模型自动适配与调优工具

AscendBridge 2.0 操作指导书

目录

1 前期准备工作 3

1.1 Agent 后端模型准备 3

1.1.1 智谱模型 3

1.1.2 MiniMax 模型 5

1.2 设备环境检查 6

1.2.1 环境登录 6

1.2.2 硬件约束 6

1.2.3 固件与驱动 7

1.2.4 docker 服务准备 7

1.3 创建个人工作目录 7

1.4 准备 Claude Code 安装配置参考文件 8

1.5 模型权重提前下载 8

1.5.1 从 HuggingFace 社区下载 8

1.5.2 从魔搭社区下载 10

1.6 创建容器 12

1.6.1 加载容器镜像 12

1.6.2 启动容器 14

2 AI 智能体安装与配置 16

2.1 自动化安装配置 16

2.1.1 Cursor 账号注册 16

2.1.2 在容器中安装 Cursor Agent 18

2.1.3 安装配置 Claude Code 20

2.2 手动安装配置 22

3 AscendBridge 2.0 工具使用 26

3.1 工具简介 26

3.2 在容器中下载工具包 28

3.3 使用方式 28

3.3.1 环境准备 28

3.3.2 典型使用方式一：Claude 自动批量编排 29

3.3.3 典型使用方式二：Claude 单模型连贯适配 30

3.3.4 典型使用方式三：Codex/Cursor 单模型连贯适配 34

4 已适配模型开源合集仓使用 37

4.1 仓库简介 37
4.2 下载开源仓 37
4.3 使用方式 38
4.3.1 独立使用 38
4.3.2 配合框架仓使用 38

1 前期准备工作

1.1 Agent 后端模型准备

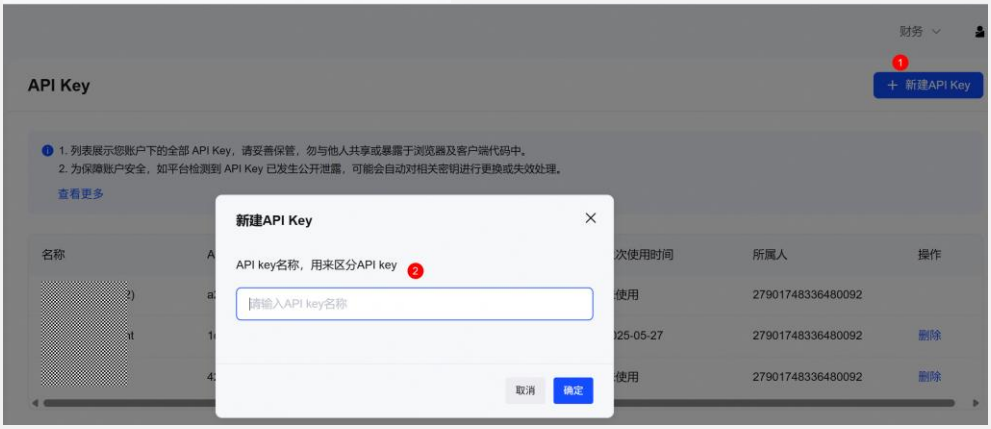
本指导书的工具使用的 Claude Code 需要配置的后端模型可自行选择，本章节主要介绍智谱和 MiniMax 两类模型的官网注册和套餐订阅方式。

1.1.1 智谱模型

登录智谱官网：https://www.bigmodel.cn/ ，点击右上角 【登录/注册】按钮

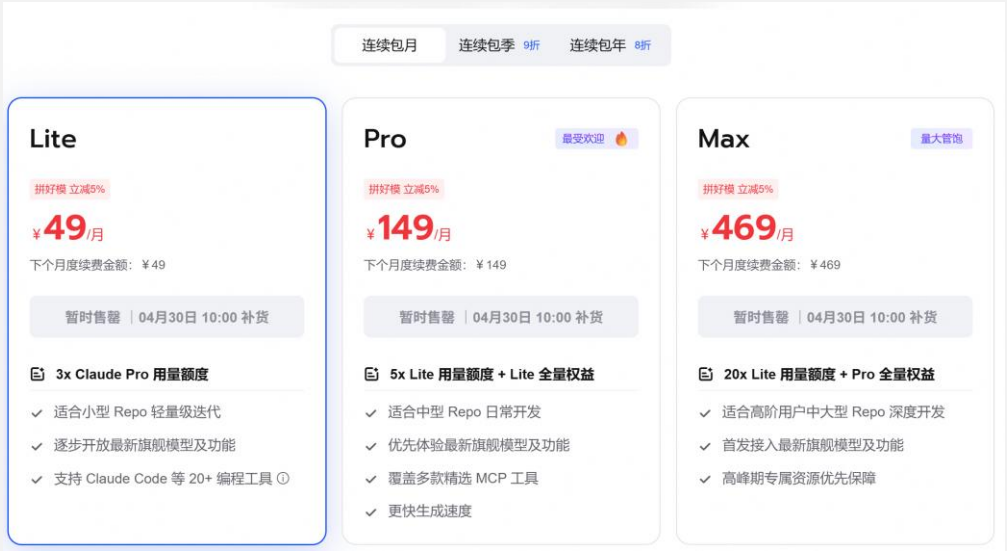
首次登录，可用手机号注册账号

登录账号后，从右上角进入【控制台】，之后点击左侧的【API Key】，新建一个API Key



API Key创建好之后，复制备用

在 Coding Plan 中选择合适的套餐进行订阅



注：Lite、Pro和Max套餐的说明可查阅https://docs.bigmodel.cn/cn/coding-plan/overview。

1.1.2 MiniMax 模型

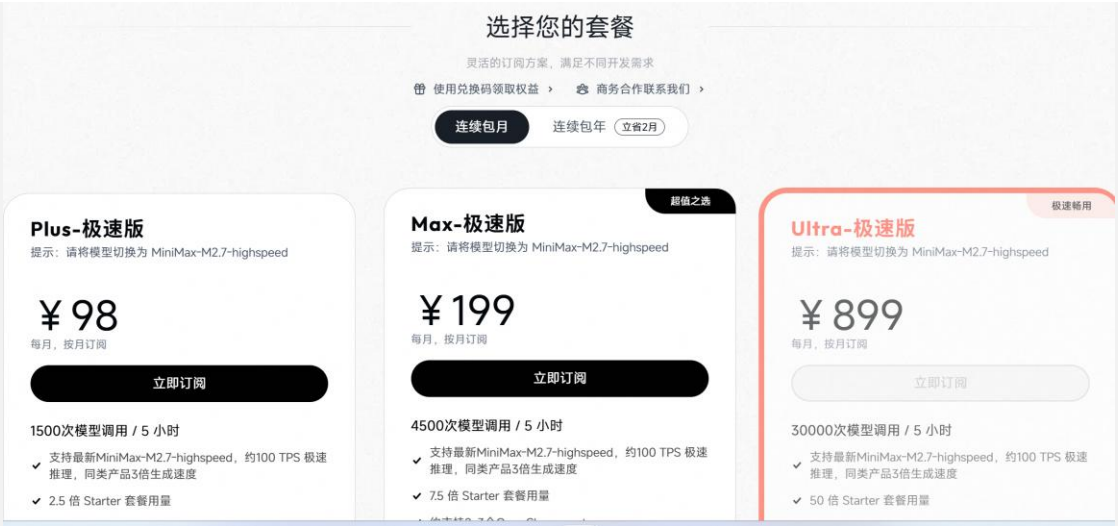
登录MiniMax官网：https://www.minimaxi.com/ ，右上角 【登录】按钮，选择【API开放平台】

首次登录，可用手机号注册账号

登录后点击左侧【接口密钥】，即可【创建新的API Key】

API Key创建之后复制备用

在 Token Plan 中选择合适的套餐订阅



注：套餐说明可参考https://platform.minimaxi.com/docs/token-plan/intro。

1.2 设备环境检查

1.2.1 环境登录

使用 Xshell / MobaXterm 软件登录昇腾服务器。

1.2.2 硬件约束

 # 内存约束：推荐大于500GB

 # 服务器型号：建议使用Atlas 800T A3/A2

执行dmidecode -s system-product-name命令可查询

```
[root@atlas-27 ~]# dmidecode -s system-product-name
Atlas 800T A3
```

 # 规格：单机8卡（单卡64GB）

执行npu-smi info命令可查询卡数、内存等相关信息

```
+------------------------------------------------------------------------------------------------+
| npu-smi 25.2.3                 Version: 25.2.3                                                 |
+---------------------------+---------------+----------------------------------------------------+
| NPU   Name                | Health        | Power(W)    Temp(C)           Hugepages-Usage(page)|
| Chip  Phy-ID              | Bus-Id        | AICore(%)   Memory-Usage(MB)  HBM-Usage(MB)        |
+===========================+===============+====================================================+
| 0     Ascend910           | OK            | 163.2       40                0    / 0           |
| 0     0                   | 0000:9D:00.0  | 0           0    / 0          3124 / 65536         |
+---------------------------+---------------+----------------------------------------------------+
| 0     Ascend910           | OK            | -           40                0    / 0           |
| 1     1                   | 0000:9F:00.0  | 0           0    / 0          2883 / 65536         |
+---------------------------+---------------+----------------------------------------------------+
```

 # 网络：昇腾物理机需打通互联网

1.2.3 固件与驱动

建议版本：25.2.3
注：此处的建议版本为写指导书时使用的环境上的固件版本，亦可选用其他版本。

# 检查方法：
echo '======Driver======' && head -1 /usr/local/Ascend/driver/version.info;
结果示例

```
[root@node-30-6 ~]# echo '======Driver======' && head -1 /usr/local/Ascend/driver/version.info;
======Driver======
Version=25.2.3
```

如果命令返回“No such file or directory”，说明驱动未安装，请参考下方链接进行安装：
https://support.huawei.com/enterprise/zh/doc/EDOC1100493505/426cffd9?idPath=23710424|251366513|254884019|261408772|252764743

1.2.4 docker 服务准备

检查docker服务是否已安装
执行docker --version命令，如果正常返回版本号信息，说明docker服务已安装

```
[root@atlas-27 ~]# docker --version
Docker version 18.09.0, build aa1eee8
```

如果命令返回“docker: command not found”说明当前环境没有docker服务，请按照下方指导安装docker环境

安装docker
参考：Ascend/ascend-deployer开源仓的安装指导
下载代码库，进入ascend_deployer目录，执行系统环境初始化（安装sys_pkg，docker）：
bash install.sh --install=sys_pkg
（执行--install命令时，请勿重复安装sys_pkg，安装成功即可查看docker版本号）

1.3 创建个人工作目录

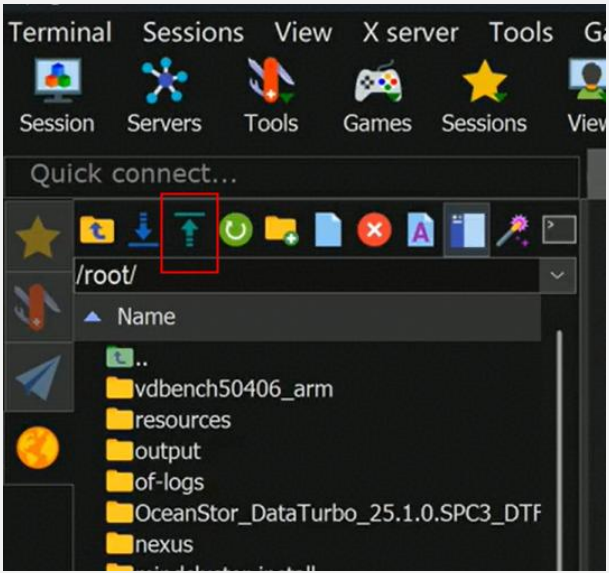
用root用户登录服务器，执行如下命令创建个人工作目录
mkdir <work_dir>
注：
<work_dir>是个人工作目录，可自定义，用于存放模型权重文件，以及后续涉及的一些脚本

1.4 准备 Claude Code 安装配置参考文件

先将下面的md文件保存到本地，然后上传到昇腾设备的<work_dir>个人工作目录下，为后续安装和配置Claude Code做准备

1：如果使用Xshell登录的环境，可点击上方的Xftp软件上传文件（第一次使用会提示先安装）

2：如果用MobaXterm登录的环境，上传文件图标如下图所示

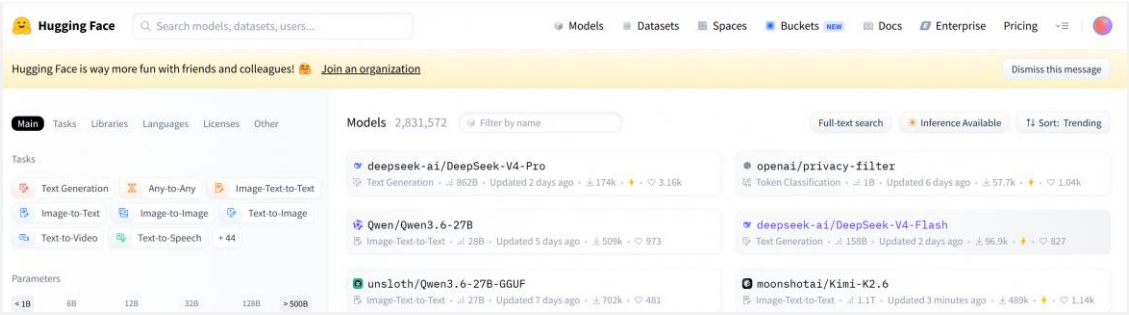


1.5 模型权重提前下载

本章节介绍两种模型权重的下载方式，根据实际情况选择其中一种即可。

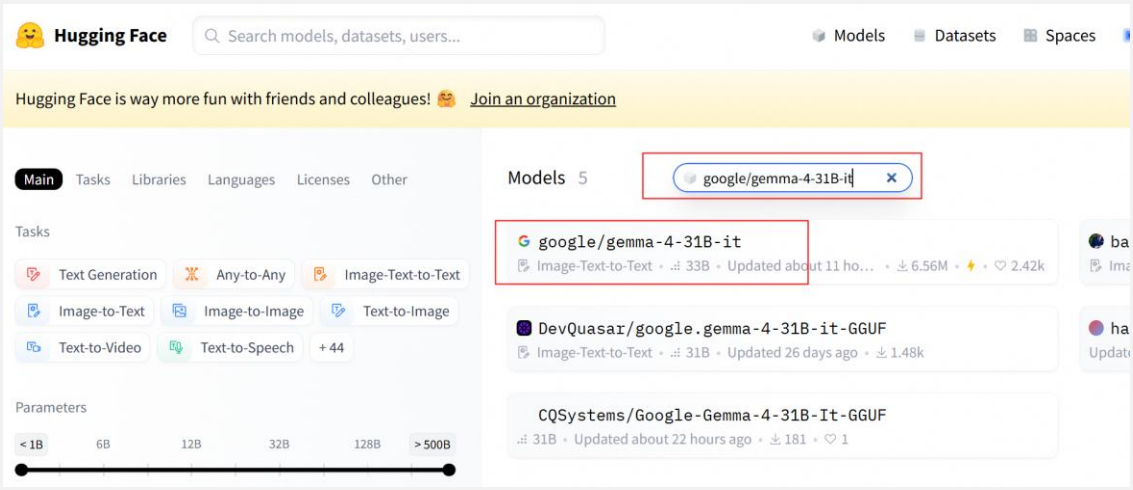
1.5.1 从 HuggingFace 社区下载

登录 HuggingFace 官网，网站如下：

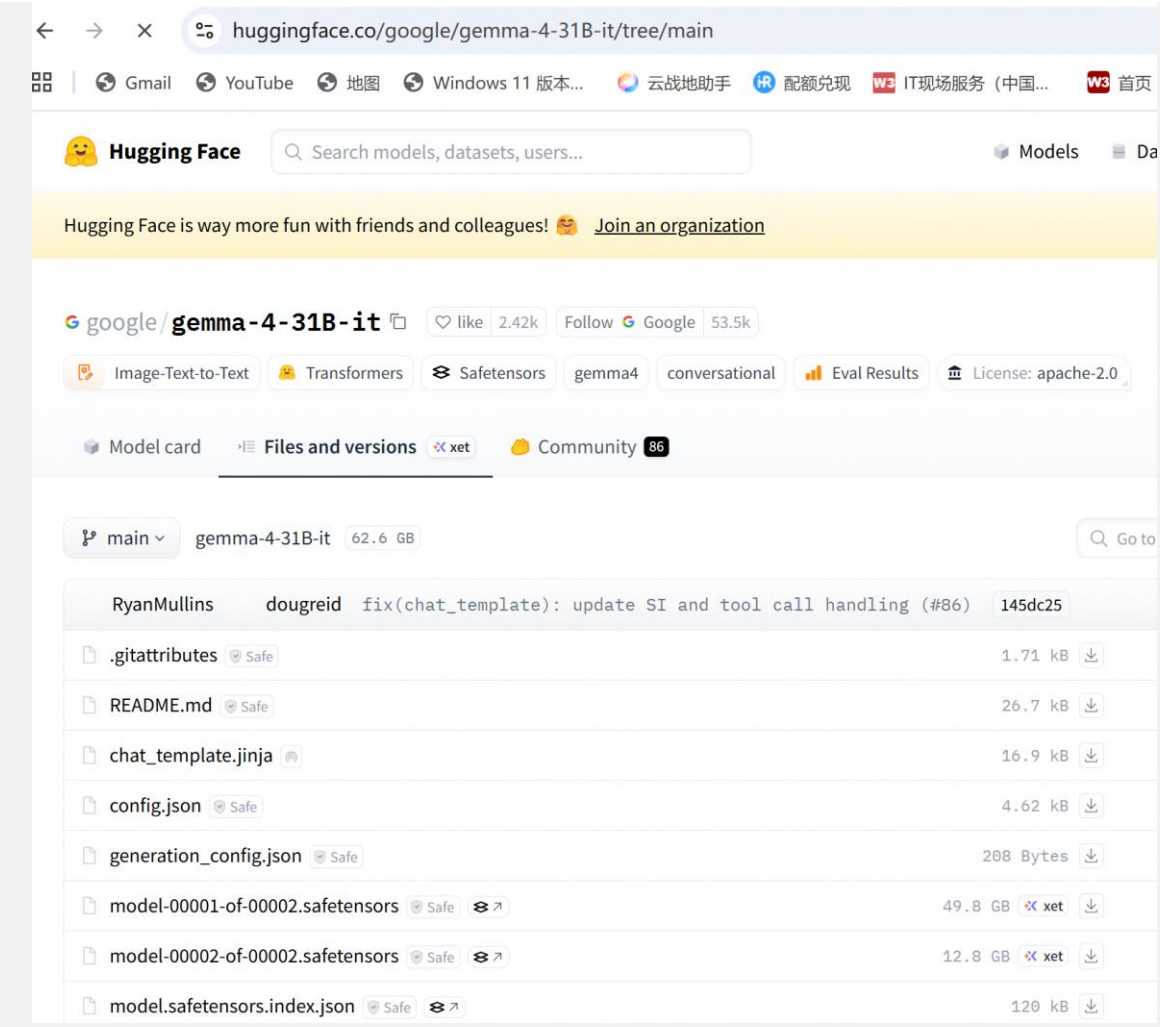


如果是第一次访问HuggingFace官网，请按照提示注册账号并登录。

搜索所需模型权重，如google/gemma-4-31B-it



权重如下：



用root用户登录服务器，执行如下命令，下载模型文件

cd <work_dir>

yum install git-lfs

git lfs install

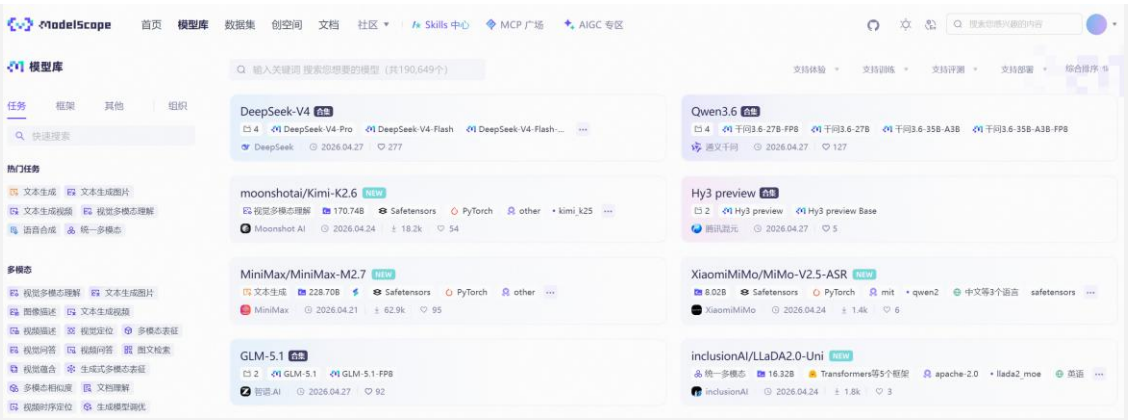
git clone https://huggingface.co/google/gemma-4-31B-it

注：

1. <work_dir> 是1.3小节中创建的个人工作目录
2. 命令中的模型名称可按实际需要进行替换，此处仅为示例
3. 如果用命令下载有问题，也可直接在模型页面点击各文件对应的【下载】图标，然后上传到服务器的<work_dir>目录下

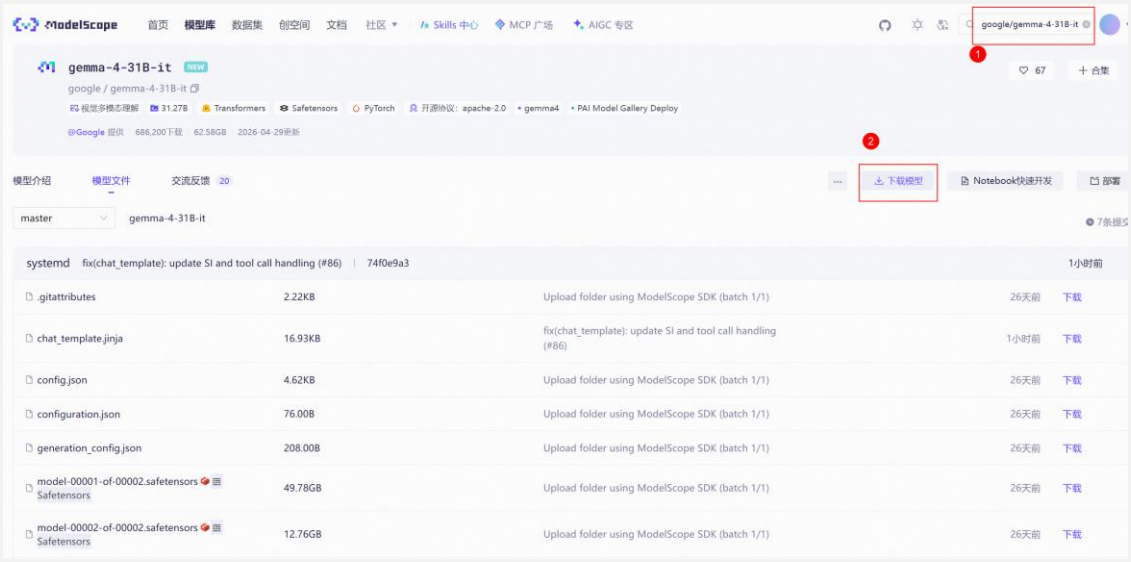
1.5.2 从魔搭社区下载

登录 ModelScope 官网，网站如下：



如果是第一次访问ModelScope官网，请按照提示注册账号并登录。

在“模型库”下搜索模型名称，如google/gemma-4-31B-it



点击“下载模型”，会弹出下载指导

用root账户登录服务器，按照下载指导，执行如下命令下载模型的所有文件
pip install modelscope
cd <work_dir>
mkdir gemma-4-31B-it
cd gemma-4-31B-it
modelscope download --model google/gemma-4-31B-it --local_dir ./

注：

1. <work_dir> 是1.3小节中创建的个人工作目录
2. 命令中的模型名称可按实际需要进行替换，此处仅为示例
3. download命令务必添加--local_dir参数，将权重文件下载到创建的模型名称目录下

1.6 创建容器

1.6.1 加载容器镜像

容器镜像建议采用包含有推理环境相关安装包的vllm-ascend镜像，可通过Docker仓库直接拉取，官方镜像托管在Quay.io平台。

下面以**v0.17.0rc1-a3**版本的镜像为例，演示从Quay.io平台加载镜像的流程。
注：也可使用其他容器镜像，镜像中需要包含CANN、Pytorch等基础软件包。

执行以下命令查看容器镜像是否已加载，若已存在该镜像，则本小节操作到此结束
docker images | grep v0.17.0rc1-a3
若镜像未加载，则继续进行下面的步骤

打开Quay.io平台：https://quay.io/repository/ascend/vllm-ascend
找到左侧导航栏的“Tags”标签，在搜索框输入“v0.17.0rc1-a3”搜索到对应的镜像，点击右侧的“下载”按钮

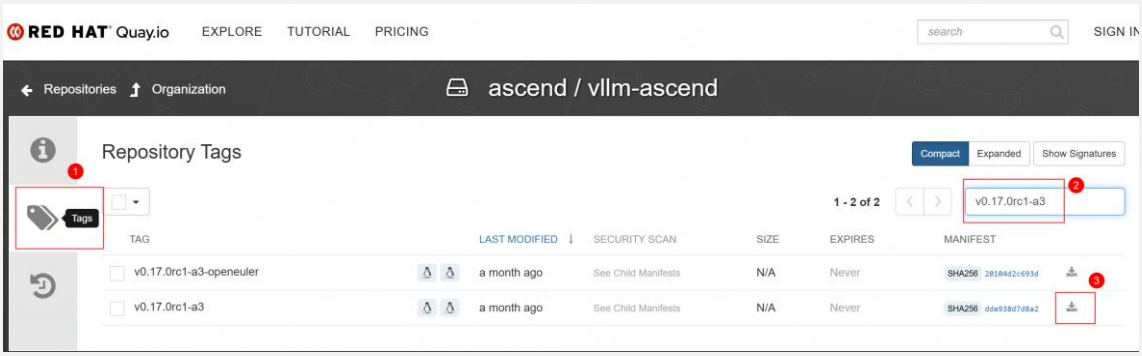


Image Format选择“Docker Pull（by tag)”，会出现一条docker pull命令

用root账户登录服务器，执行docker pull命令即可加载对应镜像

```
[root@atlas-27 ~]# docker pull quay.io/ascend/vllm-ascend:v0.17.0rc1-a3
v0.17.0rc1-a3: Pulling from ascend/vllm-ascend
c36472b34583: Already exists
9ff1473d4ee0: Already exists
3f80d0667f7a: Already exists
0a48a7b0869e: Already exists
307a6071cd9e: Already exists
63fe0193e142: Already exists
e0ec65d22c2c: Pull complete
1eb3048993c7: Pull complete
d2f3bf08ddcb: Pull complete
ca7661c4d9d5: Pull complete
4e8920cfc2e1: Pull complete
5efffd612257: Pull complete
2bba52cbe594: Pull complete
6c0e49f21133: Pull complete
c3f49a23e98a: Pull complete
07f9ae0ee986: Pull complete
004c60e60af8: Pull complete
Digest: sha256:dde938d7d8a24786648363fdb69d71bf25c77b2b56d3f26fd32d29427f375178
Status: Downloaded newer image for quay.io/ascend/vllm-ascend:v0.17.0rc1-a3
```

再次执行docker images命令，查看镜像是否加载成功

```
[root@atlas-27 ~]# docker images | grep v0.17.0rc1-a3
quay.io/ascend/vllm-ascend   v0.17.0rc1-a3       f7b939d9e47c        6 weeks ago         15.9GB
```

红框中是image id

由于直接从镜像仓库拉取镜像耗时较长，对于已经存放于服务器本地的镜像压缩包，也可直接通过如下命令进行镜像的加载

docker load -i <image_dir>/vllm-ascend-v0.17.0rc1-a3.tar

注：

<image_dir> 为服务器上存放镜像压缩包的路径

1.6.2 启动容器

用root用户登录服务器

执行命令检查容器镜像是否正常

docker images | grep v0.17.0rc1-a3

进入1.3小节中创建的个人工作目录

cd <work_dir>

创建容器启动脚本

vim docker_start.sh

输入i，进入编辑模式，粘贴下面脚本内容，编辑完成后按Esc进入命令模式再输入 :wq! 保存退出

注：要把宿主机的<work_dir>目录挂载到容器内同目录下

```
#!/bin/sh
NAME=AB2.0                  # 容器名称，自定义
PORT=8000                   # 容器与宿主机通信的端口号，不能冲突
IMAGE=f7b939d9e47c          # 1.6.1小节加载的容器镜像ID
docker run -itd -u 0 --ipc=host --privileged \
-e VLLM_USE_MODELSCOPE=True -e PYTORCH_NPU_ALLOC_CONF=max_split_size_mb:25
6 \
--name $NAME \
--device /dev/davinci_manager \
--device /dev/devmm_svm \
--device /dev/hisi_hdc \
-v /usr/local/dcmi:/usr/local/dcmi \
-v /usr/local/bin/npu-smi:/usr/local/bin/npu-smi \
-v /usr/local/Ascend/driver/lib64/:/usr/local/Ascend/driver/lib64/ \
-v /usr/local/Ascend/driver/version.info:/usr/local/Ascend/driver/version.info \
-v /etc/ascend_install.info:/etc/ascend_install.info \
-v /opt/data/verification/models:/root/.cache \
-v /home:/home \
-v /mnt:/mnt \
-v <work_dir>:<work_dir> \
-p $PORT:$PORT \
$IMAGE bash

# 创建容器
bash docker_start.sh

# 检查容器是否成功启动
docker ps | grep AB2.0
```

```
[root@atlas-27 lll]# docker ps
CONTAINER ID        IMAGE                                 COMMAND                  CREATED             STATUS              PORTS                    NAMES
f74eceb0f170        f7b939d9e47c                          "/bin/bash -c '    s…"   3 seconds ago       Up 2 seconds        0.0.0.0:8012->8012/tcp   AB2.0
```

2 AI 智能体安装与配置

2.1 自动化安装配置

2.1.1 Cursor 账号注册

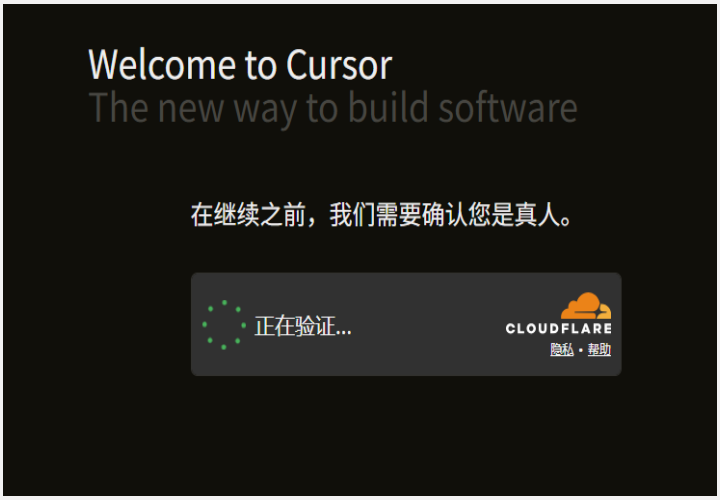
前提条件：准备邮箱地址

登录Cursor官网：https://cursor.com/cn/cli ，点击右上角【登录】按钮

第一次使用，需要注册账号，可以选择 图中的三种方式，也可以使用邮箱进行注册，下面主要以邮箱为例，进行注册

填写姓名和邮箱，点击继续

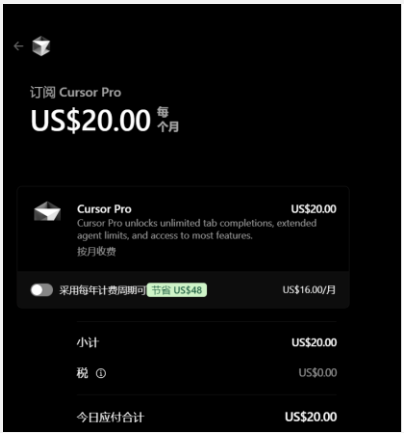
Cursor会进行人机验证，确认是不是真人

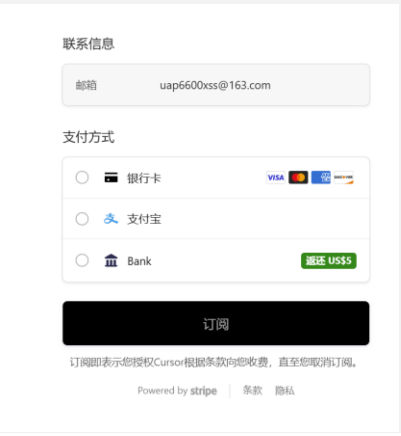


Cursor会给邮箱发送验证码，填入正确的验证码

输入自己的手机号码，注意手机格式，前面填 +86，后面填自己的手机号码，系统会自动对前三位加上括号，这是正常的，例如：(137) 1111-1111

输入收到的验证码即可，注意：Cursor默认会有基础模型/自动模式（Auto），可满足基本的日常开销，如果Auto模式额度不够或想使用更高级的模型，请根据需要购买付费版会员服务





2.1.2 在容器中安装 Cursor Agent

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 执行以下命令，安装Cursor Agent
curl https://cursor.com/install -fsS | bash
```

```
root@f74eceb0f170:/workspace# curl https://cursor.com/install -fsS | bash

Cursor Agent Installer

✓ Detected linux/arm64
✓ Package downloaded and extracted
✓ Package installed successfully
✓ Bin directory ready
✓ Symlink created

✨ Installation Complete!


Next Steps

1. Add ~/.local/bin to your PATH:
   For bash:
   echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
   source ~/.bashrc

2. Start using Cursor Agent:
   agent


Happy coding! 🚀
```

根据回显的信息，执行上图中蓝色的两条命令，导入环境变量

```
root@f74eceb0f170:/workspace# echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
root@f74eceb0f170:/workspace# source ~/.bashrc
```

输入“agent”，回车

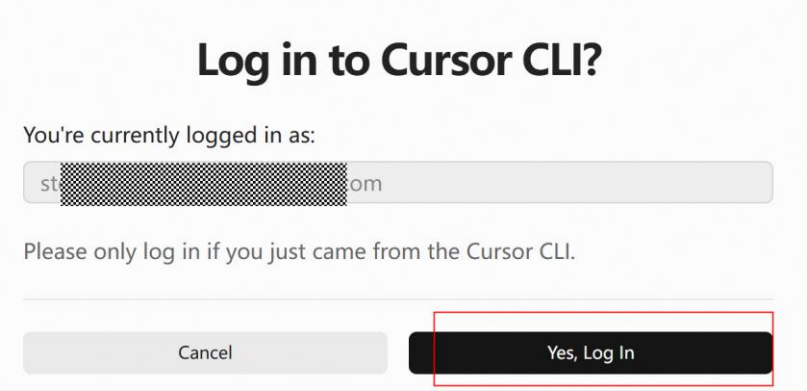
```
  Cursor Agent
  v2026.04.28-e984b46
  Press any key to log in...
```

根据提示，按任意键，回显如下

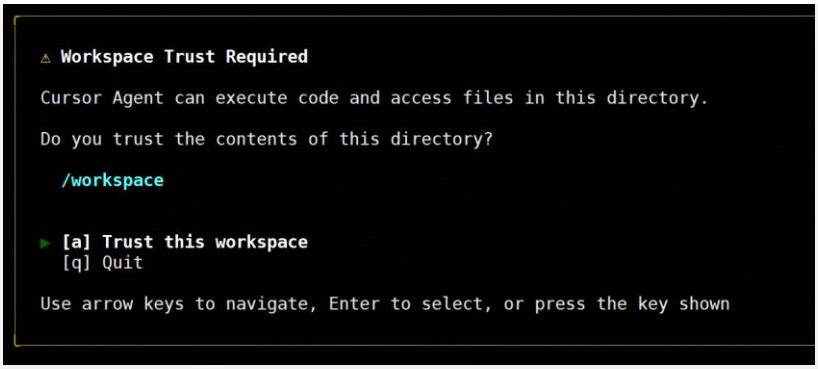
```
Signing in
If your browser didn't open, click this link to log in:

https://cursor.com/loginDeepControl?challenge=9pVl43fMM1pIGGQveNk7wuLjezIwZ-10Ot_rmnk0R-Y&uuid=87654ea6-437f-4aed-8279-6cd36ca52ea8&mode=login&redirectTarget=cli
```

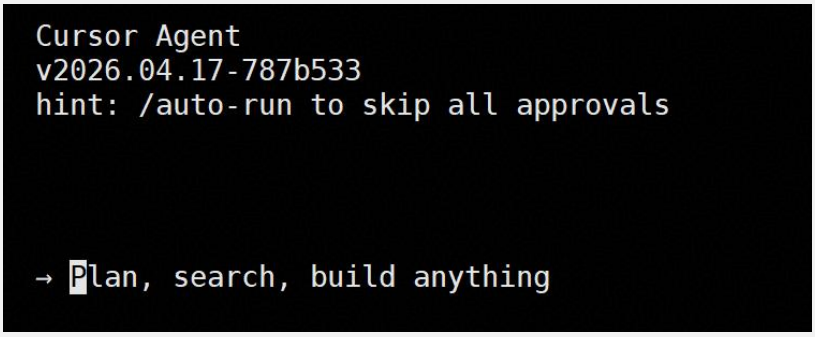
复制回显的链接地址，到浏览器中打开，并登录cursor账号（在2.2.1小节中注册）



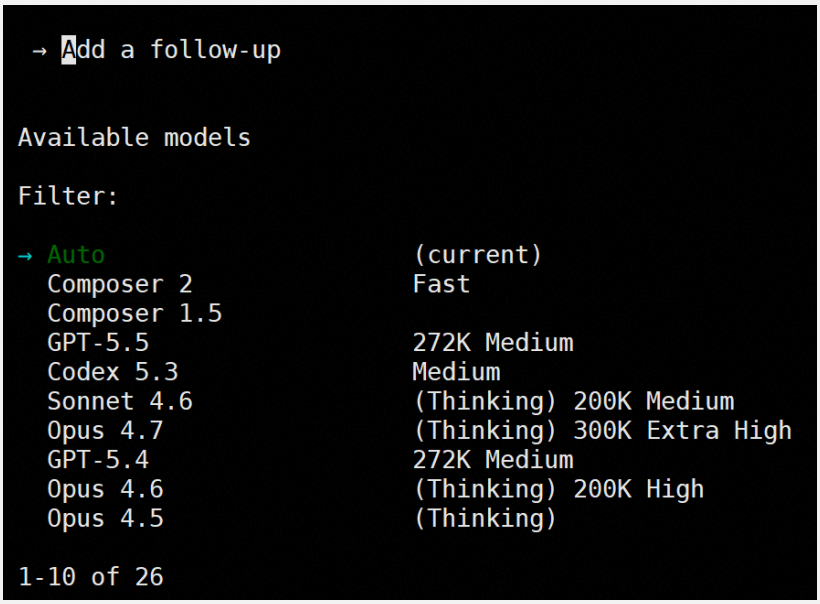
登录成功后，回到之前的昇腾服务器窗口，可以看到如下界面



默认选择**[a]**，回车即可进入cursor的CLI界面



在正式对话开始前输入“/model”回车，会展示cursor后端可使用的模型列表，默认选择Auto即可



保持在当前cursor对话界面先不要退出，接着安装和配置Claude Code

2.1.3 安装配置 Claude Code

在2.1.2小节的cursor对话界面，输入以下提示词，自动安装Claude Code并配置后端模型

提示词示例：

参考<work_dir>/claude-code-agent-teams-guide.md内容，搭建Claude Code + Agent Teams多智能体协作环境，搭建完成之后进行验证，API_KEY="替换为真实的API_Key"，MODEL="glm-5"。

注：

1. 提示词示例中，claude配置的后端模型是智谱的glm-5，<work_dir>和API_KEY需要替换成真实值，API-Key的获取方式请见1.1.1小节

2. 除了glm-5，也可以配置其他后端模型，例如MiniMax-M2.7，届时请修改提示词中的相关信息：

MODEL="MiniMax-M2.7"

API_KEY替换为真实值，API-Key的获取方式请见1.1.2小节

3. 安装配置过程大约需要3分钟左右，请耐心等待

4. 由于提示词中填写了真实的API Key，配置完成后界面可能会出现安全提示，例如：你在对话里提供了 API Key，且已写入本机 settings.json。该 Key已等同于泄露，建议在智谱控制台轮换/作废并换新 Key。此时可以根据提示信息，**在控制台创建新的API Key，并更新~/.claude/settings.json里的 ANTHROPIC_AUTH_TOKEN**，同时将旧的API Key废弃即可。

如出现以下选项，选择 “Auto-run everything”

```
Run this command?
Not in allowlist: uname -m, node --version, npm --version
   Run (once) (y)
   Add Shell(uname), Shell(node), Shell(npm --version) to allowlist? (tab)
→ Auto-run everything (shift+tab)
   Skip (esc or n)
```

安装结束返回示例

```
已按你给的指南完成了 Claude Code + Agent Teams 环境搭建，并做了验证。

• 已安装 Node.js：v22.14.0，npm：10.9.2
• 已安装 Claude Code：2.1.123（满足 Agent Teams 最低版本要求）
• 已创建配置目录和文件：/root/.claude/settings.json、/root/.claude.json
• 已写入你提供的参数（BASE_URL、MODEL、API_KEY）并开启：
  • CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1
  • teammateMode: "in-process"

验证结果：

• claude --version 正常返回，安装成功。
• 对 GLM Anthropic 兼容接口进行了实际联通测试，请求已到达服务端，但返回：
  • 余额不足或无可用资源包,请充值。（错误码 1113）

这说明环境本身已配置正确，当前阻塞点是该 API Key 对应账户额度不足，不是安装或配置问题。

你现在只需要充值/开通可用资源包后，再执行以下命令即可二次验证：

• cd /workspace
• claude -p "请只回复：验证成功"
```

注：

模型资源包需要提前订阅，否则会无法正常使用；

GLM Coding Plan和MiniMax Token Plan的订阅方式请分别参考1.1.1小节和1.1.2小节。

安装完成后，按两次【Ctrl+C】退出当前cursor对话界面回到容器内，输入claude，回车后选择【信任当前文件夹】，即可进入claude对话界面

```
root@f74eceb0f170:/workspace# claude

 Accessing workspace:

 /workspace

 Quick safety check: Is this a project you created or one you trust? (Like your own code,
 review what's in this folder first.

 Claude Code'll be able to read, edit, and execute files here.

 Security guide

 ❯ 1. Yes, I trust this folder
   2. No, exit
```

```
root@f74eceb0f170:/workspace# claude
 ╭─── Claude Code v2.1.123 ──────────────────────────────────────────────────────╮
 │                                │ Tips for getting started                     │
 │        Welcome back!           │ Ask Claude to create a new app or clone a re │
 │                                │ pository                                     │
 │                                │ ──────────────────────────────────────────── │
 │                                │ What's new                                   │
 │                                │ Check the Claude Code changelog for updates  │
 │   glm-5 · API Usage Billing    │                                              │
 │          /workspace            │                                              │
 ╰────────────────────────────────────────────────────────────────────────────────╯

 ❯
```

至此，Claude Code安装和后端模型配置全部完成

2.2 手动安装配置

进入1.6.2小节创建的容器

docker exec -it AB2.0 bash

执行docker命令返回如下时，说明已进入容器内

```
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
```

执行以下命令，安装nvm

curl -fsSL https://gitee.com/mirrors/nvm/raw/v0.39.7/install.sh | NVM_SOURCE=https://gitee.com/mirrors/nvm.git bash

```
[root@Atlas-800T-A3-trainServer-21 lll]# docker exec -it AB2.0 bash
root@d873b84ee033:/workspace# curl -fsSL https://gitee.com/mirrors/nvm/raw/v0.39.7/install.
sh | NVM_SOURCE=https://gitee.com/mirrors/nvm.git bash
=> Downloading nvm from git to '/root/.nvm'
=> Cloning into '/root/.nvm'...
remote: Enumerating objects: 427, done.
remote: Counting objects: 100% (427/427), done.
remote: Compressing objects: 100% (354/354), done.
remote: Total 427 (delta 58), reused 222 (delta 45), pack-reused 0 (from 0)
Receiving objects: 100% (427/427), 415.34 KiB | 563.00 KiB/s, done.
Resolving deltas: 100% (58/58), done.
* (HEAD detached at FETCH_HEAD)
  master
=> Compressing and cleaning up git repository

=> Appending nvm source string to /root/.bashrc
=> Appending bash_completion source string to /root/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_c
ompletion
```

执行红框中的命令

```
root@d873b84ee033:/workspace# export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && . "$NVM_DIR/bash_completion"
```

依次执行以下命令安装Node.js 18+（建议 20 LTS）

nvm install 20

nvm use 20

node -v

npm -v

```
root@d873b84ee033:/workspace# nvm install 20
Downloading and installing node v20.20.2...
Downloading https://nodejs.org/dist/v20.20.2/node-v20.20.2-linux-arm64.tar.xz...
################################################################################ 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v20.20.2 (npm v10.8.2)
Creating default alias: default -> 20 (-> v20.20.2)
root@d873b84ee033:/workspace# nvm use 20
Now using node v20.20.2 (npm v10.8.2)
root@d873b84ee033:/workspace# node -v
v20.20.2
root@d873b84ee033:/workspace# npm -v
10.8.2
```

用 npm 全局安装 Claude Code

source ~/.bashrc

npm install -g @anthropic-ai/claude-code

```
root@d873b84ee033:/workspace# source ~/.bashrc
root@d873b84ee033:/workspace# npm install -g @anthropic-ai/claude-code

added 2 packages in 5m
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.14.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.14.1
npm notice To update run: npm install -g npm@11.14.1
npm notice
```

验证Claude Code安装完毕

claude --version

claude doctor

```
root@d873b84ee033:/workspace# claude --version
2.1.138 (Claude Code)
root@d873b84ee033:/workspace# claude doctor

 Diagnostics
 ├ Currently running: npm-global (2.1.138)
 ├ Commit: d6d494651eb4
 ├ Platform: linux-arm64
 ├ Path: /root/.nvm/versions/node/v20.20.2/lib/node_modules/@anthropic-ai/claude-code/bi
   n/claude.exe
 ├ Config install method: not set
 └ Search: OK (bundled)

 Updates
 ├ Auto-updates: enabled
 ├ Update permissions: Yes
 ├ Auto-update channel: latest
 ├ Stable version: 2.1.126
 └ Latest version: 2.1.138
```

在容器内输入如下命令，配置Claude后端模型

curl -O "https://cdn.bigmodel.cn/install/claude_code_env.sh" && bash ./claude_code_env.sh

```
root@d873b84ee033:/workspace# curl -O "https://cdn.bigmodel.cn/install/claude_code_env.sh"
&& bash ./claude_code_env.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5389  100  5389    0     0  33483      0 --:--:-- --:--:-- --:--:-- 33472
🚀 Starting claude_code_env.sh
☑Node.js is already installed: v20.20.2
☑Claude Code is already installed: 2.1.138 (Claude Code)
◇  Configuring Claude Code...
   You can get your API key from: https://open.bigmodel.cn/usercenter/proj-mgmt/apikeys
🔑 Please enter your ZHIPU API key:
☑Claude Code configured successfully

☑🎉 Installation completed successfully!
```

命令执行过程中会提示输入API key，请输入一开始准备的智谱、MiniMax等大模型服务的API Key，此处以MiniMax为例。

命令执行完毕后，打开~/.claude/settings.json文件，可以看到"ANTHROPIC_AUTH_TOKEN"已配置为刚才输入的API Key，"ANTHROPIC_BASE_URL"为默认的"https://open.bigmodel.cn/api/anthropic"

对~/.claude/settings.json文件做如下修改：

1. "ANTHROPIC_BASE_URL"修改为MiniMax的url："https://api.minimaxi.com/anthropic"
2. 添加如下四条环境变量：

"ANTHROPIC_MODEL": "MiniMax-M2.7",

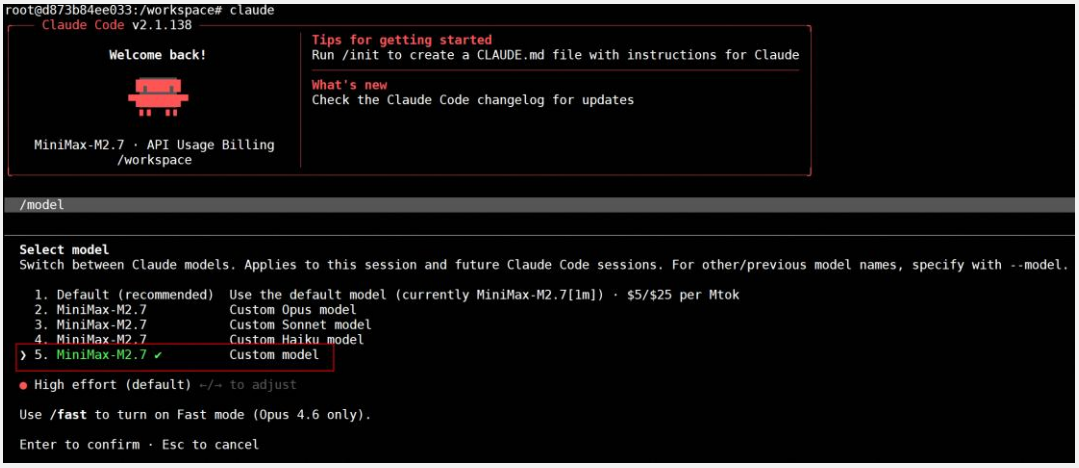
"ANTHROPIC_DEFAULT_HAIKU_MODEL": "MiniMax-M2.7",

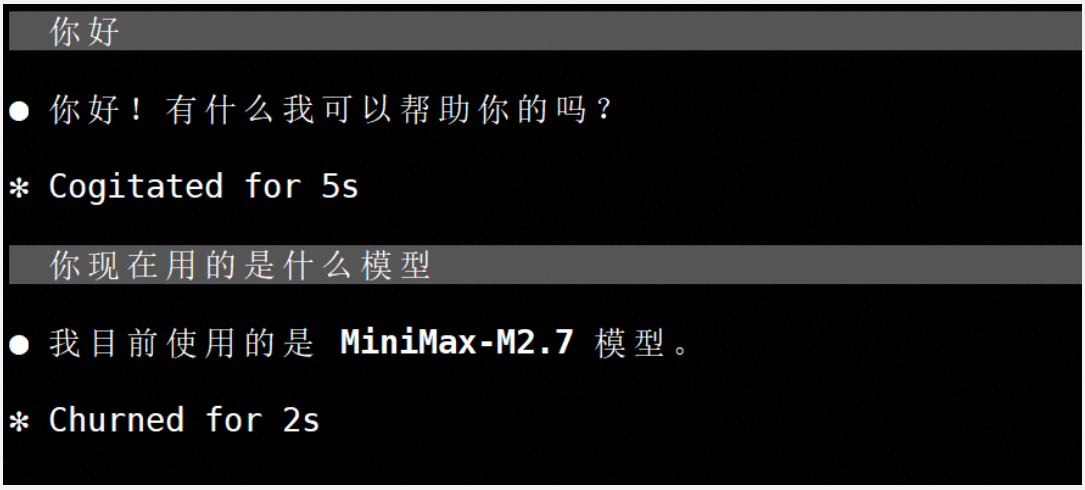
"ANTHROPIC_DEFAULT_SONNET_MODEL": "MiniMax-M2.7",

"ANTHROPIC_DEFAULT_OPUS_MODEL": "MiniMax-M2.7"

修改完成后的配置文件内容如下：

修改完成后，输入claude，回车即可进入对话界面，输入”/model”命令可以查看/切换模型





更多Claude Code配置相关指导请参考：

1. 智谱：https://docs.bigmodel.cn/cn/coding-plan/tool/claude#%E6%AD%A5%E9%AA%A4%E4%BA%8C%EF%BC%9A%E9%85%8D%E7%BD%AE-glm-coding-plan
2. MiniMax：https://platform.minimaxi.com/docs/token-plan/claude-code

3 AscendBridge 2.0 工具使用

3.1 工具简介

SLAI-AscendBridge2 是一款面向昇腾 NPU 的自动化智能体编排与单模型适配框架，用于将 PyTorch 模型迁移到 Ascend。它支持从模型发现、环境治理、代码适配、精度评测到 NPU 性能优化的闭环流程；如果需要，也可以继续扩展到第四阶段 business_benchmark。

仓库定位：

SLAI-AscendBridge2

开源地址：https://gitcode.com/SLAI/SLAI-AscendBridge2

- 框架、脚本、检查器、dashboard、.claude agents / skills / agent-memory
- 适合批量调度、多阶段闭环、统一状态管理

SLAI-AscendBridge2-Adaptations

开源地址：https://gitcode.com/SLAI/SLAI-AscendBridge2-Adaptations

- 是 SLAI-AscendBridge2 的已适配模型开源合集仓，存放模型级 adaptations/{name}/ 产物集合
- 适合单模型复用、逐步公开、按需筛选发布

仓库主要文件架构如下：

```
SLAI-AscendBridge2/
├── .claude/
│   ├── agent-memory/                # 智能体阶段记忆与规则沉淀
│   ├── agents/                      # 智能体定义
│   │   ├── team-lead.md             # 团队主控：负责任务分配、并发编排、状态推进
│   │   ├── adapter.md               # 适配执行：模型迁移到Ascend、保证demo可运行
│   │   ├── model-crawler.md         # 模型发现：抓取并注册新模型到 board
│   │   ├── benchmark-runner.md      # 评测执行：生成/运行 accuracy_run
│   │   ├── npu-optimizer.md         # 优化执行：生成 accuracy_run_perf
│   │   └── business-benchmark.md    # 真实业务测评与 NPU/CUDA 对比证据沉淀
│   └── skills/                      # 技能库
│       ├── ascend-adaptation/
│       │   └── SKILL.md             # 通用 Ascend 适配方法与操作清单
│       ├── ascend-diffusers-adaptation/   # diffusers类模型迁移到NPU的专项指导
│       ├── ascend-diffusers-benchmark/    # diffusers类模型 benchmark 编写与执行规范
│       ├── ascend-diffusers-optimization/ # diffusers类模型推理优化策略
│       ├── benchmark-analysis/      # 聚合报告、输出对比、trace/profiling 分析说明
```

```
│       ├── benchmark-script/          # accuracy_run.py 生成与执行主技能
│       ├── torchair-graph-mode/       # TorchAir 图模式启用/调优/排障总指南等
│       ├── torch-npu-optimization/ # 通用 torch_npu 优化策略与执行步骤等
│       └── …
├── .agents -> .claude                          # 兼容部分 agent 工具的目录入口
├── AGENTS.md -> CLAUDE.md                      # 兼容 AGENTS.md 约定
├── adaptation/
│   └── scripts/
│       ├── adaptation_manager.py               # 第一阶段运行与产物管理
│       ├── check_adaptation.py                 # demo.py / adaptation 完成度检查
│       ├── package_adaptations.py              # adaptation 打包
│       └── run_completed_adaptations.py        # 批量下载已完成适配模型的权重
├── benchmark/                                  # 精度/性能评测模块
│   ├── scripts/
│       ├── benchmark_tool.py                   # 聚合、对比、trace/profiling 分析
│       ├── benchmark_manager.py                # benchmark 运行与产物管理
│       └── check_accuracy_run.py               # accuracy_run.py 规范检查
│   ├── figures/                                # 图表产物
│   └── reports/                                # 汇总报告
├── optimization/                               # NPU 推理优化模块
│   ├── scripts/
│       ├── optimization_tool.py                # 优化数据聚合与对比
│       ├── optimization_manager.py             # 优化运行管理
│       ├── check_accuracy_run_perf.py          # accuracy_run_perf.py 结构检查
│       └── check_optimization_notes.py
│   └── reports/                                # 优化评测结果
├── business_benchmark/                         # 业务测评模块
│   ├── scripts/                                # 运行管理、汇总、completed gate 检查
│   └── templates/                              # 模板文件
├── dashboard/                                  # Web 看板
├── prompts/                                    # team-lead prompt 模板
├── scripts/
│   ├── board_ops.py                    # 看板 CRUD、心跳、任务分配
│   ├── get_model_info.py               # 模型元数据提取
│   ├── download_datasets.py            # 数据集下载
│   ├── dataset_mapping.py              # 模型 -> 数据集映射
│   ├── ensure_agent_symlinks.sh        # 恢复 .agents / AGENTS.md 兼容软链接
│   └── sanitize_repo_paths.py          # 路径清洗工具
├── adaptations/                        # 默认空目录，可挂载或拷贝单模型adaptation
├── CLAUDE.md                           # 项目上下文说明
```

```
├── pyproject.toml                # Python/nv项目配置
├── uv.lock                       # uv锁文件
└── README.md                     # 项目说明
```

3.2 在容器中下载工具包

```
# 用root用户登录服务器

# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 从开源仓下载工具代码到个人工作目录
cd <work_dir>
git clone https://gitcode.com/SLAI/SLAI-AscendBridge2
```

3.3 使用方式

3.3.1 环境准备

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 在容器中安装uv（如之前未安装）
curl -LsSf https://astral.sh/uv/install.sh | sh

# 初始化框架环境
uv sync
uv run python scripts/board_ops.py init

# 推荐环境变量
export HF_ENDPOINT=https://hf-mirror.com
```

```
export TASK_QUEUE_ENABLE=1

# 权重准备建议
```
建议提前把模型权重准备到本地路径，避免 agent 在适配过程中临时下载导致等待时间过长或受网络影响。如果不提前准备，agent 也可以自行下载，但整体链路通常更慢。
权重文件下载可参考1.4小节。

3.3.2 典型使用方式一：Claude 自动批量编排

适用场景：

- 一次要处理多个模型
- 希望通过 .claude/agents、.claude/skills、.claude/agent-memory 配合 team-lead 自动推进多个阶段
- 需要 board.db + dashboard 的统一状态管理

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 进入仓库根目录
cd <work_dir>/SLAI-AscendBridge2

# 初始化空数据库
uv run python scripts/board_ops.py init

# 把模型注册进 board.db
uv run python scripts/board_ops.py register_model \
  --model_id "google/gemma-4-31B-it" \
  --source huggingface \
  --url "https://huggingface.co/google/gemma-4-31B-it" \
  --description "batch adaptation example"

注：如果是多个模型，可以循环执行 register_model

# 启动批量编排有两种方式：自动批量调度和手动进入Claude team-lead
1. 自动批量调度
./run_auto_team_lead.sh
```

注：如只跑某个阶段，可通过切换PROMPT_MODE实现
PROMPT_MODE=benchmark ./run_auto_team_lead.sh
PROMPT_MODE=optimization ./run_auto_team_lead.sh
PROMPT_MODE=business ./run_auto_team_lead.sh

长期运行时，可交给看门狗脚本
nohup ./team_lead_watchdog.sh > <work_dir>/slai_team_lead.log 2>&1 &

2. 如需手动进入Claude team-lead，可在仓库根目录启动
cd <work_dir>/SLAI-AscendBridge2
IS_SANDBOX=1 claude --dangerously-skip-permissions --agent team-lead

提示词示例：
请检查 board.db 中的pending模型，优先推进adaptation -> benchmark -> optimization。
只处理当前看板中的任务，不要新建无关目录。
每完成一个模型后给出阶段结论与风险摘要。

查看执行状态
uv run python scripts/board_ops.py list_adaptation_tasks --status pending
uv run python scripts/board_ops.py list_benchmark_tasks --status pending
uv run python scripts/board_ops.py list_optimization_tasks --status pending

注：如本地有board.db，也可直接打开dashboard/index.html 查看看板。

3.3.3 典型使用方式二：Claude 单模型连贯适配

适用场景：

- 只想处理一个模型
- 希望一个对话连续完成 Adaptation -> Benchmark -> Optimization
- 需要 agent 在同一上下文里连续修复问题，而不是切到批量队列

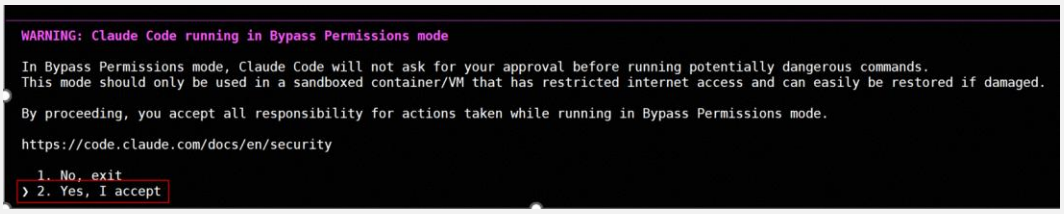
下面用 google/gemma-4-31B-it 模型作为示例演示流程。

进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

执行docker命令返回如下时，说明已进入容器内
```
root@5078b464d7ef:/workspace# docker
bash: docker: command not found
```

进入工具根目录下
cd <work_dir>/SLAI-AscendBridge2

# 执行以下命令，启动Claude
IS_SANDBOX=1 claude --dangerously-skip-permissions
出现如下选项，选择 2

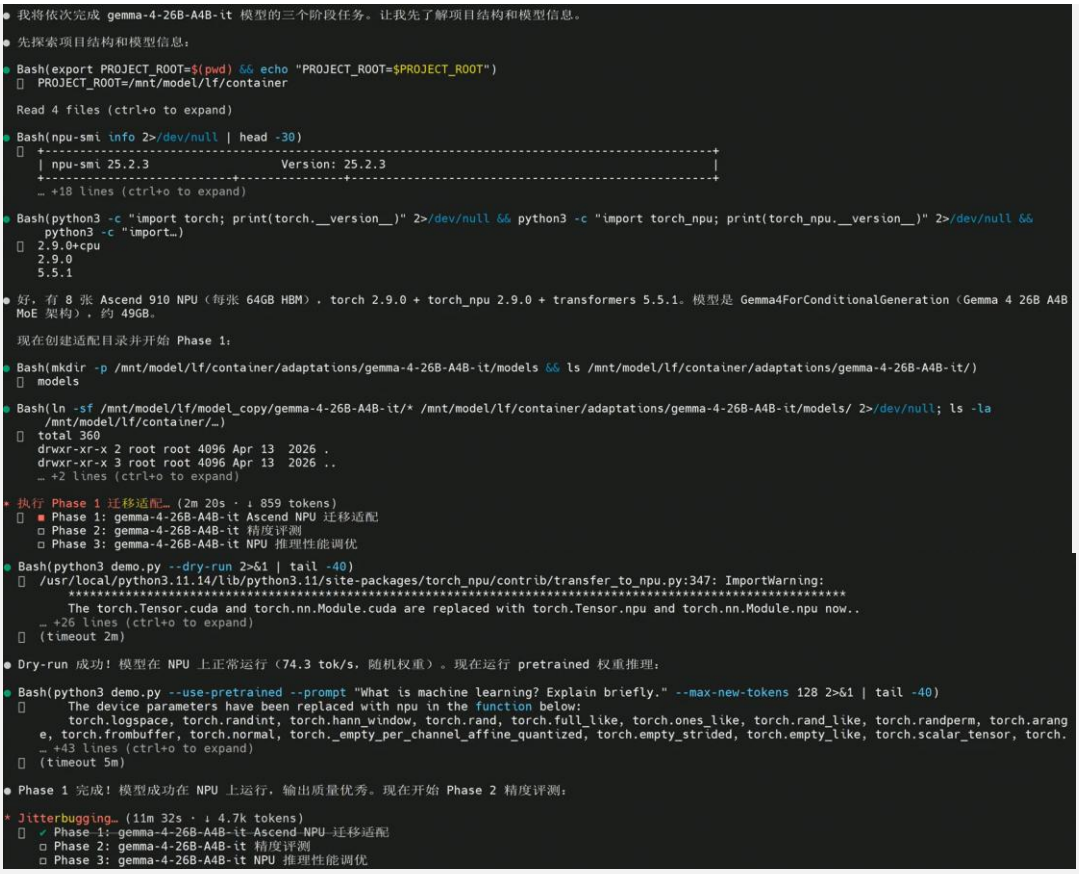


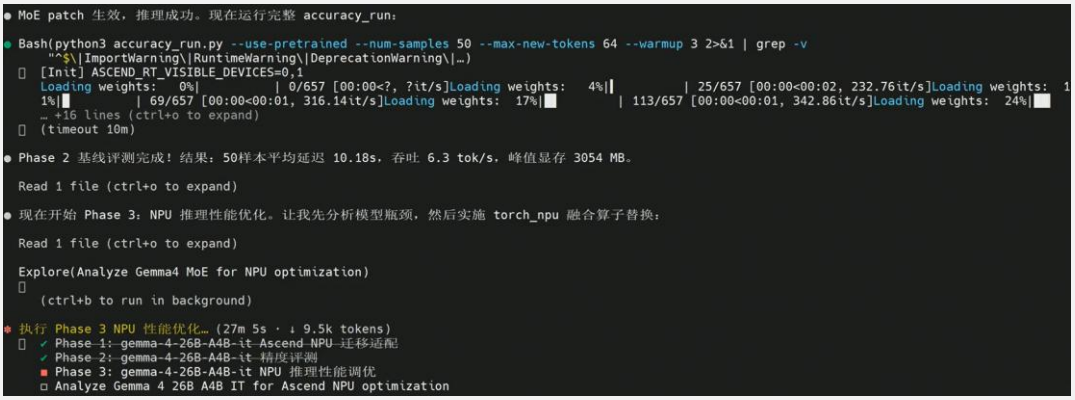
注：
若直接用“claude”命令启动，后续agent有很多操作都需要人工允许才可继续往下进行，用上述命令启动，表示信任claude的所有操作，无需人为干预。

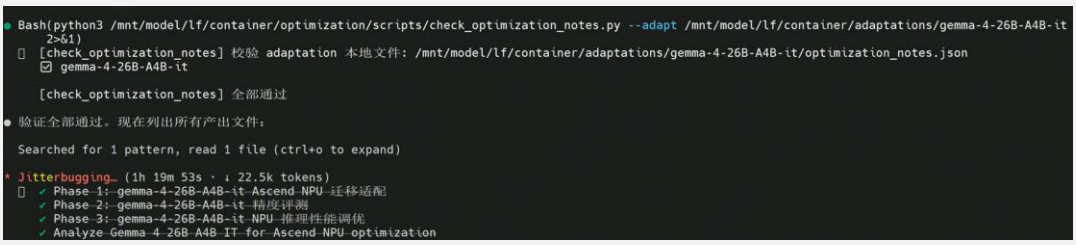
输入提示词，开始模型的适配调优
提示词示例：
你当前在一台昇腾910C服务器上，你的任务是完成google/gemma-4-31B-it模型的迁移适配、精度对齐和推理性能调优，请依次完成指定的这三个阶段的任务，并给我输出一份完整详细的任务报告并落盘，报告中一定要包含性能调优前后的数据对比，过程中生成的文件请在adaptations目录下新建一个模型的文件夹进行存放，模型权重文件在/mnt/model/gemma-4-31B-it。

注：
1. 如更换其他模型进行适配调优，请同步修改提示词中的模型名称以及权重文件所在目录，权重文件建议提前下载，参考1.5小节。
2. 如果实际服务器型号与示例不同，也请修改提示词。
3. 如果没有提前下载模型权重，在提示词中无需指定权重文件路径，Agent会自行下载。

执行过程示例


我将依次完成 gemma-4-26B-A4B-it 模型的三个阶段任务。让我先了解项目结构和模型信息。
先探索项目结构和模型信息：
Bash(export PROJECT_ROOT=$(pwd) && echo "PROJECT_ROOT=$PROJECT_ROOT")
PROJECT_ROOT=/mnt/model/lf/container
Read 4 files (ctrl+o to expand)
Bash(npu-smi info 2>/dev/null | head -30)
| npu-smi 25.2.3 Version: 25.2.3 |
… +18 lines (ctrl+o to expand)
Bash(python3 -c "import torch; print(torch.__version__)" 2>/dev/null && python3 -c "import torch_npu; print(torch_npu.__version__)" 2>/dev/null &&
python3 -c "import…)
2.9.0+cpu
2.9.0
5.5.1
好，有 8 张 Ascend 910 NPU（每张 64GB HBM），torch 2.9.0 + torch_npu 2.9.0 + transformers 5.5.1。模型是 Gemma4ForConditionalGeneration（Gemma 4 26B A4B MoE 架构），约 49GB。
现在创建适配目录并开始 Phase 1：
Bash(mkdir -p /mnt/model/lf/container/adaptations/gemma-4-26B-A4B-it/models && ls /mnt/model/lf/container/adaptations/gemma-4-26B-A4B-it/)
models
Bash(ln -sf /mnt/model/lf/model_copy/gemma-4-26B-A4B-it/* /mnt/model/lf/container/adaptations/gemma-4-26B-A4B-it/models/ 2>/dev/null; ls -la
/mnt/model/lf/container/…)
total 360
drwxr-xr-x 2 root root 4096 Apr 13 2026 .
drwxr-xr-x 3 root root 4096 Apr 13 2026 ..
… +2 lines (ctrl+o to expand)
* 执行 Phase 1 迁移适配… (2m 20s · ↓ 859 tokens)
■ Phase 1: gemma-4-26B-A4B-it Ascend NPU 迁移适配
□ Phase 2: gemma-4-26B-A4B-it 精度评测
□ Phase 3: gemma-4-26B-A4B-it NPU 推理性能调优
Bash(python3 demo.py --dry-run 2>&1 | tail -40)
/usr/local/python3.11.14/lib/python3.11/site-packages/torch_npu/contrib/transfer_to_npu.py:347: ImportWarning:
The torch.Tensor.cuda and torch.nn.Module.cuda are replaced with torch.Tensor.npu and torch.nn.Module.npu now..
… +26 lines (ctrl+o to expand)
(timeout 2m)
Dry-run 成功！模型在 NPU 上正常运行（74.3 tok/s，随机权重）。现在运行 pretrained 权重推理：
Bash(python3 demo.py --use-pretrained --prompt "What is machine learning? Explain briefly." --max-new-tokens 128 2>&1 | tail -40)
The device parameters have been replaced with npu in the function below:
torch.logspace, torch.randint, torch.hann_window, torch.rand, torch.full_like, torch.ones_like, torch.rand_like, torch.randperm, torch.arang
e, torch.frombuffer, torch.normal, torch._empty_per_channel_affine_quantized, torch.empty_strided, torch.empty_like, torch.scalar_tensor, torch.
… +43 lines (ctrl+o to expand)
(timeout 5m)
Phase 1 完成！模型成功在 NPU 上运行，输出质量优秀。现在开始 Phase 2 精度评测：
* Jitterbugging… (11m 32s · ↓ 4.7k tokens)
✓ Phase 1: gemma-4-26B-A4B-it Ascend NPU 迁移适配
□ Phase 2: gemma-4-26B-A4B-it 精度评测
□ Phase 3: gemma-4-26B-A4B-it NPU 推理性能调优


MoE patch 生效，推理成功。现在运行完整 accuracy_run：
Bash(python3 accuracy_run.py --use-pretrained --num-samples 50 --max-new-tokens 64 --warmup 3 2>&1 | grep -v
"^$\|ImportWarning\|RuntimeWarning\|DeprecationWarning\|…)
[Init] ASCEND_RT_VISIBLE_DEVICES=0,1
Loading weights: 0%| | 0/657 [00:00<?, ?it/s]Loading weights: 4%| | 25/657 [00:00<00:02, 232.76it/s]Loading weights: 1
1%| | 69/657 [00:00<00:01, 316.14it/s]Loading weights: 17%| | 113/657 [00:00<00:01, 342.86it/s]Loading weights: 24%|
… +16 lines (ctrl+o to expand)
(timeout 10m)
Phase 2 基线评测完成！结果：50样本平均延迟 10.18s，吞吐 6.3 tok/s，峰值显存 3054 MB。
Read 1 file (ctrl+o to expand)
现在开始 Phase 3：NPU 推理性能优化。让我先分析模型瓶颈，然后实施 torch_npu 融合算子替换：
Read 1 file (ctrl+o to expand)
Explore(Analyze Gemma4 MoE for NPU optimization)
(ctrl+b to run in background)
* 执行 Phase 3 NPU 性能优化… (27m 5s · ↓ 9.5k tokens)
✓ Phase 1: gemma-4-26B-A4B-it Ascend NPU 迁移适配
✓ Phase 2: gemma-4-26B-A4B-it 精度评测
■ Phase 3: gemma-4-26B-A4B-it NPU 推理性能调优
□ Analyze Gemma 4 26B A4B IT for Ascend NPU optimization


Bash(python3 /mnt/model/lf/container/optimization/scripts/check_optimization_notes.py --adapt /mnt/model/lf/container/adaptations/gemma-4-26B-A4B-it
2>&1)
[check_optimization_notes] 校验 adaptation 本地文件: /mnt/model/lf/container/adaptations/gemma-4-26B-A4B-it/optimization_notes.json
☑ gemma-4-26B-A4B-it
[check_optimization_notes] 全部通过
验证全部通过。现在列出所有产出文件：
Searched for 1 pattern, read 1 file (ctrl+o to expand)
* Jitterbugging… (1h 19m 53s · ↓ 22.5k tokens)
✓ Phase 1: gemma-4-26B-A4B-it Ascend NPU 迁移适配
✓ Phase 2: gemma-4-26B-A4B-it 精度评测
✓ Phase 3: gemma-4-26B-A4B-it NPU 推理性能调优
✓ Analyze Gemma 4 26B A4B IT for Ascend NPU optimization

结果总结输出

三个阶段全部完成。现在输出完整的任务报告：

Gemma 4 26B A4B IT 昇腾 NPU 迁移适配、精度评测与性能调优报告

一、模型概述

项目	详情
模型	Google Gemma 4 26B A4B IT
架构	Gemma4ForConditionalGeneration（多模态） → 提取 Gemma4ForCausalLM（纯文本）
MoE 配置	128 专家，top-8 路由，~26B 总参数 / ~4B 活跃参数
层数	30 层（25 sliding_attention + 5 full_attention）
隐藏维度	2816
注意力头	16 heads / 8 KV heads (GQA)
权重路径	/mnt/model/lf/model_copy/gemma-4-26B-A4B-it (49GB)
硬件	华为昇腾 910C NPU × 8（每张 64GB HBM）
软件栈	PyTorch 2.9.0 + torch_npu 2.9.0 + transformers 5.5.1

二、Phase 1：迁移适配

2.1 关键技术挑战与解决

挑战	解决方案
多模态权重提取	模型原为 Gemma4ForConditionalGeneration，包含视觉编码器。通过 model.language_model.* → model.* 重映射，分离出 657 个文本权重键（丢弃 356 个视觉权重键）
MoE torch._grouped_mm 不支持	Ascend NPU 不支持 torch._grouped_mm（CUDA 专有算子），运行时 ACL 报错 507018。通过 monkey-patch transformers.integrations.moe._can_use_grouped_mm 强制返回 False，使用 loop-based fallback
Gemma4 layer_types 长度校验	config.num_hidden_layers 必须等于 layer_types 数组长度。dry-run 时将两者同步截断到 2
AutoConfig.for_model 参数冲突	model_type 字段不能同时作为位置参数和关键字参数传入，需先 pop 再传入

2.2 产出文件

- demo.py — 主推理脚本（支持 --dry-run、--use-pretrained、--cpu）
- pyproject.toml — 依赖配置
- output.txt — 首次推理输出记录

2.3 验证结果

```
[Mode] PRETRAINED (full weights)
[Run] Output: At its simplest, **machine learning (ML)** is a field of artificial intelligence...
[Metrics] 128 tokens in 24.02s (5.3 tok/s)
[Success] Inference completed on npu:0
```

☑ 模型在 Ascend NPU 上成功运行，输出语义正确、流畅。

三、Phase 2：精度评测（Baseline）

3.1 测试配置

参数	值
样本数	50
每样本最大生成 token	64
Warmup 次数	3
精度	bfloat16
设备	npu:0（跨 2 张 NPU device_map="auto"）

3.2 Baseline 性能数据

指标	值
平均延迟	10.18s
平均吞吐	6.29 tok/s
峰值显存	3,054 MB (3.0 GB)
总生成 token	3,200
总延迟	509.21s

3.3 产出文件

- accuracy_run.py — 评测脚本
- benchmark_metrics_google_gemma-4-26b-a4b-it_baseline.json
- outputs_google_gemma-4-26b-a4b-it_baseline.pt

四、Phase 3：NPU 推理性能调优

4.1 优化策略

采用 torch_npu 融合算子 monkey-patch 方式，在 model_files/__init__.py 中实现：

优化项	实例数	说明
npu_rms_norm	300+	替换 Gemma4RMSNorm（每层 10 个：input/post_attn/pre_ffw/post_ffw + q/k/v norm + router norm + MoE 额外 norm）
npu_gelu (Dense MLP)	25	替换 Gemma4TextMLP 中的 F.gelu(approximate='tanh')
npu_gelu (MoE Experts)	128 专家 × 5 层	替换 Gemma4TextExperts.act_fn，保持路由逻辑不变
npu_rotary_mul	60	替换 apply_rotary_pos_emb（每层 2 次：Q 和 K）
环境变量	-	TASK_QUEUE_ENABLE=1、COMBINED_ENABLE=1、ACL_OP_COMPILER_CACHE_MODE=1

跳过的优化：

优化	原因
npu_fusion_attention	模型使用 softcapping + 混合 sliding/full attention + GQA + per-head QK norm，掩码构建复杂，稳定性风险高
npu_grouped_matmul	MoE 专家的 per-token 路由导致批次极小，overhead 可能超过收益

4.2 调优过程

尝试	配置	结果	分析
✗ 全量重写 experts forward	rms_norm + gelu + rotary + experts 路由重写	18.73s (0.54x)	性能退化：重写 experts forward 路由逻辑引入额外开销
☑ 核心3项	rms_norm + dense gelu + rotary	9.35s (1.09x)	有效加速
☑ 最终方案	rms_norm + dense gelu + rotary + 轻量 experts act_fn	9.07s (1.12x)	轻量 act_fn 替换进一步优化

4.3 优化后性能数据

指标	Baseline	Optimized	变化
平均延迟	10.18s	9.07s	-10.9%
平均吞吐	6.29 tok/s	7.06 tok/s	+12.2%
峰值显存	3,054 MB	3,054 MB	持平
Speedup	1.00x	1.12x	-

3.3.4 典型使用方式三：Codex/Cursor 单模型连贯适配

适用场景：

- 你不想走 Claude team-lead 的批量队列
- 你更习惯在终端或 IDE 内用一个 agent 连续完成单模型适配
- 你希望 agent 在本地仓库里直接改代码、跑命令、迭代修复

一、Codex CLI

推荐做法：

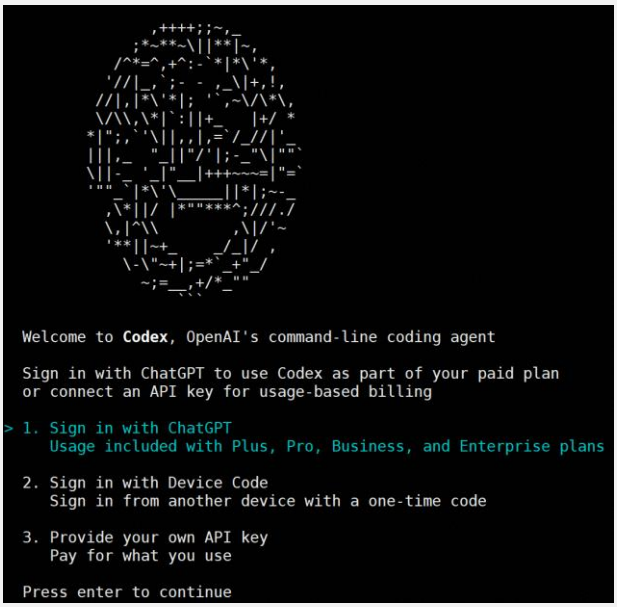
- 第一次运行先完成登录
- 单模型任务尽量保持在一个 Codex 会话里完成
- 在 prompt 中显式点名关键目录，例如 scripts/、adaptation/、benchmark/、optimization/、adaptations/<name>/
- 如果要切换模型或推理强度，可在会话中使用 Codex CLI 的 /model

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 进入工具根目录下
cd <work_dir>/SLAI-AscendBridge2

# 启动Codex CLI
npm i -g @openai/codex
codex
```



根据实际情况选择登录方式，按照屏幕提示操作

```
# 建议使用“目标 / 上下文 / 约束 / 完成标准”结构来下达任务
```

示例：

```
Goal:
完成 google/gemma-4-31B-it 的单模型连贯适配，覆盖 Adaptation -> Benchmark -> Optimization。

Context:
- 仓库根目录是当前工作目录
- 权重路径：<work_dir>/gemma-4-31B-it
- 重点目录：scripts/, adaptation/, benchmark/, optimization/

Constraints:
- 只处理一个模型
- 不修改其他 adaptation
- benchmark / optimization 产物生成后必须跑对应 checker

Done when:
- demo.py 可运行
- accuracy_run.py / accuracy_run_perf.py 均通过 checker
- 给出最终性能对比摘要
```

二、Cursor

推荐流程：

1. 用 Cursor 打开 SLAI-AscendBridge2 仓库根目录
2. 进入 Agent mode 或 Agents Window
3. 明确告诉 agent：只处理一个模型、只改一个 adaptation 目录
4. 先让 agent 制定计划，再让它落地修改和运行命令
5. 每个阶段完成后先 review diff，再继续下一阶段

```
# 进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 启动Cursor Agent
cd <work_dir>/SLAI-AscendBridge2
agent

# 提示词可直接复用上面的 Codex / Claude 单模型模板
```

提示词示例：

```
你当前在一台昇腾910C服务器上，你的任务是完成google/gemma-4-31B-it模型的迁移适配、精度对齐和推理性能调优，请依次完成指定的这三个阶段的任务，并给我输出一份完整详细的任务报告，一定要包含性能调优前后的数据对比，过程中生成的文件请在adaptations目录下新建一个模型的文件夹进行存放，模型权重文件在/mnt/model/gemma-4-31B-it。
```

4 已适配模型开源合集仓使用

4.1 仓库简介

SLAI-AscendBridge2-Adaptations 仓库是 SLAI-AscendBridge2 的已适配模型开源合集仓。用于在现有适配工作基础上，逐步开源已经完成适配、清理和核验的模型 adaptation。adaptations/ 是统一的模型适配目录入口，后续会按模型逐步加入对应的 <sanitized_model_id>/ 子目录。

仓库主要文件架构如下：

```
SLAI-AscendBridge2-Adaptations/
├── adaptations/                    # 各模型适配产物目录（按模型拆分）
│   └── {sanitized_model_id}/       # 单模型产物目录
│       ├── demo.py                 # 主推理脚本（支持 --dry-run）
│       ├── accuracy_run.py         # 精度评测脚本
│       ├── accuracy_run_perf.py    # NPU 优化版评测脚本
│       ├── pyproject.toml          # 依赖配置（含 cuda/ascend extra）
│       ├── model_files/            # NPU 优化版模型monkey patch文件
│       ├── profiling/              # NPU profiling 数据
│       ├── optimization_notes.json # 优化记录
│       ├── README.md               # 脚本执行指导
│       └── ...
```

4.2 下载开源仓

```
# 用root用户登录服务器

# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 从开源仓下载
cd <work_dir>
git clone https://gitcode.com/SLAI/SLAI-AscendBridge2-Adaptations
```

4.3 使用方式

4.3.1 独立使用

只做推理或局部验证时，可独立使用。

```
# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 在容器中安装uv（如之前未安装）
curl -LsSf https://astral.sh/uv/install.sh | sh

# 进入adaptations目录
cd <work_dir>/SLAI-AscendBridge2-Adaptations/adaptations/
可以先用“ll”命令查看当前目录下都有哪些模型的脚本
[root@Atlas-800T-A3-trainServer-21 SLAI-AscendBridge2-Adaptations]# cd adaptations/
[root@Atlas-800T-A3-trainServer-21 adaptations]# ll
total 260
drwxr-xr-x 3 root root 4096 May  9  2026 01_ai_yi_1_5_34b_chat
drwxr-xr-x 3 root root 4096 May  9  2026 aleph_alpha_tfree_research_vocab_32k_fineweb_steps_370k
drwxr-xr-x 3 root root 4096 May  9  2026 alibaba_nlp_gte_large_en_v1_5
drwxr-xr-x 3 root root 4096 May  9  2026 alibaba_nlp_gte_multilingual_base
drwxr-xr-x 3 root root 4096 May  9  2026 awsteam7052_industrial_design_extreme_material_sdxl_v1_0
drwxr-xr-x 3 root root 4096 May  9  2026 baai_bge_base_en_v1_5
drwxr-xr-x 4 root root 4096 May  9  2026 baai_bge_large_en_v1_5

# 进入某个模型的目录下，即可执行脚本
cd <model_name>
uv run python demo.py（跑推理）
uv run python accuracy_run.py --use-pretrained（精度评测）
uv run python accuracy_run_perf.py --use-pretrained（性能调优）

注：<work_dir>为个人工作目录，<model_name>为某个模型子目录名称
```

4.3.2 配合框架仓使用

如果你需要：

- 统一看板
- 批量分配与编排
- benchmark / optimization / business benchmark 的完整工具链
- board.db 状态管理

- .claude/agents、.claude/skills、.claude/agent-memory 对应的多智能体编排能力

建议配合 SLAI-AscendBridge2 框架仓一起使用。

```
# 执行命令进入1.6.2小节创建的容器
docker exec -it AB2.0 bash

# 执行docker命令返回如下时，说明已进入容器内
root@5078b464d7ef:/workspace# docker
bash: docker: command not found

# 在容器中安装rsync
yum install -y rsync

# 将单模型目录按需同步到框架仓
cd <work_dir>/SLAI-AscendBridge2
rsync -av <work_dir>/SLAI-AscendBridge2-Adaptations/adaptations/<model_name>/ ./adaptations/<name>/

注：<work_dir>为个人工作目录，<model_name>为某个模型子目录名称

# 在框架仓中继续跑 checker、benchmark、optimization 和 dashboard等操作，具体操作方法参见第3章的说明
```