面向工业应用的柔性可扩展的边缘计算软件系统

软件架构文档

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 15/10/2021 | 0.1 | 软件架构文档初稿 | 董彦君 |
| 26/10/2021 | 1.0 | 软件架构文档第一版 | 董彦君、张俸铭 |
| 4/11/2021 | 2.0 | 根据评审建议完善质量设计和部署视图 | 董彦君 |
|  |  |  |  |

目录

1. 简介 6

1.1 目的 6

1.2 范围 6

1.3 定义、首字母缩写词和缩略语 6

1.4 参考资料 6

1.5 概述 6

2. 用例视图 7

2.1 云端管理人员用例 7

2.1.1 边缘端管理 7

2.1.2 流程配置与下发 7

2.1.3 协议规则管理 7

2.1.4 数据管理 7

2.1.5 机器学习模型管理 7

2.2 边缘端管理人员用例 7

2.2.1 设备注册与管理 7

2.2.2 生产流程调度 7

2.2.3 协议规则应用 7

2.2.4 设备数据采集与处理 7

2.2.5 机器学习模型配置 8

2.3 安全管理人员用例 8

2.3.1 用户权限管理 8

3. 逻辑视图 8

4. 部署视图 9

5. 实现视图 10

6. 技术视图 12

6.1 技术栈 12

6.2 编程语言 13

6.3 开发工具 13

6.4 开发框架 13

7. 数据视图 13

8. 质量属性的设计 14

8.1 易用性 14

8.1.1 培训时间 14

8.1.2 易用性需求 14

8.2 可靠性 14

8.2.1 可用性 14

8.2.2 平均故障间隔时间（MTBF） 14

8.2.3 平均修复时间（MTTR） 14

8.2.4 最高错误或缺陷率 14

8.3 性能 14

8.3.1 响应时间 14

8.3.2 吞吐量 14

8.3.3 数据量 14

8.4 可支持性 14

8.4.1 兼容性 14

8.4.2 可维护性 15

8.4.3 编码标准 15

8.4.4 命名约定 15

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 范围

本文档适用于高级软件开发与管理课程第一小组正在开发的面向工业应用的柔性可扩展的边缘计算软件系统，将影响和指导该系统的开发过程。

## 定义、首字母缩写词和缩略语

|  |  |  |
| --- | --- | --- |
| 定义 | 缩略语 | 含义 |
| 边缘计算软件系统 | ECSS | 在靠近数据源（设备端）采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供有限的低时延服务，并接入云平台进行管理和计算的软件系统。 |
| 业务流程管理标记 | BPMN | 一套图形化的流程建模标准，用于以业务流程模型详细说明各种业务流程。 |
| 协议集成系统 | PIS | 将各类协议或规则集成到相互关联的、统一和协调的系统之中，使资源达到充分共享，实现集中、高效、便利的管理。 |
| 工业异构设备 | IHD | 工业生产场景下应用不同通信协议、使用不同数据解析方法的高度异构化设备。 |
| 容器 |  | 一种运行于独立命名空间的进程，为应用提供隔离的环境 |

## 参考资料

1. 《面向工业应用的柔性可扩展的边缘计算软件系统—立项建议书》, 高级软件开发与管理课程（2021秋季）第1小组。
2. 《面向工业应用的柔性可扩展的边缘计算软件系统需求规约文档》, 高级软件开发与管理课程（2021秋季）第1小组。
3. RUP的软件结构文档模板
4. 《边缘计算参考架构3.0》, 边缘计算产业联盟、工业互联网产业联盟。
5. 《5G 与工业互联网融合应用发展白皮书》, 工业互联网产业联盟、5G 应用产业方阵。

## 概述

本文档将明确面向工业应用的柔性可扩展的边缘计算软件系统的架构表示方式、架构的模板和约束、大小和性能、并通过以下一系列视图来表示平台的软件架构：用例视图、逻辑视图、部署视图、实现视图、技术视图和数据视图。

# 用例视图

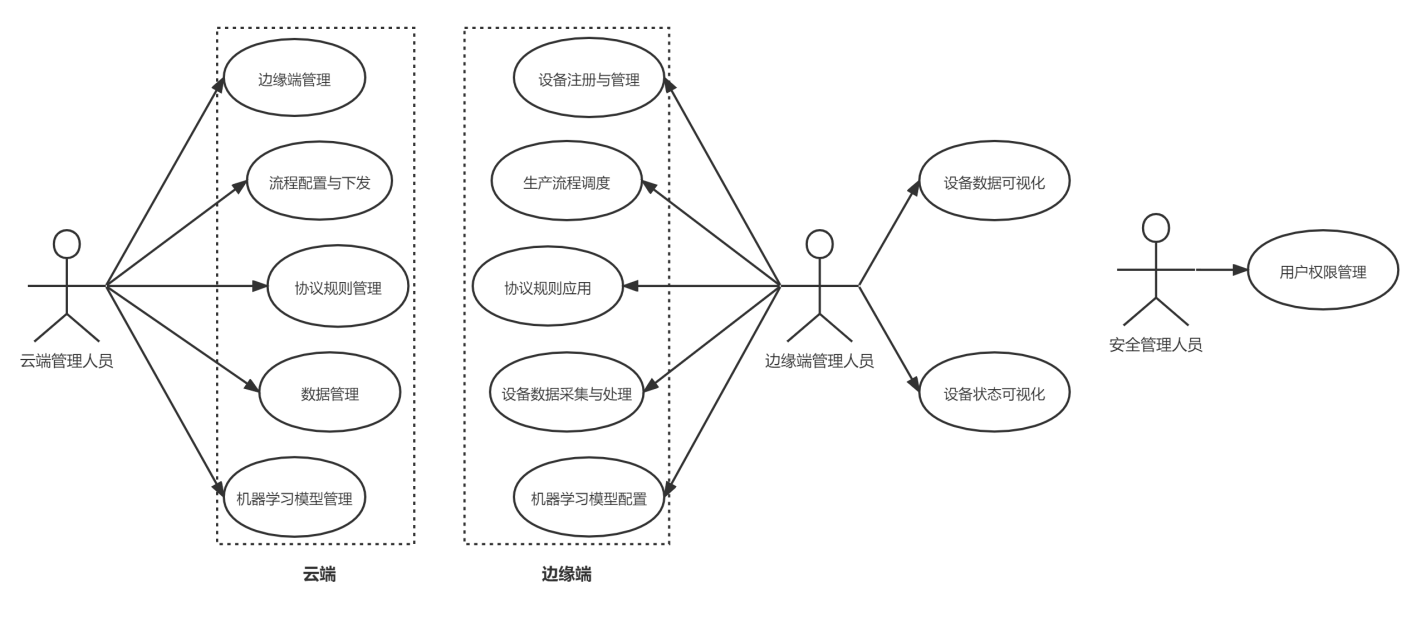


图1 用例视图

## 云端管理人员用例

### 边缘端管理

云端管理人员注册、编辑、查看和删除边缘节点。

### 流程配置与下发

云端管理人员自定义生产流程，将流程中的任务节点与相应设备功能标签和机器学习模型进行绑定，形成生产流程实例。

云端管理人员将流程实例与边缘端绑定，下发至边缘端。

### 协议规则管理

云端管理人员配置与设备型号对应的命令转换规则。

### 数据管理

云端管理人员查看、下载、删除边缘端上传的数据包。

### 机器学习模型管理

云端管理人员上传机器学习模型，配置环境参数、训练参数和模型更新周期，保存模型。

## 边缘端管理人员用例

### 设备注册与管理

边缘端管理人员注册、编辑、查看和删除设备。

### 生产流程调度

边缘端人员为流程中的任务节点指定执行设备，为流程中的条件判断节点选择协议规则，开启和停止流程。

### 协议规则应用

边缘端人员查看边缘端已有设备的协议规则。

### 设备数据采集与处理

边缘端人员配置设备正确的数据上传内容、周期和方式。

### 机器学习模型配置

边缘端人员配置运行模型使用的脚本路径，配置推理结果与条件判断节点的接口，保存配置内容。

## 安全管理人员用例

### 用户权限管理

安全管理人员根据用户名或权限等级查询用户、添加新用户、编辑用户权限等级。

# 逻辑视图

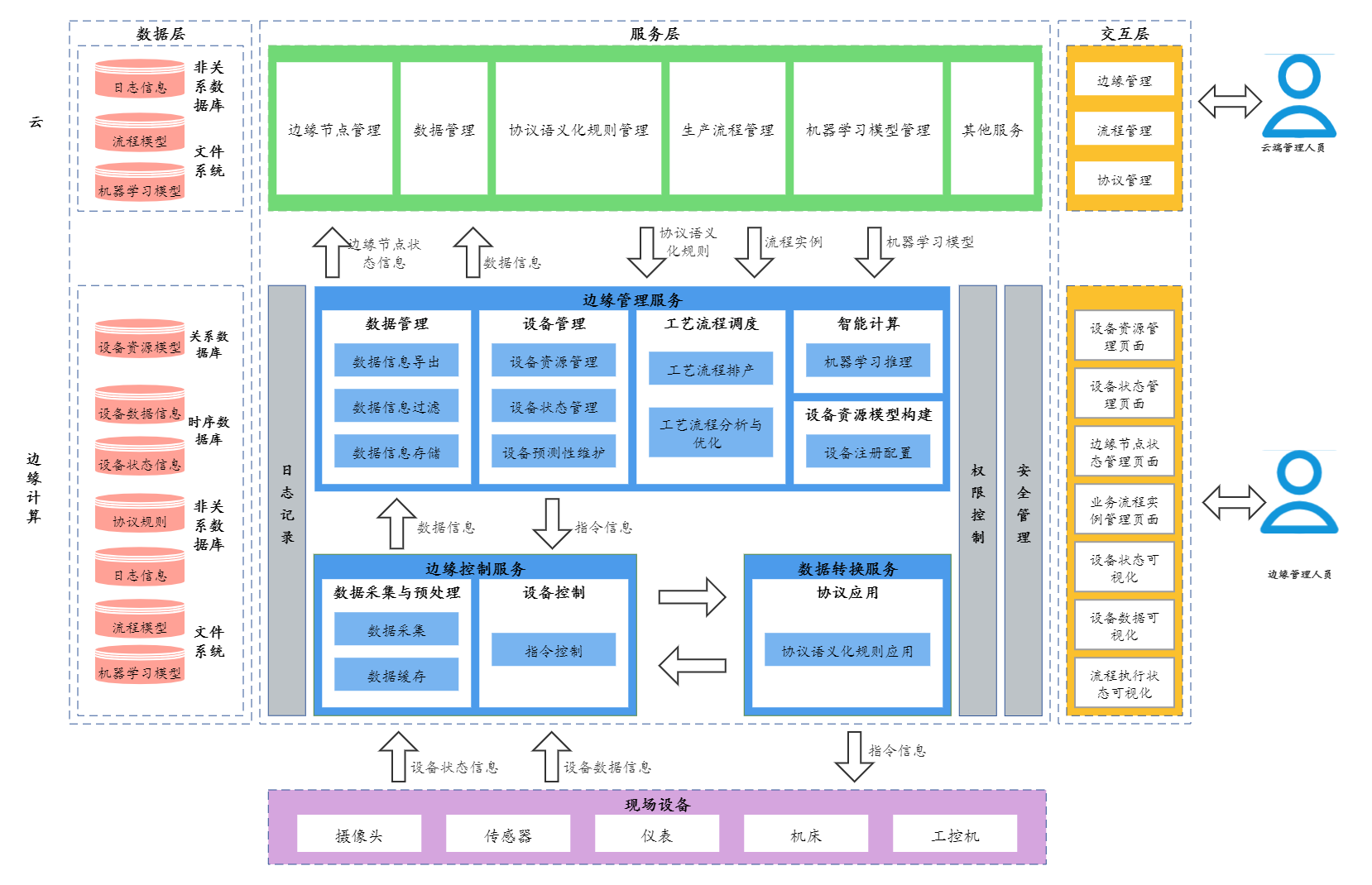


图2 逻辑视图

逻辑视图基于三层架构，云端和边缘端均设计为数据层、服务层和交互层。

1. 数据层：数据层存储的数据资源为与数据视图中对应的持久性数据，使用MySQL，MongoDB，Redis，IoTDB等数据库分别存储结构化、非结构化、临时性和工业设备数据，使用HDFS文件系统存储机器学习模型、数据包、脚本等文件数据，为服务层提供数据支撑。
2. 服务层：服务层是整个系统的核心支撑部分，通过对不同功能模块的实现，服务层响应交互层的请求，提供云端和边缘端的相关功能服务。同时云端和边缘端通过相互调用相应的Api完成数据的传递。
3. 交互层：用户在交互层与系统进行交互。

# 部署视图

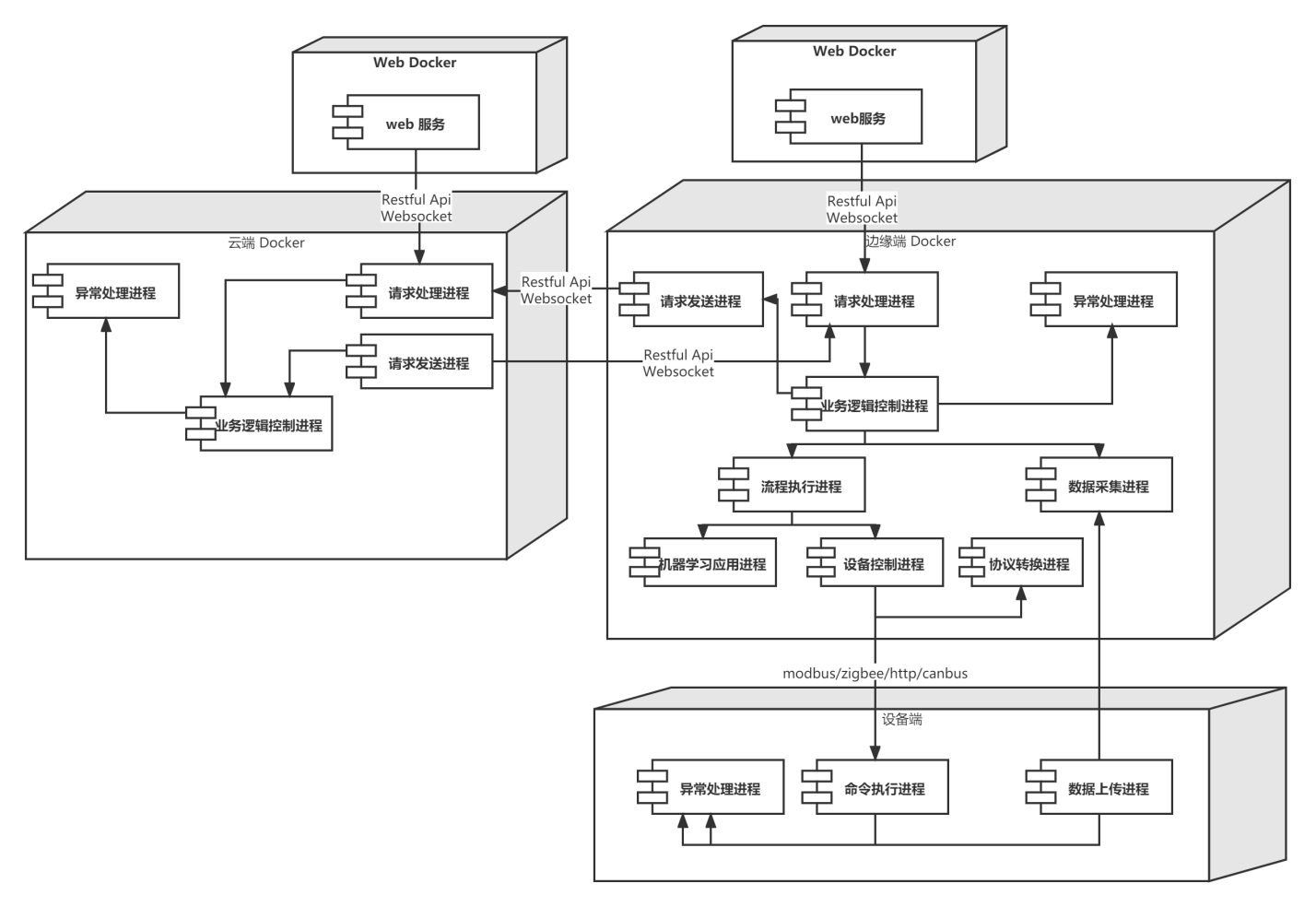


图3 部署视图

1. Web端

云端和边缘端管理人员可以在Web页面进行边缘端注册、设备端注册等操作，通过Http协议与云端和边缘端服务器进行通信，并实时更新云端与边缘端服务器通过Websocket协议推送的消息。云端的Web应用部署在远程服务器的Docker中，边缘端的Web应用部署在生产线附近的一台独立服务器的Docker中

1. 云端

云端处理前端和边缘端发送的请求，通过业务逻辑控制进程提供相应的功能，异常处理进程处理全局异常，请求处理进程解析请求并包装回传响应信息。云端通过调用边缘端的相应api向边缘端下发流程、机器学习模型、协议规则。云端部署在远端服务器的Docker中。

1. 边缘端

边缘端处理前端和云端发送的请求，通过业务逻辑控制进程提供相应的功能，异常处理进程处理全局异常，请求处理进程解析请求并包装回传响应信息。边缘端通过调用云端的相应api向云端上传设备功能标签、数据包。边缘端通过Modbus、Zigbee、Http、Canbus协议与现场工业设备进行通信。边缘端部署在靠近生产线的一台独立服务器的Docker中。

1. 设备端

设备端是运行在智能工业设备上的软件，通过Modbus、Zigbee、Http、Canbus协议与边缘端通信。

# 实现视图

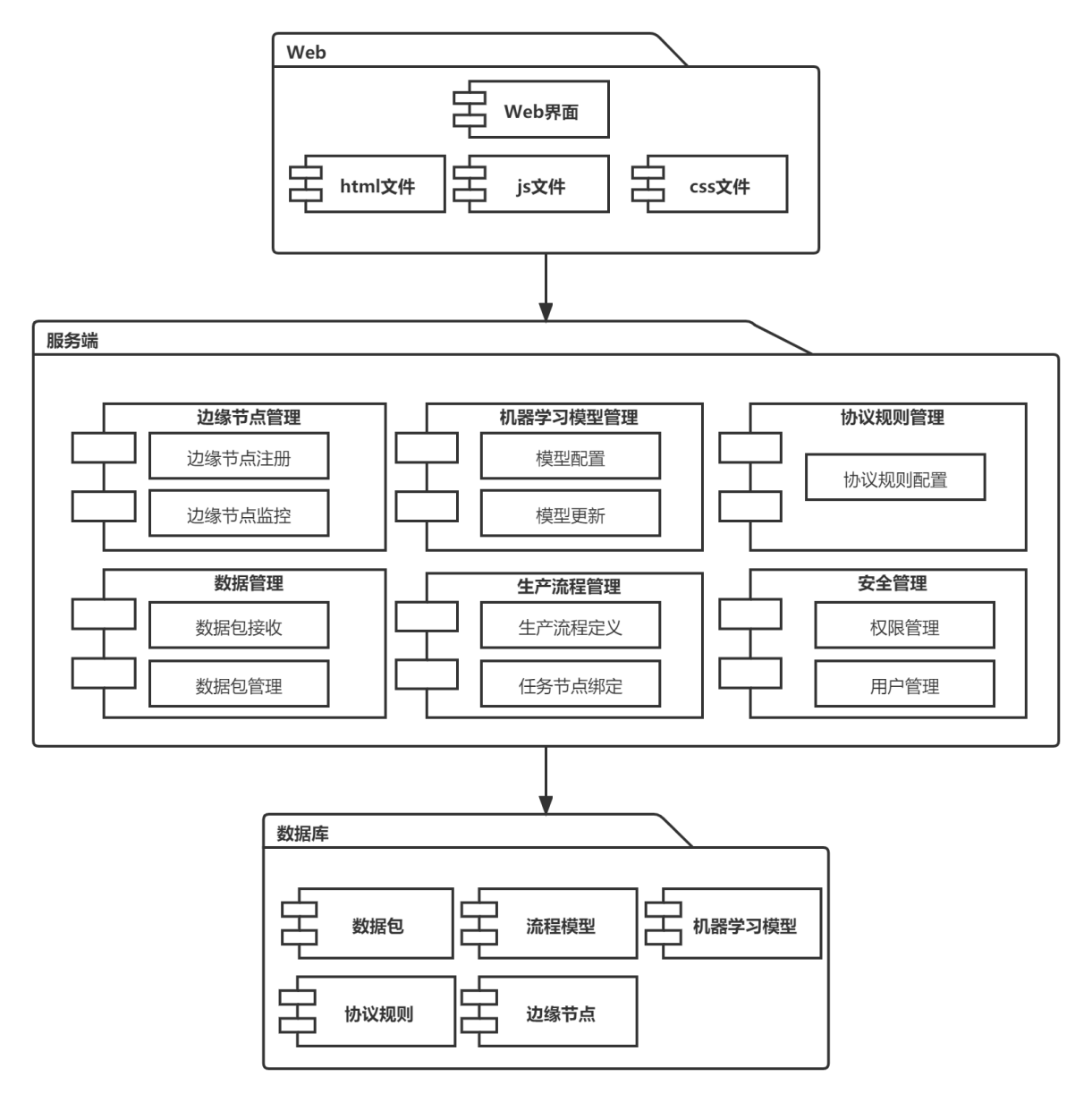


图4 云端实现视图

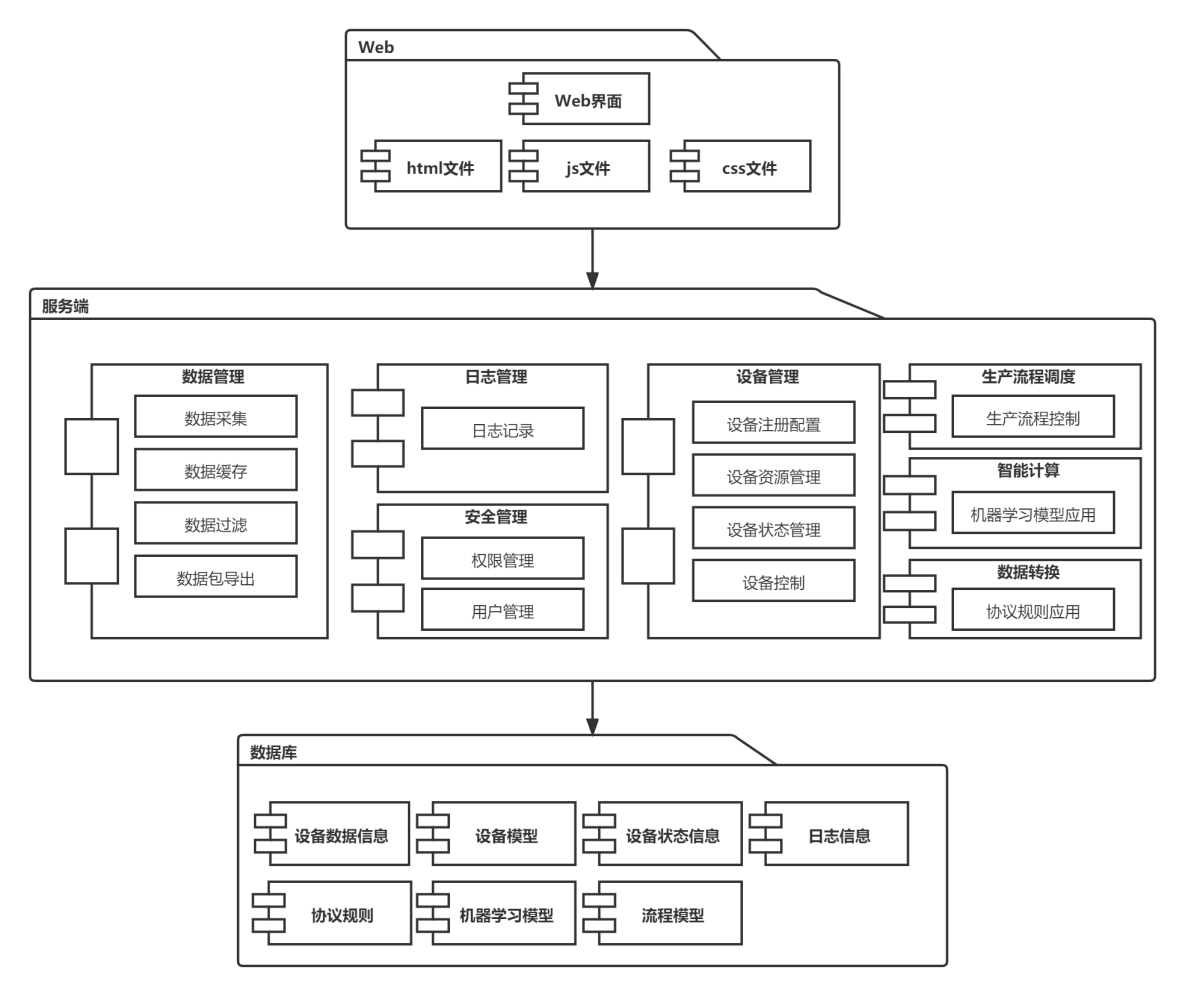


图6 边缘端实现视图

云端和边缘端都主要分为三大部分，web界面，服务端和数据库。Web使用js，html，css进行界面开发。在服务端将多个实现单元进行分类。云端分为边缘节点管理、数据管理、机器学习模型管理、生产流程管理、协议规则管理、安全管理这6个子系统进行开发，数据库被分为数据包、流程模型、机器学习模型、协议规则、边缘节点分别进行设计和开发。边缘端分为数据管理、日志管理、安全管理、设备管理、生产流程调度、智能计算、数据转换7个子系统进行开发，数据库被分为设备数据信息、设备模型、设备状态信息、日志信息、协议规则、机器学习模型、流程模型分别进行设计和开发。

# 技术视图

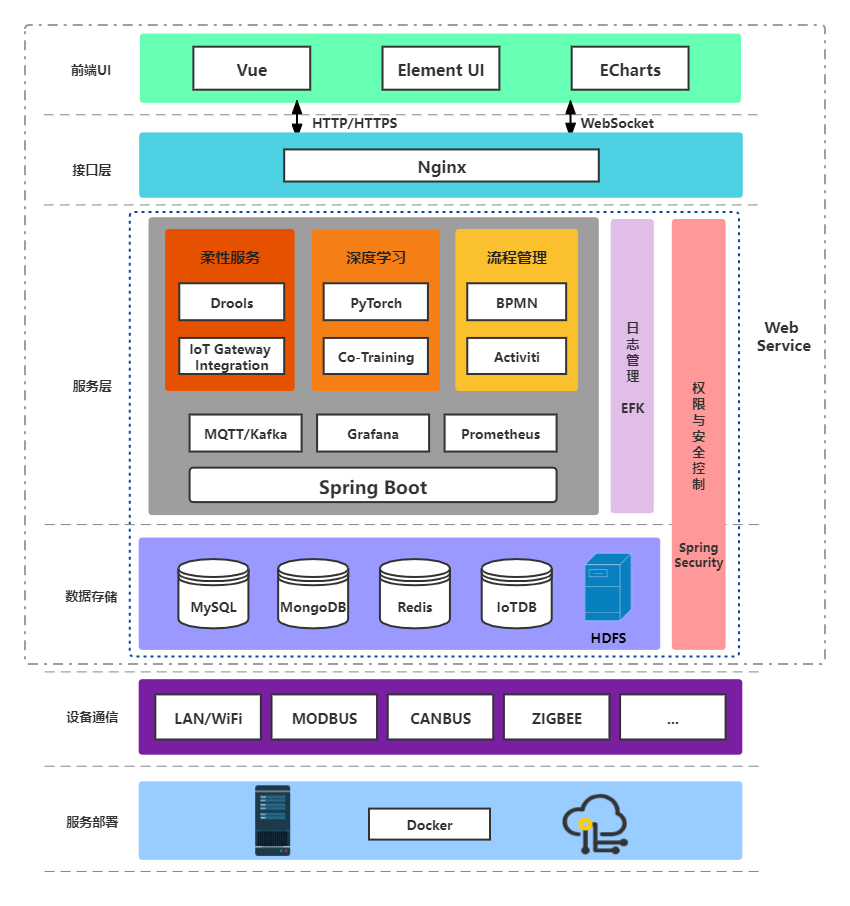


图7 技术视图

## 技术栈

在横向技术栈上可以分为数据存储、服务层、接口层、前端UI等几个部分：

1. 数据存储：使用MySQL，MongoDB，Redis，IoTDB等数据库分别存储结构化、非结构化、临时性和工业设备数据，使用HDFS文件系统存储机器学习模型、数据包、脚本等文件数据。
2. 服务层：后端的核心技术栈。整体使用Spring Boot进行后端开发，使用MQTT进行边缘端至云端的数据上传，并通过Grafana+Prometheus对服务资源状态进行监控。在细粒度的技术选型上：
   1. 使用Drools作为逻辑推理工具，服务于用户可配置的协议转换规则。
   2. 在设备端使用物联网网关集成，提供设备指令下发和数据传输协议的可扩展性。
   3. 机器学习模块使用PyTorch框架，并提供与Tensorflow进行模型转换的方法。
   4. 通过云边协同训练的方式，在模型滞后影响不大的情况下保证边缘端的低时延响应。
   5. 通过BPMN+Activiti实现用户可配置的流程管理和下发，提高系统的适用性。
3. 接口层：使用HTTP/HTTPS和WebSocket进行服务请求或有状态的长连接；使用Nginx作为反向代理，解决跨域访问问题。
4. 前端UI：使用Vue框架，通过Element UI和ECharts等提供的组件提高开发效率和界面美观度、易用性。

在纵向技术栈上，通过EFK进行服务层的日志管理；通过Spring Security进行权限管理，保证数据和服务的安全。

本系统适配和支持工业设备以LAN/WIFI，MODBUS，CANBUS，ZIGBEE等多种工业标准通信协议进行接入。

通过Docker镜像完成边缘端和云端的部署，保证服务独立稳定、容易修复。

## 编程语言

本项目开发在框架上使用前后端分离的B/S架构。后端业务逻辑使用Java开发，浏览器前端使用HTML/CSS + JavaScript进行开发。

## 开发工具

项目后端使用IntelliJ IDEA作为开发IDE，使用maven作为项目构建生命周期管理工具。前端使用VS Code作为开发IDE。

## 开发框架

项目后端基于Spring Boot框架进行开发，前端基于Vue框架进行开发，并采用RESTful API作为接口设计规范。

# 数据视图

# 数据视图 (6)

图8 数据视图

平台主要使用MongoDB和IoTDB储存持久化数据，数据视图如上图所示。

设备（Device）及其关联的一系列信息，包括设备型号（DeviceModel）、功能标签（Task）、协议类型（Protocol）、设备状态（DeviceState）、设备属性（DeviceProperty）、设备事件（DeviceEvent）、设备功能（DeviceFunction）。其中，设备属性（DeviceProperty）对应着设备所需采集的数据项，这涉及到对应不同协议的采集器（Collector）和调度器（Scheduler），设备事件（DeviceEvent）对应着上传云端的数据，同样涉及到调度器（Scheduler），设备功能（DeviceFunction）对应着系统对设备下发指令，涉及命令执行器（Actutor）。采集到的数据项（Properties）形成数据包（DataPacket）发送至云端。

边缘端（Edge）注册到云端，与生产流程（Process）和机器学习模型（MachineLearningModel）绑定，同时在执行生产流程（Process）的过程中产生日志记录（Log）。协议规则（ConversionRule）被注册到云端，与功能标签（Task）进行绑定。

用户数据（User）被用于进行权限管理。

# 质量属性的设计

## 易用性

前端界面排版清晰，风格统一。表单输入时，当用户输入错误，界面会有错误提示，引导用户修改。

## 可用性

异常的捕获和抛出：对重要的地方添加异常的捕获，使得出现异常后可以及时的反馈信息，不会造成系统奔溃和停止，提高可用性。

使用SpringBoot框架，增加事务，并对事务进行异常的捕捉，在捕捉到异常后， 会对事务进行回滚操作。

## 性能

采集设备数据时使用并发。

## 可支持性

### 可维护性

以基于UML的面向对象方法作为软件开发方法，以实现整个系统的灵活可扩展与高可维护性。

### 编码标准

前端：符合ESLint代码规范。

后端：符合Google Java Style代码规范。

### 命名约定

前端符合 Airbnb JavaScript Style Guide 指南的要求。

后端：《Effective Java》第三版的命名约定。

## 可测试性

使用主流框架进行前后端开发，将不同的代码写在不同的层，便于测试各层的代码，易于发现问题。