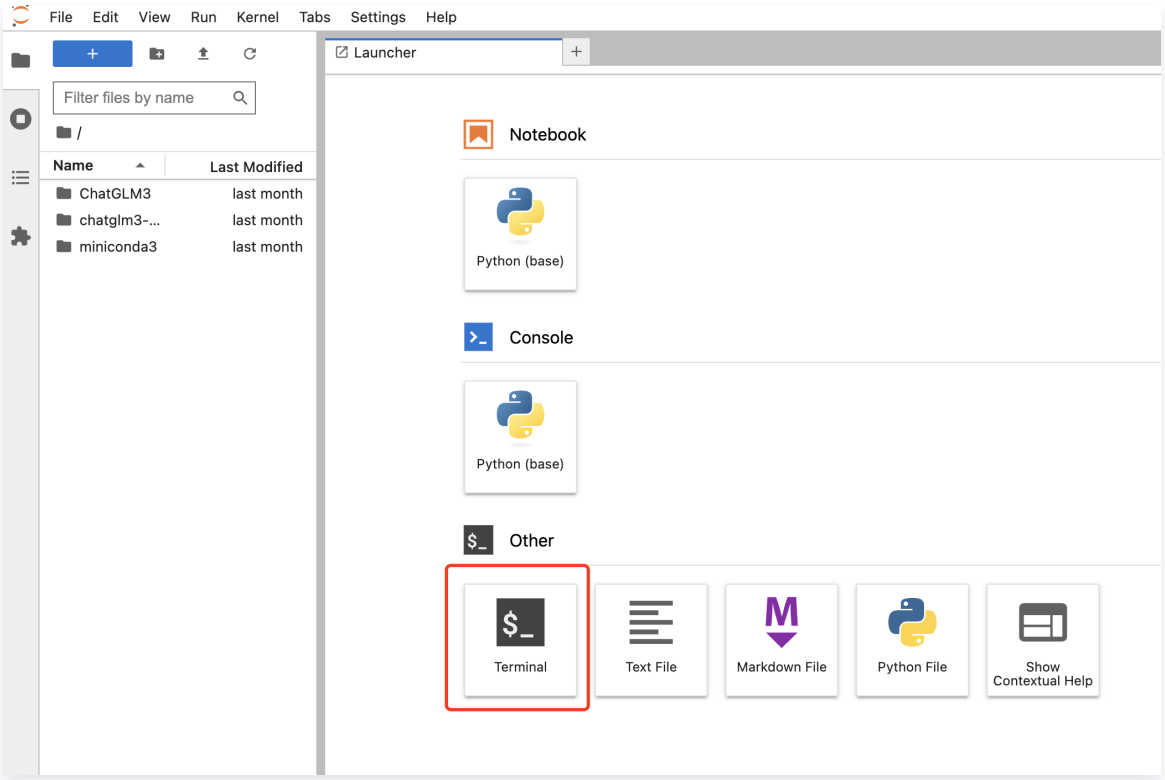
部署示例

本次将通过简单的示例代码，用 `transfomers` 库加载预装的 ChatGLM3-6B 模型并进行文本生成，体验将 ChatGLM3-6B 集成到您自己的代码工作流。

步骤1：释放显存

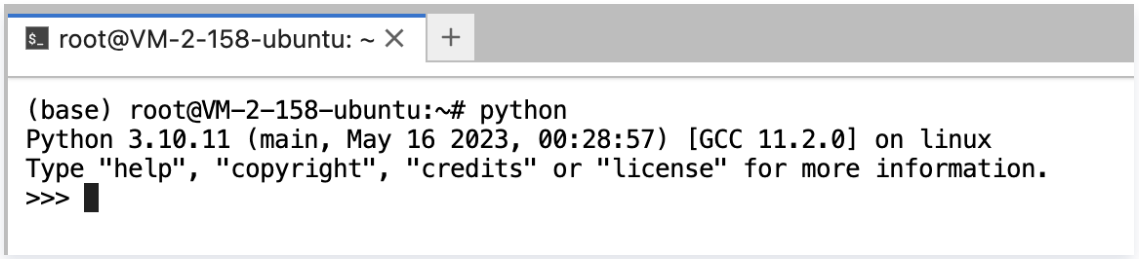
1. 登录 高性能应用服务控制台 。
2. 在**算力管理页**中选择**算力连接** > **JupyterLab** 。

3. 进入 **JupyterLab** ，打开**终端**，参见 手动中断与重启 Gradio WebUI 服务 ，中断 **Gradio WebUI** 以释放显存。



步骤2：部署实例代码

1. 在终端输入 `python` ，进入 python 命令行。



2. 从 `transformers` 库导入 `AutoModelForCausalLM` 和 `AutoTokenizer` 模块。

```
>>> from transformers import AutoModelForCausalLM, AutoTokenizer
```

3. 导入本地预装的 ChatGLM3-6B 模型和模型分词器，模型路径为 `/root/chatglm3-6b-model` 。

```
>>> model = AutoModelForCausalLM.from_pretrained("/root/chatglm3-6b-model", device_map="auto", trust_remote_code=True).to("cuda")
>>> tokenizer = AutoTokenizer.from_pretrained("/root/chatglm3-6b-model", padding_side="left", trust_remote_code=True)
```

4. 设置输入的文本，并获取文本生成的结果。

```
>>> model_inputs = tokenizer(["A list of colors: red, blue"],
return_tensors="pt").to("cuda")
>>> generated_ids = model.generate(**model_inputs)
>>> tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

得到的如下输出。

```
>>> tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
'[gMASK] sop A list of colors: red, blue, green, yellow, orange, purple, pink'
>>>
```

到此为止，您已经体验了利用大语言模型进行代码开发的一个简单完整流程！利用腾讯云高性能应用服务 HAI 的预装环境，实现即开即用，开发和部署灵活简单。

部署 Pytorch 2.0.0 快速创建

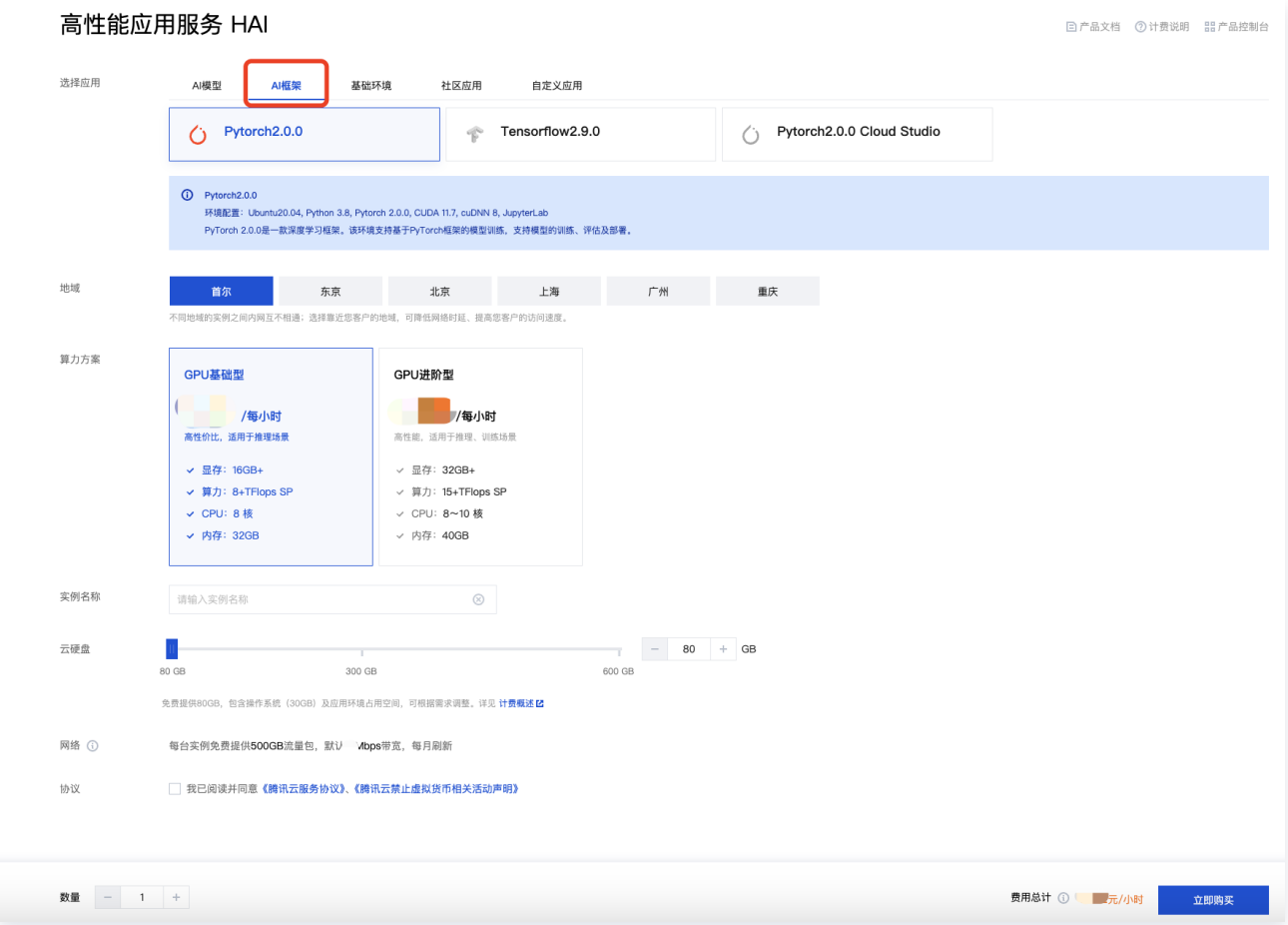
最近更新时间：2025-06-26 14:57:42

操作场景

本次介绍 腾讯云高性能应用服务 HAI Pytorch 2.0.0 应用实例的购买流程。

操作步骤

1. 登录 高性能应用服务控制台 。
2. 单击**新建**，进入 高性能应用服务购买页面 。



- **选择应用：**选择 AI 框架，应用选择 Pytorch 2.0.0。
- **地域：**建议选择靠近自己实际地理位置的地域，降低网络延迟、提高您的访问速度。
- **算力方案：**本次算力方案选择进阶型，生成图片效率更高。
- **实例名称：**自定义实例名称，若不填则默认使用实例 ID 替代。
- **硬盘：**默认提供 80GB 免费空间，可根据实际使用需求进行调整。
- **网络：**每台实例每月免费提供 500GB 流量包，默认10Mbps 带宽，每月刷新。

- **购买数量**：默认1台。

3. 单击**立即购买**。
4. 核对配置信息后，单击**提交订单**，并根据页面提示完成支付。
5. 等待创建完成。单击实例**任意位置**并进入该实例的详情页。

6. 您可以在此页面查看 Pytorch 2.0.0 应用实例详细的配置信息，到此为止，说明您的 Pytorch 2.0.0 应用实例购买成功。

访问方式

最近更新时间：2024-11-22 14:57:33

操作场景

购买 Pytorch 2.0.0 应用实例后，我们有 JupyterLab 和 SSH 2 种方式访问该应用实例，以下将逐一介绍。

前提条件

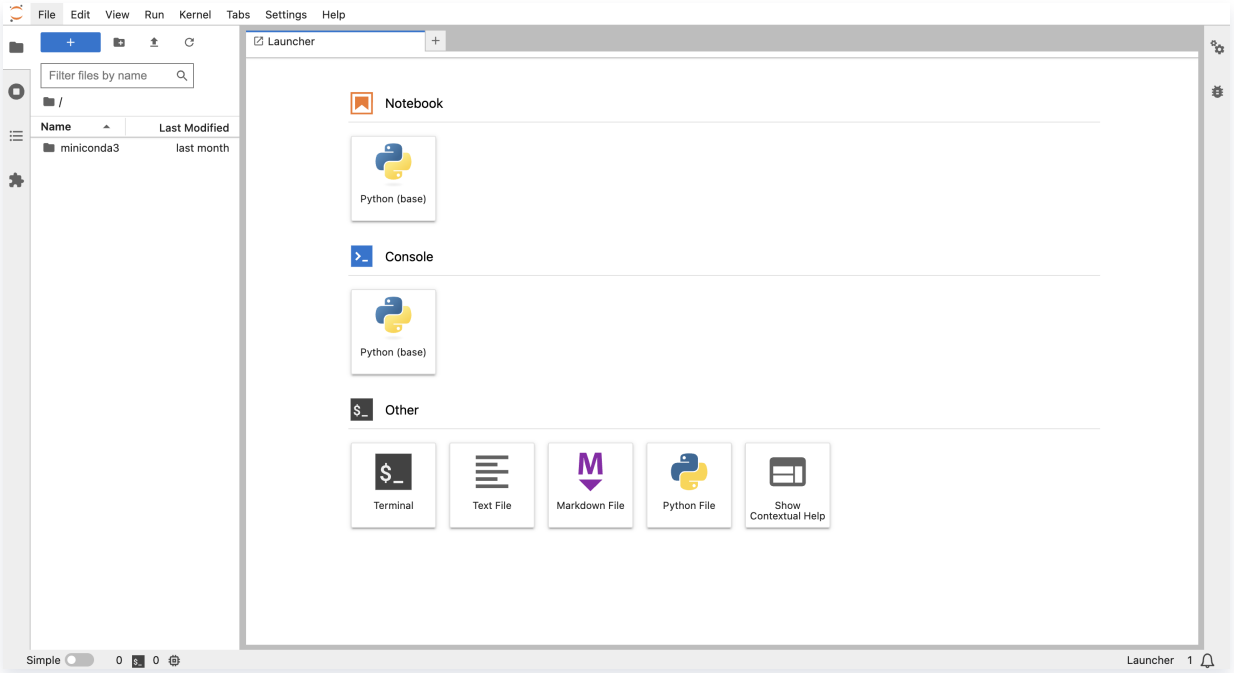
已经购买了 Pytorch 2.0.0 应用实例（参见 Pytorch 2.0.0 的快速创建），能够登录腾讯云高性能应用服务控制台。

操作步骤

方法1：通过 JupyterLab 访问

1. 登录 高性能应用服务控制台。
2. 在**算力管理**页中选择**算力连接** > **JupyterLab**。

3. 弹出如下图所示的新的页面，说明通过 **JupyterLab** 的访问已建立，您可以在这里快速创建您自己的代码项目或应用服务。



方法2：通过 SSH 访问

1. 在控制台的上边栏的右侧，找到并点击 站内信 ，找到应用实例创建时推送的消息，单击消息的任意位置进入该条消息。

2. 在该条消息中，找到**默认用户名**，**登录密码**，**IP地址（公）**这三个配置信息，并根据这些信息，进行 SSH 登录。

管理系统 CUDA

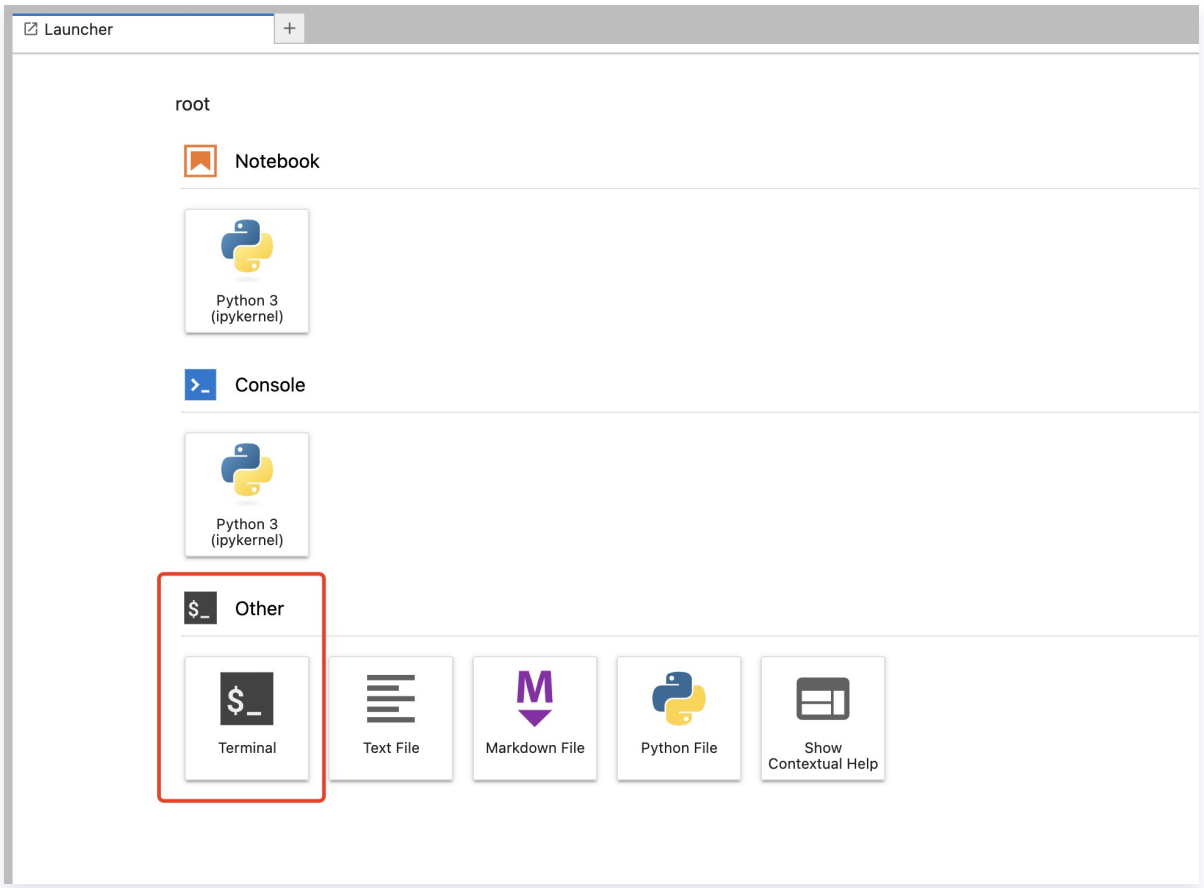
最近更新时间：2024-11-22 14:57:33

CUDA 是 NVIDIA 开发的一种编程技术，它允许开发者使用 NVIDIA 的图形处理单元（GPU）进行高效的并行计算。CUDA 具有很多发行版本，用户在使用时 Pytorch 2.0.0 应用实例时，可能有对系统 CUDA 版本进行查看和修改的需求。本次，我们将介绍如何在 Pytorch 2.0.0 应用实例中查看和修改系统 CUDA。

查看系统支持的最高 CUDA 版本

1. 登录 高性能应用服务 HAI 控制台。
2. 选择需要连接的算力，然后单击**算力连接**。在下拉菜单中，单击 **JupyterLab**。

3. 进入 JupyterLab 后，选择 **Other** > **Terminal**，进入终端界面。



4. 在终端输入如下命令，查看系统 GPU 驱动版本。系统 GPU 驱动版本和系统支持的最高 CUDA 版本存在对应关系。

```
nvidia-smi
```

得到的终端输出如下，在这个例子中，Driver Version 为 `525.105.17`，CUDA Version 为 `12.0`。这里显示的 CUDA Version 指的是系统支持的最高 CUDA 版本，兼容 CUDA <= 12.0 的任何版本。

```
(base) root@VM-0-9-ubuntu:~# nvidia-smi
Wed May 15 07:03:12 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| 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            On   | 00000000:00:09.0 Off |                    0 |
| N/A   30C    P8     9W /  70W |      2MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

查看系统 CUDA 版本

在终端输入如下命令，查看系统实际安装的 CUDA 的版本。

```
nvcc --version
```

得到的终端输出如下，我们可以读出，这个例子中，系统实际安装的 CUDA 的版本为 `11.7` 。

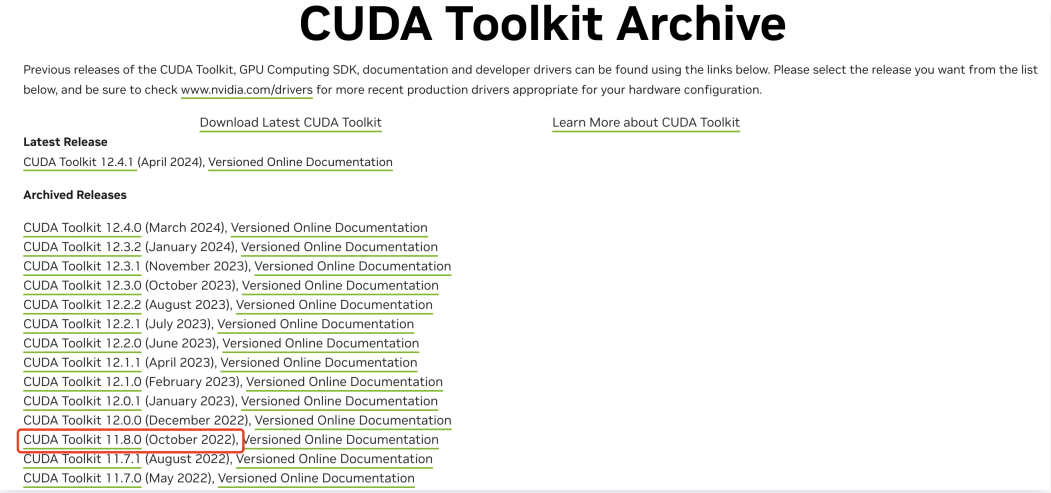
```
(base) root@VM-2-42-ubuntu:~# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Jun__8_16:49:14_PDT_2022
Cuda compilation tools, release 11.7, V11.7.99
Build cuda_11.7.r11.7/compiler.31442593_0
```

升级系统 CUDA 版本

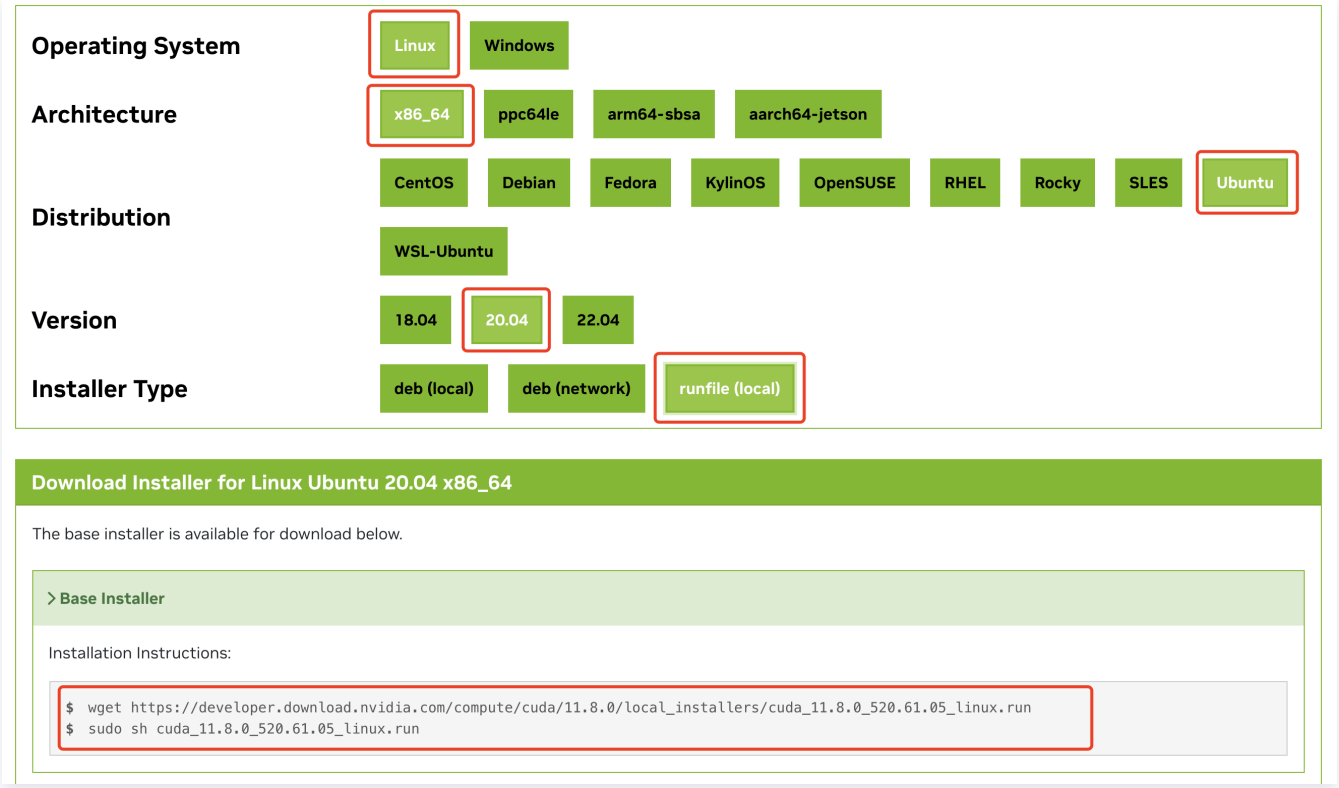
不同项目需要的系统 CUDA 版本依赖是不同的。为了适配不同项目的需求，我们可能需要升级系统 CUDA 到更新的版本。下面讲展示如何将系统 CUDA 版本升级到 `11.8` 。

> **注意：**
> 升级后的系统 CUDA 版本，不能超过系统支持的最高 CUDA 版本。例如，在本例中，用户可以对 `11.7` 版本的 CUDA 进行升级，但升级后的 CUDA 版本不能超过 `12.0` 。

1. 在英伟达的 CUDA 发行版本 页面，找到目标 CUDA 版本，这里我们选择 `11.8` 的版本。



2. 选择合适的系统信息，之后会生成安装命令。
   - **Operating System：** 选择 Linux。
   - **Architecture：** 选择 x86_64。
   - **Distribution：** 应用搭载 Ubuntu 20.04，因此选择 Ubuntu。
   - **Version：** 选择 20.04。
   - **Installer Type：** 选择 runfile (local)。



3. 将上一步得到的安装命令复制到终端执行。这一步，`wget` 命令拉取安装文件，之后 `sh` 命令安装下载文件。注意删去 `sudo`，因为HAI实例里用户默认已经是 `root` 用户了。

```
wget
https://developer.download.nvidia.com/compute/cuda/11.8.0/local_instal
lers/cuda_11.8.0_520.61.05_linux.run
sh cuda_11.8.0_520.61.05_linux.run
```

4. `sh` 命令执行后，稍等片刻，终端显示以下内容。输入 `accept`，回车键确认。

```
End User License Agreement
--------------------------

NVIDIA Software License Agreement and CUDA Supplement to
Software License Agreement. Last updated: October 8, 2021

The CUDA Toolkit End User License Agreement applies to the
NVIDIA CUDA Toolkit, the NVIDIA CUDA Samples, the NVIDIA
Display Driver, NVIDIA Nsight tools (Visual Studio Edition),
and the associated documentation on CUDA APIs, programming
model and development tools. If you do not agree with the
terms and conditions of the license agreement, then do not
download or use the software.

Last updated: October 8, 2021.


Preface
-------

Do you accept the above EULA? (accept/decline/quit):
```

5. 弹出以下内容，请注意，不要直接安装，**取消**默认选中的 Driver 复选框，然后再选择 Install。使用**上下箭头键**移动光标，回车键进行**勾选或取消**勾选。

```
CUDA Installer
- [ ] Driver
     [ ] 520.61.05
+ [X] CUDA Toolkit 11.8
  [X] CUDA Demo Suite 11.8
  [X] CUDA Documentation 11.8
- [ ] Kernel Objects
     [ ] nvidia-fs
  Options
  Install

Up/Down: Move | Left/Right: Expand | 'Enter': Select | 'A': Advanced options
```

6. 弹出以下内容，询问是否替代原有的 CUDA 安装。选择 `Yes` 。

```
A symlink already exists at /usr/local/cuda. Update to this installation?
Yes
No

Up/Down: Move | 'Enter': Select
```

7. 弹出以下内容，说明安装完成。

```
(base) root@VM-0-9-ubuntu:~# sh cuda_11.8.0_520.61.05_linux.run
===========
= Summary =
===========

Driver:   Not Selected
Toolkit:  Installed in /usr/local/cuda-11.8/

Please make sure that
 -   PATH includes /usr/local/cuda-11.8/bin
 -   LD_LIBRARY_PATH includes /usr/local/cuda-11.8/lib64, or, add /usr/local/cuda-11.8/lib64 to /etc/ld.so.conf and run ldconfig as root

To uninstall the CUDA Toolkit, run cuda-uninstaller in /usr/local/cuda-11.8/bin
***WARNING: Incomplete installation! This installation did not install the CUDA Driver. A driver of version at least 520.00 is required for CUDA 11
.8 functionality to work.
To install the driver using this installer, run the following command, replacing <CudaInstaller> with the name of this run file:
    sudo <CudaInstaller>.run --silent --driver

Logfile is /var/log/cuda-installer.log
```

8. 在终端输入 `nvcc --version` ，查看系统当前安装的 CUDA 的版本，得到现安装版本是11.8，这验证了我们的安装是成功的。

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

管理系统 cuDNN

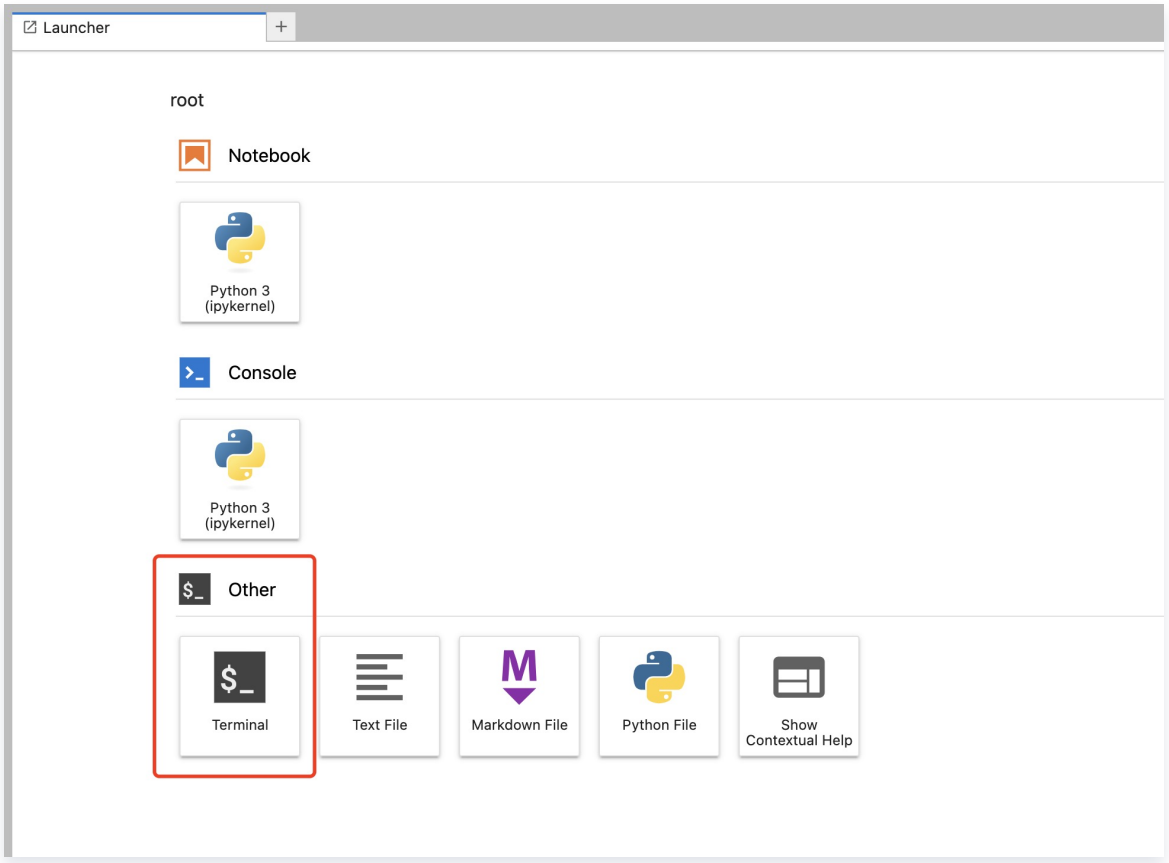
最近更新时间：2024-11-22 14:57:33

cuDNN 是 NVIDIA 提供的一个深度神经网络库，它能够在 NVIDIA 的 GPU 上高效地运行深度学习算法。cuDNN 也具有很多发行版本，用户在使用时 Pytorch 2.0.0 应用实例时，可能有对系统 cuDNN 版本进行查看和修改的需求。本次，我们将介绍如何在 Pytorch 2.0.0 应用实例中查看和修改系统 cuDNN。

查看系统现有 cuDNN

1. 登录 高性能应用服务 HAI 控制台。
2. 选择需要连接的算力，然后单击**算力连接**。在下拉菜单中，单击 **JupyterLab**。

3. 进入 JupyterLab 后，选择 **Other** > **Terminal**，进入终端界面。



4. 在终端输入如下命令，查看系统实际安装的 cuDNN 的版本。

```
dpkg -l | grep cudnn
```

得到的终端输出如下，我们可以读出，这个例子中，系统当前安装的 cuDNN 的版本为 `8.5.0` 。

```
(base) root@VM-0-9-ubuntu:~# dpkg -l | grep cudnn
hi  libcudnn8                    8.5.0.96-1+cuda11.7          amd64        cuDNN runtime libraries
ii  libcudnn8-dev                8.5.0.96-1+cuda11.7          amd64        cuDNN development libraries and headers
```

查找可安装的 cuDNN

在终端输入如下命令，查找可安装的 cuDNN 版本。

```
apt-cache policy libcudnn8
```

得到以下输出，在其中找到和系统 CUDA 版本适配的 cuDNN 版本。假设系统 CUDA 已经更新至 `11.8` 版本，我们可以选择版本 `8.9.7` 为的 cuDNN。

```
(base) root@VM-0-9-ubuntu:~# apt-cache policy libcudnn8
libcudnn8:
  Installed: 8.5.0.96-1+cuda11.7
  Candidate: 8.9.7.29-1+cuda12.2
  Version table:
     8.9.7.29-1+cuda12.2 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.7.29-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.6.50-1+cuda12.2 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.6.50-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.5.30-1+cuda12.2 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.5.30-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.5.29-1+cuda12.2 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.5.29-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.4.25-1+cuda12.2 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.4.25-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.3.28-1+cuda12.1 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.3.28-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.2.26-1+cuda12.1 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.2.26-1+cuda11.8 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
     8.9.1.23-1+cuda12.1 500
        500 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  Packages
```

安装指定版本的 cuDNN

1. 根据上一步查找得到的 cuDNN 版本，在命令行中输入如下命令。

```
apt-get update
apt-get install libcudnn8=8.9.7.29-1+cuda11.8
```

2. 安装中途，如果遇到询问，输入 y 即可。

```
(base) root@VM-0-9-ubuntu:~# apt-get install libcudnn8=8.9.7.29-1+cuda11.8
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  libcudnn8-dev
The following held packages will be changed:
  libcudnn8
The following packages will be upgraded:
  libcudnn8
1 upgraded, 0 newly installed, 1 to remove and 73 not upgraded.
Need to get 441 MB of archives.
After this operation, 850 MB disk space will be freed.
Do you want to continue? [Y/n] y
```

3. 完成安装后，再次在终端输入 `dpkg -l | grep cudnn` ，返回如下结果，可以看到系统 cuDNN 的版本已经更新为 `8.9.7` 。

```
(base) root@VM-0-9-ubuntu:~# dpkg -l | grep cudnn
ii  libcudnn8                      8.9.7.29-1+cuda11.8              amd64        cuDNN runtime libraries
```

管理 Python 虚拟环境依赖

最近更新时间：2025-06-26 14:57:42

应用实例内置的 Python 版本是 `3.8`，内置的 Python 环境管理工具是 Miniconda。您可以使用 `conda` 命令管理 Python 环境。

创建新的虚拟环境

您可以创建新的虚拟环境，并在新的虚拟环境中指定 Python 版本。

```
# -n new_env：指定新环境的名称，指定为new-env
conda create -n new_env python==3.9

# 查看所有虚拟环境
conda env list

# 切换虚拟环境
conda activate new_env

# 退出虚拟环境
conda deactivate
```

安装 Python 依赖

Python 依赖可以通过 `conda` 或 `pip` 命令安装，已经安装的 Python 依赖的版本等信息可以通过 `conda list` 或 `pip list` 查看。

```
# 通过conda安装python依赖示例
conda install numpy

# 通过pip安装python依赖示例
pip install numpy

# 查看虚拟环境中的依赖
conda list
pip list

# 安装其他版本的pytorch，在https://pytorch.org/get-started/previous-versions/查找相关的安装指令
conda install pytorch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2
pytorch-cuda=11.8 -c pytorch -c nvidia
```

配置 conda 环境级别的 CUDA 和 cuDNN

如果在同一个机器里部署的不同工程对 CUDA 和 cuDNN 的版本要求不同，可以在单个虚拟环境里，通过 `conda` 命令部署 CUDA 和 cuDNN。

- 如果在 conda 环境和系统中都安装了CUDA 和 cuDNN，一般默认优先使用 conda 环境里的 CUDA 和 cuDNN。
- 涉及编译的，只能使用系统 CUDA，conda 环境级别的 CUDA 或 cuDNN 在编译时会报错。

```
conda install cudatoolkit==11.2
conda install cudnn==8.1
```