# 环境搭建及CI/CD实践说明文档

朱子墨　　赵英棋　　杨正菲　　陈梓萌

October 15, 2025

## 1　作业环境搭建

### 1.1　ECS的配置选择

为完成本次作业，小组购买了华为ECS弹性云服务器服务。具体配置如下：

| 配置项目 | 规格选择 | 配置说明 |
|---|---|---|
| 操作系统 | Ubuntu 24.04.3 LTS | 原计划采用Windows，后考虑Linux环境更便于部署ElasticSearch等数据库 |
| 区域 | 华北-北京四 | 自动选择，降低网络延迟 |
| CPU核心 | 2核 | 满足开发测试需求 |
| 内存容量 | 8GB | 保证ElasticSearch等服务稳定运行 |
| 系统盘 | 40GB | 系统盘标准配置 |
| 网络带宽 | 5 Mbit/s | 适合中小流量应用 |
| 服务期限 | 3个月（90天） | 基于课程持续时间与成本的综合考量 |
| 购买价格 | 915元 | |

Table 1: ECS云服务器环境配置清单

### 1.2　关于配置的说明

由于项目对于时间延迟的容忍度相对较高，而对于存储需求要求比较大，因此我们选择了内存8GB+系统盘40GB的设置，这是由于RAG知识库、ElasticSearch的运行均会占用一定空间，为了保证项目的正常运行小组选择了此配置。

小组原来考虑选择Windows操作系统，但是由于ElasticSearch等工具在Windows上安装配置比较麻烦，在Linux环境中的配置较为简单，最终选择了较为熟悉的Ubuntu作为ECS的操作系统。

考虑到购买长时间的服务价格较高，因此小组选择购买3个月的服务，这样保证预算可控在1000元以下且足够覆盖课程的学习时间。

## 1.3 安全问题

华为云ECS环境易受到外部攻击，理想的方案是采用密钥作为密保防范攻击；项目中由于在购买和配置时的设置，采用了相对简单的密码进行防护，不过仍包含了数字、字母、特殊符号等要素，但是安全性比起密钥仍有欠缺。

## 1.4 项目的迁移与部署

原项目在windows主机上开发。在ECS中，小组先通过github获得了项目的源代码，然后配置了java、Nodejs、mySQL、Elasticsearch的工作环境，并且配置了RAG知识库。随后，小组成员进行了数据库的迁移。在此之后，进行了简单的初步环境测试，确认通过后开始了CI/CD流程。

# 2 CI/CD pipeline流程

考虑到github的CI/CD流程的操作较为简单，因此小组利用github进行了CI/CD流程。

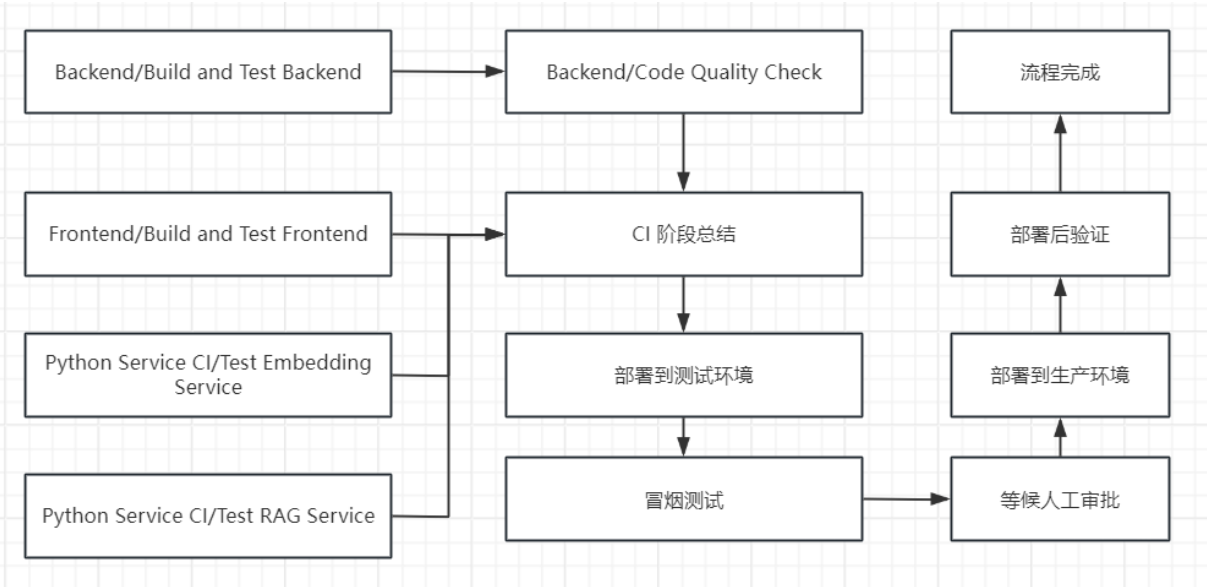


Figure 1: CI/CD流程的示意图

## 2.1 CI 阶段流程

### 2.1.1 后端持续集成流程

后端的持续集成包括两个提交触发流程:
**Build and Test Backend**: 编译、构建后端应用并运行单元/集成测试。这个过程涉及初始化CI 环境、配置Java 17 环境、使用Maven 编译项目、运行单元测试或集成测试、生成代码覆盖率报告（JaCoCo）、上传覆盖率报告到CI 系统和外部服务、归档构建产物、清理JDK 相关资源以及清理代码检出相关资源等过程，耗时2分钟56秒。

**Code Quality Check**：进行静态代码分析、安全检查、代码规范检查等过程，作为质量门禁。

这里的设计意在确保后端代码的任何提交都满足功能正确性和代码质量要求。

### 2.1.2 前端持续集成流程

这里主要进行了Build and Test Frontend过程：
**Build and Test Frontend**：打包前端资源（如JavaScript、CSS），运行前端测试套件，主要包括初始化CI 运行环境、从Git 仓库中拉取最新的前端代码、安装并配置Node.js环境、运行npm install安装项目依赖包、检查代码风格和质量问题、执行前端单元测试和组件测试、将测试覆盖率报告上传到CI 系统、构建用于生产环境的前端资源、将构建产物存档和清理临时资源等过程。

这里的设计意在独立验证前端代码的健壮性，确保用户界面功能正常。其独立性意味着前端和后端开发可以并行进行。

### 2.1.3 Python服务持续集成流程

这个流程专门针对Embedding Service和RAG service两个微服务，它们采用了Python技术栈。它们在项目的语义搜索功能中起到重要作用。
**Test Embedding Service**：专门测试向量嵌入服务，包括嵌入模型的功能、性能和准确性测试，主要步骤包括初始化CI运行环境、从Git仓库中拉取最新的Python代码、安装并配置Python环境、安装项目依赖包、使用flake8进行代码风格和质量检查、使用pytest框架运行单元测试和集成测试、上传测试覆盖率报告到CI系统或第三方服务和清理Python环境相关资源等。
**Test RAG Service**：专门测试检索增强生成服务，主要步骤包括初始化CI运行环境、从Git仓库中拉取最新的Python代码、安装并配置Python环境、安装项目依赖包、使用flake8进行代码风格和质量检查、使用pytest框架运行单元测试和集成测试、上传测试覆盖率报告到CI系统或第三方服务和清理Python环境相关资源等。

这里的设计意在将微服务核心能力作为独立的服务进行质量管控，确保Python微服务功能的可靠性和准确性。

### 2.1.4 CI 阶段总结

这是一个CI流程的汇聚阶段，主要作用是：
**聚合验证**：收集并验证所有并行CI子流程的结果
**质量决策**：基于各组件状态做出整体通过/失败决策
**流程推进**：为后续CD阶段提供明确的"绿灯信号"

该阶段通过后，当前代码质量已达到发布标准，持续部署流程已具备触发条件，main分支的这次推送将进入自动化部署环节，这个阶段总结实现了从持续集成到持续部署的过渡，代码处于可部署状态，准备进入CD阶段。

## 2.2 CD 阶段流程

CD流程从"CI 阶段总结"之后开始，包含了一个标准的部署流水线。

### 2.2.1 部署到测试环境

这一阶段包含两个部分：配置SSH密钥和布置到测试服务器。这是CD流程的第一步，将在CI阶段构建并测试通过的制品（jar包等）自动部署到一个模拟生产环境的测试环境。这个环境接近实际的生产环境。

这一步主要有两个目的：
**环境验证**：验证构建产物能否在一个“真实”的环境中正常启动和运行。
**集成测试准备**：为后续的冒烟测试和更全面的集成测试提供部署好的目标。

### 2.2.2 冒烟测试

在测试环境部署成功后，自动执行一系列关键路径的自动化测试用例。这些测试是高频、核心的业务流程。

其目的主要包括：
**快速反馈**：在几分钟内确认此次部署没有导致应用“冒烟”（即出现致命错误）。
**质量门禁**：如果冒烟测试失败，流程会立即中止，不会进入后续更耗时的阶段，从而节省时间和资源。

### 2.2.3 等候人工审批

流程中的一个手动门禁。在自动化流程成功部署到测试环境并通过冒烟测试后，流水线会暂停，并进行手动的审批。

这一步是为部署到生产环境增加最后一道安全阀。可以在此期间进行额外的手动验证，包括检查监控指标、查看测试报告和基于业务需求决定是否继续发布。

### 2.2.4 部署到生产环境

获得人工审批后，流水线继续，将同一个构建产物部署到真正的生产环境。这一步是自动化的，以确保部署过程的一致性和可重复性。

### 2.2.5 部署后验证

在生产环境部署完成后，立即运行一系列自动化健康检查。

其目的主要包括：
**确保稳定性**：第一时间发现因部署引入的线上问题，以便快速回滚。
**形成闭环**：确认部署不仅是成功的，而且应用是健康的。

### 2.2.6 流程完成

一个总结性步骤。当所有上述步骤都成功完成后，整个CI/CD流水线标记为完成。完成后会生成最终报告。

# 3 项目URL

这里我们给出了项目的URL：http://1.94.200.25/login

以及项目的源代码：https://github.com/tomorrowonce1010/new-voyage-mate.git