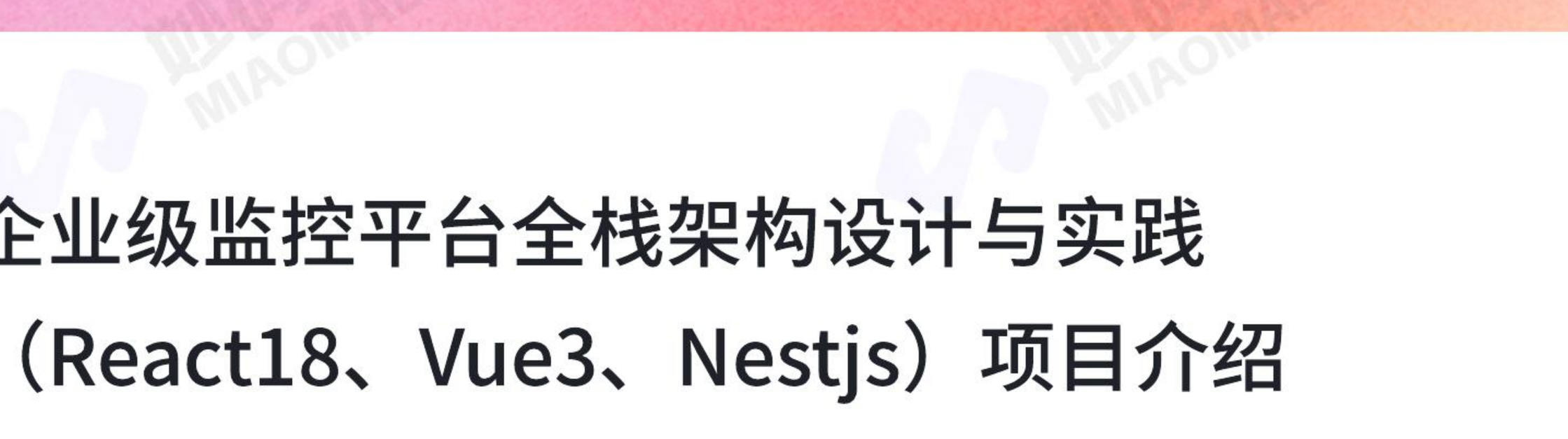
企业级监控平台全栈架构设计与实践（React18、Vue3、Nestjs）项目介绍



需求背景

项目概述

本项目旨在打造一个完整的前端监控解决方案，涵盖 SDK（适配 vanilla、React、Vue）、数据服务后端（Nest.js 实现）以及数据可视化平台（基于 React 和 Nest.js 构建）。项目的核心目标是帮助开发

团队对应用进行实时性能监控和异常捕获，支持迅速响应和修复问题，以提高产品的稳定性和用户体验。

业务需求

- **性能监控**：现代化前端应用在用户体验方面对性能要求越来越高。该项目的性能监控部分需要深入跟踪页面加载时间、渲染时间、首屏渲染时间等重要指标。此外，还需关注用户交互时的响应速度，如点击、滚动、表单提交等事件的处理时间。通过实时监控这些性能指标，可以识别并优化用户体验中存在的瓶颈。
- **异常监控**：现代化前端应用越来越复杂，异常监控需求随之增强。项目需能够捕获前端 JavaScript 错误（包括但不限于 Uncaught Errors 和 Promise Rejection），并提供丰富的错误上下文信息，如错误堆栈、错误来源文件、行号、调用堆栈等。异常监控系统还应支持自定义错误上报，便于开发团队添加业务逻辑中可能遇到的特定错误类型。
- **数据可视化与分析**：收集的数据不仅需要展示给开发者，还要能通过图表、报表等形式让团队成员轻松理解数据趋势。可视化平台需支持实时数据更新、数据过滤、事件追踪等功能。平台需具备自定义数据筛选和分组能力，使团队可以按项目、页面、时间等维度深度分析性能和异常问题，从而作出更具数据驱动的决策。

技术需求

- **多框架 SDK 支持**：考虑到不同业务线和技术栈的项目需求，SDK 应提供对多种前端框架（vanilla、React、Vue、Node）的支持，方便集成到不同项目中。SDK 应实现高效的数据采集、支持插件化扩展，并在性能上做到低影响。
- **数据服务后端**：使用 Nest.js 构建数据服务后端，数据服务需能高效接收大量前端数据上报请求并稳定运行。同时，需具备数据的预处理功能，将不同类型的数据进行过滤、分类【kafka topics】和清洗，并支持按需存储【clickhouse】，以便在可视化展示时减少数据冗余。
- **数据可视化平台**：基于 React 和 Nest.js 构建的数据可视化平台应支持丰富的数据展示功能，包括实时数据流、数据折线图、柱状图、热力图等。同时，前端需实现用户友好的交互，支持不同数据维度的分析和过滤功能。

竞品分析

竞品分析概述

目前市场上成熟的前端监控解决方案有 **Sentry**、New Relic、Datadog 等。这些产品通常具有完善的异常监控、性能监控和用户行为分析功能，支持实时数据上报和高效数据处理。它们提供了丰富的 SDK 和灵活的数据可视化接口，能够满足大部分企业的监控需求。

竞品功能对比

- **监控功能**：大部分竞品均提供全面的监控功能，如 JavaScript 错误捕获、资源加载失败监控、性能监控（如 FCP、LCP、CLS 指标），并支持自定义事件的上报。某些高级功能包括错误的自动聚合、优先级标记等。
- **SDK 支持**：在 SDK 方面，大多数竞品支持主流框架（React、Vue、Angular 等）和原生 JavaScript 集成。它们通常以低代码方式接入，能在不干扰项目的前提下实现监控功能，但在某些场景下可能存在性能开销。此外，Sentry 等产品还支持跨平台集成，适配移动端和后端应用。
- **数据可视化**：在数据可视化方面，竞品通常提供图表、仪表盘、报表生成等功能，使用户能够实时查看关键性能和异常指标的变化情况。以 Datadog 为例，它具备强大的自定义监控和告警配置能力，能满足企业对实时告警的需求。

项目差异化

- **高度可定制化**：相比市面上成熟的产品，本项目将提供更多自定义选项，尤其是在 SDK 的数据采集方式和后端数据处理流程上。企业可以根据自己的业务需求定制监控功能，避免不必要的性能开销。
- **内建系统与成本控制**：通过自主开发的数据处理和可视化系统，项目可以有效控制成本，减少对第三方监控服务的依赖，避免长期的服务费用。企业能够灵活扩展系统并避免外部服务中断带来的风险。
- **垂直化功能设计**：本项目聚焦前端性能和异常监控，提供更深度的性能优化建议和异常管理功能，特别是针对不同业务场景定制的告警策略和数据展示方式。

项目价值

1. **实时问题发现与处理**：通过构建实时的前端性能和异常监控系统，能够帮助企业在问题发生时立刻捕捉，便于快速定位问题根源并加以修复。通过实时告警机制，开发团队可以在问题发生的瞬间收到通知，从而减少对用户体验的负面影响。

2. **全生命周期监控**：项目将覆盖从数据采集、处理、存储到展示的全链路监控流程，为企业提供一个从开发到运维的一体化解决方案。团队可以通过系统随时掌握应用在用户环境中的真实表现，并基于监控数据对产品进行周期性优化。

3. **灵活集成与扩展**：提供对多种前端框架的支持，确保企业能轻松将监控集成到现有项目中。同时，系统具备可扩展性，能够根据业务需求轻松扩展监控项和数据采集点，为企业的未来发展提供灵活支持。

4. **数据驱动决策**：借助强大的数据可视化和分析功能，团队能够深入了解应用的性能和用户行为，利用数据驱动的决策为产品优化提供依据。项目的可视化平台支持多维度数据筛选和对比分析，使团队可以清晰地了解性能变化趋势，识别问题并进行预防性维护。

项目架构设计

该项目为全栈类型项目，包含 SDK、SDK 数据服务、SDK 数据 SaaS 共计三个应用，包含前后端实现共计 4 个项目。分别为：

- packages/* SDK 相关开发
- apps/backend/dsn-server 数据处理与存储服务后端
- apps/backend/monitor 数据监控可视化与 SaaS 平台后端
- apps/frontend/monitor 数据监控可视化与 SaaS 平台前端

项目工程化设计

pnpm monorepo 管理

- **项目管理**：采用 pnpm monorepo 进行管理，可实现多模块统一依赖管理、快速安装以及高效的包缓存。monorepo 管理方式还方便了跨项目依赖共享和版本同步更新，使开发团队可以集中管理各个模块。
- **文件结构**：项目的核心结构包括两个主要文件夹 `packages`、`apps`、`demos`：
  - **packages**：存放 SDK 相关模块，包含具体实现和工具函数等
  - **apps**：存放数据采集服务等业务应用代码
  - **demos**：用于演示不同应用的 SDK 使用

子模块开发与发布

- 每个模块均可单独开发、测试和发布，并且可以通过 `pnpm` 轻松地实现包之间的引用。这样可以确保每个模块在独立开发的同时依赖于其他模块的更新。
- 各模块包的详细介绍：
  - **browser**：包含核心 SDK 的基础功能，负责与浏览器进行交互并采集性能、异常数据。
  - **browser-utils**：为 `browser` 包提供工具函数库，包含数据处理、格式转换等常用工具。
  - **core**：项目的核心逻辑模块，包括数据采集、处理和上报等核心功能。
  - **react**：React 适配包，提供与 React 应用无缝集成的功能，如性能监控组件、错误边界等。
  - **types**：包含项目所有共享的 TypeScript 类型定义文件，便于维护和统一数据结构。
  - **utils**：包含通用工具函数，可供各模块调用，以提高代码复用性。
  - **vue**：Vue 适配包，提供与 Vue 应用的集成功能，支持性能监控和异常捕获。

技术选型

前端 SDK 技术选型与设计

架构与实现：

- 采用 Sentry 的架构设计模式，核心包实现基本功能，各适配包负责集成不同框架。
- 使用 **TypeScript** 提供静态类型支持，提升代码的可靠性和可维护性。
- SDK 适配多种框架（vanilla、React、Vue），实现监控采集数据，包括性能指标和异常数据。

监控内容：

- 使用 `PerformanceObserver` API 收集 web 性能指标，如 FCP、LCP、CLS。
- 使用 `window.onerror` 、 `window.onunhandledrejection` 捕获全局 JavaScript 异常，并允许开发者自定义异常采集。
- SDK 通过 HTTP 协议上报数据到后端服务，并支持插件化设计，方便未来扩展更多监控项和上报逻辑。

数据采集服务（apps/dsn-server）技术选型与设计

- **后端框架**：使用 `Nest.js` 构建后端服务。Nest.js 是一个基于 TypeScript 的 Node.js 框架，具有模块化、依赖注入等特性，便于构建高可维护性和高扩展性的应用

- **消息队列**：使用 `Kafka` 实现消息队列，定义不同 `topics` 以及 `partition` 提升服务稳定性与性能
- **数据处理和存储**：在应用中设计数据处理服务，进行数据清洗、聚合等操作。集成 Clickhouse 列式数据库进行数据存储，方便后续的数据查询和分析
- **通信协议**：数据采集服务通过 HTTP 接口接收 SDK 上报的数据

监控可视化与 SaaS 平台技术选型

- **前端框架**：选用 `React` 构建监控可视化与 SaaS 平台，结合 `shadcn/ui` 进行数据展示与界面设计
- **后端服务**：数据可视化模块也使用 `Nest.js` 构建后端服务。后端服务负责与数据库交互，提供 RESTful 接口供前端查询数据。主要消费 Clickhouse 数据

课程目标

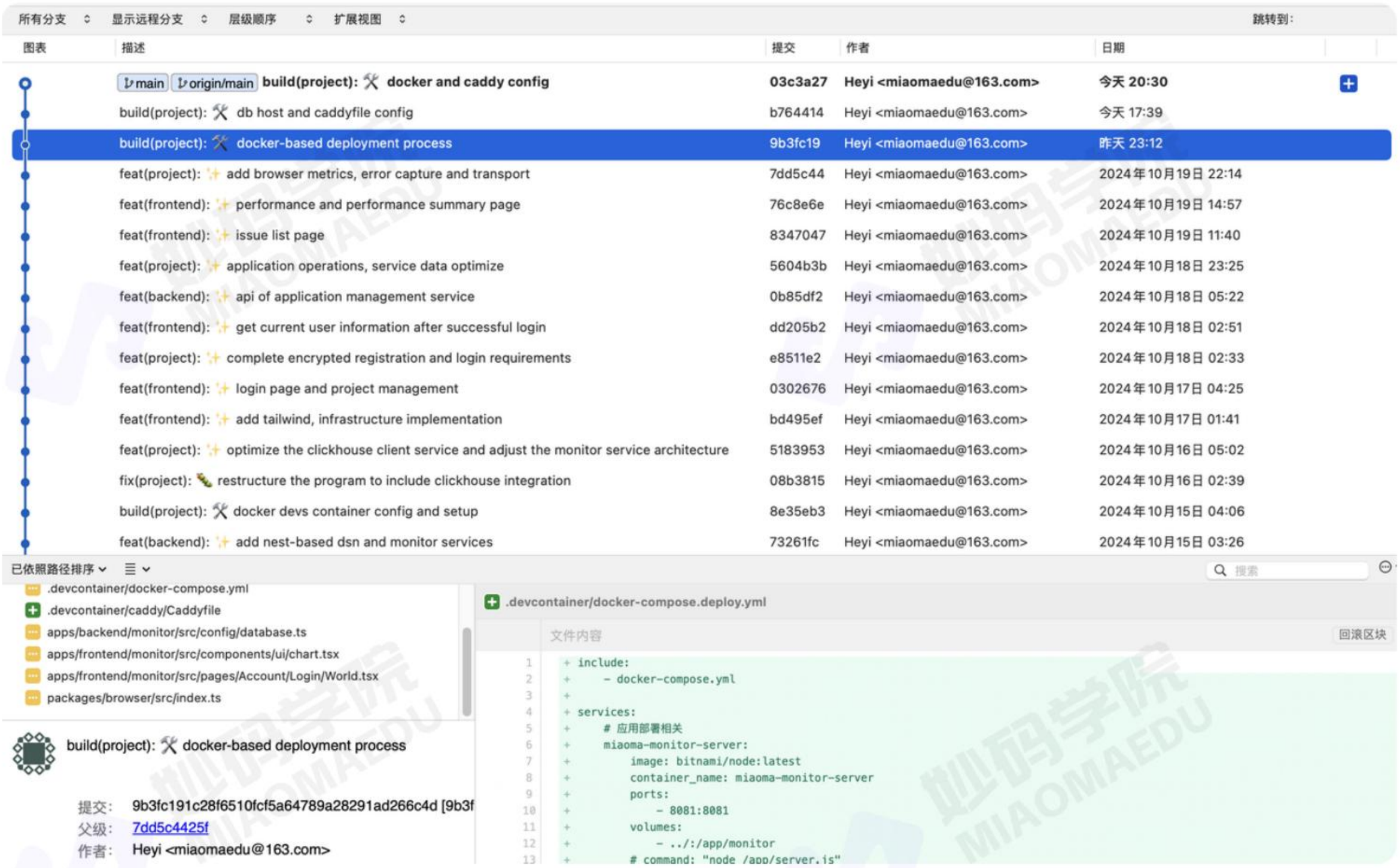
💡
- 初中级：
  a. **前端监控需求分析与方案评审**：能够充分理解前端异常与性能监控的需求，理解监控需求的重要性
  b. **项目架构设计分析**：掌握项目架构的设计原则，能够根据监控需求对项目进行架构规划，基于 monorepo 设计全栈项目
  c. **Performance 指标与 Web Vital 基础**：理解 Performance 指标的基本概念，掌握 web vital 指标，能够在项目中应用这些指标对性能进行基础监控
  d. **性能与异常指标分类与采集**：掌握性能指标（如加载时间、交互响应时间）和异常指标（如错误率、异常次数）的分类，并能够完成基础数据采集
- 高级：
  a. **前端监控方案的深入理解与优化**：深入理解并分析前端异常与性能监控需求，优化现有监控方案，提出有针对性的解决方案
  b. **高级项目架构设计与实施**：具备从零开始设计复杂前端监控架构的能力，包括将监控功能集成到项目中，同时遵循性能和扩展性的最佳实践，插件化机制的运用
  c. **Performance 指标优化与高级分析**：能够使用 web vital 指标对性能问题进行深度分析
  d. **数据采集与上报流程优化**：熟悉性能和异常数据采集的高级方法，能够设计通用数据协议，封装上报数据

- 初中级：
  a. **深入 Web Vital 运用**：深入理解 Performance 指标的基本概念，掌握 web vital 指标，能够在项目中场景性能指标
  b. **掌握指标统一协议设计思路**：掌握性能指标（如加载时间、交互响应时间）和异常指标（如错误率、异常次数）的分类，并根据设计统一上报数据协议
  c. **理解无痕埋点方案与实现**：理解无痕埋点的方案设计与代码实现，能根据项目所需，采集指定事件信息
  d. **掌握数据上报队列设计**：掌握数据上报实现与数据上报队列设计以支持弱网与网络异常处理，提升监控稳定性
  e. **了解监控 SDK 服务基础架构设计**：掌握基于 nest 的监控服务基础架构设计，了解核心技术点：nest、docker、clickhouse
- 高级：
  a. **深入理解指标数据协议设计**：深入理解不同前端监控指标要素与数据打包协议设计
  b. **掌握 SDK 与监控平台数据流向**：掌握数据从 SDK 采集到 dsn 服务、入库 clickhouse、数据统计、监控平台服务数据消费全流程设计
  c. **掌握监控 SDK 服务设计**：掌握基于 nest 的监控服务架构设计，掌握核心技术点：nest 模块设计、docker compose 服务编排、clickhouse 存储表与物化视图应用

- 初中级：
  - **掌握 Docker Compose 的基础配置与使用**：理解如何编写 `docker-compose.yml` 文件，部署 Node.js 应用和 Caddy 反向代理服务。
  - **熟悉 Docker 容器的基本管理**：学会启动、停止、重启容器，并使用日志工具排查运行问题。
  - **掌握服务端与静态资源的托管**：配置 Caddy 处理 HTTP/HTTPS 请求及托管静态文件。
  - **理解阿里云服务器与域名的基础使用**：能够连接远程服务器，配置域名解析等。
- 高级：
  - **深入掌握多容器架构的搭建与优化**：配置 Kafka、ClickHouse、PostgreSQL 和 Redis 等服务，并解决服务间依赖问题。
  - **学会 Caddy 的进阶配置**：掌握路径重写、API 代理、静态文件处理与缓存控制策略的应用。
  - **优化 Docker 网络性能**：使用阿里云镜像加速器或 SSH 隧道解决 Docker 拉取镜像缓慢的问题。

- **掌握云端与本地环境的差异化部署**：灵活管理本地与线上服务的配置，并通过 Docker Compose 快速部署和调试完整系统。
- **掌握阿里云效 CI/CD**：掌握通过阿里云效定义构建流水线，包括代码检测卡点，代码构建，代码部署全流程

代码 commit 截图



演示地址

https://monitor.miaomaedu.com

性能与监控平台 | 妙码学院

“企业级项目”指采用企业常用技术栈及开发流程的教学模拟项目，不等同于商业生产环境使用的项目