

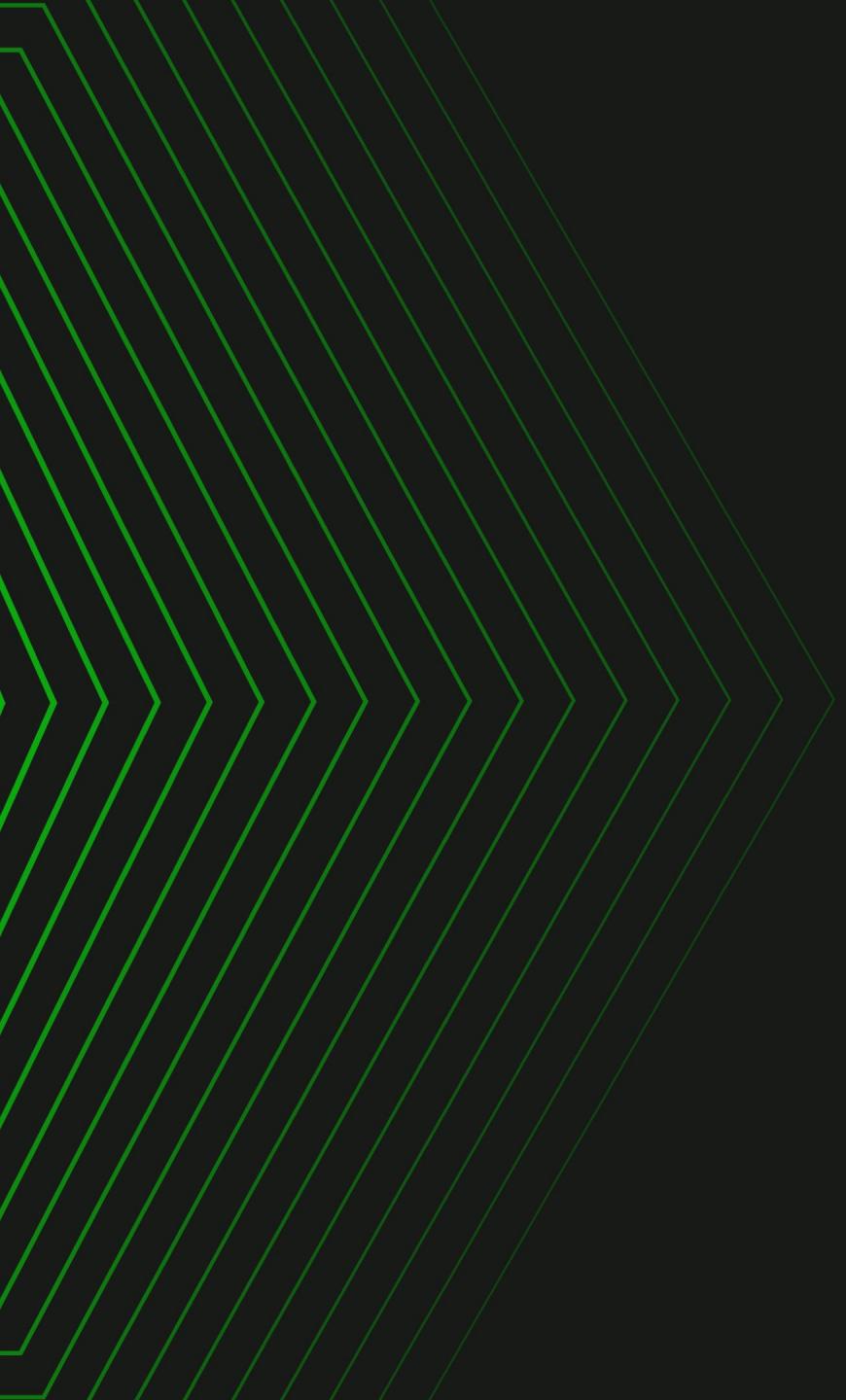
# 大语言模型部署技术概述与选型

NDX - 卢传佳



# Content 目录

1. 写在前面
2. 本地大模型推理体验矩阵
3. 服务端大模型体验矩阵
4. 大规模推理的关键技术点



# Part 01

## 写在前面

## ■ 写在前面的话

大模型狂飙时代，你还在只做“调包侠”吗？

模型周周新，只调 API 懂原理吗？

断网能不能跑？数据敢不敢传？

“由于速度太快，我们是否错过了引擎的轰鸣？”



# ■ 文本生成：Ollama



<https://ollama.com/search>

Search models

Cloud Embedding Vision Tools Thinking Popular

**gpt-oss**  
OpenAI's open-weight models designed for powerful reasoning, agentic tasks, and versatile developer use cases.  
tools thinking cloud 20b 120b  
4.7M Pulls 5 Tags Updated 1 month ago

**qwen3-vl**  
The most powerful vision-language model in the Qwen model family to date.  
vision tools cloud 2b 4b 8b 30b 32b 235b  
498.3K Pulls 59 Tags Updated 3 weeks ago

**deepseek-r1**  
DeepSeek-R1 is a family of open reasoning models with performance approaching that of leading models, such as O3 and Gemini 2.5 Pro.  
tools thinking 1.5b 7b 8b 14b 32b 70b 671b  
72.2M Pulls 35 Tags Updated 4 months ago

**qwen3-coder**  
Alibaba's performant long context models for agentic and coding tasks.  
tools cloud 30b 480b  
800.4K Pulls 10 Tags Updated 2 months ago

快捷体验本地模型

```
Ollama

# pull model at first
ollama pull gpt-oss:20b

# run model
ollama run gpt-oss:20b

# as backend service
ollama serve # or gui tool
> run at http://127.0.0.1:11434
```

使用 Modelfile 自定义

```
Ollama Modelfile

cat > Modelfile << EOF
FROM mistral:7b
SYSTEM """
You answer like a senior cloud-native architect.
"""

PARAMETER temperature 0.2
ADAPTER <path to safetensor adapter> # LORA
EOF

ollama create arch-assistant -f Modelfile
```

# ■ LM Studio

<https://lmstudio.ai/models>

The screenshot shows the LM Studio Model Catalog interface. It displays a list of local models that can be run on your own machine. The models listed are:

- Olmo 3**: 7B, 7B, 32B. Status: Updated 3 days ago.
- olmOCR 2**: 7B. Status: Updated 5 days ago.
- minimax-m2**: 230B. Status: Updated 19 days ago.
- gpt-oss-safeguard**: 20B, 120B. Status: Updated 26 days ago.
- Qwen3-VL**: 2B, 4B, 8B, 30B, 32B. Status: Updated 28 days ago.

Each model entry includes its name, size variants, and a status indicating when it was last updated.

The screenshot shows the LM Studio developer interface. The top bar indicates the application is running and shows the reachable address as <http://127.0.0.1:1234>. The developer logs pane shows the following log entries:

```
2025-11-25 12:03:37 [INFO] [Plugin(lmstudio/js-code-sandbox)] stdout: [Tools Prvdr.] Register with LM Studio
2025-11-25 12:03:37 [INFO] [Plugin(lmstudio/rag-v1)] stdout: [PromptPreprocessor] Register with LM Studio
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] Success! HTTP server listening on port 1234
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] Supported endpoints:
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] -> GET http://localhost:1234/v1/models
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] -> POST http://localhost:1234/v1/responses
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] -> POST http://localhost:1234/v1/chat/completions
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] -> POST http://localhost:1234/v1/completions
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] -> POST http://localhost:1234/v1/embeddings
2025-11-25 12:04:17 [INFO] [LM STUDIO SERVER] Logs are saved into /Users/samzonglu/.cache/lm-studio/server-logs
2025-11-25 12:04:17 [INFO] Server started.
2025-11-25 12:04:17 [INFO] Just-in-time model loading inactive.
```

The screenshot shows the LM Studio CLI interface. It provides usage instructions and a list of command options:

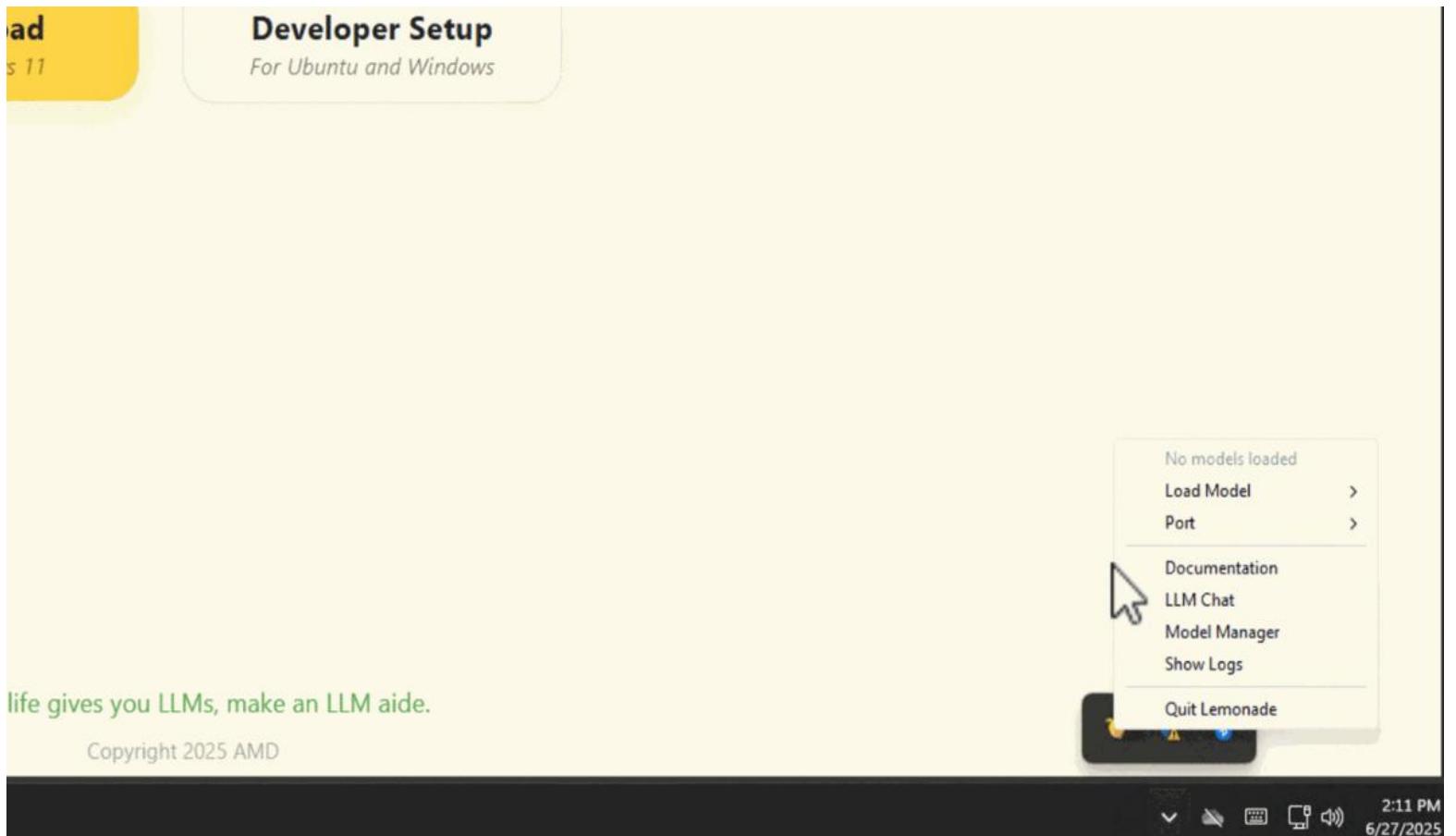
- Usage:** Usage: lms [options] [command]
- LM Studio CLI**
- Options:**
  - h, --help display help for command
- Manage Models:**
  - get Searching and downloading a model from online.
  - import Import a model file into LM Studio
  - ls List all downloaded models
- Use Models:**
  - chat Open an interactive chat with the currently loaded model.
  - load 载入一个模型 Load a model
  - ps List all loaded models
  - server Commands for managing the local server
  - unload 卸载一个模型 Unload a model
- Develop & Publish Artifacts:**
  - clone 克隆一个 artifact from LM Studio Hub to a local folder.
  - create Create a new project with scaffolding
  - dev Starts the development server for the plugin in the current folder.
  - login Authenticate with LM Studio
  - push Uploads the plugin in the current folder to LM Studio Hub.
- System Management:**
  - bootstrap Bootstrap the CLI
  - flags Set or get experiment flags
  - log Log operations. Currently only supports streaming logs from LM Studio via 'lms log stream'
- Commands:**
  - runtime Manage runtime engines
  - status Prints the status of LM Studio
  - version Prints the version of the CLI

# lemonade

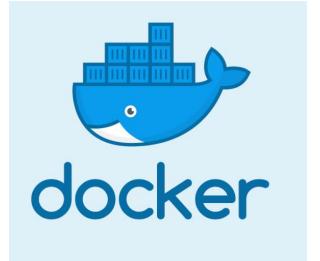
<https://github.com/lemonade-sdk/lemonade>

for Windows and MacOS

- ✓ **自动侦测 CPU/GPU/NPU 多后端自动切换**,  
CPU 用 OGA, GPU 用 llama.cpp Vulkan 或  
ROCM, 苹果用 Metal, AMD NPU 用 FLM。
- ✓ **模型管理**: 内置图形化管理页面, 可直接拉取  
HuggingFace 模型, 支持 GGUF 与 ONNX,  
界面比很多“命令行爱好者作品”友好。
- ✓ **OpenAI API 兼容**: 本地起个  
`http://localhost:8000/api/v1` 就能骗过各种客  
户端, 以为连的是 OpenAI。
- ✓ **跨平台支持广**: Windows、Linux、macOS 全覆  
盖, 还支持 AMD NPU (Ryzen AI)。
- ✓ **CLI 工具完善**: `lemonade-server run|pull|list`  
指令简单粗暴, 适合喜欢命令行又不想翻文档的  
人。
- ✓ **模型生态接入丰富**: 官方列了能接 OpenWebUI、  
Continue、AnythingLLM、GAIA、AI-Dev-  
Gallery 等生态。
- ✓ **支持分体式推理、KV 缓存**: 官方路线图里明确持  
续做推理加速, 不只是包装一层 GUI。
- ✓ **安装门槛低**: Windows 给 MSI, 其他平台给 pip  
或源码路径。比某些“编译三小时”的开源推  
理库友善。



# Docker Model Runner



1. 像拉镜像一样拉模型 docker model pull ai/my-llama:latest
2. 本地 serve 模型，跑起来之后就直接提供 OpenAI 风格的 REST API，
3. 把 GGUF 打包成 OCI Artifact，本地有 GGUF，能直接 package 并 push，
4. 支持 Docker Desktop GUI 管理，有图形界面可视化模型列表、日志、交互记录，

The screenshot shows the Docker Desktop interface with the 'Models' section selected. On the left sidebar, 'Models' is highlighted under the 'Docker Hub' category. The main area displays a grid of six models:

- qwen3-reranker-vlm: Multilingual reranking model for text retrieval, scoring document relevance across 119 languages. Pull count: 371.
- qwen3-reranker: Multilingual reranking model for text retrieval, scoring document relevance across 119 languages. Pull count: 159.
- snowflake-arctic-embed-l-v2-vlm: Snowflake's Arctic-Embed v2.0 boosts multilingual retrieval and efficiency. Pull count: 1.9K.
- qwen3-embedding-vlm: Qwen3 Embedding: multilingual models for advanced text/ranking tasks like retrieval & clustering. Pull count: 4.5K.
- qwen3-embedding: Qwen3 Embedding: multilingual models for advanced text/ranking tasks like retrieval & clustering. Pull count: 4.9K.
- gpt-oss: OpenAI's open-weight models designed for powerful reasoning, agentic tasks. Pull count: 100K+.

At the bottom, there are status indicators: 'Engine running', 'RAM 2.33 GB CPU 0.50%', 'Disk: 97.06 GB used (limit 329.70 GB)', and a version 'v4.51.0'.

把模型构建为 OCI 镜像

The terminal window shows the usage of Docker Model Runner commands:

```
# Download a model file in GGUF format, for example from HuggingFace
curl -L -o model.gguf https://huggingface.co/TheBloke/Mistral-7B-v0.1-GGUF/resolve/main/mistral-7b-v0.1.Q4_K_M.gguf

# Package it as OCI Artifact and push it to Docker Hub
docker model package --gguf "$(pwd)/model.gguf" --push myorg/mistral-7b-v0.1:Q4_K_M
```

Usage: docker model COMMAND

Docker Model Runner

Commands:

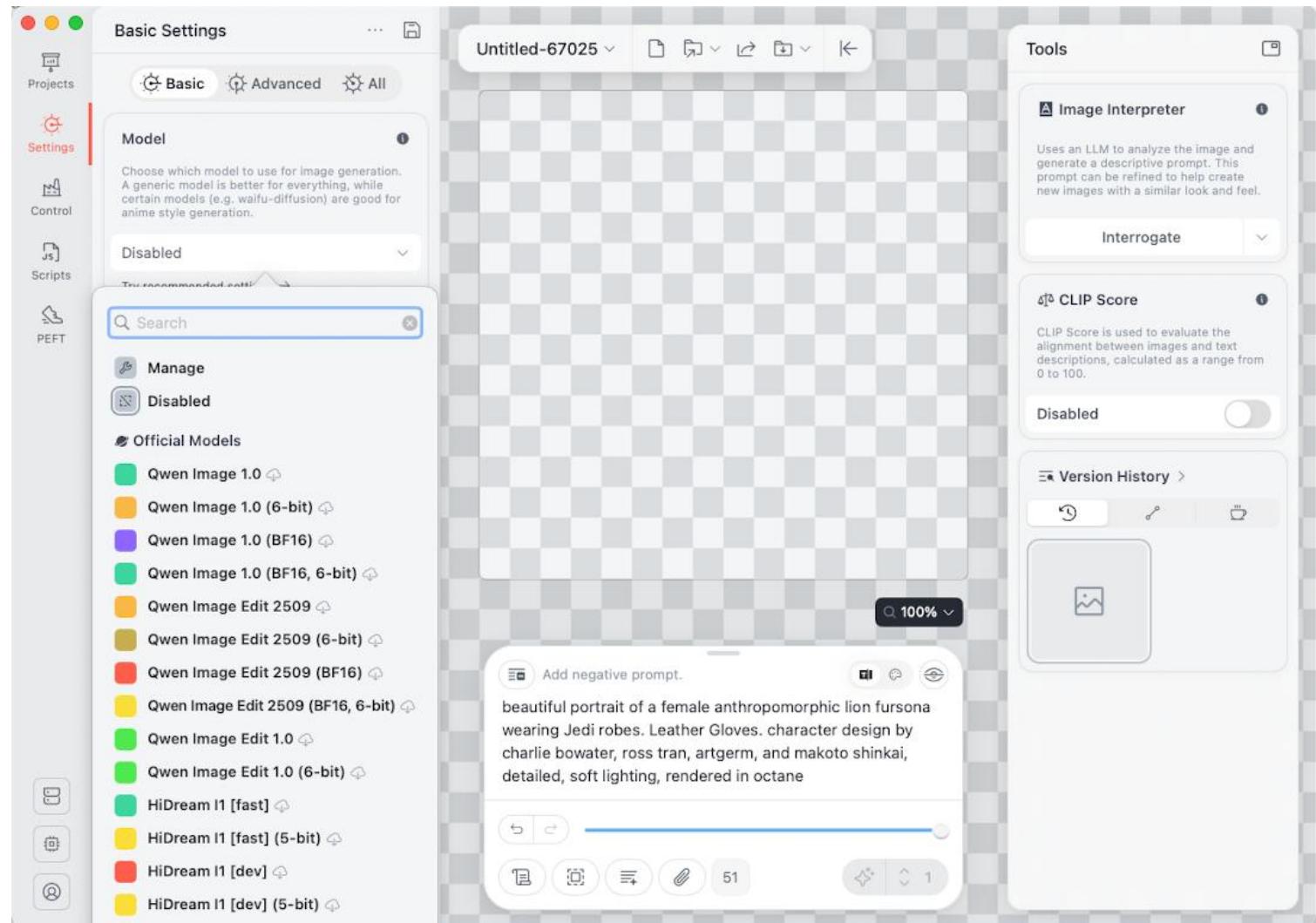
df	Show Docker Model Runner disk usage
inspect	Display detailed information on one model
install-runner	Install Docker Model Runner (Docker Engine only)
list	List the models pulled to your local environment
logs	Fetch the Docker Model Runner logs
package	Package a GGUF file, Safetensors directory, or existing model into a Docker model OCI artifact.
ps	List running models
pull	Pull a model from Docker Hub or HuggingFace to your local environment
push	Push a model to Docker Hub
reinstall-runner	Reinstall Docker Model Runner (Docker Engine only)
requests	Fetch requests+responses from Docker Model Runner
restart-runner	Restart Docker Model Runner (Docker Engine only)
rm	Remove local models downloaded from Docker Hub
run	Run a model and interact with it using a submitted prompt or chat mode
start-runner	Start Docker Model Runner (Docker Engine only)
status	Check if the Docker Model Runner is running
stop-runner	Stop Docker Model Runner (Docker Engine only)
tag	Tag a model
uninstall-runner	Uninstall Docker Model Runner (Docker Engine only)
unload	Unload running models
version	Show the Docker Model Runner version

Run 'docker model COMMAND --help' for more information on a command.

# ■ 图片生成：Draw things

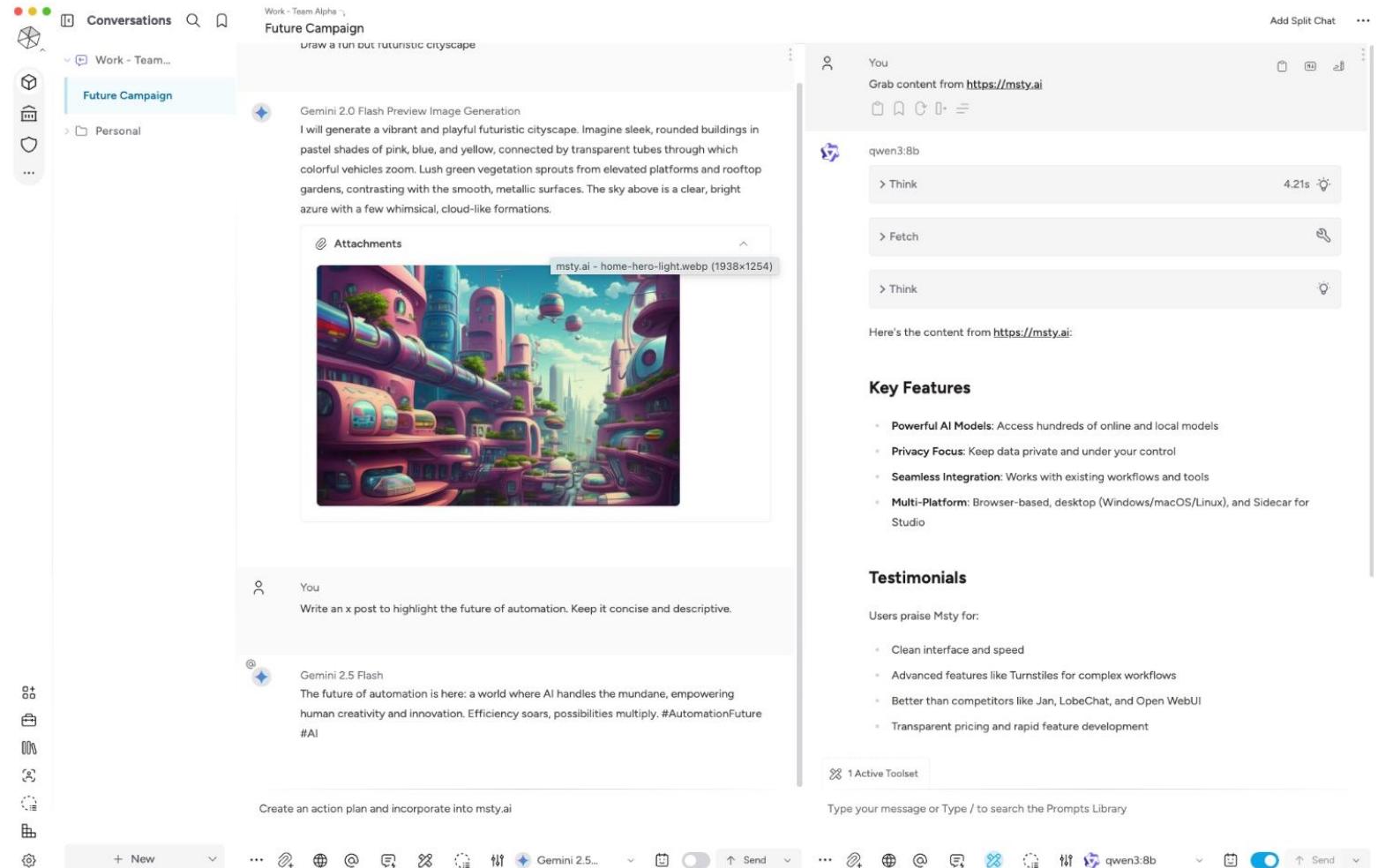
<https://drawthings.ai>

- ✓ 本地离线推理，隐私不外泄
- ✓ 支持多类型模型，含文本生图、LoRA、ControlNet
- ✓ 支持 LoRA 本地训练
- ✓ 无限画布，任意填充补图
- ✓ 图转视频，直接生成动效
- ✓ 多风格滤镜与图像重绘
- ✓ iPhone、iPad、Mac 全平台原生支持
- ✓ 云端 compute 可选，用于加速或大模型
- ✓ 免费版可本地跑，社区版/高级版提升算力与云特性



# Msty

- ✓ 本地优先，支持本地模型运行，零遥测，数据不出设备
- ✓ 全端形态，桌面应用、Web 版统一界面，
- ✓ 大模型集成广，在线模型、本地模型、Ollama、厂商 API 都能接
- ✓ 知识库增强，Stacks 做持久上下文、多层结构、语义检索、自动 PII 清洗
- ✓ MCP 工具体系，可挂外部 API、数据库、脚本，实时调用
- ✓ Personas，定义助手行为、能力、工具权限
- ✓ 工作流，Turnstiles 与自动化执行复杂多步任务
- ✓ 分屏与多会话，适合比较推理、并行任务，
- ✓ 隐私默认开启，不需要账号、不开日志
- ✓ 性能取向，对大模型推理场景的 UI、交互、工作流做了优化



# ComfyUI

<https://github.com/comfyanonymous/ComfyUI>

基于节点的可视化工作流。

支持主流文生图、图生图、修图、视频、音频、3D 模型。

大模型兼容广，SD1.x 到 SD3、Flux、Hunyuan、Wan、LTX 等。

智能缓存与部分重算，推理加速明显。

超低显存运行策略，1GB 也能跑。

完全离线，可手动管理模型路径。

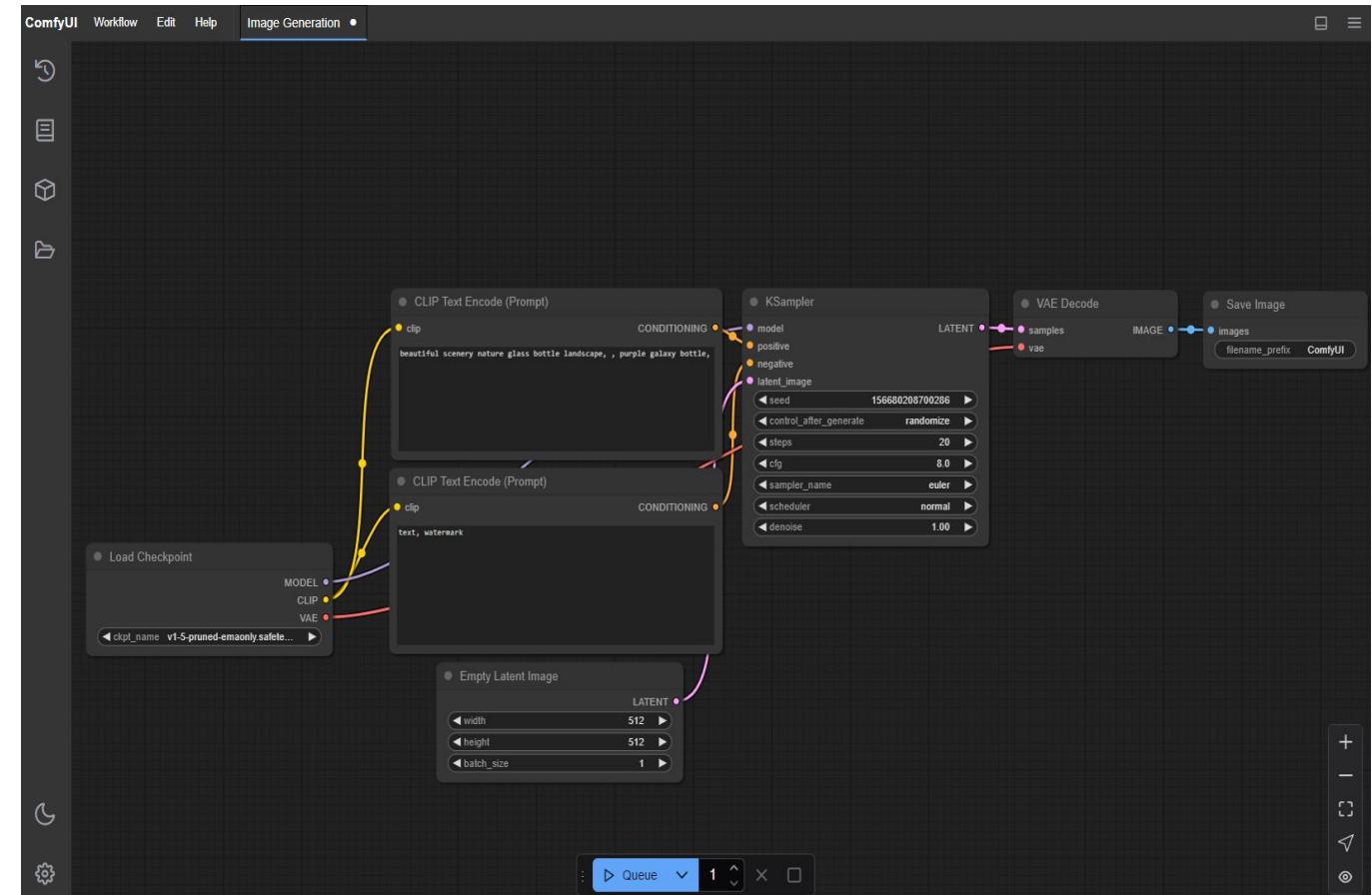
跨平台，Windows macOS Linux 都能跑。

支持 API 节点，可接入外部商用模型。

高级玩法丰富，如 ControlNet、LoRA、Inpaint、Upscale、

Area Composition。

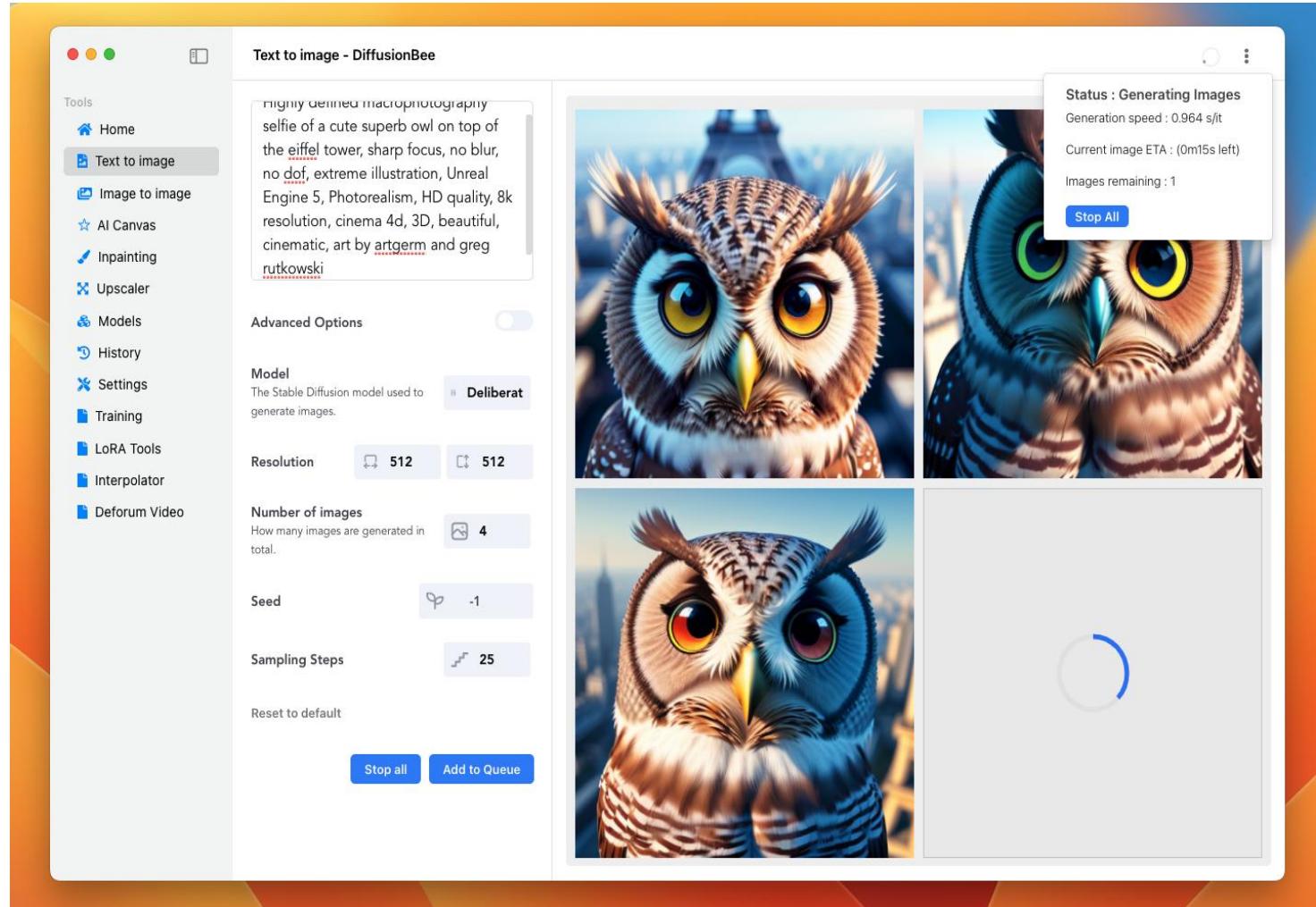
工作流可导入导出，支持从 PNG/WebP 反解析完整流程。



# diffusionbee-stable-diffusion-ui

<https://github.com/divamgupta/diffusionbee-stable-diffusion-ui>

- ✓ 本地运行 Stable Diffusion，无需云端，隐私可控
- ✓ 一键安装，零依赖，适合小白
- ✓ 支持 SD 1.x、2.x、SDXL、Inpainting、ControlNet、LoRA
- ✓ 文生图、图生图、局部重绘、扩图等常用生成能力
- ✓ 内置模型下载与管理
- ✓ 历史记录、放大、尺寸可选
- ✓ 针对 M1/M2 优化，推理速度较快



# ■ 工具分享：ai-icon-generator

<https://github.com/samzong/ai-icon-generator> 利用大模型 API 实现图片 logo 生成

The image shows two screenshots of the AI Icon Generator tool. The left screenshot displays the main interface with a dark theme. It features a text input field for describing the icon ('Describe your icon'), a 'Get Suggestion' button, and a 'Generate' button. Below the input field are several category buttons: Technology (Simple cloud computing icon, Modern database icon, Elegant leaf icon, Simple mountain icon), Business (Professional chart icon, Clean document icon, Chat bubble icon, User avatar icon), and Social. At the bottom are style buttons: Flat (selected), Line, Solid, Gradient, Windows, and Fluent. An 'Icon preview area' is at the bottom. The right screenshot shows an 'API Configuration' dialog box. It includes a 'Security Notice' section stating that the API key is stored locally in the browser and not sent to other servers. The 'Service Provider' dropdown is set to 'console.d.run'. Under 'console.d.run', there are fields for 'API Key' (containing 'sk-...'), 'Base URL' (set to 'https://chat.d.run'), and 'Model' (set to 'public/hidream-i1-dev'). There are 'Cancel' and 'Save' buttons at the bottom.

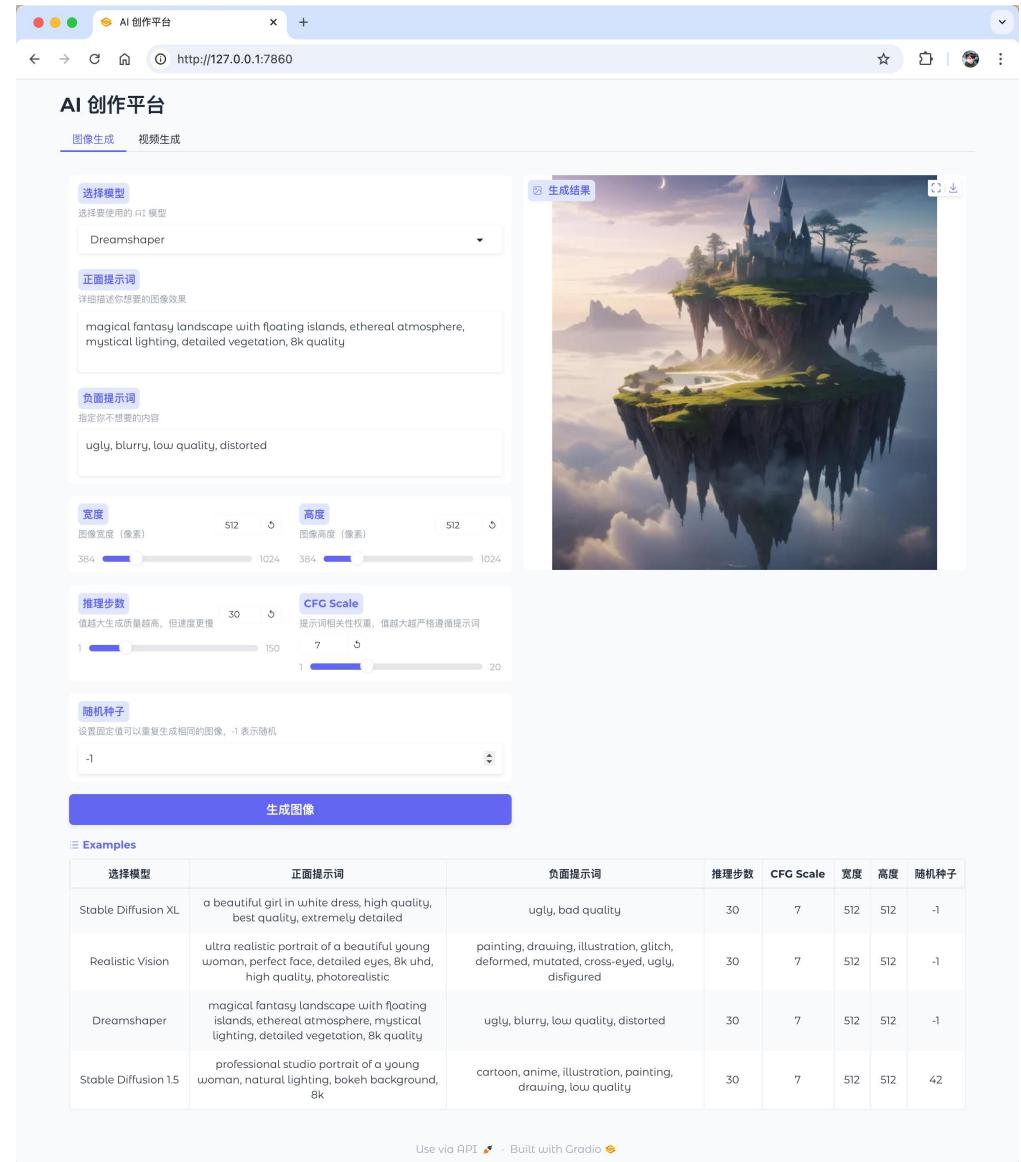
- ✓ 基于 gpt-image-1 做图标生成，描述一句话就能出图。
- ✓ 多风格，可控，可批量。
- ✓ 支持 PNG、ICO、ICNS、JPEG 导出。
- ✓ 中英双语界面，自动切语言。
- ✓ 本地保存生成历史，缓存避免重复请求。
- ✓ 可配自定义 API Endpoint，兼容第三方代理。
- ✓ 自带速率限制与用量控制。
- ✓ 响应式 UI，实时预览，暗黑/亮色主题。

# sd-chat

<https://github.com/samzong/sd-chat>

一个基于 Stable Diffusion 的 AI 创作平台，支持图像生成和视频生成功能，本项目集成了多个先进的 AI 模型，提供简单易用的 Web 界面，让用户可以轻松创作高质量的 AI 图像和视频，

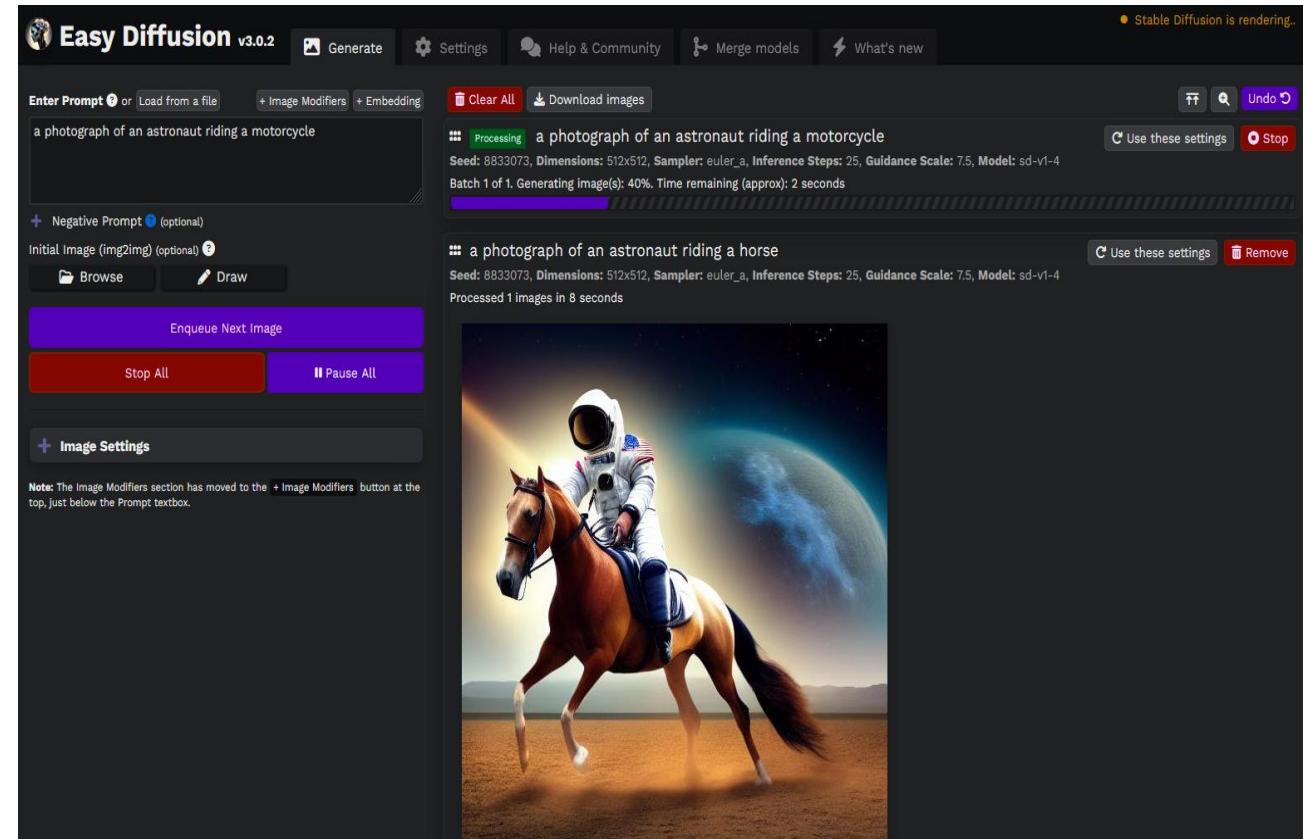
```
1. 安装 Conda (如已安装可跳过) :  
```bash  
# 下载 Miniforge3 (适用于 Apple Silicon)  
curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-MacOSX-arm64.sh"  
bash Miniforge3-MacOSX-arm64.sh  
```  
  
2. 创建并激活环境:  
```bash  
conda create -n diffusers python=3.12  
conda activate diffusers  
```  
  
3. 安装 PyTorch (针对 MPS 加速) if Mac:  
```bash  
conda install -y pytorch torchvision torchaudio -c pytorch-nightly  
```  
  
4. 安装依赖:  
```bash  
conda install -y environment.yml  
```  
  
5. 启动服务:  
```bash  
python run.py # 首次运行时会自动下载所需模型, 这可能需要一些时间, 具体取决于网络状况。  
```
```



# ■ 大模型推理很吸引人，想玩怎么办？

<https://github.com/easydiffusion/easydiffusion>

- ✓ 一键安装，零依赖，本地直接跑
- ✓ 支持文本生图、图生图、Inpainting
- ✓ 内置任务队列，批量生成
- ✓ 自动模型识别，免手动配置
- ✓ 支持 SD1.5、SD2.1、SDXL、Flux Beta
- ✓ 内置 ControlNet、多采样器、负面提示、权重提示
- ✓ 支持 GFPGAN、RealESRGAN、放大与人脸修复
- ✓ 实时预览生成过程
- ✓ 模型管理、子目录与搜索
- ✓ 插件系统，可扩展 UI 功能
- ✓ 多 GPU 自动分配，支持 CPU 模式
- ✓ 安全特性：picklescan 与 safetensors
- ✓ 自动更新机制



## ■ 服务端怎么跑？vllm

Easy, fast, and cheap LLM serving for everyone

```
1 docker run --runtime nvidia --gpus all \
2     -v ~/.cache/huggingface:/root/.cache/huggingface \
3     --env "HF_TOKEN=$HF_TOKEN" \
4     -p 8000:8000 \
5     --ipc=host \
6     vllm/vllm-openai:latest \
7     --model Qwen/Qwen3-0.6B
```



<https://docs.vllm.ai/en/latest/deployment/k8s/#deployment-with-gpus>

# ■ sglang

<https://docs.sglang.io>



```
1 docker run --gpus all \
2   --shm-size 32g \
3   -p 30000:30000 \
4   -v ~/.cache/huggingface:/root/.cache/huggingface \
5   --env "HF_TOKEN=<secret>" \
6   --ipc=host \
7   lmsysorg/sqlang:latest \
8   python3 -m sglang.launch_server --model-path meta-llama/Llama-3.1-8B-Instruct --host
0.0.0.0 --port 30000
```

# ■ hf-model-downloer

跨平台 GUI，一键下载模型，**无需任何依赖**。

支持 Hugging Face 与 ModelScope

自动处理 token 登录

显示实时下载进度

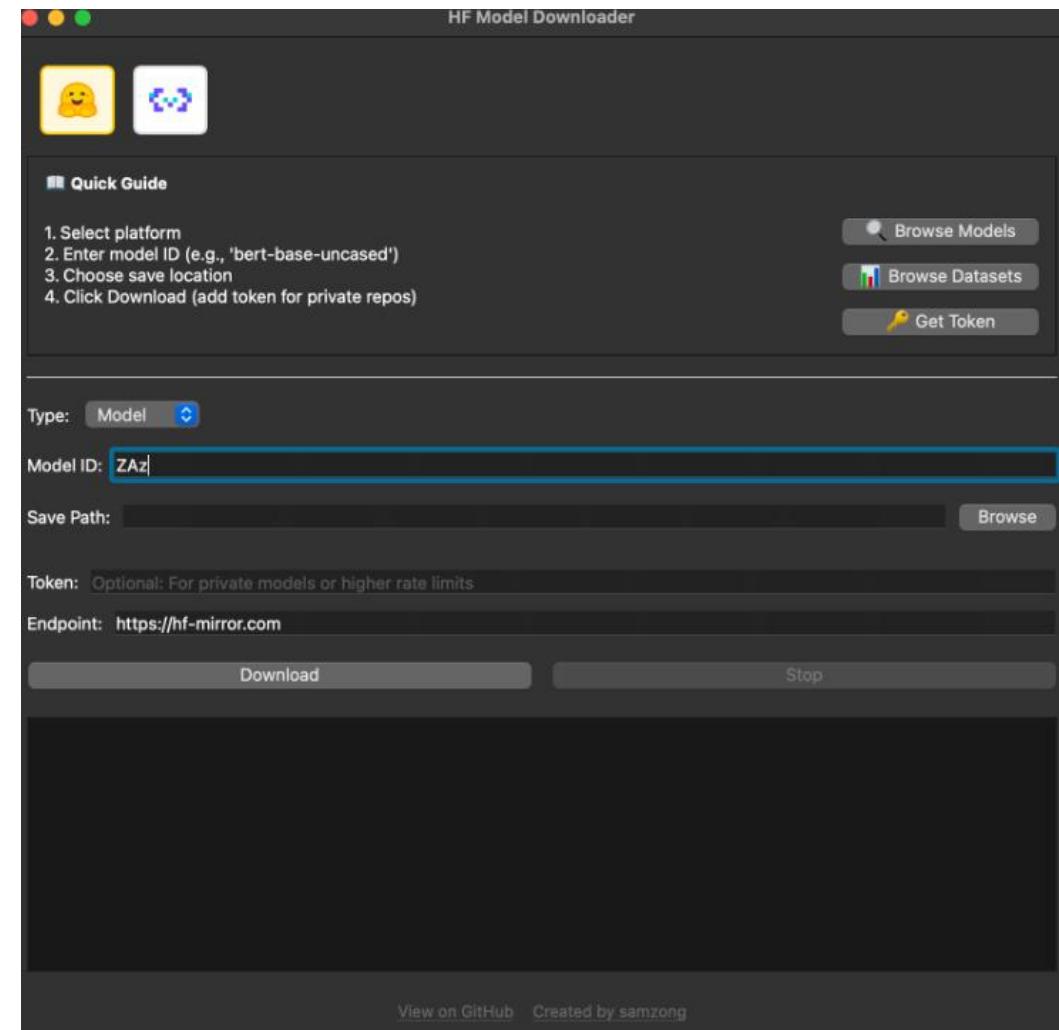
纯本地可执行文件，免安装依赖

Windows macOS Linux 全支持

开箱即用，无需命令行

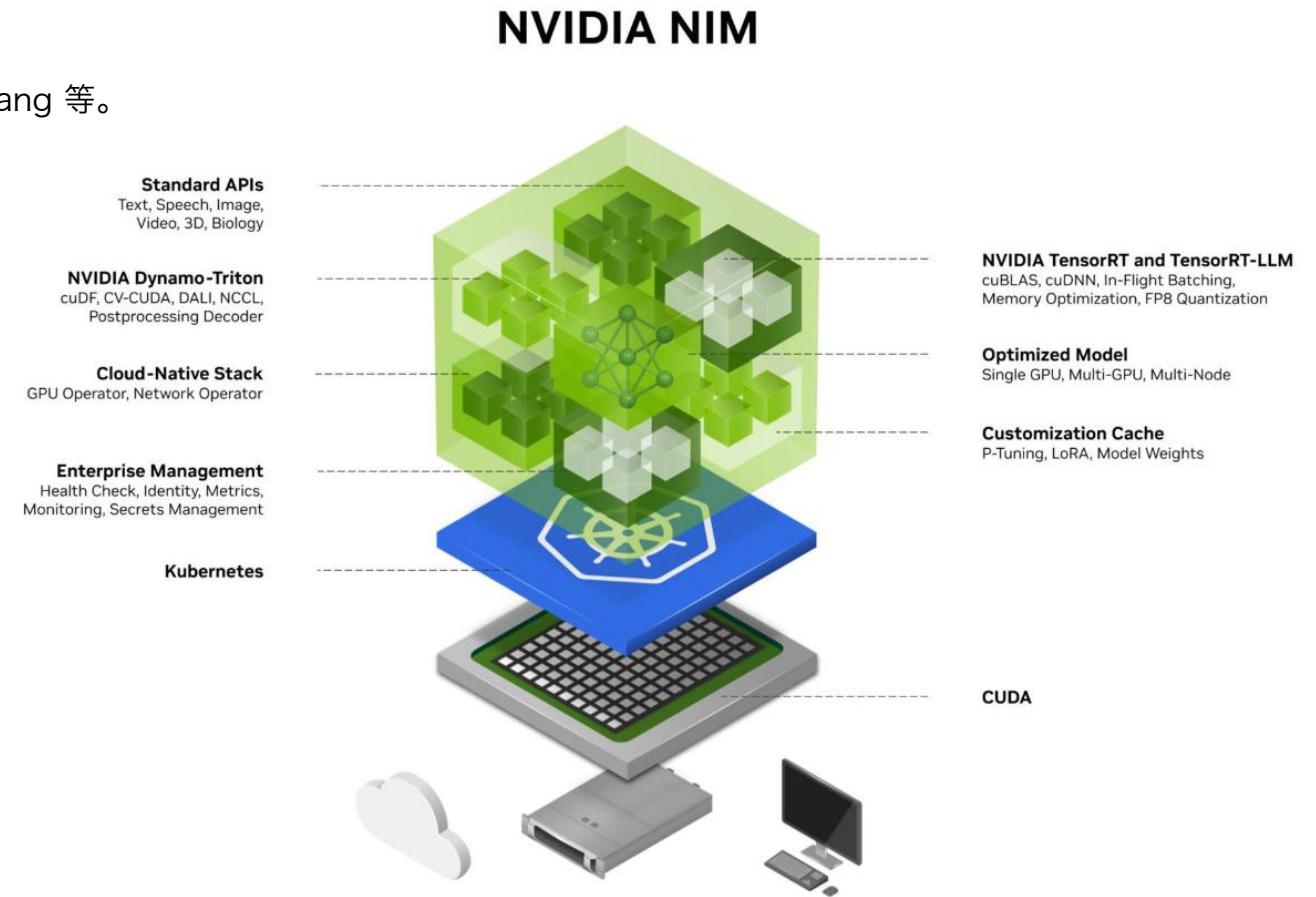
支持大模型断点续传与多文件下载

<https://github.com/samzong/hf-model-downloader>



# Nvidia NIM

- ✓ 预封装推理微服务。开箱即用的 GPU 加速容器。
- ✓ 覆盖主流推理引擎。TensorRT、TensorRT-LLM、vLLM、SGLang 等。
- ✓ 标准化 API。文本、图像、语音等统一接口。
- ✓ 针对 GPU 优化。低延迟，高吞吐，支持多 GPU、多节点。
- ✓ 支持模型定制。LoRA、P-tuning、自定义权重缓存。
- ✓ 自托管部署。PC、工作站、数据中心、云均可运行。
- ✓ 可观测与运维组件。指标、健康检查、Helm、K8s 集成。
- ✓ 海量模型选择。社区模型、企业模型、官方优化模型均可用。



# ■ nvidia trtllm-serve & trtllm-build

<https://github.com/NVIDIA/TensorRT-LLM>

## trtllm-serve

- ✓ 启动 OpenAI API 兼容服务，支持 /v1/models、/v1/completions、/v1/chat/completions
- ✓ 支持 TensorRT 引擎、HF Checkpoint、本地目录三种模型源
- ✓ 支持 C++ backend 与 PyTorch backend，可切换
- ✓ 开箱即用的健康检查、版本、metrics 监控端点
- ✓ 支持 TP、PP、EP、多节点 Slurm 分布式部署
- ✓ 支持 KV Cache 管理、block reuse、free\_gpu\_memory\_fraction 配置
- ✓ 原生多模态（图像、视频）推理能力
- ✓ 支持 reasoning parser (DeepSeek-R1)
- ✓ 支持额外 YAML 配置覆盖运行参数
- ✓ 与 genai-perf 等基于 OpenAI API 的基准工具兼容

## trtllm-build

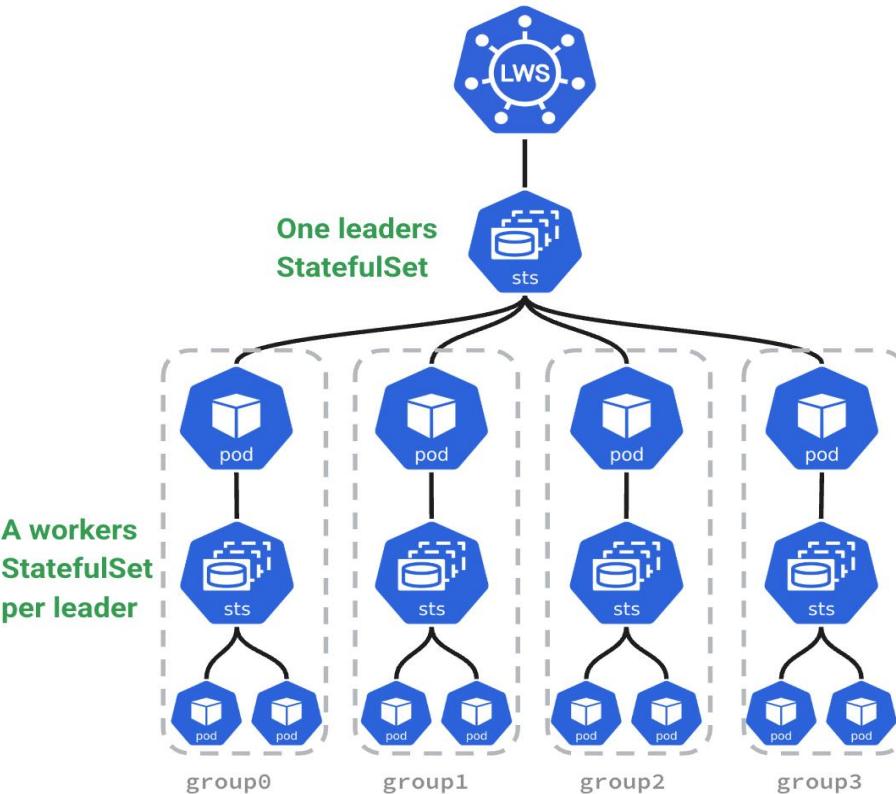
- ✓ 将 TensorRT-LLM checkpoint 转成可部署的 TensorRT 引擎
- ✓ 支持自定义模型、LoRA、多 LoRA、DoRA、MoE 等多类型模型结构
- ✓ 支持 FP16 FP32 FP8 等多精度构建
- ✓ 可开 timing cache、strip plan，加速构建或减少体积
- ✓ 自动并行 (auto\_parallel) 支持多节点 GPU 构建规划
- ✓ 可选多 profile、可视化 ONNX 导出、dry-run 调试
- ✓ 构建产物统一输出到 engine\_outputs 目录

```
1 docker run --rm -it \
2   --ipc host \
3   --gpus all \
4   --ulimit memlock=-1 \
5   --ulimit stack=67108864 \
6   -p 8000:8000 \
7   nvcr.io/nvidia/tensorrt-llm/release:1.2.0rc4
```

# ■ 大规模分布式：LWS (LeaderWorkerSet)

<https://github.com/kubernetes-sigs/lws>

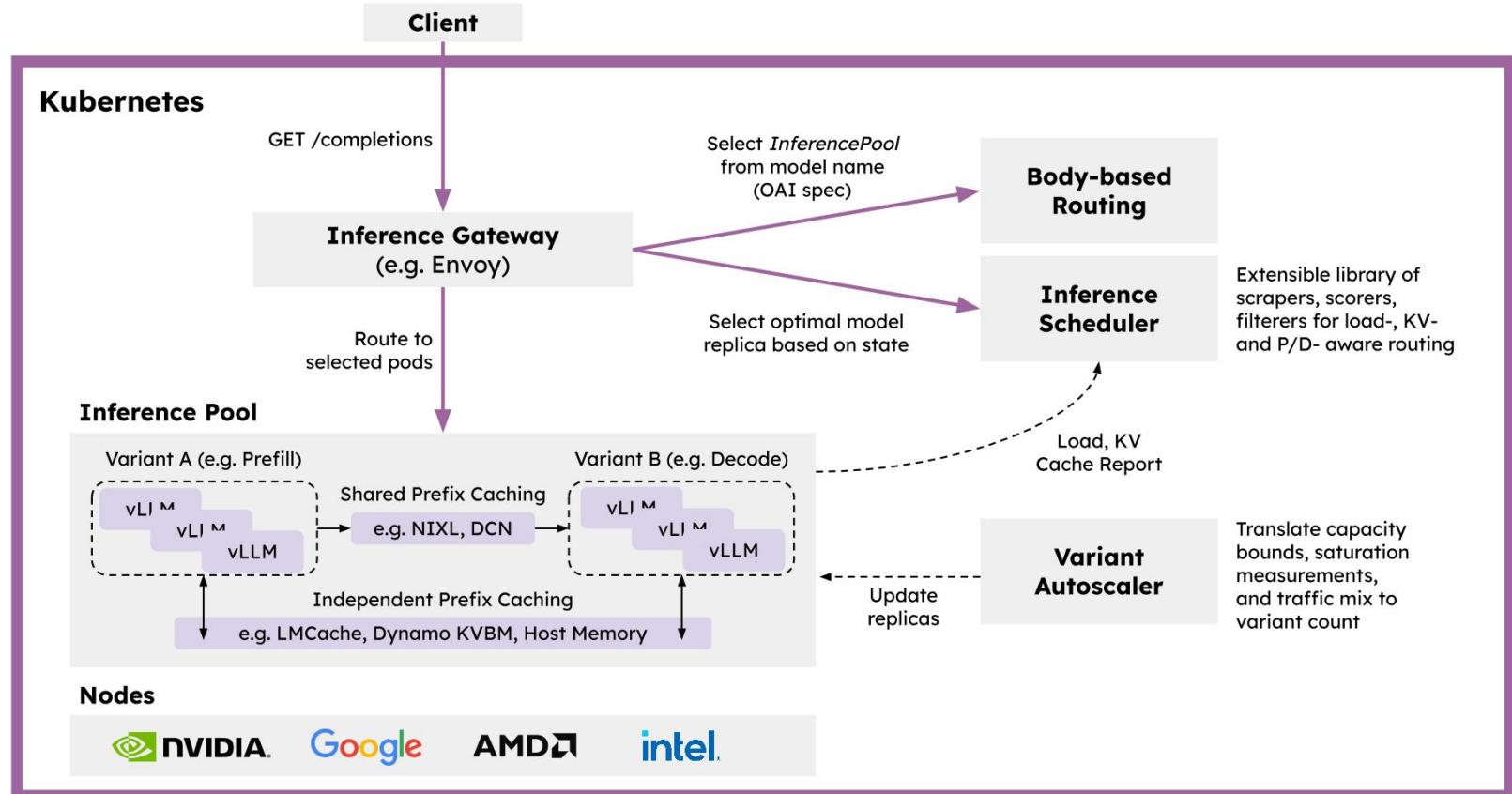
- ✓ 将一组 Pod 当成一个“超 Pod”执行单元，统一生命周期管理
- ✓ 支持 Leader 与 Worker 双模板，适合多机协同的推理任务
- ✓ 每个 Pod 有唯一 index，利于分片、跨节点调度
- ✓ 多 Pod 并行创建，提高大规模推理任务的启动速度
- ✓ Gang Scheduling，可选的全成组调度，保证同组全部成功
- ✓ 以“组”为单位滚动升级、扩缩容，与 HPA 对接
- ✓ 拓扑感知调度，保证同组 Pod 同拓扑放置
- ✓ 失败全组重建，可选的 all-or-nothing 恢复策略
- ✓ 支持多 replica，多组相同配置的独立协作单元
- ✓ 专为多主机 LLM 推理、分布式 AI 应用设计



# ■ LLM-D

<https://github.com/llm-d/llm-d>

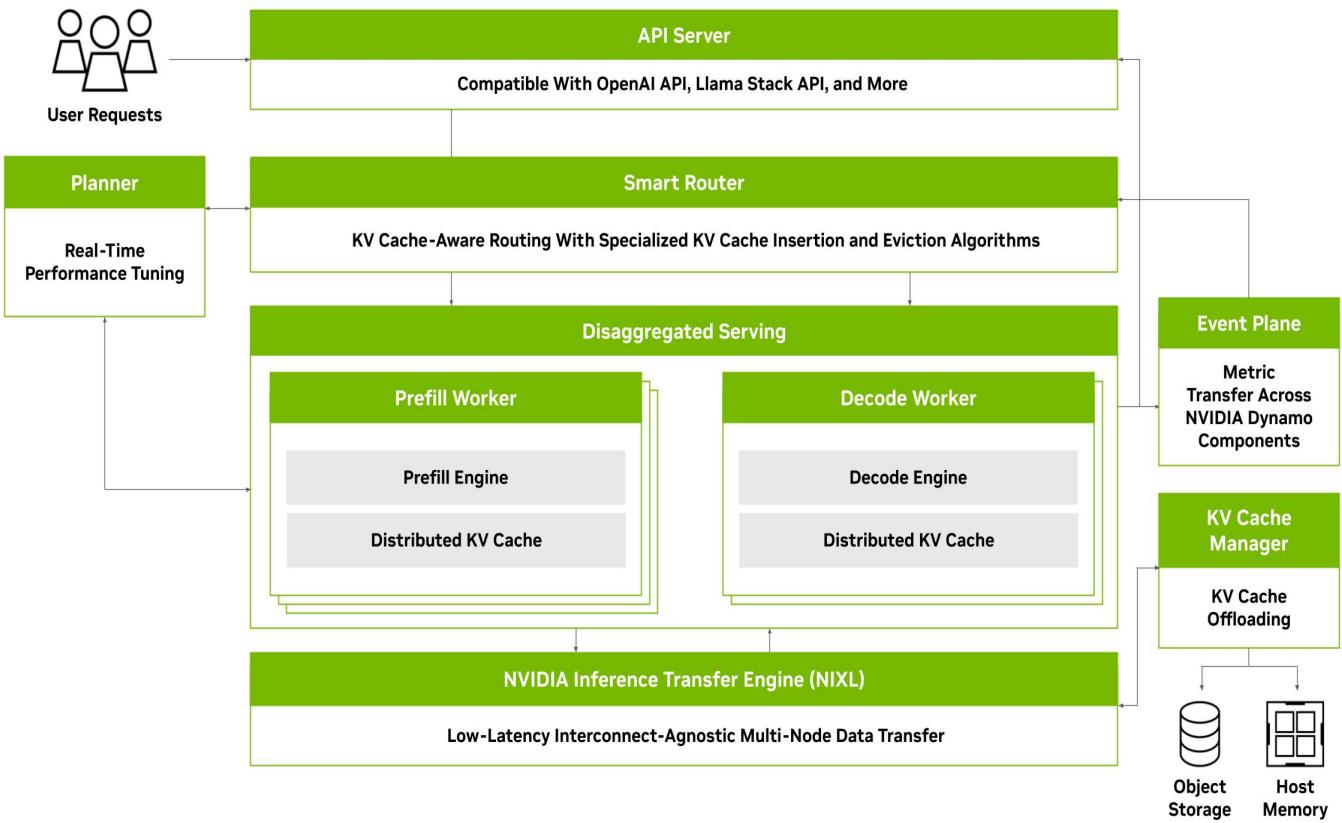
- ✓ Kubernetes 原生的大模型推理栈
- ✓ 统一调度 vLLM、IGW、NIXL、Envoy
- ✓ 支持多硬件 GPU/TPU/XPU/AMD
- ✓ 提供推理调度、前后分离、Prefix Cache
- ✓ 提供专家并行 MoE 部署方案
- ✓ 带 Helm Charts，可直接部署
- ✓ 面向高吞吐、低延迟的大模型在线推理
- ✓ 针对数据中心高速网络优化
- ✓ 有完善的指南、架构和 Northstar 设计



# Nvidia Dynamo

- ✓ 多节点多 GPU 推理。高吞吐，低延迟。
- ✓ 分体式推理。prefill 和 decode 解耦提高并发。
- ✓ KV 路由与缓存复用，减少重复计算。
- ✓ 多引擎统一框架。支持 vLLM、SGLang、TensorRT-LLM。
- ✓ NATS + etcd 做集群控制面。
- ✓ Rust + Python 混合架构，性能与扩展性兼顾。
- ✓ 支持本地跑，也支持集群级部署（K8s）。
- ✓ 内置预处理、路由、前端 API（OpenAI 兼容）。
- ✓ 数据传输优化。NIXL、缓存卸载等加速路径。

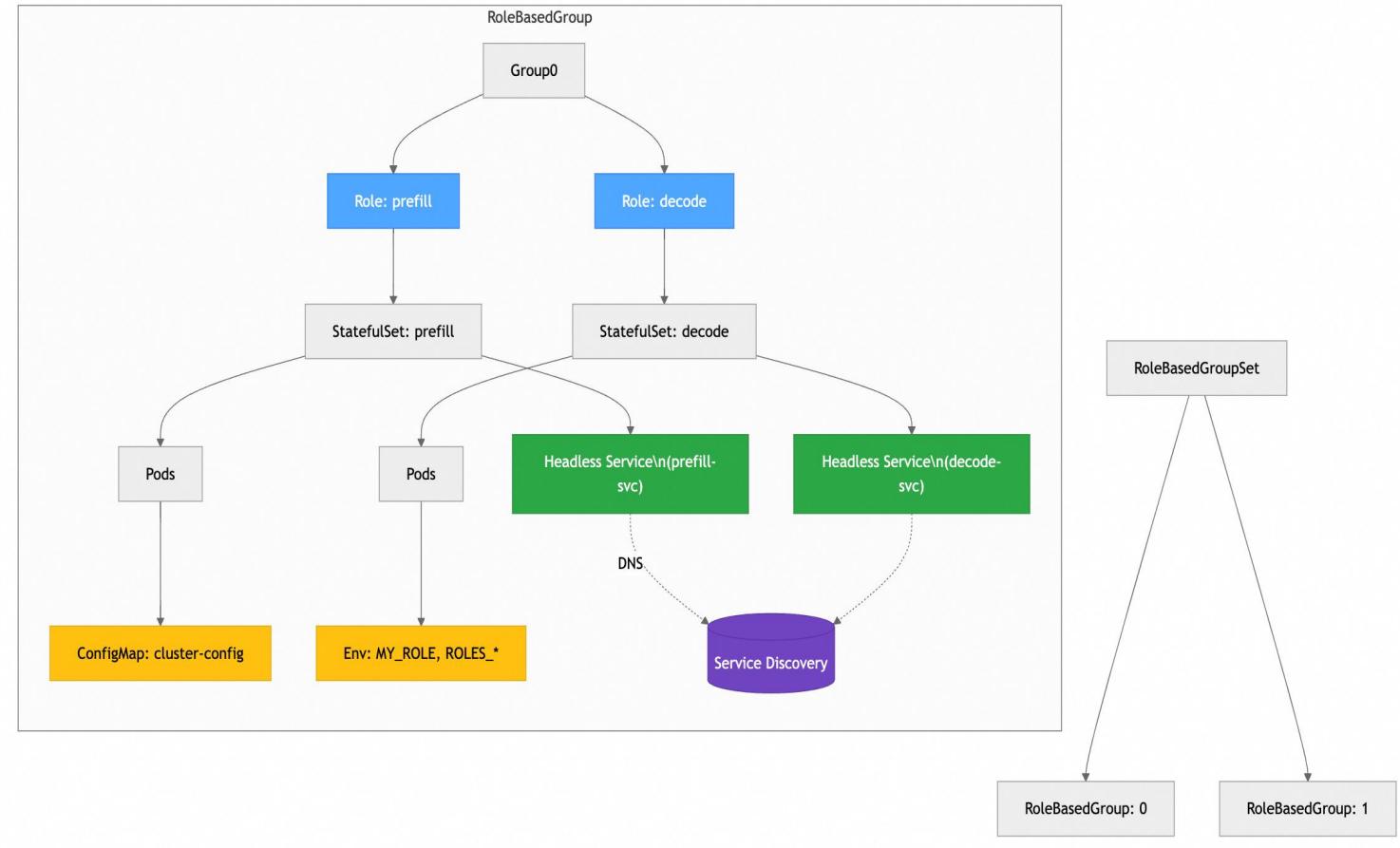
<https://github.com/ai-dynamo/dynamo>



# ■ sgl-project/rbg

<https://github.com/sgl-project/rbg>

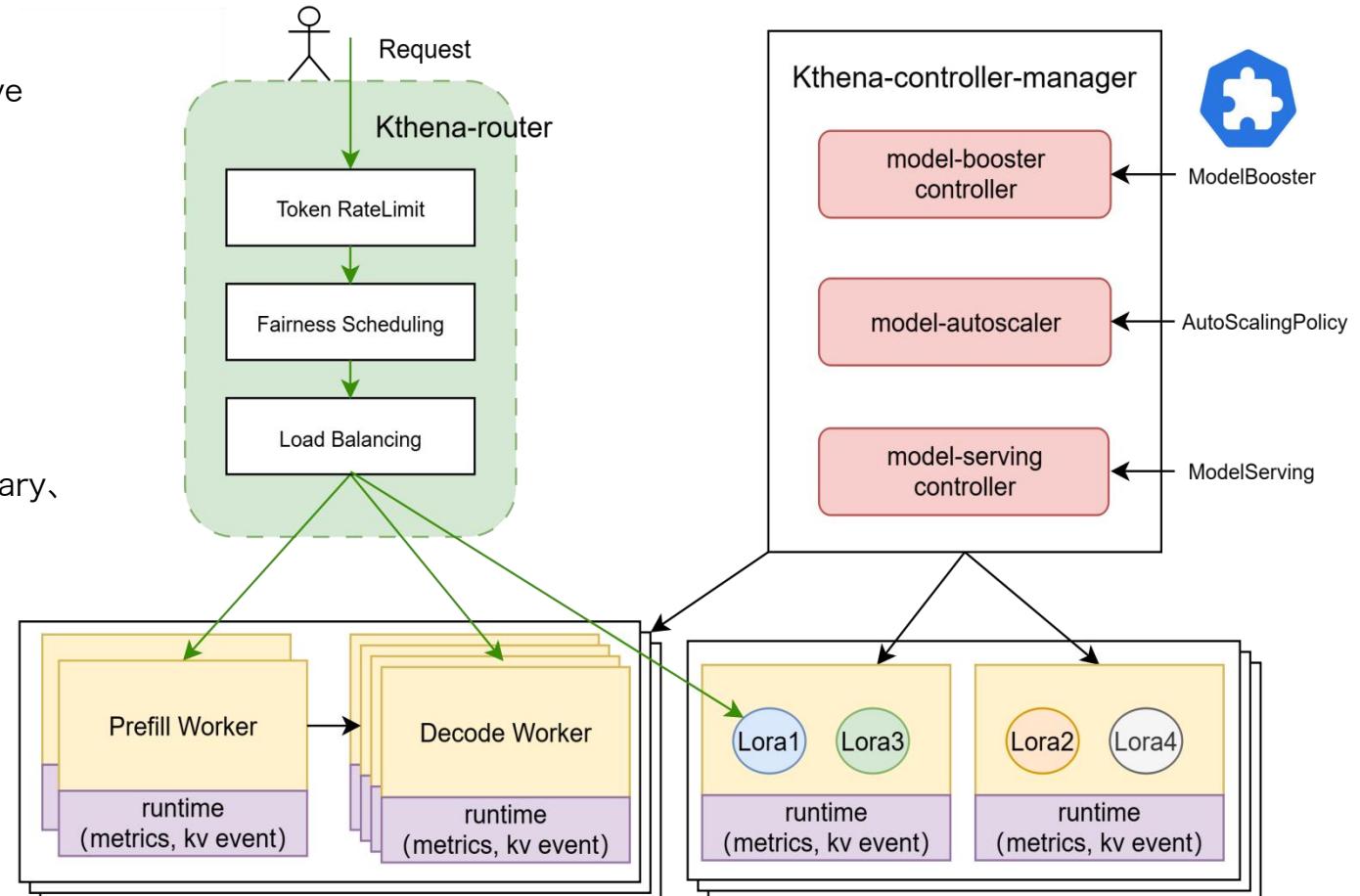
- ✓ 多角色工作负载编排，统一管理 prefill、decode 等角色
- ✓ 跨角色启动顺序控制，确保依赖角色先启动
- ✓ 自动服务发现，注入拓扑与角色信息
- ✓ 角色级弹性伸缩
- ✓ 角色级原子滚动升级，整组同步更新
- ✓ 故障原子恢复，任一 Pod 异常触发整组重建
- ✓ 支持多种工作负载类型 StatefulSet、Deployment、LeaderWorkerSet
- ✓ 拓扑感知调度，保证角色在同一拓扑域内部署



# kthena

- ✓ Kubernetes 原生的 LLM 推理平台
- ✓ 统一 CRD 管理模型、实例、路由、LoRA 等生命周期
- ✓ 支持多推理引擎，含 vLLM、SGLang、Triton、TorchServe
- ✓ Prefill Decode 分体式推理，提升吞吐与利用率
- ✓ 成本驱动的自动扩缩，多指标、多策略
- ✓ 零中断模型滚动升级
- ✓ LoRA 动态热插拔
- ✓ 拓扑感知调度，靠近数据和带宽域
- ✓ Gang 调度，保障 xPyD 等分布式推理原子部署
- ✓ 智能路由，支持多模型、KV cache aware、流量权重、canary、失败切换
- ✓ 控制面和数据面分离，架构清晰可独立部署

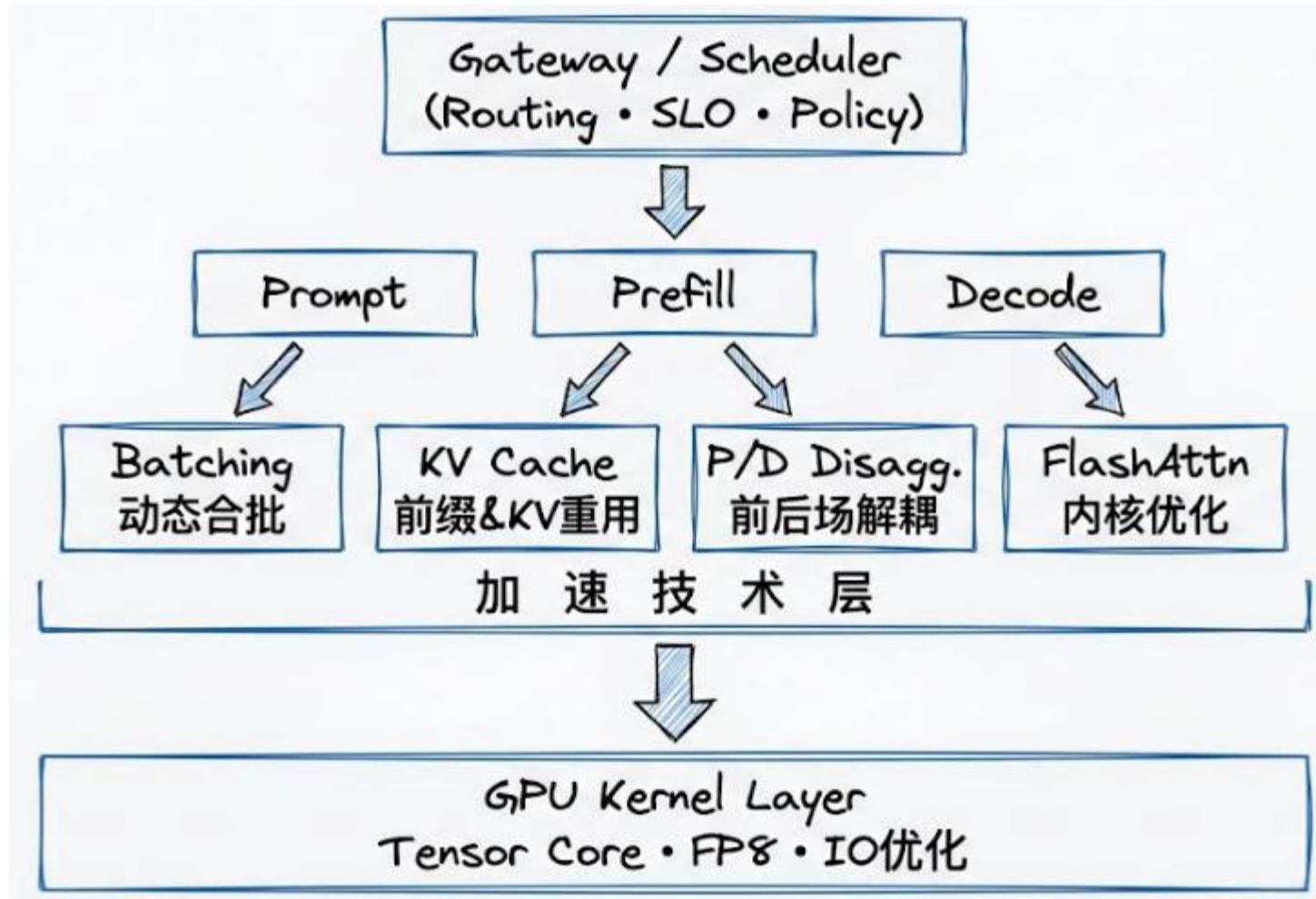
<https://github.com/volcano-sh/kthena>



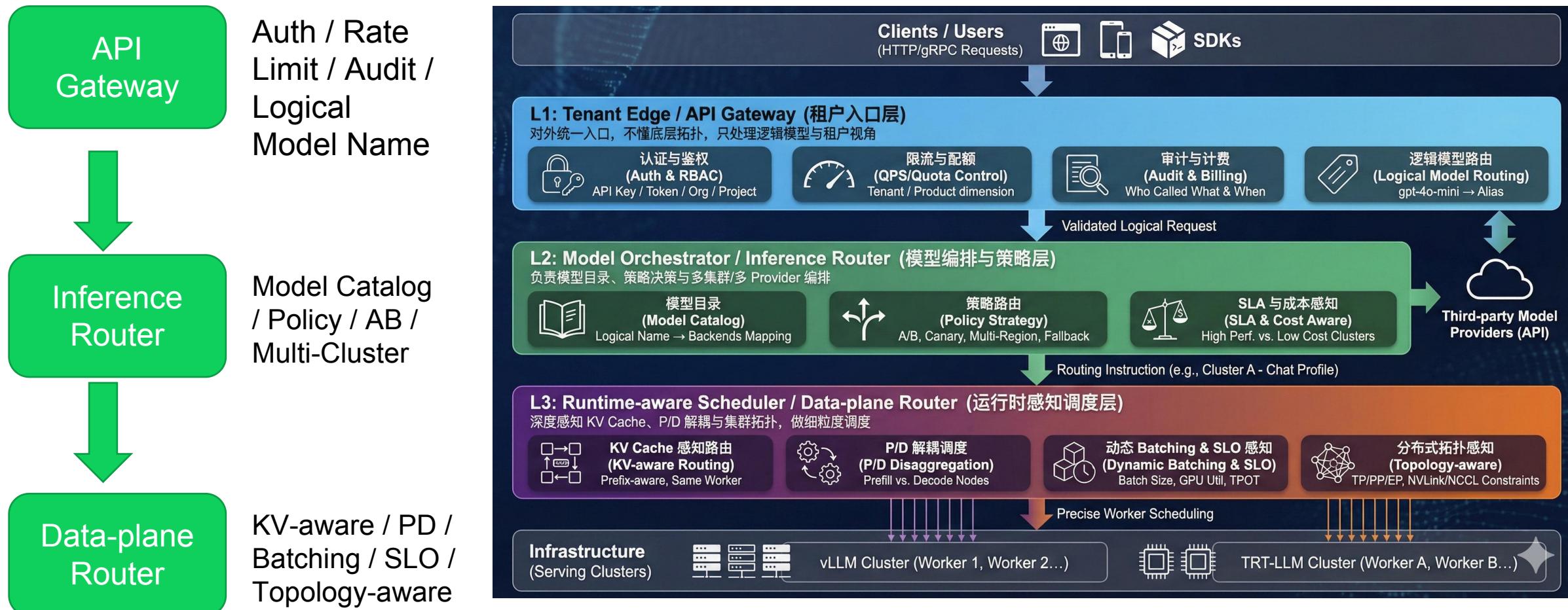
## ■ 大规模分布式推理带来的挑战

当把大模型从单个 POD / 单个节点转移多节点，甚至根据角色进行分离时，大模型部署的复杂度急剧提升，不仅仅要考虑 runtime 的支持情况，同时还有涉及到网络、调度、KV Cache 管理、模型权重管理等等问题。

这里的复杂程度就像是再造一个 K8s，操作系统。



# ■ 入口与调度层：Gateway / Router / Scheduler



# ■ KV Cache 体系：本地 / 跨节点 / 分层存储

场景：Prefill 在 GPU A 上算完 KV， Decode 在 GPU B 上消费

依赖组件：

- ✓ NIXL 这种 Data Movement Library，实现 GPU↔GPU / GPU↔CPU / GPU↔存储的高吞吐低延迟传输，支持 RDMA、NVLink、GDS 等后端
- ✓ KV metadata：维护 KV block 的 handle / location / TTL

关注点：

- ✓ 0拷贝 / RDMA，减少 host 参与
- ✓ 支持非连续数据块、异步传输（NIXL 明确针对这点优化）

## KV 分层存储与 offload

Local KV store: GPU HBM → host DRAM → local NVMe

- ✓ Remote KV store: 对象存储 / 分布式文件系统 + GDS / NIXL 插件
- ✓ 专用 KV 管理器（例如 Dynamo KV Cache Manager、LMCache 等）
- ✓ 与 Gateway / Scheduler 对齐，使“KV 命中率”对调度可见

## ■ 模型 & 权重管理 Model Catalog

- ✓ 保存模型元数据：名称、版本、格式、精度、所需资源
- ✓ Model Storage / Snapshotting / Lora
- ✓ 冷存：对象存储（S3 / 兼容）
- ✓ 热存：NVMe 集群、本地 cache
- ✓ 版本管理：支持 rollback / canary
- ✓ Model Loader (Sidecar) 负责把权重下发到指定节点，管理本地 cache、GC、配额
- ✓ 与调度层打通：只有权重 ready 的 replica 才能接流量

<https://github.com/samzong/modelfs>

抽象专门的模型管理层，支持模型多版本管理，和模型信息



<https://github.com/BaizeAI/dataset/>

提供统一的存储层和预热能力

## ■ 资源规划与弹性

### Workload-aware Planner

- ✓ 输入: ISL/OSL 分布、模型 profile (prefill heavy / decode heavy) 、GPU 拓扑
- ✓ 输出: 每种 “variant” 该给多少 GPU、prefill:decode 比例、每个池子多大
- ✓ 典型例子: NVIDIA Dynamo Planner, 对 disaggregated serving 做推理感知扩缩

### Autoscaler

- ✓ 根据 SLO (P95 TTFT / TPOT) 和资源利用率联合决策, 而不是只看 QPS
- ✓ 与 Gateway/Scheduler 打通, 知道当前队列长度 / backlog shape

### Placement Controller

- ✓ 把 “需要多少 prefill/ decode GPU” 翻译成 pod 部署、节点亲和、network 拓扑约束
- ✓ 与 GPU Operator / Node Feature Discovery / topology-aware scheduler 集成

# ■ NEXT ?



在用道客技术之前



用了道客技术之后

# Thanks.