aiedumate 项目软件配置与运维报告

1. 项目概述

项目采用 Git 进行版本控制，并托管在 GitHub 上。利用 GitHub Actions 实现了一套完整的自动化 CI/CD (持续集成/持续部署) 流程，旨在提高开发效率和部署的可靠性。

技术栈核心：

- **语言:** Java 17
- **构建工具:** Apache Maven
- **CI/CD:** GitHub Actions

---

2. 配置管理

2.1 源代码管理

项目所有源代码均使用 Git 进行版本控制，并托管在 GitHub 仓库中。

2.2 应用配置

项目的配置遵循"配置与代码分离"的原则，分为开发环境配置和生产环境配置。

- **开发环境配置**:
    - 配置文件（如 `application.properties` 或 `application.yml`）位于各个模块的 `src/main/resources` 目录下。
    - 这些文件包含开发时所需的数据库连接、服务器端口等信息，不应包含任何生产环境的敏感信息。
- **生产环境配置**:
    - 生产环境的配置在部署时动态注入，而不是硬编码在代码或配置文件中。
    - 通过 GitHub Actions 的部署脚本，利用 Java 启动参数 (`-Dproperty=value`) 来覆盖 `application.properties` 中的默认配置。
    - 这种方式极大地增强了安全性，避免了将生产密钥、密码等敏感信息提交到代码库。

2.3 敏感信息管理

所有敏感信息（如服务器凭据、数据库密码等）都存储在 GitHub 仓库的 `Settings > Secrets and variables > Actions` 中。CI/CD 流程通过 `${{ secrets.SECRET_NAME }}` 的语法来安全地引用这些信息。

本项目使用的 Secrets 列表如下：

Secret Key	用途
`SERVER_HOST`	生产服务器的 IP 地址或域名
`SERVER_USERNAME`	用于 SSH 登录的用户名

Secret Key	用途
`SSH_PRIVATE_KEY`	用于免密登录服务器的 SSH 私钥
`SERVER_PORT`	服务器的 SSH 端口（默认为 22）
`PROD_DB_URL`	生产环境数据库的 JDBC URL
`PROD_DB_USERNAME`	生产环境数据库的用户名
`PROD_DB_PASSWORD`	生产环境数据库的密码
`SERVER_API_URL`	生产环境中 `aiedumate-server` 的访问地址，供 `aiedumate-client` 调用

---

3. 版本控制

3.1 分支策略

项目采用基于 `main` 分支的简单高效工作流。

- **`main` 分支**:
  - 作为项目的主分支，代表了稳定且可随时部署的生产版本。
  - 禁止直接向 `main` 分支推送代码。所有更改必须通过 Pull Request 合并。
- **功能分支**:
  - 所有新功能开发、错误修复都应在单独的功能分支上进行（例如 `feat/user-login`, `fix/db-connection-bug`）。
  - 开发完成后，向 `main` 分支发起 Pull Request。

3.2 Pull Request 流程

1. 开发者从 `main` 分支创建新的功能分支。
2. 在功能分支上完成开发和本地测试。
3. 向 `main` 分支发起 Pull Request。
4. GitHub Actions 会自动触发 CI 流程，对该 PR 进行构建和测试（注意：当前配置中部署构建跳过了测试，建议为 PR 流程单独配置一个强制测试的 Job）。
5. 代码审查通过且 CI 检查成功后，项目负责人将 PR 合并到 `main` 分支。
6. PR 合并后，自动触发向生产环境的部署流程。

---

4. 持续集成与持续部署 (CI/CD)

项目的 CI/CD 流程由 `.github/workflows/aiedumate-ci-cd.yml` 文件定义，基于 GitHub Actions 实现。

4.1 触发条件

- **`push` 到 `main` 分支**: 当有代码合并到 `main` 分支时，触发完整的构建和部署流程。
- **向 `main` 分支发起 `pull_request`**: 当有新的 PR 指向 `main` 分支时，触发构建流程（不部署），用于验证代码的正确性。

4.2 流程详解

Job: `build-and-deploy`

1. **环境准备 (Checkout & Set up JDK)**
    - 使用 `actions/checkout@v4` 拉取最新代码。
    - 使用 `actions/setup-java@v4` 配置一个基于 `temurin` 发行版的 Java 17 环境，并启用 Maven 缓存以加速后续构建。
2. **构建 (Build with Maven)**
    - 执行命令 `mvn -B clean package -DskipTests`。
    - `-B` (batch mode) 使 Maven 以非交互模式运行。
    - `clean package` 会清理旧的构建产物，然后编译代码、运行测试（如果未跳过）、并打包成可执行的 JAR 文件。
    - 注意: `-DskipTests` 参数跳过了测试环节。这是为了加速部署流程。在 PR 环节，应确保测试被完整执行。
    - 成功后，会在 `aiedumate-client/target` 和 `aiedumate-server/target` 目录下生成 `aiedumate-client-1.0.0.jar` 和 `aiedumate-server-1.0.0.jar`。
3. **部署 (Deploy to Server via SCP)**
    - 条件: 此步骤仅在代码被 `push` 到 `main` 分支时执行。
    - 使用 `appleboy/scp-action` 插件。
    - 将构建好的两个 JAR 文件 (`aiedumate-client-1.0.0.jar`, `aiedumate-server-1.0.0.jar`) 安全地传输到生产服务器的 `/opt/aiedumate` 目录下。
    - `overwrite: true` 确保每次都用新版本覆盖旧文件。
4. **重启服务 (Secure Restart on Server)**
    - 条件: 此步骤同样仅在代码被 `push` 到 `main` 分支时执行。
    - 使用 `appleboy/ssh-action` 插件，在生产服务器上执行一段 Shell 脚本来完成应用的重启。
    - **脚本逻辑**: a. **停止旧服务**: * 使用 `pgrep -f "aiedumate-..."` 根据 JAR 文件名查找正在运行的 Java 进程的 PID。 * 如果找到 PID，使用 `kill -15 $PID` 发送一个关闭信号，并等待 5 秒钟让程序处理关闭逻辑。 b. **启动新服务**: * 使用 `nohup java -jar ... > logfile 2>&1 &` 的方式在后台启动新的 JAR 文件。 * `nohup ... &` 确保即使 SSH 会话关闭，应用进程也能继续运行。 * `> /opt/aiedumate/server.log 2>&1` 将标准输出和标准错误都重定向到指定的日志文件中。 * **关键**: 在启动命令中，通过 `-D` 参数将 GitHub Secrets 中存储的生产配置（数据库、API 地址）注入到应用中，实现配置覆盖。

---

5. 部署与运维计划

5.1 环境要求

- **操作系统**: Linux (工作流中使用 `ubuntu-latest`)
- **运行时**: Java 17 (JRE 或 JDK)
- **部署目录**: `/opt/aiedumate`

5.2 日志管理

- Server 端日志: `/opt/aiedumate/server.log`
- Client 端日志: `/opt/aiedumate/client.log`
- **日常运维**:
  - 实时查看日志: `tail -f /opt/aiedumate/server.log`
  - 搜索关键字: `grep "ERROR" /opt/aiedumate/server.log`

5.3 监控与健康检查

- **基础监控**:
  - 检查进程是否存活: `ps -ef | grep aiedumate` 或 `pgrep -f "aiedumate-..."`
  - 检查服务端口是否在监听: `netstat -tuln | grep <端口号>`

5.4 灾难恢复

- **代码**: 所有代码安全托管在 GitHub，是主要的恢复源。
- **构建产物 (JARs)**: 即使本地和服务器上的 JAR 包丢失，也可以通过 GitHub Actions 的历史运行记录找到并下载之前的构建产物。
- **服务器配置**: 服务器的访问凭据和应用配置都保存在 GitHub Secrets 中，可随时用于重新配置新服务器。
- **数据**: 数据库的备份和恢复是独立于本应用运维体系之外的关键任务，需要有专门的数据库备份策略。