项目申请书

项目名称：Cloudpods 大模型一键部署

项目导师：空心

项目申请人：程萬哲

邮箱：cwz_eikoh@163.com

项目背景

项目基本需求

1. 能通过 Cloudpods 容器主机一键部署主流大语言模型服务(deepseek, qwen, llama 等）。

2. 能通过 Cloudpods 容器主机一键部署 Dify LLM开发应用平台，能通过 Dify 访问部署的大语言模型服务。

3. 代码以 PR 形式提交到 https://github.com/yunionio/cloudpods 仓库
4. 产出项目开发使用文档

项目相关仓库

- Cloudpods 主仓库：https://github.com/yunionio/cloudpods
- Cloudpods 前端页面仓库：https://github.com/yunionio/dashboard

相关技术的了解程度

golang 与 k8s 相关

在学校实验室实习时负责过一站式云原生基础设施解决方案平台的开发，期间一直用 **golang** 语言进行开发，同时自己动手在多台机器上部署过 **k8s** 集群，对 **k8s** 也有一定程度的了解。

大模型部署相关

在学校实验室实习时主持过大模型低依赖智能体框架的开发，手动部署过市面上主流的几乎所有开源大模型，并且在自己的电脑上常态化部署着一个 **ollama** 服务以使用大模型进行翻译，因此有一定的大模型部署的熟练度。

Linux 相关

本人自大三以来的主力开发系统就是 **Arch Linux**，有日常的 **Linux** 使用经验。

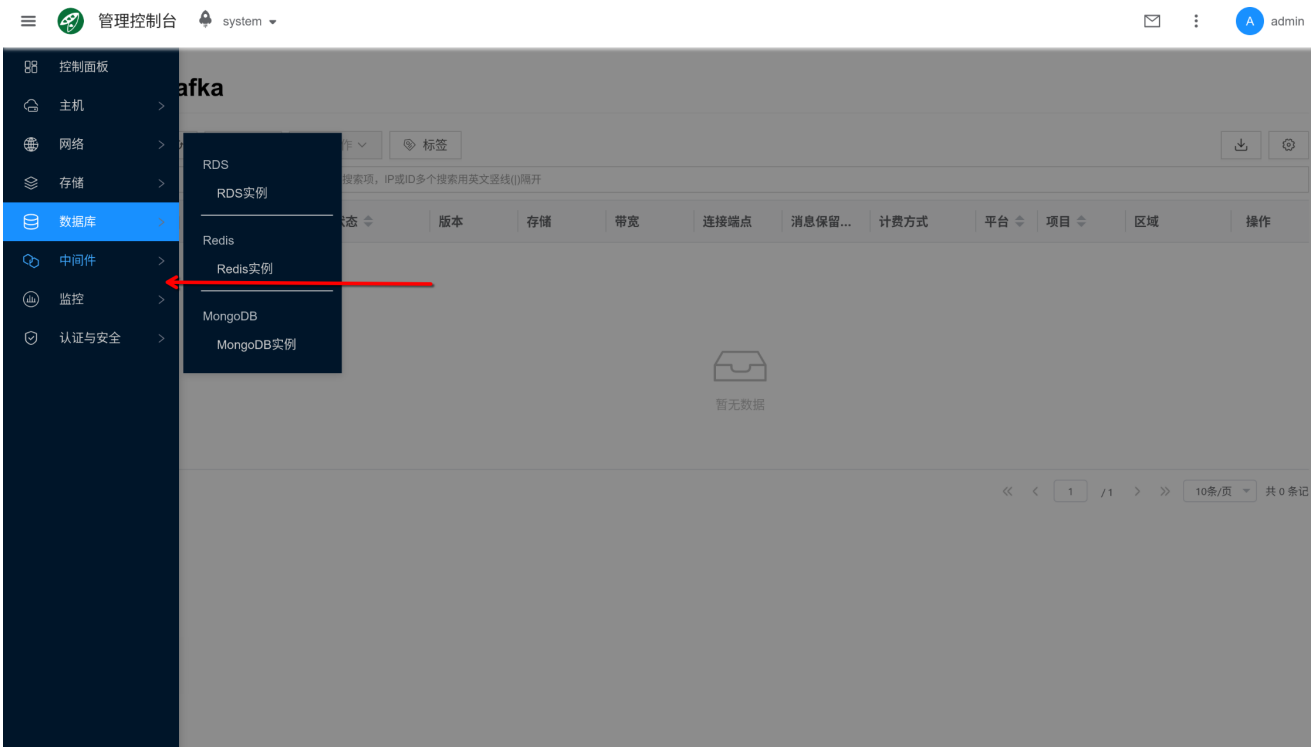
Container 相关

本人在开发学校实验室项目时参与制作了每一版项目发布的相关镜像，能够熟练完成 `Dockerfile` 的编写。

实现项目的技术方案

为大模型一键部署提供前端页面的访问接口

在“中间件”的下方提供“大模型”的菜单项，其中包括 “开源大模型” 分组下的 “**LLM** 实例” 与 “大模型开发应用平台” 分组下的 “**Dify** 实例” 两个子菜单选项。



其中，“**LLM** 实例”点击后跳转到与“**RDS** 实例”类似的页面，表头上的操作按钮相同，表头的内容会进行变动。“**Dify** 实例”点击后跳转的页面也类似。

在后端实现大模型实例一键部署与管理，并提供相关接口与前端进行对接

首先对 **ollama** 的一键安装脚本进行修改，从而与 **cloudpods** 平台的 **GPU** 透传功能进行对接，让部署的 **ollama** 能够调用到指定的 **GPU**。

在安装 **ollama** 后就可以调用 **ollama** 的命令拉取常见的大模型并进行部署，这里可以考虑参照 在 Kubernetes 中 Autoscale LLM 的实践 这篇博客实现一个镜像加载的缓存来缓解网络的压力。

然后就是根据这个安装脚本以及后续的命令制作 **docker** 镜像并编写对应的 `Helm` 文件，从而实现在 **k8s** 集群上的一键部署。

最后在 **cloudpods** 的数据库表中添加大模型实例管理的相关表，并在相应的模块上实现部署与管理的接口，最后提供给前端进行调用。

对于 **huggingface** 上无法通过 **ollama** 一键部署的模型，可以提供一个 `Modelfile` 的创建功能，从而通过 **ollama** 运行 **huggingface** 上的众多大模型。

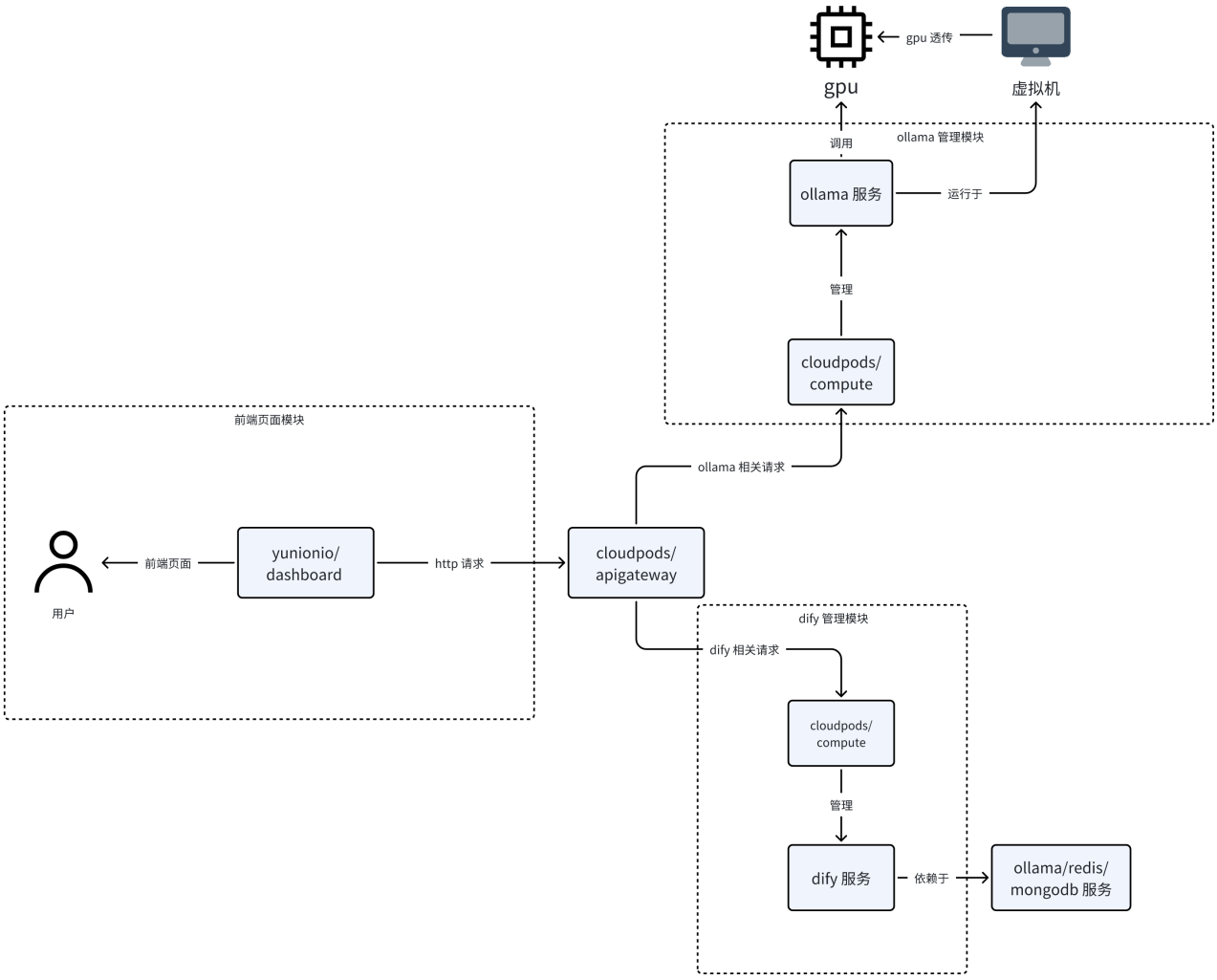
在后端实现 Dify 实例的一键部署与管理，并提供相关接口与前端进行对接

制作 **dify** 部署的 `Helm` 文件，直接用官方的 **docker** 镜像即可，不过部署时需要考虑依赖问题（**redis mongodb ollama**），这里可以让用户自行在 **cloudpods** 平台上部署相关依赖，并由用户在部署 **dify** 时选择。

然后制作管理相关的数据库表并在 **cloudpods** 的相应模块上实现部署与管理的接口，最后提供给前端进行调用。

总体模块设计

总体的模块设计图见下方，主要分为三个模块，每个模块实现对应的需求



时间规划

> *因为是大四而且保研本校，因此有相当多的空闲时间保障项目进度。*

镜像制作与测试阶段（第1～3周）

- ◯ 制作一键部署 **ollama** 的 `helm` 文件
  - ◯ 编写能够调用透传 **GPU** 的 **ollama** 一键部署脚本
- ◯ 制作一键部署 **dify** 的 `helm` 文件

后端管理接口的开发与测试阶段（第4～8周）

- ◯ 设计相关数据库表结构
- ◯ 编写前后端的接口文档
- ◯ 在后端实现管理 **ollama** 服务的相关功能
- ◯ 在后端实现管理 **dify** 服务的相关功能
- ◯ 测试并完善相关接口

用一周时间设计相关数据库表结构并编写前后端的接口文档，之后用三周时间实现后端接口的实际功能（**service** 与 **model** 层），最后用一周时间测试与完善接口。

前端页面的实现与总体功能的测试阶段（第9～12周）

- ◯ 开发管理 **ollama** 服务的前端页面
- ◯ 开发管理 **dify** 服务的前端页面
- ◯ 对实现的功能进行功能性与非功能行测试
- ◯ 整理出开发文档

用一周时间开发前端页面，在最后两周时间进行功能性与非功能性测试并整理出开发文档，完成项目的结项。