

第六课

基础作业

环境配置

- 1. 创建开发机和 conda 环境
  - 2. 安装 Lagent 和 AgentLego
  - 3 安装其他依赖
  - 4 准备 Tutorial
- A. Lagent Web Demo
- 1. Lagent Web Demo
    - 1.1 使用 LMDeploy 部署
    - 1.2 启动并使用 Lagent Web Demo
  - 2 用 Lagent 自定义工具
    - 2.1 创建工具文件
    - 2.2 获取 API KEY
    - 2.3 体验自定义工具效果

B. AgentLego

- 1. 直接使用 AgentLego
- 2 作为智能体工具使用
  - 2.1 修改相关文件
  - 2.2 使用 LMDeploy 部署
  - 2.3 启动 AgentLego WebUI
  - 2.4 使用 AgentLego WebUI
- 3. 用 AgentLego 自定义工具
  - 3.1 创建工具文件
  - 3.2 注册新工具
  - 3.3 体验自定义工具效果

进阶作业

- 1.完成 AgentLego WebUI 使用（已完成）
- 2.使用 Lagent 或 AgentLego 实现自定义工具并完成调用

大作业选题

- 算法方向
- 应用方向

# 基础作业

完成以下任务，并将实现过程记录截图：

## 环境配置

### 1. 创建开发机和 conda 环境

在创建开发机界面选择镜像为 Cuda12.2-conda，并选择 GPU 为30% A100。

开发机名称	资源配置	状态	工具	操作
 RGB128_AgentLego id: 20240422-d5085e0-40069428	 30% A100 升降配置	 排队中 位于队列第 1	资源监控 操作日志 自定义服务	SSH 连接 进入开发机 停止   删除

进入**开发机**后，配置环境以同时满足 Lagent 和 AgentLego 运行时所需依赖。

在开始配置环境前，创建用于存放 Agent 相关文件的目录

```
mkdir -p /root/agent
```

配置 conda 环境

```
studio-conda -t agent -o pytorch-2.1.2
```

显示下图就是初始化环境成功了。

```
Downloading and Extracting Packages:
Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
| #
# To activate this environment, use
#
#     $ conda activate agent
#
# To deactivate an active environment, use
#
#     $ conda deactivate
/ [2/2] 同步当前conda环境至jupyterlab kernel

Installed kernelspec agent in /root/.local/share/jupyter/kernels/agent
conda环境: agent安装成功!

=====
ALL DONE!
=====
(base) root@intern-studio-40069428:~#
```

## 2. 安装 Lagent 和 AgentLego

Lagent 和 AgentLego 都提供了两种安装方法，一种是通过 pip 直接进行安装，另一种则是从源码进行安装。为了方便使用 Lagent 的 Web Demo 以及 AgentLego 的 WebUI，我们选择直接从源码进行安装。此处附上源码安装的相关帮助文档：

- Lagent: [https://lagent.readthedocs.io/zh-cn/latest/get\\_started/install.html](https://lagent.readthedocs.io/zh-cn/latest/get_started/install.html)
- AgentLego: [https://agentlego.readthedocs.io/zh-cn/latest/get\\_started.html](https://agentlego.readthedocs.io/zh-cn/latest/get_started.html)

可以执行如下命令进行安装：

```
cd /root/agent
conda activate agent
git clone https://gitee.com/internlm/lagent.git
cd lagent && git checkout 581d9fb && pip install -e . && cd ..
git clone https://gitee.com/internlm/agentlego.git
cd agentlego && git checkout 7769e0d && pip install -e . && cd ..
```

```
Collecting rapidfuzz<4.0.0,>=3.0.0 (from thefuzz->agentlego==0.2.0)
  Using cached https://pypi.tuna.tsinghua.edu.cn/packages/47/d8/4b23351344ac5cc90398912485a82001c80619fc67f0a620191fd5862e23/rapidfuzz-3.8.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.4 MB)
Requirement already satisfied: mdurl~=0.1 in /root/.conda/envs/agent/lib/python3.10/site-packages (from markdown-it-py>=2.2.0->rich->agentlego==0.2.0) (0.1.2)
Installing collected packages: addict, tqdm, rapidfuzz, pydantic-core, annotated-types, thefuzz, pydantic, openapi-pydantic, agentlego
  Running setup.py develop for agentlego
Successfully installed addict-2.4.0 agentlego-0.2.0 annotated-types-0.6.0 openapi-pydantic-0.4.0 pydantic-2.7.0 pydantic-core-2.18.1 rapidfuzz-3.8.1 thefuzz-0.22.1 tqdm-4.66.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
(agent) root@intern-studio-40069428:~/agent#
```

### 3 安装其他依赖

在这一步中，我们将会安装其他将要用到的依赖库，如 LMDeploy，可以执行如下命令：

```
conda activate agent
pip install lmdeploy==0.3.0
```

```
Successfully installed accelerate-0.29.3 fastapi-0.110.2 fire-0.6.0 fsspec-2024.3.1 huggingface-hub-0.22.2 importlib-metadata-7.1.0 lmdeploy-0.3.0 mmengine-lite-0.10.3 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-runtime-cu12-12.4.127 nvidia-cuda-toolkit-12.4.127 nvidia-cufile-cu12-1.10.3.5 nvidia-nccl-cu12-2.21.5 peft-0.9.0 pynvml-11.5.0 safetensors-0.4.3 sentencepiece-0.2.0 shortuuid-1.0.13 starlette-0.37.2 termcolor-2.4.0 tokenizers-0.15.2 transformers-4.38.2 uvicorn-0.29.0 yapf-0.40.2 zipp-3.18.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
(agent) root@intern-studio-40069428:~/agent#
```

### 4 准备 Tutorial

由于后续的 Demo 需要用到 tutorial 已经写好的脚本，因此我们需要将 tutorial 通过 git clone 的方法准备好，以备后续使用：

```
cd /root/agent
git clone -b camp2 https://gitee.com/internlm/Tutorial.git
```

```
(agent) root@intern-studio-40069428:~/agent# git clone -b camp2 https://gitee.com/internlm/Tutorial.git
Cloning into 'Tutorial'...
remote: Enumerating objects: 1321, done.
remote: Counting objects: 100% (1321/1321), done.
remote: Compressing objects: 100% (628/628), done.
remote: Total 1321 (delta 663), reused 1306 (delta 648), pack-reused 0
Receiving objects: 100% (1321/1321), 61.55 MiB | 9.07 MiB/s, done.
Resolving deltas: 100% (663/663), done.
Updating files: 100% (157/157), done.
(agent) root@intern-studio-40069428:~/agent#
```

## A. Lagent Web Demo

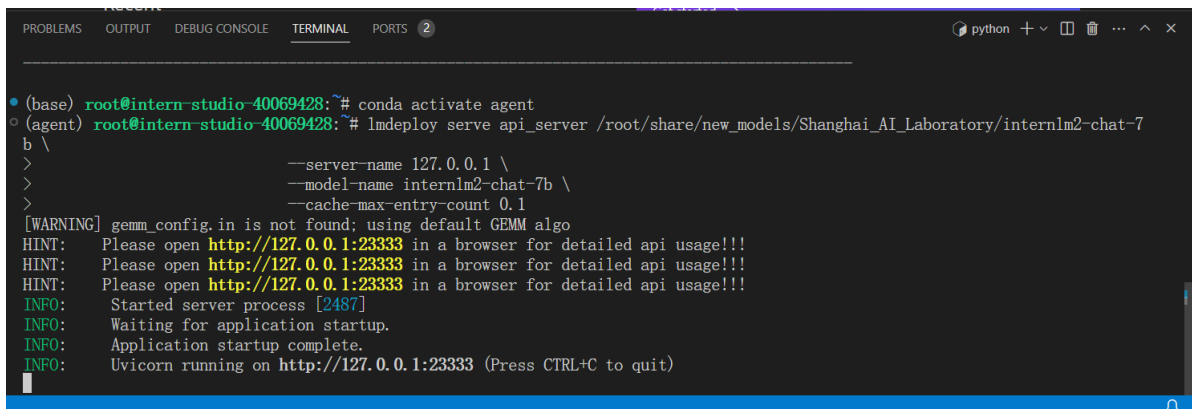
文档可见 [Lagent Web Demo](#)

### 1. Lagent Web Demo

#### 1.1 使用 LMDeploy 部署

在 vscode terminal 中执行如下代码使用 LMDeploy 启动一个 api\_server。

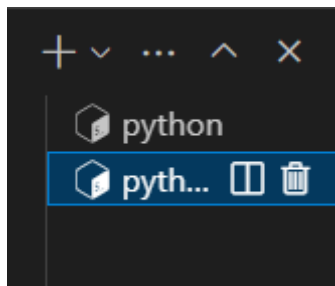
```
conda activate agent
lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
--server-name 127.0.0.1 \
--model-name internlm2-chat-7b \
--cache-max-entry-count 0.1
```



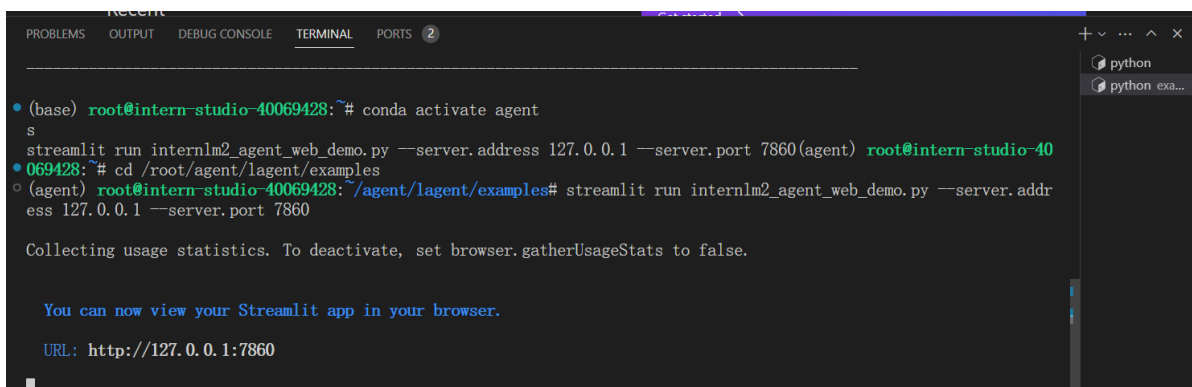
```
(base) root@intern-studio-40069428:~# conda activate agent
(agent) root@intern-studio-40069428:~# lmdeploy serve api_server /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
>
> --server-name 127.0.0.1 \
> --model-name internlm2-chat-7b \
> --cache-max-entry-count 0.1
[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [2487]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)
```

## 1.2 启动并使用 Lagent Web Demo

新建一个 terminal 以启动 Lagent Web Demo



```
conda activate agent
cd /root/agent/lagent/examples
streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --
server.port 7860
```



```
(base) root@intern-studio-40069428:~# conda activate agent
s
streamlit run internlm2_agent_web_demo.py --server.address 127.0.0.1 --server.port 7860(agent) root@intern-studio-40
069428:~# cd /root/agent/lagent/examples
(agent) root@intern-studio-40069428:~/agent/lagent/examples# streamlit run internlm2_agent_web_demo.py --server.addr
ess 127.0.0.1 --server.port 7860

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

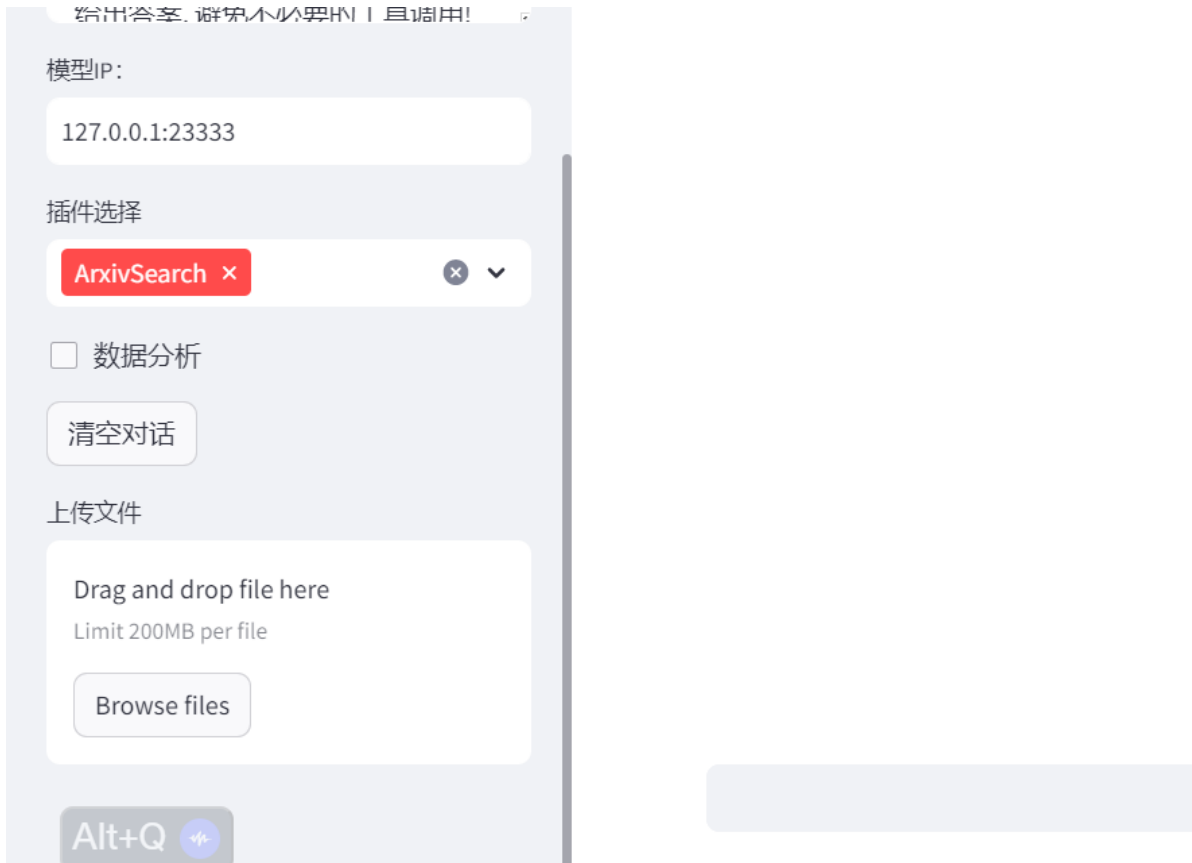
URL: http://127.0.0.1:7860
```

在等待 LMDeploy 的 api\_server 与 Lagent Web Demo 完全启动后，在本地进行端口映射，将 LMDeploy api\_server 的23333端口以及 Lagent Web Demo 的7860端口映射到本地。

```
C:\windows\System32\OpenSSH\ssh -Cng -L 7860:127.0.0.1:7860 -L
23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 44841
```

```
(base) PS C:\Users\LTstatu>
(base) PS C:\Users\LTstatu> C:\Windows\System32\OpenSSH\ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 44841
The authenticity of host '[ssh.intern-ai.org.cn]:44841 ([8.130.47.207]:44841)' can't be established.
ECDSA key fingerprint is SHA256:edNH1U5xEglF1LC5tJcLSbigdQ2pYQjH0D49L6GcSAY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[ssh.intern-ai.org.cn]:44841,[8.130.47.207]:44841' (ECDSA) to the list of known hosts.
```

接下来在本地的浏览器页面中打开 <http://localhost:7860> 以使用 Lagent Web Demo。首先输入模型 IP 为 127.0.0.1:23333，**在输入完成后按下回车键以确认。** 并选择插件为 ArxivSearch，以让模型获得在 arxiv 上搜索论文的能力。



输入“请帮我搜索 InternLM2 Technical Report”以让模型搜索书生·浦语2的技术报告。



## 2 用 Lagent 自定义工具

Lagent 中关于工具部分的介绍文档位于 <https://lagent.readthedocs.io/zh-cn/latest/tutorials/action.html>。使用 Lagent 自定义工具主要分为以下几步：

1. 继承 BaseAction 类
2. 实现简单工具的 run 方法；或者实现工具包内每个子工具的功能

3. 简单工具的 run 方法可选被 tool\_api 装饰；工具包内每个子工具的功能都需要被 tool\_api 装饰

## 2.1 创建工具文件

首先通过 `touch /root/agent/lagent/lagent/actions/weather.py`（大小写敏感）新建工具文件，该文件内容如下：

```
import json
import os
import requests
from typing import Optional, Type

from lagent.actions.base_action import BaseAction, tool_api
from lagent.actions.parser import BaseParser, JsonParser
from lagent.schema import ActionReturn, ActionStatusCode

class WeatherQuery(BaseAction):
    """Weather plugin for querying weather information."""

    def __init__(self,
                 key: Optional[str] = None,
                 description: Optional[dict] = None,
                 parser: Type[BaseParser] = JsonParser,
                 enable: bool = True) -> None:
        super().__init__(description, parser, enable)
        key = os.environ.get('WEATHER_API_KEY', key)
        if key is None:
            raise ValueError(
                'Please set Weather API key either in the environment '
                'as WEATHER_API_KEY or pass it as `key`')
        self.key = key
        self.location_query_url = 'https://geoapi.qweather.com/v2/city/lookup'
        self.weather_query_url = 'https://devapi.qweather.com/v7/weather/now'

    @tool_api
    def run(self, query: str) -> ActionReturn:
        """一个天气查询API。可以根据城市名查询天气信息。

        Args:
            query (:class:`str`): The city name to query.
        """
        tool_return = ActionReturn(type=self.name)
        status_code, response = self._search(query)
        if status_code == -1:
            tool_return.errmsg = response
            tool_return.state = ActionStatusCode.HTTP_ERROR
        elif status_code == 200:
            parsed_res = self._parse_results(response)
            tool_return.result = [dict(type='text', content=str(parsed_res))]
            tool_return.state = ActionStatusCode.SUCCESS
        else:
            tool_return.errmsg = str(status_code)
            tool_return.state = ActionStatusCode.API_ERROR
        return tool_return

    def _parse_results(self, results: dict) -> str:
        """Parse the weather results from QWeather API.
```

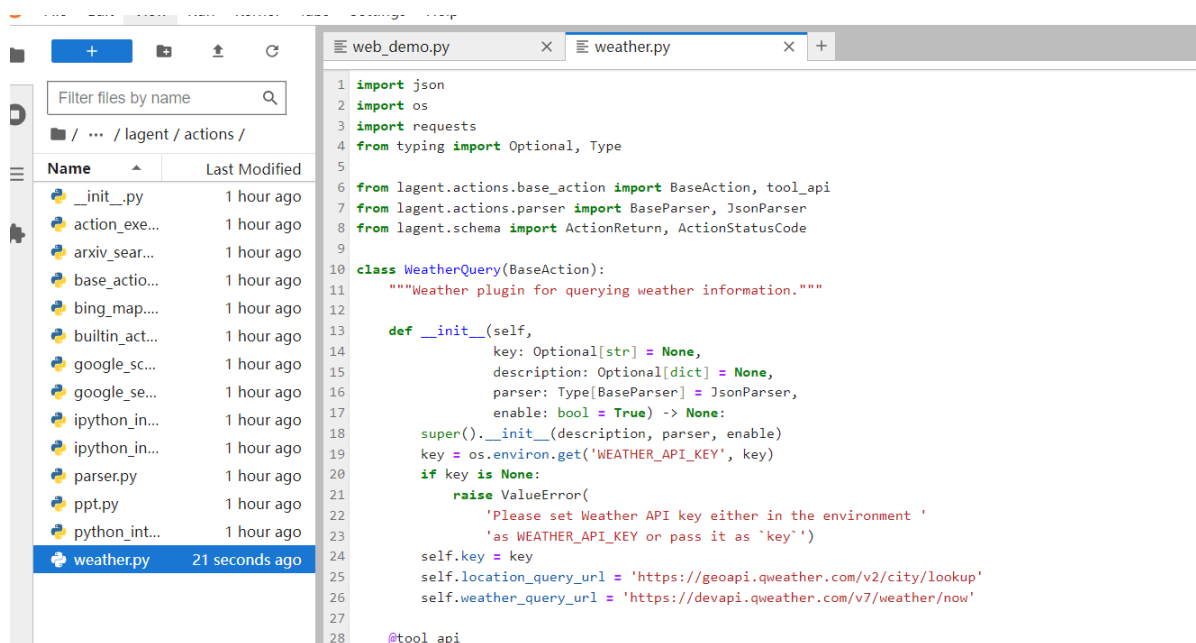
```

Args:
    results (dict): The weather content from Qweather API
    in json format.

Returns:
    str: The parsed weather results.
"""
now = results['now']
data = [
    f'数据观测时间: {now["obsTime"]}',
    f'温度: {now["temp"]}°C',
    f'体感温度: {now["feelsLike"]}°C',
    f'天气: {now["text"]}',
    f'风向: {now["windDir"]}, 角度为 {now["wind360"]}°',
    f'风力等级: {now["windScale"]}, 风速为 {now["windSpeed"]} km/h',
    f'相对湿度: {now["humidity"]}',
    f'当前小时累计降水量: {now["precip"]} mm',
    f'大气压强: {now["pressure"]} 百帕',
    f'能见度: {now["vis"]} km',
]
return '\n'.join(data)

def _search(self, query: str):
    # get city_code
    try:
        city_code_response = requests.get(
            self.location_query_url,
            params={'key': self.key, 'location': query}
        )
    except Exception as e:
        return -1, str(e)
    if city_code_response.status_code != 200:
        return city_code_response.status_code, city_code_response.json()
    city_code_response = city_code_response.json()
    if len(city_code_response['location']) == 0:
        return -1, '未查询到城市'
    city_code = city_code_response['location'][0]['id']
    # get weather
    try:
        weather_response = requests.get(
            self.weather_query_url,
            params={'key': self.key, 'location': city_code}
        )
    except Exception as e:
        return -1, str(e)
    return weather_response.status_code, weather_response.json()

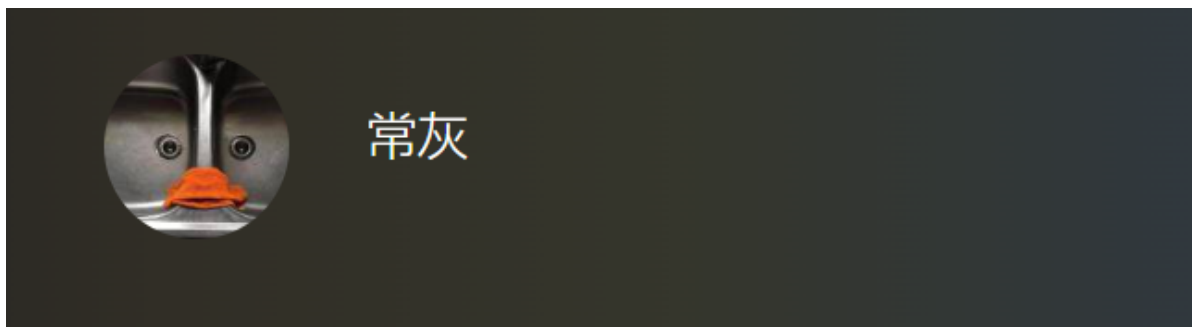
```



## 2.2 获取 API KEY

为了获得稳定的天气查询服务，我们首先要获取 API KEY。首先打开 <https://dev.qweather.com/docs/api/> 后，点击右上角控制台。

创建项目之前要绑定手机号和邮箱

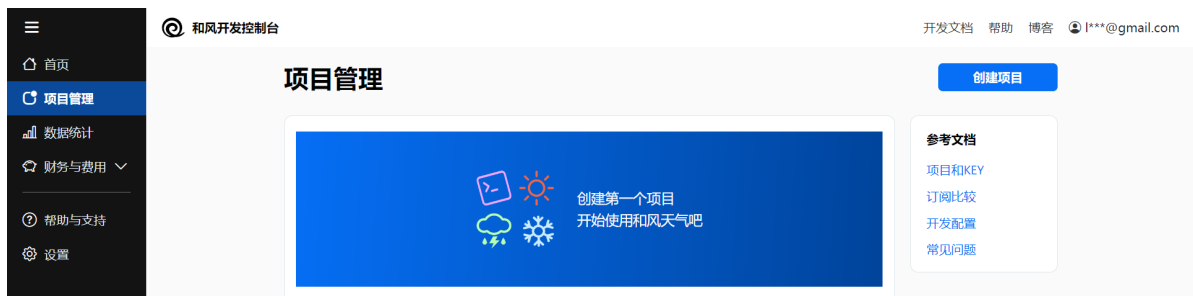




更多和风天气产品的账号设置



进入控制台创建项目



项目名称，免费订阅，**Web API**，key 的名称

RGB128-Lagent13/20

选择订阅

查看订阅文档对比订阅

项目需要绑定一种订阅方案，绑定后不支持更改。

☒ 免费订阅（剩余 1）

☐ 标准订阅（按量计费）

☐ 高性能订阅（联系我们）

• 免费

• 1,000次请求/天

• 仅限免费数据

• 无全球加速

• 按量计费

• 无请求量限制

• 全部数据

• 全球加速

• 全部标准订阅功能

• 更强大的性能和网络

• 自定义API域名

• CDN边缘加速

设置KEY

现在开始设置第一个KEY，你可以稍后创建更多类型的KEY。不同平台的KEY不能混用，即Web API的KEY不能用于SDK获取数据，反之亦然。

适用平台

☒ Web API☐ iOS SDK☐ Android SDK

KEY的名称

weather7/20

复制保存

RGB128-Lagent	免费订阅	添加KEY	编辑	参考文档
KEY名称	Public ID	KEY	类型	操作
weather	HE2404221251131155	<a href="#">查看</a>	Web API	<a href="#">项目和KEY</a> <a href="#">订阅比较</a> <a href="#">开发配置</a> <a href="#">常见问题</a>

## 2.3 体验自定义工具效果

在两个 terminal 中分别启动 LMDeploy 服务和 Tutorial 已经写好的用于这部分的 Web Demo：

注意，确保 1.1 节中的 LMDeploy 服务以及 1.2 节中的 Web Demo 服务已经停止（即 terminal 已关闭），否则会出现 CUDA Out of Memory 或是端口已占用的情况！

```
conda activate agent
lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
--server-name 127.0.0.1 \
--model-name internlm2-chat-7b \
--cache-max-entry-count 0.1
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
- model-name internlm2-chat-7b \
- cache-max-entry-count 0.1(agent) root@intern-studio-40069428:~/agent# lmdeploy serve a
pi_server /root-7b \e/new_models/Shanghai_AI_Laboratory/internlm2-chat
>
>
>
- server-name 127.0.0.1 \
- model-name internlm2-chat-7b \
- cache-max-entry-count 0.1

[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [28391]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)
```

```
export WEATHER_API_KEY=bcf76aab7771431b81bdd005e9cc50c1
# 比如 export WEATHER_API_KEY=1234567890abcdef
conda activate agent
cd /root/agent/Tutorial/agent
streamlit run internlm2_weather_web_demo.py --server.address 127.0.0.1 --
server.port 7860
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
conda activate agent
cd /root/agent/Tutorial/agent
streamlit run internlm2_weather_web_demo.py --se (base) root@intern-studio-40069428:~# # 比如 export WEATHER_API_KEY=
1234567890abcdef
(base) root@intern-studio-40069428:~# conda activate agent
rver.address 127.0.0.1 --server.port 7860(agent) root@intern-studio-40069428:~# cd /root/agent/Tutorial/agent
(agent) root@intern-studio-40069428:~/agent/Tutorial/agent# streamlit run internlm2_weather_web_demo.py --server.add
ress 127.0.0.1 --server.port 7860

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

URL: http://127.0.0.1:7860
```

并在本地执行如下操作以进行端口映射：

```
C:\windows\System32\OpenSSH\ssh -CNg -L 7860:127.0.0.1:7860 -L
23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 44841
```

## 测试效果



## B. AgentLego

文档可见 [直接使用 AgentLego](#)

## 1. 直接使用 AgentLego

首先下载 demo 文件：

```
cd /root/agent
wget http://download.openmmlab.com/agentlego/road.jpg
```

```
(base) root@intern-studio-40069428:~# cd /root/agent
(base) root@intern-studio-40069428:/agent# wget http://download.openmmlab.com/agentlego/road.jpg
--2024-04-23 09:19:44-- http://download.openmmlab.com/agentlego/road.jpg
Resolving proxy.intern-ai.org.cn (proxy.intern-ai.org.cn)... 172.18.128.194
Connecting to proxy.intern-ai.org.cn (proxy.intern-ai.org.cn)|172.18.128.194|:50000... connected.
Proxy request sent, awaiting response... 200 OK
Length: 284676 (278K) [image/jpeg]
Saving to: 'road.jpg'

road.jpg                               100%[=====>] 278.00K  --KB/s  in 0.006s

2024-04-23 09:19:44 (48.7 MB/s) - 'road.jpg' saved [284676/284676]

(base) root@intern-studio-40069428:~/agent#
```

由于 AgentLego 在安装时并不会安装某个特定工具的依赖，因此我们接下来准备安装目标检测工具运行时所需依赖。

AgentLego 所实现的目标检测工具是基于 mmdet (MMDetection) 算法库中的 RTMDet-Large 模型，因此我们首先安装 mim，然后通过 mim 工具来安装 mmdet。

```
conda activate agent
pip install openmim==0.3.9
mim install mmdet==3.3.0
```

在安装完成后，**可能会**观察到以下现象（如下图所示），但请放心，这是正常现象，这并不会影响到我们的使用。

```
Successfully uninstalled requests-2.31.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
urllib 2.1.0 requires requests==2.31.0, but you have requests 2.28.2 which is incompatible.
mymterlab-server 2.26.0 requires requests>2.31, but you have requests 2.28.2 which is incompatible.
Successfully installed aliyun-python-sdk-core-2.15.1 aliyun-python-sdk-kms-2.16.2 crcmod-1.7 cryptography-42.0.5 jmespath-0.10.0 markdown-3.6 model-index-0.1.11 opendatalab-0.0.10 openmim-0.3.9 openlab-0.0.38 ordered-set-4.1.0 oss2-2.17.0 pycryptodome-3.20.0 pytz-2023.4 requests-2.28.2 rich-13.4.2 setuptools-60.2.0 tabulate-0.9.0 tqdm-4.65.2 urllib3-1.26.18
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

然后通过 `touch /root/agent/direct_use.py` 新建 `direct_use.py` 以直接使用目标检测工具，填充 `direct_use.py` 代码

```
import re

import cv2
from agentlego.apis import load_tool

# load tool
tool = load_tool('ObjectDetection', device='cuda')

# apply tool
visualization = tool('/root/agent/road.jpg')
print(visualization)

# visualize
image = cv2.imread('/root/agent/road.jpg')

preds = visualization.split('\n')
pattern = r'(\w+) \((\d+), (\d+), (\d+), (\d+)\)', score (\d+)'

for pred in preds:
    name, x1, y1, x2, y2, score = re.match(pattern, pred).groups()
```

```

x1, y1, x2, y2, score = int(x1), int(y1), int(x2), int(y2), int(score)
cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 1)
cv2.putText(image, f'{name} {score}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 255, 0), 1)

cv2.imwrite('/root/agent/road_detection_direct.jpg', image)

```

```

direct_use.py X
agent > direct_use.py > ...
1  import re
2
3  import cv2
4  from agentlego.apis import load_tool
5
6  # load tool
7  tool = load_tool('ObjectDetection', device='cuda')
8
9  # apply tool
10 visualization = tool('/root/agent/road.jpg')
11 print(visualization)
12
13 # visualize
14 image = cv2.imread('/root/agent/road.jpg')
15
16 preds = visualization.split('\n')
17 pattern = r'(\w+) \((\d+), (\d+), (\d+), (\d+)\)', score (\d+)'
18
19 for pred in preds:
20     name, x1, y1, x2, y2, score = re.match(pattern, pred).groups()
21     x1, y1, x2, y2, score = int(x1), int(y1), int(x2), int(y2), int(score)
22     cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 1)
23     cv2.putText(image, f'{name} {score}', (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 1)
24
25 cv2.imwrite('/root/agent/road_detection_direct.jpg', image)

```

使用第四节的python文件打印agent文件树结构

```
python /root/tree.py /root/agent
```

```

/root/agent
├─ agentlego
│   ├─ agentlego
│   ├─ docs
│   ├─ examples
│   └─ LICENSE
├─ agent
│   ├─ docs
│   ├─ examples
│   ├─ agent
│   └─ LICENSE
├─ Tutorial
│   ├─ assets
│   ├─ agent
│   ├─ helloworld
│   ├─ huixiangdou
│   └─ ...
├─ direct_use.py
└─ road.jpg

```

接下来在执行 `python /root/agent/direct_use.py` 以进行推理。在等待 RTMDet-Large 权重下载并推理完成后，可以看到如下输出以及一张位于 `/root/agent` 名为 `road_detection_direct.jpg` 的图片：



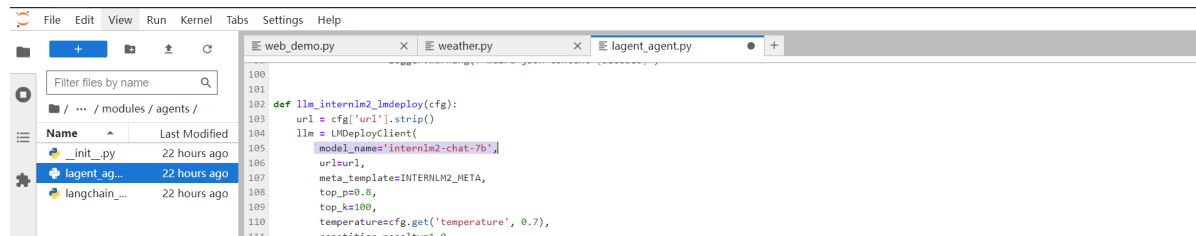




## 2 作为智能体工具使用

### 2.1 修改相关文件

由于 AgentLego 算法库默认使用 InternLM2-Chat-20B 模型，因此我们首先需要修改 /root/agent/agentlego/webui/modules/agents/lagent\_agent.py 文件的第 105行位置，将 internlm2-chat-20b 修改为 internlm2-chat-7b，即



### 2.2 使用 LMDeploy 部署

由于 AgentLego 的 WebUI 需要用到 LMDeploy 所启动的 api\_server，因此我们首先按照下图指示在 vscode terminal 中执行如下代码使用 LMDeploy 启动一个 api\_server。

```
conda activate agent
lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
--server-name 127.0.0.1 \
--model-name internlm2-chat-7b \
--cache-max-entry-count 0.1
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
• (base) root@intern-studio-40069428:~# conda activate agent
ot/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
  --server-name 127.0.0.1 \
  --model-name internlm2-chat-7b \
  --cache-max-entry-count 0.1
◦ (agent) root@intern-studio-40069428:~# lmdeploy serve api_server /root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
>
>
>
Convert to turbomind format: 25% | 8/32 [00:10<00:
Convert to turbomind format: 28% | 9/32 [00:11<00:
Convert to turbomind format: 31% | 10/32 [00:12<00:
Convert to turbomind format: 34% | 11/32 [00:12<00:
Convert to turbomind format: 38% | 12/32 [00:14<00:
Convert to turbomind format: 41% | 13/32 [00:15<00:
Convert to turbomind format: 44% | 14/32 [00:15<00:
Convert to turbomind format: 47% | 15/32 [00:16<00:
Convert to turbomind format: 50% | 16/32 [00:17<00:
Convert to turbomind format: 53% | 17/32 [00:19<00:
```

## 2.3 启动 AgentLego WebUI

新建一个 terminal 以启动 AgentLego WebUI

```
conda activate agent
cd /root/agent/agentlego/webui
python one_click.py
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
5. 将conda环境一键添加到jupyterlab:
lab add {YOUR_CONDA_ENV_NAME}

• (base) root@intern-studio-40069428:~# conda activate agent
i
• python one_click.py(agent) root@intern-studio-40069428:~# cd /root/agent/agentlego/webui
◦ (agent) root@intern-studio-40069428:~/agent/agentlego/webui# python one_click.py
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting gradio>=4.13.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/94/e6/48f65e81840f83aec83a805a60f81cbd60d85de6d2e6d6dc604217fdc6d5/gradio-4.27.0-py3-none-any.whl (17.1 MB)
    17.1/17.1 MB 9.2 MB/s eta 0:00:00
Collecting langchain
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/ed/3e/93045d37eba24e0b5eb05312e30cd9e12805ea5f1ae9ba51ec8a7d2f5372/langchain-0.1.16-py3-none-any.whl (817 kB)
    817.7/817.7 kB 6.2 MB/s eta 0:00:00
Collecting langchain-openai
```

等待 LMDeploy 的 api\_server 与 AgentLego WebUI 完全启动状态

### LMDeploy api\_server

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
Convert to turbomind format: 72% | 23/32 [00:25<00:
Convert to turbomind format: 75% | 24/32 [00:26<00:
Convert to turbomind format: 78% | 25/32 [00:27<00:
Convert to turbomind format: 81% | 26/32 [00:28<00:
Convert to turbomind format: 84% | 27/32 [00:29<00:
Convert to turbomind format: 88% | 28/32 [00:29<00:
Convert to turbomind format: 91% | 29/32 [00:30<00:
Convert to turbomind format: 94% | 30/32 [00:33<00:
Convert to turbomind format: 97% | 31/32 [00:34<00:
Convert to turbomind format: 100% | 32/32 [00:35<00:

[WARNING] gemm_config.in is not found; using default GEMM algo
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
HINT: Please open http://127.0.0.1:23333 in a browser for detailed api usage!!!
INFO: Started server process [58785]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:23333 (Press CTRL+C to quit)
```

### AgentLego WebUI



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 3
Attempting uninstall: requests
Found existing installation: requests 2.28.2
Uninstalling requests-2.28.2:
Successfully uninstalled requests-2.28.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
openlab 0.0.38 requires requests=2.28.2, but you have requests 2.31.0 which is incompatible.
Successfully installed SQLAlchemy-2.0.29 aiofiles-23.2.1 aiohttp-3.9.5 aiosignal-1.3.1 async-timeout-4.0.3 dataclass-es-json-0.6.4 ffmpeg-0.3.2 frozenlist-1.4.1 gradio-4.27.0 gradio-client-0.15.1 greenlet-3.0.3 importlib-resources-6.4.0 jsonpatch-1.33 langchain-0.1.16 langchain-community-0.0.34 langchain-core-0.1.45 langchain-openai-0.1.3 langchain-text-splitters-0.0.1 langsmith-0.1.49 marshmallow-3.21.1 multidict-6.0.5 mypy-extensions-1.0.0 openai-1.23.2 orjson-3.10.1 packaging-23.2 pydub-0.25.1 python-multipart-0.0.9 requests-2.31.0 ruff-0.4.1 semantic-version-2.10.0 shellingham-1.5.4 tomlkit-0.12.0 typer-0.12.3 typing-inspect-0.9.0 urllib3-2.2.1 websockets-11.0.3 yarll-1.9.4
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[09:42:23] INFO Loading tool `Calculator`
Running on local URL: http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 4
(base) root@intern-studio-40069428:~# conda activate agent
i
python one_click.py(agent) root@intern-studio-40069428:~# cd /root/agent/agentlego/webui
(agent) root@intern-studio-40069428:~/agent/agentlego/webui# python one_click.py
[17:08:47] INFO Loading tool `Calculator`
INFO Loading tool `ObjectDetection`
Loads checkpoint by http backend from path: https://download.openmmlab.com/mmdetection/v3.0/rtmdet/rtmdet_l_8xb32-300e_coco/rtmdet_l_8xb32-300e_coco_20220719_112030-5a0be7c4.pth
The model and loaded state dict do not match exactly

unexpected key in source state_dict: data_preprocessor.mean, data_preprocessor.std

04/23 17:10:17 - mmengine - WARNING - Failed to search registry with scope "mmdet" in the "function" registry tree. As a workaround, the current "function" registry in "mmengine" is used to build instance. This may cause unexpected failure when running the built modules. Please check whether "mmdet" is a correct scope, or whether the registry is initialized.
/root/.conda/envs/agent/lib/python3.10/site-packages/mmengine/visualization/visualizer.py:196: UserWarning: Failed to add <class 'mmengine.visualization.vis_backend.LocalVisBackend'>, please provide the `save_dir` argument.
warnings.warn(f'Failed to add {vis_backend.__class__}',
Running on local URL: http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

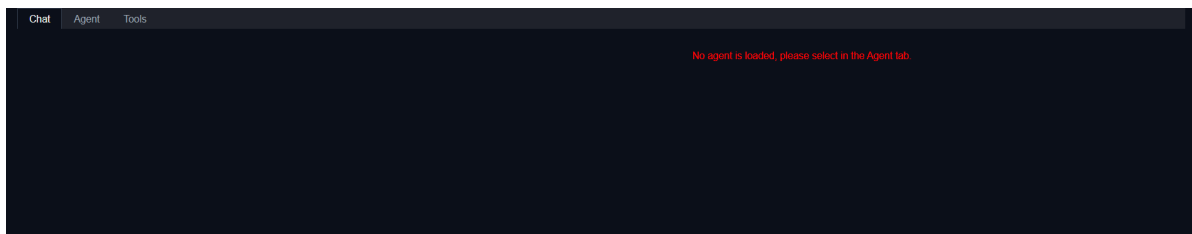
在本地进行端口映射，将 LMDeploy api\_server 的23333端口以及 AgentLego WebUI 的7860端口映射到本地

```
C:\windows\System32\OpenSSH\ssh -CNg -L 7860:127.0.0.1:7860 -L 23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 44841
```

## 2.4 使用 AgentLego WebUI

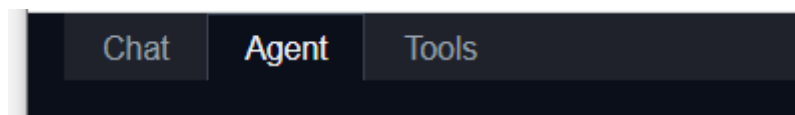
接下来在本地的浏览器页面中打开 <http://localhost:7860> 以使用 AgentLego WebUI。

初始状态



配置 Agent，按照以下步骤。

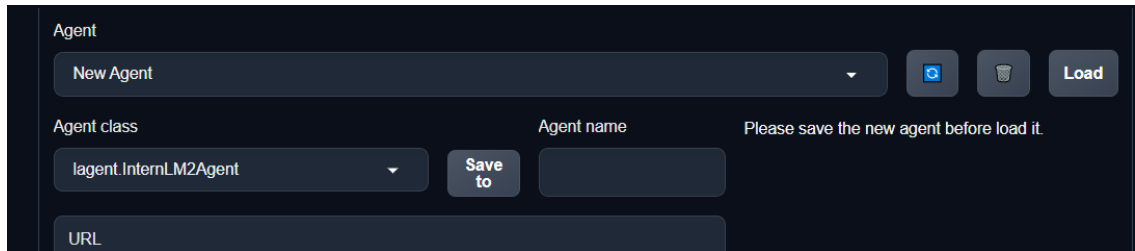
1. 点击上方 Agent 进入 Agent 配置页面。



2. 点击 Agent 下方框，选择 New Agent。

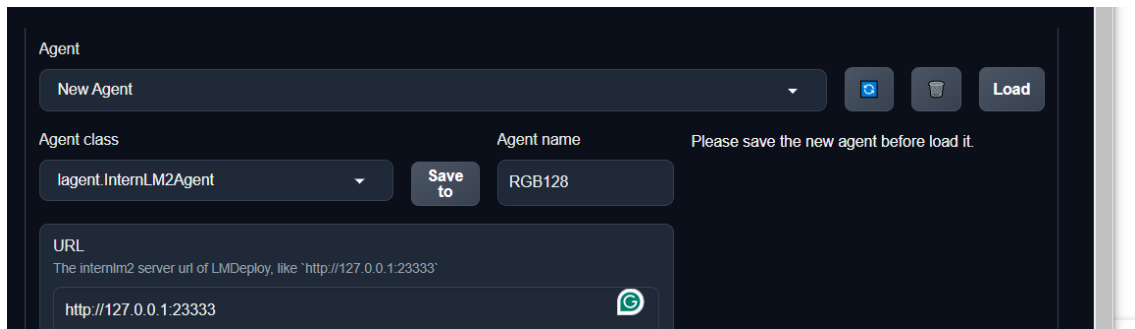


3. 选择 Agent Class 为 `lagent.InternLM2Agent`。



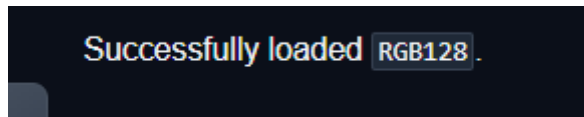
4. 输入模型 URL 为 <http://127.0.0.1:23333>。

5. 输入 Agent name，自定义即可，图中输入了 RGB128



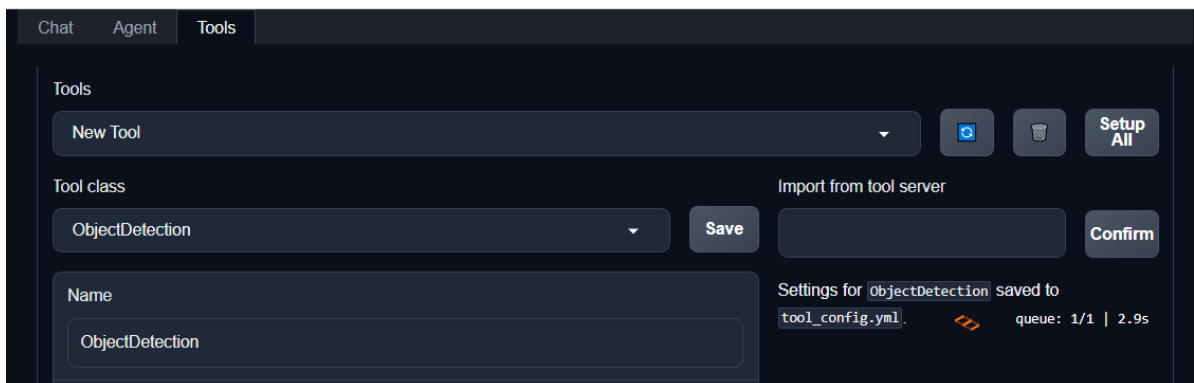
6. 点击 save to 以保存配置，这样在下次使用时只需在第2步时选择 Agent 为 internlm2 后点击 load 以加载就可以了。

7. 点击 load 以加载配置。

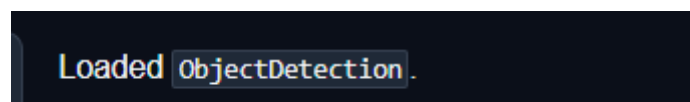


然后配置工具，如下图所示。

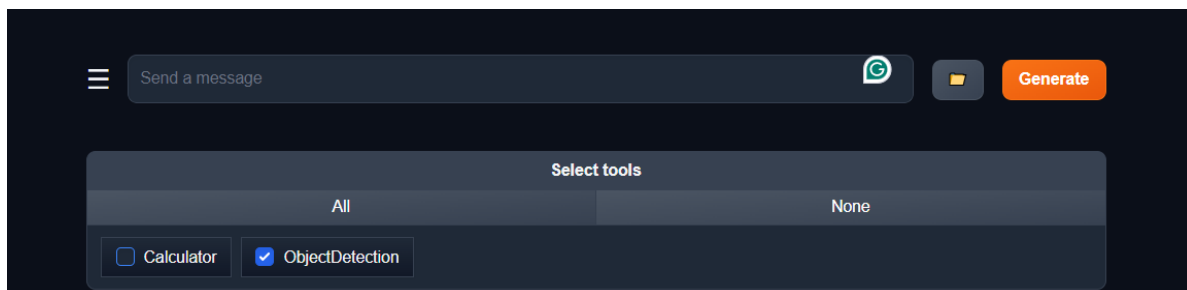
1. 点击上方 Tools 页面进入工具配置页面。（如①所示）
2. 点击 Tools 下方框，选择 New Tool 以加载新工具。（如②所示）
3. 选择 Tool Class 为 `ObjectDetection`。（如③所示）
4. 点击 save 以保存配置。（如④所示）



等待工具加载完成



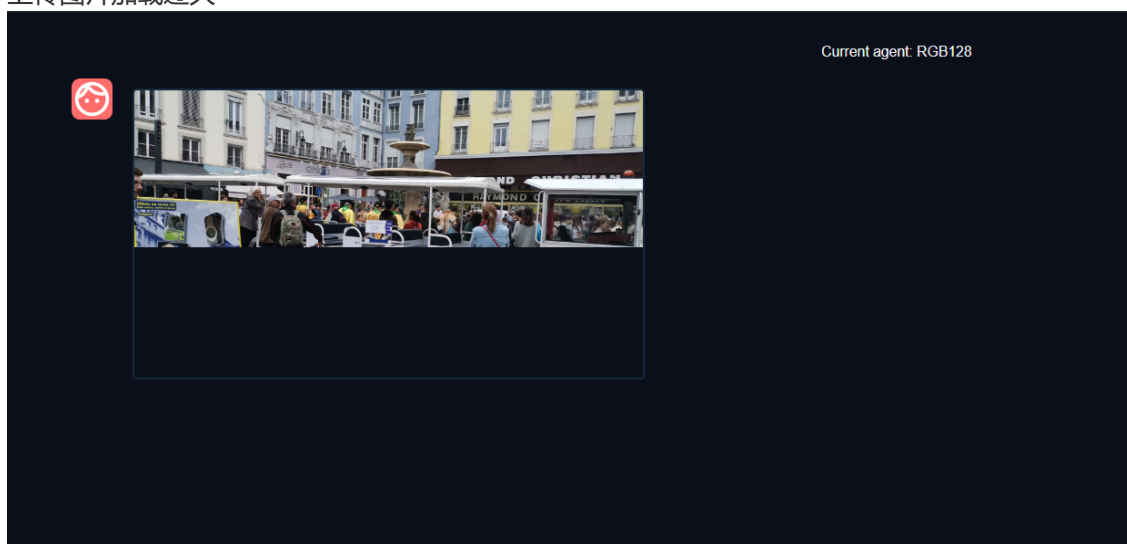
点击上方 Chat 以进入对话页面。在页面下方选择工具部分只选择 ObjectDetection 工具，如下图所示。为了确保调用工具的成功率，请在使用时确保仅有这一个工具启用。



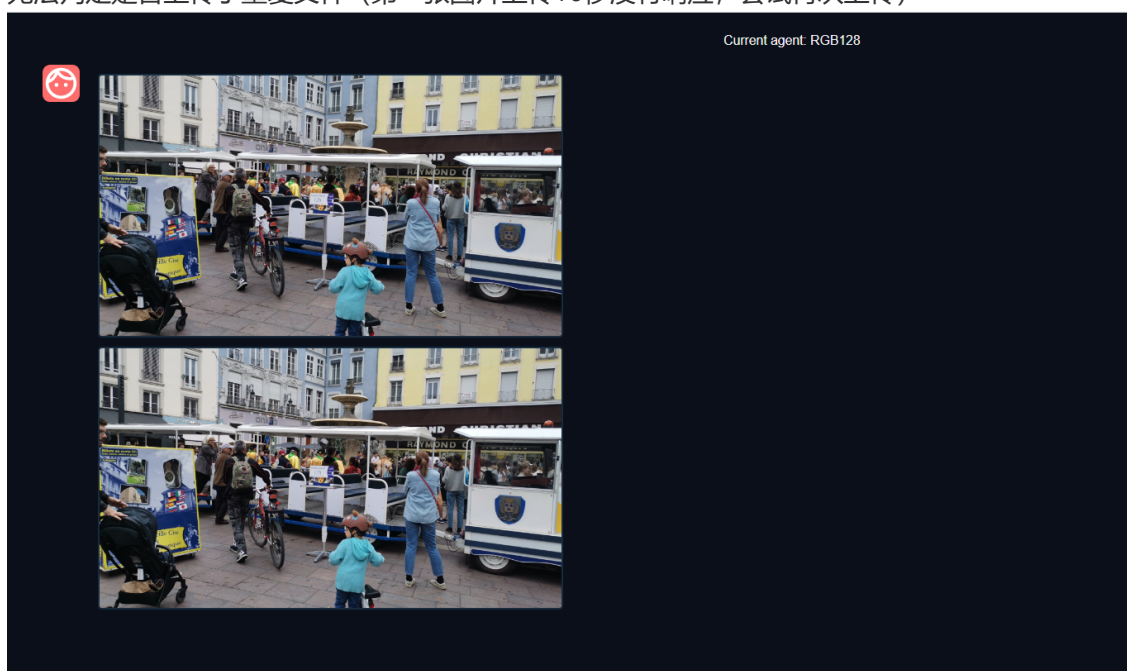
点击右下角文件夹以上传图片，点击 generate 以得到模型回复。如下图所示

观察：

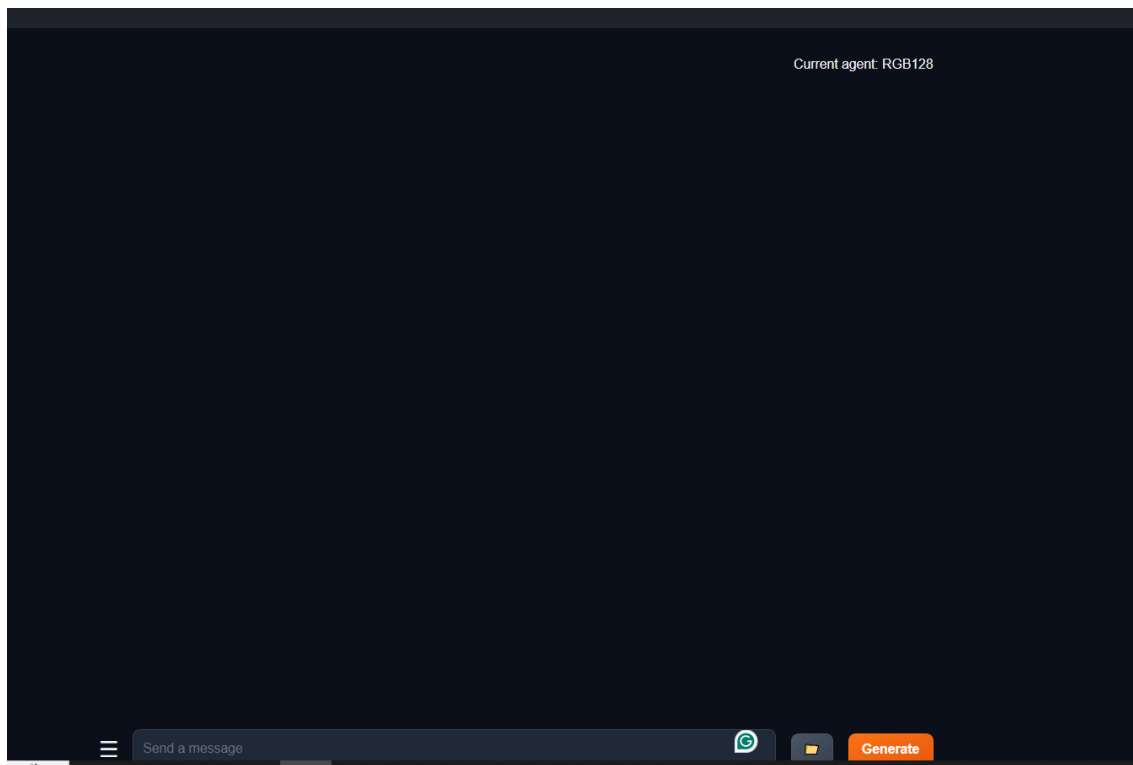
- 刷新响应非常慢
- 刷新后需要重新选择ools配置
- 上传图片加载过久



- 无法判定是否上传了重复文件（第一张图片上传10秒没有响应，尝试再次上传）



- 点击生成按钮无响应（上传图片被清除）



### 3. 用 AgentLego 自定义工具

在本节中，我们将基于 AgentLego 构建自己的自定义工具。AgentLego 在这方面提供了较为详尽的文档，文档地址为 <https://agentlego.readthedocs.io/zh-cn/latest/modules/tool.html>。自定义工具主要分为以下几步：

1. 继承 BaseTool 类
2. 修改 default\_desc 属性（工具功能描述）
3. 如有需要，重载 setup 方法（重型模块延迟加载）
4. 重载 apply 方法（工具功能实现）

其中第一二四步是必须的步骤。下面我们将实现一个调用 MagicMaker 的 API 以实现图像生成的工具。

MagicMaker 是汇聚了优秀 AI 算法成果的免费 AI 视觉素材生成与创作平台。主要提供图像生成、图像编辑和视频生成三大核心功能，全面满足用户在各种应用场景下的视觉素材创作需求。体验更多功能可以访问 <https://magicmaker.openxlab.org.cn/home>。

#### 3.1 创建工具文件

首先通过 `touch /root/agent/agentlego/agentlego/tools/magicmaker_image_generation.py`（大小写敏感）的方法新建工具文件。该文件的内容如下：

```
import json
import requests

import numpy as np

from agentlego.types import Annotated, ImageIO, Info
from agentlego.utils import require
from .base import BaseTool

class MagicMakerImageGeneration(BaseTool):
```

```

default_desc = ('This tool can call the api of magicmaker to '
                'generate an image according to the given keywords.')

styles_option = [
    'dongman', # 动漫
    'guofeng', # 国风
    'xieshi',  # 写实
    'youhua',  # 油画
    'manghe',  # 盲盒
]

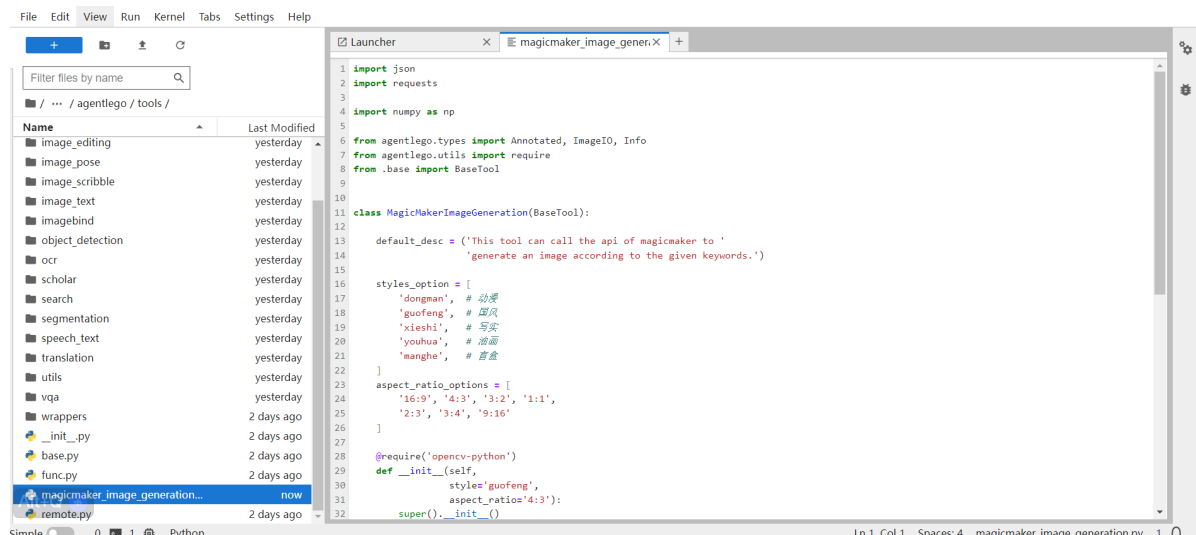
aspect_ratio_options = [
    '16:9', '4:3', '3:2', '1:1',
    '2:3', '3:4', '9:16'
]

@require('opencv-python')
def __init__(self,
              style='guofeng',
              aspect_ratio='4:3'):
    super().__init__()
    if style in self.styles_option:
        self.style = style
    else:
        raise ValueError(f'The style must be one of {self.styles_option}')

    if aspect_ratio in self.aspect_ratio_options:
        self.aspect_ratio = aspect_ratio
    else:
        raise ValueError(f'The aspect ratio must be one of {self.aspect_ratio_options}')

def apply(self,
           keywords: Annotated[str,
                               Info('A series of Chinese keywords separated by
comma.')]
           ) -> ImageIO:
    import cv2
    response = requests.post(
        url='https://magicmaker.openxlab.org.cn/gw/edit-anything/api/v1/bff/sd/generate',
        data=json.dumps({
            "official": True,
            "prompt": keywords,
            "style": self.style,
            "poseT": False,
            "aspectRatio": self.aspect_ratio
        }),
        headers={'content-type': 'application/json'})
    image_url = response.json()['data']['imgUrl']
    image_response = requests.get(image_url)
    image = cv2.imdecode(np.frombuffer(image_response.content, np.uint8),
                        cv2.IMREAD_COLOR)
    return ImageIO(image)

```



## 3.2 注册新工具

接下来修改 `/root/.agentlego/agentlego/tools/init.py` 文件，将我们的工具注册在工具列表中。如下所示，我们将 `MagicMakerImageGeneration` 通过 `from .magicmaker_image_generation import MagicMakerImageGeneration` 导入到了文件中，并且将其加入了 `__all__` 列表中。

```
from .base import BaseTool
from .calculator import Calculator
from .func import make_tool
from .image_canny import CannyTextToImage, ImageToCanny
from .image_depth import DepthTextToImage, ImageToDepth
from .image_editing import ImageExpansion, ImageStylization, ObjectRemove,
ObjectReplace
from .image_pose import HumanBodyPose, HumanFaceLandmark, PoseToImage
from .image_scribble import ImageToScribble, ScribbleTextToImage
from .image_text import ImageDescription, TextToImage
from .imagebind import AudioImageToImage, AudioTextToImage, AudioToImage,
ThermalToImage
from .object_detection import ObjectDetection, TextToBbox
from .ocr import OCR
from .scholar import * # noqa: F401, F403
from .search import BingSearch, GoogleSearch
from .segmentation import SegmentAnything, SegmentObject, SemanticSegmentation
from .speech_text import SpeechToText, TextToSpeech
from .translation import Translation
from .vqa import VQA
+ from .magicmaker_image_generation import MagicMakerImageGeneration

__all__ = [
    'CannyTextToImage', 'ImageToCanny', 'DepthTextToImage', 'ImageToDepth',
    'ImageExpansion', 'ObjectRemove', 'ObjectReplace', 'HumanFaceLandmark',
    'HumanBodyPose', 'PoseToImage', 'ImageToScribble', 'ScribbleTextToImage',
    'ImageDescription', 'TextToImage', 'VQA', 'ObjectDetection', 'TextToBbox',
    'OCR',
    'SegmentObject', 'SegmentAnything', 'SemanticSegmentation',
    'ImageStylization',
    'AudioToImage', 'ThermalToImage', 'AudioImageToImage', 'AudioTextToImage',
    'SpeechToText', 'TextToSpeech', 'Translation', 'GoogleSearch', 'Calculator',
    - 'BaseTool', 'make_tool', 'BingSearch'
    + 'BaseTool', 'make_tool', 'BingSearch', 'MagicMakerImageGeneration'
]
```

```

15 from .segmentation import SegmentAnything, SegmentObject, SemanticSegmentation
16 from .speech_text import SpeechToText, TextToSpeech
17 from .translation import Translation
18 from .vqa import VQA
19 from .magicmaker_image_generation import MagicMakerImageGeneration
20
21 __all__ = [
22     'CannyTextToImage', 'ImageToCanny', 'DepthTextToImage', 'ImageToDepth',
23     'ImageExpansion', 'ObjectRemove', 'ObjectReplace', 'HumanFaceLandmark',
24     'HumanBodyPose', 'PoseToImage', 'ImageToScribble', 'ScribbleTextToImage',
25     'ImageDescription', 'TextToImage', 'VQA', 'ObjectDetection', 'TextToBbox', 'OCR',
26     'SegmentObject', 'SegmentAnything', 'SemanticSegmentation', 'ImageStylization',
27     'AudioToImage', 'ThermalToImage', 'AudioImageToImage', 'AudioTextToImage',
28     'SpeechToText', 'TextToSpeech', 'Translation', 'GoogleSearch', 'Calculator',
29     - 'BaseTool', 'make_tool', 'BingSearch'
30 + 'BaseTool', 'make_tool', 'BingSearch', 'MagicMakerImageGeneration'
31 ]

```

### 3.3 体验自定义工具效果

与2.2, 2.3以及2.4节类似，我们在两个 terminal 中分别启动 LMDeploy 服务和 AgentLego 的 WebUI 以体验我们自定义的工具的效果。

Important

注意，确保 2.2 节中的 LMDeploy 服务以及 2.3 节中的 Web Demo 服务已经停止（即 terminal 已关闭），否则会出现 CUDA Out of Memory 或是端口已占用的情况！

```

conda activate agent
lmdeploy serve api_server
/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-7b \
    --server-name 127.0.0.1 \
    --model-name internlm2-chat-7b \
    --cache-max-entry-count 0.1

```

```

conda activate agent
cd /root/agent/agentlego/webui
python one_click.py

```

并在本地执行如下操作以进行端口映射：

```

C:\Windows\System32\OpenSSH\ssh -CNg -L 7860:127.0.0.1:7860 -L
23333:127.0.0.1:23333 root@ssh.intern-ai.org.cn -p 44841

```

在 Tool 界面选择 MagicMakerImageGeneration 后点击 save 后，回到 Chat 页面选择 MagicMakerImageGeneration 工具后就可以开始使用了。为了确保调用工具的成功率，请在使用时确保仅有这一个工具启用。下图是一个例子。可以看到模型成功地调用了工具并得到了结果。

ChatAgentTools

Tools

MagicMakerImageGeneration

Tools

Tool class

MagicMakerImageGeneration

Save

Import from tool server

Confirm

Name

MagicMakerImageGeneration


Description

This tool can call the api of magicmaker to generate an image according to the given keywords.


☒ Enable


Initialize arguments

Loaded MagicMakerImageGeneration.



Send a message






Generate


Select tools

AllNone

☐ Calculator☐ ObjectDetection☒ MagicMakerImageGeneration

Send a message





Generate


Select tools

AllNone

☐ Calculator☐ ObjectDetection☒ MagicMakerImageGeneration

Past chats

Save



# 进阶作业



## 1.完成 AgentLego WebUI 使用（已完成）

---

文档可见 [AgentLego WebUI](#)

具体步骤详见 B.AgentLego 2.4 小节

## 2.使用 Lagent 或 AgentLego 实现自定义工具并完成调用

---

- [用 Lagent 自定义工具](#)
- [用 AgentLego 自定义工具](#)（已完成）

具体步骤详见 B.AgentLego 3.3小节

## 大作业选题

---

### 算法方向

---

1. 在 Lagent 或 AgentLego 中实现 RAG 工具，实现智能体与知识库的交互。
2. 基于 Lagent 或 AgentLego 实现工具的多轮调用，完成复杂任务。如：智能体调用翻译工具，再用搜索工具，最后调用生成工具，完成一个完整的任务。
3. ...

### 应用方向

---

1. 基于 Lagent 或 AgentLego 实现一个客服智能体，帮助用户解决问题。
2. 基于 Lagent 或 AgentLego 实现一个智能体，实现艺术创作，如生成图片、视频、音乐等。
3. ...