冷链设备监控平台概述

学习目标：
1. 能够了解物流行业的背景知识
2. 能够了解冷链行业的背景知识
3. 能够了解冷链监控的业务背景
4. 能够掌握冷链监控的业务架构
5. 能够掌握冷链监控的技术架构
6. 能够掌握冷链监控平台的数据架构
7. 能够掌握此技术架构的适用场景
8. 能够掌握冷链监控的中间件

1. 业务背景

1.1 物流行业

物流是指从供应地向接收地的实体流动过程。根据实际需要，将运输、储存、装卸、搬运、包装、流通加工、配送、信息处理等基本功能实施有机结合。

” 评价物流体系有五个主要因素，它们是：

- 品质： 指物流过程中，物料的品质保持不变
- 数量： 指符合经济性的数量要求和运输活动中往返运输载重尽可能满载等
- 时间： 指以合理费用及时送达为原则做到的快速
- 地点： 指选择合理的集运地及仓库，避免两次无效运输及多次转运
- 价格：指保证质量及满足时间要求的前提下尽可能降低物流费用

物流行业包括：

- 电商系
- 快递系
- 物流产业园
- 货代系
- ...

1.2 冷链行业

冷链（cold chain）是指某些食品原料、经过加工的食品或半成品、特殊的生物制品和药品在经过收购、加工、灭活后，在产品**加工、贮藏、运输、分销和零售、使用**过程中，其各个环节始终处于产品所必需的**特定低温环境**下，减少损耗，防止污染和变质，以保证产品食品安全、生物安全、药品安全的特殊供应链系统。

冷链由冷冻加工、冷冻贮藏、冷藏运输及配送、冷冻销售四个方面构成：

1. **冷冻加工**：包括肉禽类、鱼类和蛋类的冷却与冻结，以及在低温状态下的加工作业过程；也包括果蔬的预冷；各种速冻食品和奶制品的低温加工等。在这个环节上主要涉及冷链装备有冷却、冻结装置和速冻装置
2. **冷冻贮藏**：包括食品的冷却储藏和冻结储藏，以及水果蔬菜等食品的气调贮藏，它是保证食品在储存和加工过程中的低温保鲜环境。在此环节主要涉及各类冷藏库/加工间、冷藏柜、冻结柜及家用冰箱等等。
3. **冷藏运输**：包括食品的中、长途运输及短途配送等物流环节的低温状态。它主要涉及铁路冷藏车、冷藏汽车、冷藏船、冷藏集装箱等低温运输工具。在冷藏运输过程中，温度波动是引起食品品质下降的主要原因之一，所以运输工具应具有良好性能，在保持规定低温的同时，更要保持稳定的温度，远途运输尤其重要。
4. **冷冻销售**：包括各种冷链食品进入批发零售环节的冷冻储藏和销售，它由生产厂家、批发商和零售商共同完成。随着大中城市各类连锁超市的快速发展，各种连锁超市正在成为冷链食品的主要销售渠道，在这些零售终端中，大量使用了冷藏/冻陈列柜和储藏库，由此逐渐成为完整的食品冷链中不可或缺的重要环节。

1.3 冷链监控

因为在冷链物资流转过程中，对物资的生产、运输、存储、销售时的温度、湿度等各种指标要求比较高，为了减轻人工成本、提高设备检测的可靠性，需要专门的软件系统来进行系统化、自动化操作。

冷眼物流设备监控系统是一款应用于对食品、药品冷链仓储、运输的环节中针对温度、湿度、电量等进行监控、预警和统计分析的系统。

实现了冷链监控环节的：

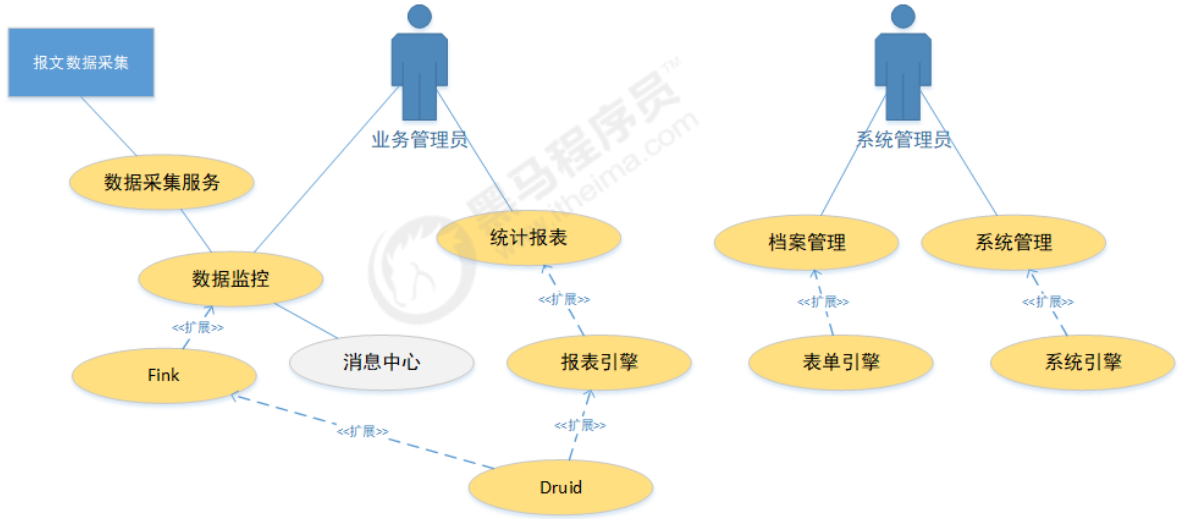
- 数据采集自动化
- 监控指标配置化
- 预警通知自动化
- 统计分析可视化

从而提升了生鲜、药品仓储、运输的安全管控水平，增强了政府、企业对业务各环节的了解和管理。

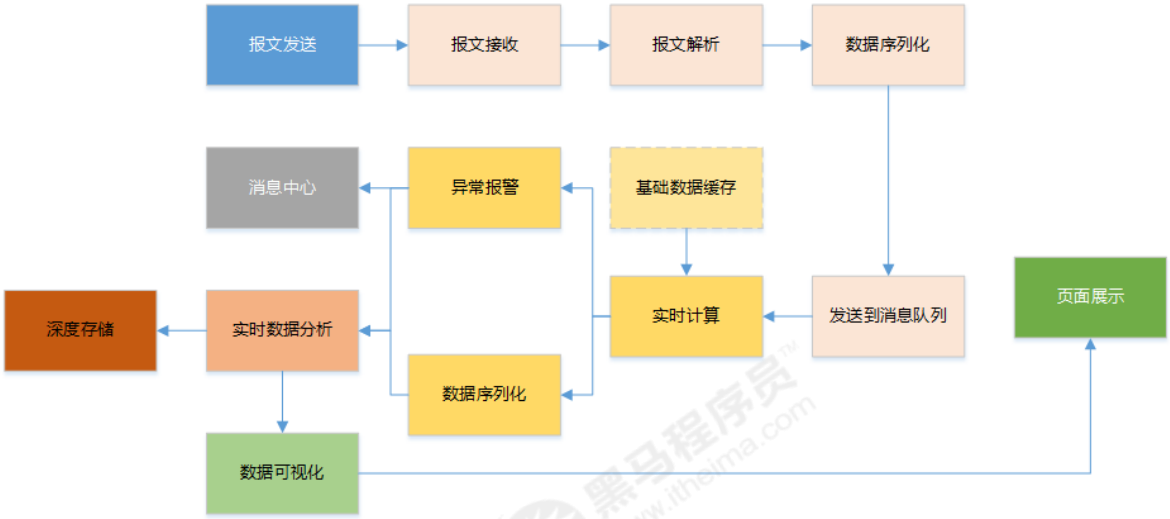
2. 系统架构

2.1. 系统用例图

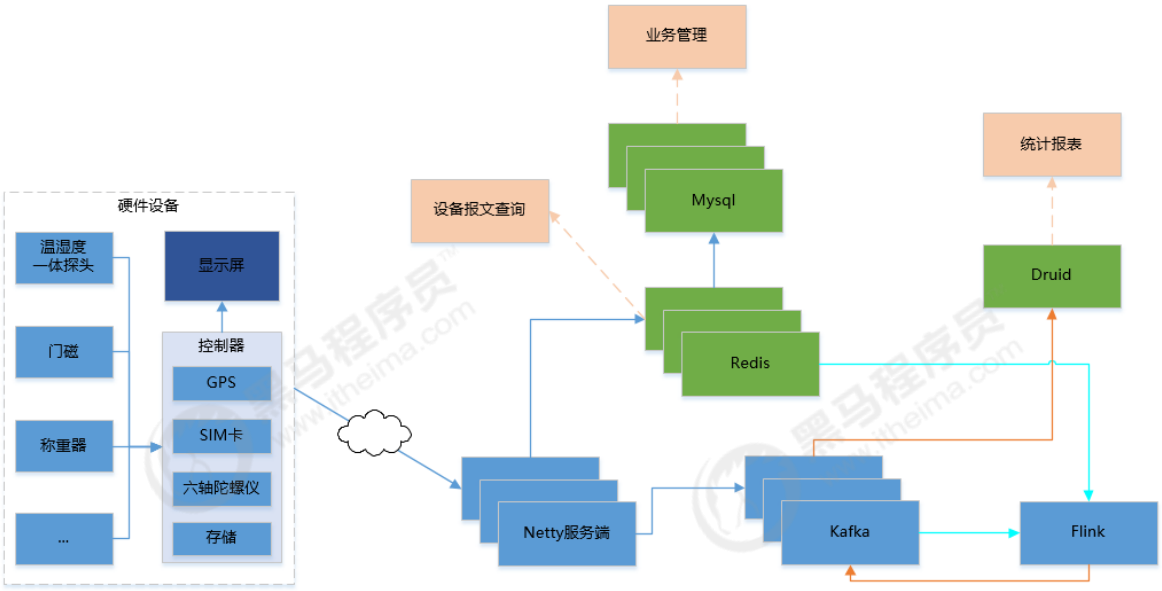
系统用例图是为了方便的说明系统中有多少个角色、每个角色拥有什么样的操作的图，如下图所示：



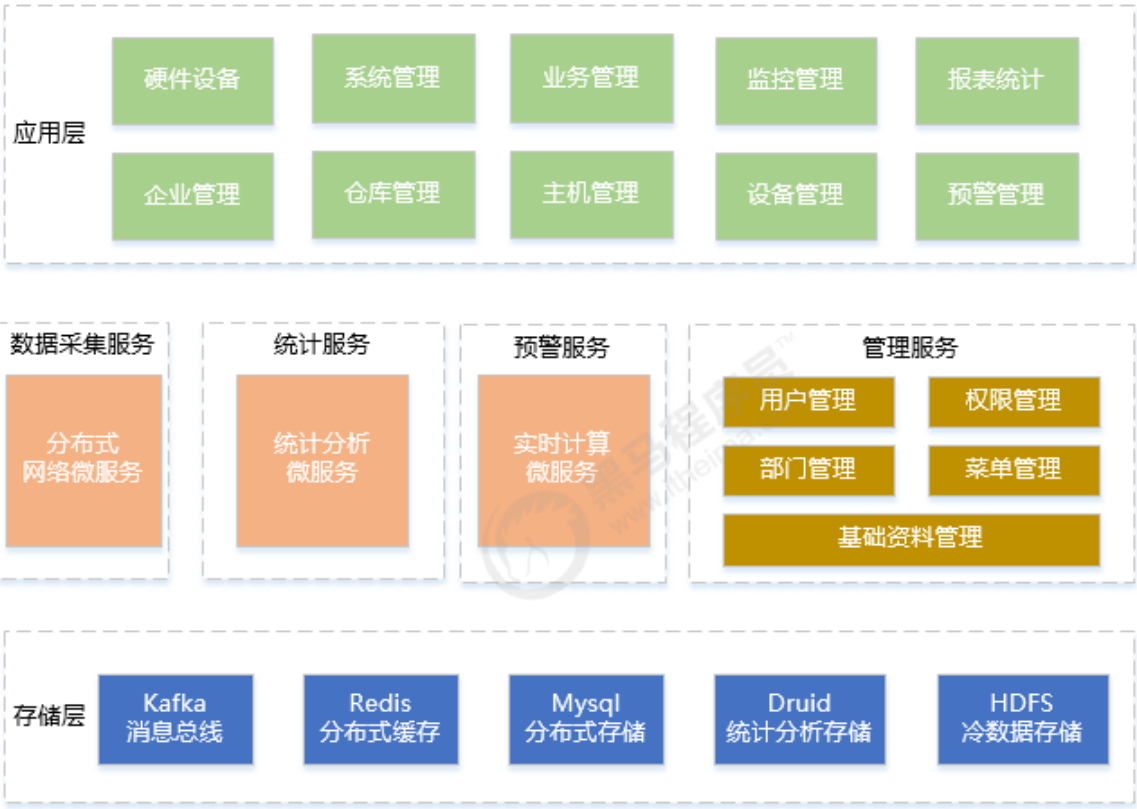
2.2. 任务流程图



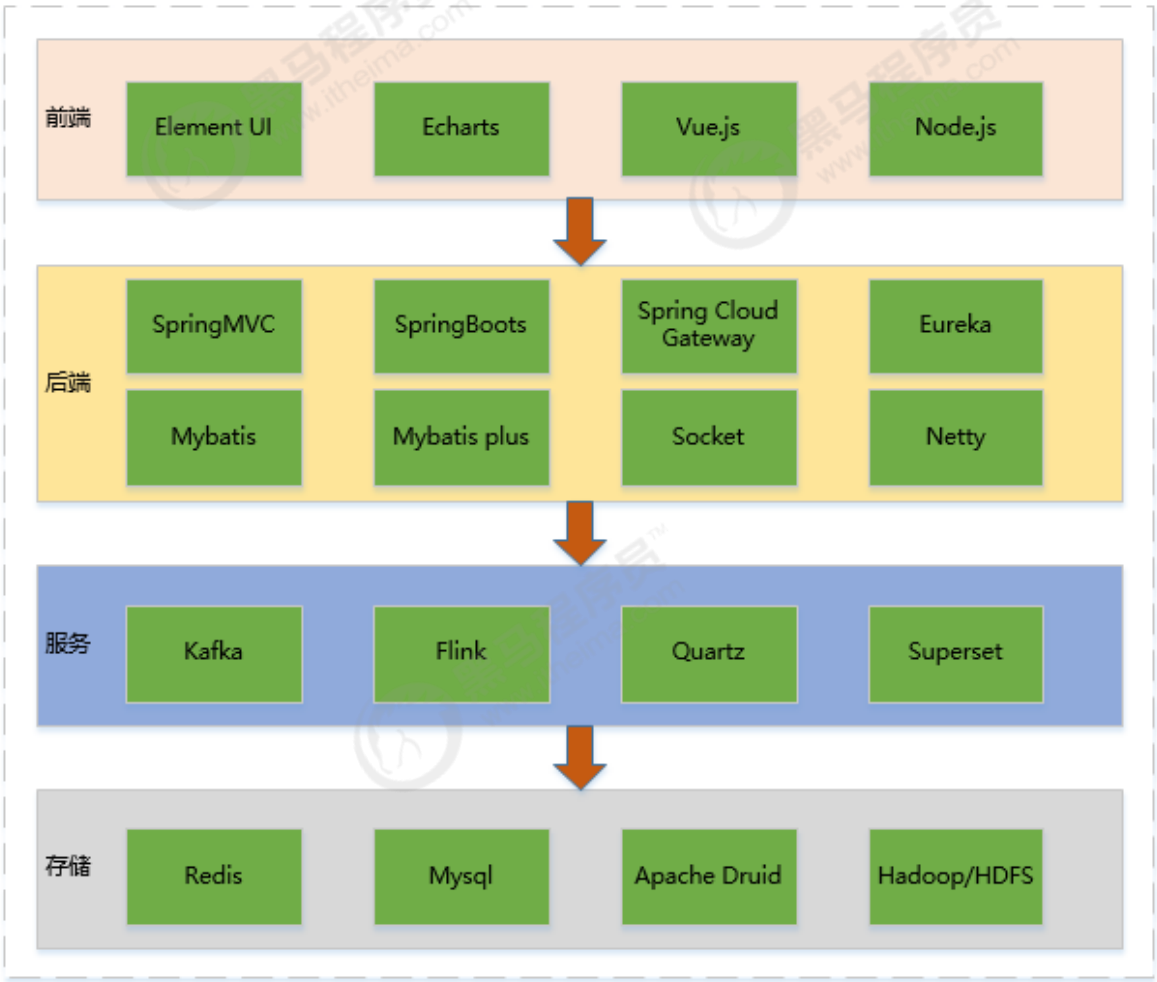
2.3. 数据流程图



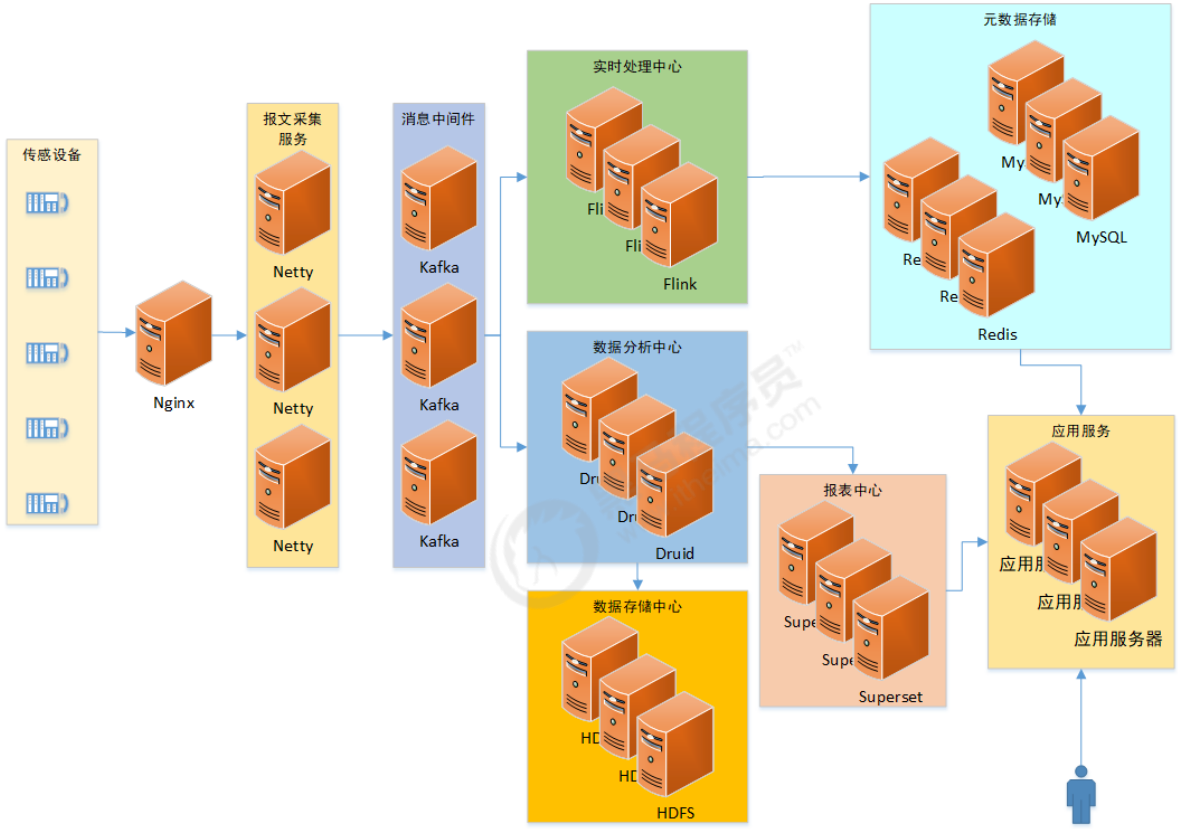
2.4. 系统架构图



2.5. 技术架构图

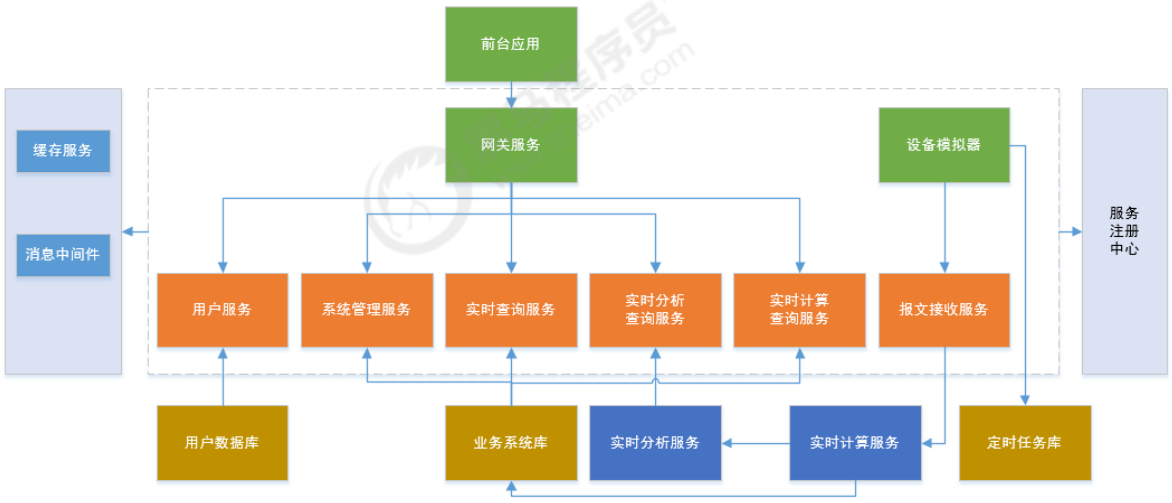


2.6. 物理部署图



2.7. 微服务拆分

```
cold-chain-monitor
├── cold-user              # 用户服务
├── cold-admin             # 业务管理、系统管理服务
└── cold-common            # 对象实体、公共组件
└── cold-monitor           # 实时数据查询
└── cold-druid             # 历史数据查询
└── cold-eureka            # Spring cloud服务注册中心
└── cold-gateway           # Spring cloud网关
└── cold-flink             # 实时数据处理
└── cold-jobs              # 分布式任务调度（硬件模拟）
└── cold-netty-server      # 设备报文接收服务

cold-ui                    # 前台页面
```



3. 系统服务

环境：

- 本地环境
- 开发环境
- 测试环境
- UAT环境
- 生产环境

服务：

- 基础组件
- 大数据服务组件
- 业务应用

3.1 系统环境

3.1.1 基础组件

- jdk 8 +

  Oracle JDK下载页面:

  ```
  https://www.oracle.com/technetwork/java/javase/archive-139210.html
  ```

  在下载页面选择Java SE 8 rpm文件， 下载过程中需要登录oracle网站，安装jdk：

  ```
  [root@node2-vm06 ~]# cd /opt/software/

  [root@node2-vm06 software]# wget -c
  https://download.oracle.com/otn/java/jdk/8u202-
  b08/1961070e4c9b4e26a04e7f5a083f551e/jdk-8u202-linux-x64.rpm?
  AuthParam=1561169291_970b059c42541ddd27d399473a89de61

  [root@node2-vm06 software]# mv jdk-8u202-linux-x64.rpm\?
  AuthParam\=1561169291_970b059c42541ddd27d399473a89de61 jdk-8u202-linux-
  x64.rpm

  [root@node2-vm06 software]# rpm -ivh jdk-8u202-linux-x64.rpm

  [root@node2-vm06 software]# java -version
  java version "1.8.0_202"
  Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
  Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
  ```

  设置JAVA_HOME环境变量：

  ```
  [root@node2-vm06 ~]# vim /etc/profile
  export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL  --此行之后加入以
  下语句

  export JAVA_HOME=/usr/java/jdk1.8.0_202-amd64/jre
  export PATH=$PATH:$JAVA_HOME/bin
  export CLASSPATH=.:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar

  [root@node2-vm06 tmp]# source /etc/profile
  ```

设置hostname：

```
[root@node2-vm06 broker]# vim /etc/hosts

127.0.0.1       node2-vm06.yjy
```

- Git
- Maven
- Idea
- nodejs

名称	端口	说明
Mysql	3306	版本: 5.7, 系统配置、指标数据库
Redis	6379	docker容器中, 版本: 3.2
zookeeper	2181	docker容器中
kafka	9092	docker容器中

3.1.2 docker

Docker 要求 CentOS 系统的内核版本高于 3.10 ，通过 uname -r 命令查看你当前的内核版本：

```
uname -r
3.10.0-1062.1.2.el7.x86_64
```

安装 Docker：

```
yum -y install docker
```

3.1.3 kafka和zookeeper

```
docker pull wurstmeister/zookeeper
docker pull wurstmeister/kafka

#启动zookeeper容器
docker run -d --name zookeeper -p 2181:2181 -t wurstmeister/zookeeper

#启动kafka容器
docker run -d --name kafka --publish 9092:9092 --link zookeeper --env
KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 --env
KAFKA_ADVERTISED_HOST_NAME=172.16.0.117 --env KAFKA_ADVERTISED_PORT=9092 --
volume /etc/localtime:/etc/localtime wurstmeister/kafka:latest
```

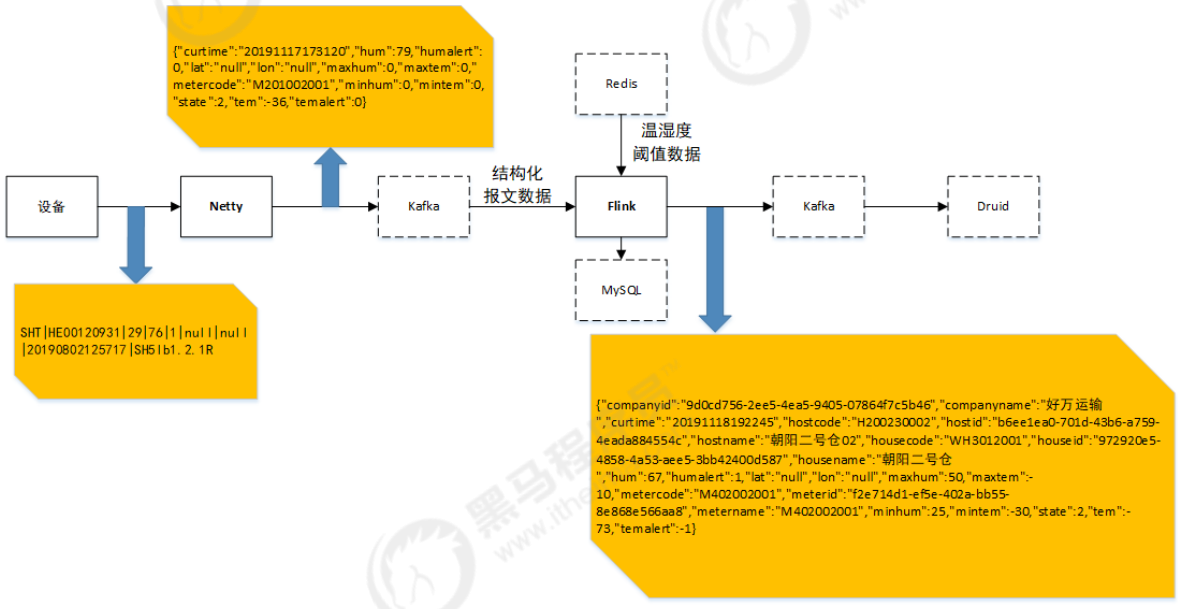
3.1.4 redis

```
docker search redis

docker pull redis

docker run -p 6379:6379 -d redis:latest redis-server
```

3.1.5 mysql

```
1、安装数据库：yum -y install mysql-server
2、启动数据库：service mysqld start
3、登录数据库：mysql -u root -p回车后输入密码（默认用户名是root，密码为空）
4、输入show databases；查看数据库
5、使用数据库：use mysql；
6、连接数据库：使用Navicat for MySQL链接数据库
（1）从user表中查询所有可以登录的用户以及支持连接的主机：select user，host，password from user；
（2）User表host字段的值为%或者localhost时的区别
（3）新建root用户，并给此用户赋予所有数据库和及其所有对象的操作权限，这个赋权语句里的%代表支持任意主机链接到mysql服务器，赋权语句：
1、 grant all privileges on . to 'root'@'%' identified by '123456' with grant option;
2、 flush privileges;(刷新权限）
注：
如果连接失败，请查看服务器的端口是否开启
防火墙是否关闭
连接信息是否填写正确
```

3.2. 大数据组件



3.2.1 Apache Flink

Apache Flink 是一个开源的分布式流式处理框架，是新的流数据计算引擎，用java实现，它能够基于同一个Flink运行时（Flink Runtime），提供支持流处理和批处理两种类型应用的功能。

现有的开源计算方案，会把流处理和批处理作为两种不同的应用类型：

- 流处理一般需要支持低延迟、Exactly-once保证
- 批处理需要支持高吞吐、高效处理

所以在实现的时候通常是分别给出两套实现方法，或者通过一个独立的开源框架来实现其中每一种处理方案。例如，实现批处理的开源方案有MapReduce、Tez、Spark，实现流处理的开源方案有Storm等。

Flink在实现流处理和批处理时，与传统的一些方案完全不同，它从另一个视角看待流处理和批处理，将二者统一起来：Flink是完全支持流处理，也就是说作为流处理看待时输入数据流是无界的；批处理被作为一种特殊的流处理，只是它的输入数据流被定义为有界的。基于同一个Flink运行时（Flink Runtime），分别提供了流处理和批处理API，而这两种API也是实现上层面向流处理、批处理类型应用框架的基础。

流处理特性

- 支持高吞吐、低延迟、高性能的流处理
- 支持带有事件时间的窗口（Window）操作
- 支持有状态计算的Exactly-once语义
- 支持高度灵活的窗口（Window）操作，支持基于time、count、session，以及data-driven的窗口操作
- 支持具有“背压”功能的持续流模型
- 支持基于轻量级分布式快照（Snapshot）实现的容错
- 一个运行时同时支持Batch on Streaming处理和Streaming处理
- Flink在JVM内部实现了自己的内存管理
- 支持迭代计算
- 支持程序自动优化：避免特定情况下Shuffle、排序等昂贵操作，中间结果有必要进行缓存

API支持

- 对Streaming数据类应用，提供DataStream API
- 对批处理类应用，提供DataSet API（支持Java/Scala）

Libraries支持

- 支持机器学习（FlinkML）
- 支持图分析（Gelly）
- 支持关系数据处理（Table）
- 支持复杂事件处理（CEP）

整合支持

- 支持Flink on YARN
- 支持HDFS
- 支持来自Kafka的输入数据
- 支持Apache HBase
- 支持Hadoop程序
- 支持Tachyon
- 支持ElasticSearch
- 支持RabbitMQ
- 支持Apache Storm
- 支持S3

3.2.2 Apache Druid

Apache Druid是一个拥有大数据实时查询和分析(OLAP)的高容错、高性能开源分布式数据存储系统，旨在快速处理大规模的数据，并能够实现快速聚合查询和分析的高并发系统。Druid的常用应用领域包括：

- 点击流分析(web和移动分析)
- 网络遥测分析(网络性能监控)

- 服务器指标存储
- 供应链分析（制造指标）
- 应用程序性能度量
- 数字营销/广告分析
- 商业智能/联机分析处理

Druid的核心架构结合了**数据仓库**，**时间序列数据库**和**日志搜索系统**的创意。Druid的一些主要特点是：

1. **列式存储格式：**Druid使用面向列的存储，这意味着它只需要加载特定的查询所需的精确列。这为仅查看几列的查询提供了巨大的速度提升。此外，每列都针对其特定数据类型进行了优化，支持快速扫描和聚合。
2. **可扩展的分布式系统：** Druid通常部署在数十到数百台服务器的集群中，可以提供数百万条记录/秒的摄取率，保留数万亿条记录，以及亚秒级到几秒钟的查询延迟。
3. **大规模并行处理：**Druid可以在整个集群中并行处理查询。
4. **实时或批量采集：** Druid可以实时流式采集数据（采集的数据可立即用于查询）或批量采集。
5. **自愈，自平衡，易于操作：**作为运营商，要将群集扩展或缩小，只需添加或删除服务器，群集将在后台自动重新平衡，无需任何停机时间。如果任何Druid服务器发生故障，系统将自动绕过损坏路由，直到可以更换这些服务器。Druid旨在全天候运行，无需任何原因计划停机，包括配置更改和软件更新。
6. **云本机，容错架构，不会丢失数据：**一旦Druid采集了数据，副本就会安全地存储在深层存储（通常是云存储，HDFS或共享文件系统）中。即使每个Druid服务器都出现故障，数据也可以从深层存储中恢复。对于仅影响少数Druid服务器的更有限的故障，复制可确保在系统恢复时仍可进行查询。
7. **用于快速过滤的索引：**Druid使用CONCISE或 Roaring压缩bitmap索引来创建索引，这些索引可以跨多个列进行快速过滤和搜索。
8. **基于时间的分区：**Druid首先按时间划分数据，并且可以基于其他字段进行额外划分。这意味着基于时间的查询将仅访问与查询的时间范围匹配的分区。这导致基于时间的数据的显着性能改进。
9. **近似算法：**Druid包括用于近似count-distinct的算法，近似排序以及近似直方图和分位数的计算的算法。这些算法提供有限的内存使用，并且通常比精确计算快得多。对于精度比速度更重要的情况，Druid还提供精确的count-distinct以及精确的排序。
10. **在采集时自动汇总：** Druid可选择在采集时支持数据汇总。提前预聚合数据，可以节省大量存储成本并提高性能。

3.2.3 Apache Superset

Apache Superset 是由 Airbnb 开源的数据分析与可视化平台，是由Python 语言构建的轻量级 BI 系统。

- 提供一个直观的数据挖掘和数据可视化的交互界面，并且可以建立一套可交互的 Dashboards（看板）。 通过Superset可以与各种主流的数据库，如sqlserver、mysql等进行实时交互，并且可以对数据表进行即时的可视化，支持建立Dashboards功能。
- 提供一套美观的数据可视化图表，superset提供了一套基于d3.js的数据可视化解决方案，也可以通过二次开发加入一些诸如echarts,antv等数据化图表。
- 简单，即使无代码基础的用户也可以通过对数据的拖曳完成图表和视图面板的制作，为进一步的数据挖掘提供可视化分析的基础
- 可以作为一个优雅的SQL IDE，运行sql指令从数据库提数，并且可以对查询结果进行实时的可视化
- 高粒度权限控制体系，通过基于flask-appbuilder的权限设计，可以对各个数据集和dashbord的访问设计权限
- 轻量级语义层设计，允许通过定义维度和指标来控制数据源向用户公开的方式
- 开箱即用，支持大部分数据库语言
- 与Druid深度集成，可快速处理海量数据
- 缓存配置设计，可快速加载Dashboard

Superset解决了什么？

- 动态Dashboards，可对数据库中的数据进行动态可视化，以满足监控或者研究的需要
- 支持目前主流的大多数sql语言，可以对各种数据库进行动态查询、管理与可视化
- 开箱即用的数据可视化解决方案

3.3. 应用服务

3.3.1 后台服务

应用	端口	说明	启动命令
cold-eureka	8001	服务注册中心	java -jar cold-eureka-0.0.1-SNAPSHOT.jar &
cold-gateway	8080	API网关，前端统一入口	java -jar cold-gateway-0.0.1-SNAPSHOT.jar &
cold-user	8185	用户服务	java -jar cold-user-1.0.0 jar &
cold-admin	8181	管理服务	java -jar cold-admin-0.0.1-SNAPSHOT.jar &
cold-druid	8182	Druid查询服务	java -jar cold-druid-1.0.0.jar &
cold-monitor	8183	实时查询服务	java -jar cold-monitor-0.0.1-SNAPSHOT.jar &
cold-jobs	8184	定时任务服务	java -jar cold-jobs-0.0.1-SNAPSHOT.jar &
cold-netty-server	10010	netty服务器	java -jar cold-netty-server-1.0-SNAPSHOT.jar
cold-ui	8000	web服务，Tomcat	service tomcat start | stop | restart

3.3.2 前台应用

前端tomcat：

- 配置文件： /etc/tomcat/
- 应用目录： /var/lib/tomcat/webapps

编译web应用

```
# 编译vue
npm install

# 本地运行web应用
npm run serve
```

部署web应用：

```
# 生成dist目录
npm run build

# 将dist目录上传至web服务器
# 更改文件夹名称为cold
[root@node2-vm06 ~]# cd /var/lib/tomcat/webapps/
[root@node2-vm06 webapps]# unzip dist.zip
[root@node2-vm06 webapps]# mv dist cold
```

4. 数据结构

4.1 用户库（cold-user）

4.1.1 用户表

表名： sys_user

名称	类型	长度	小数点	是否为NULL	主键	注释
user_id	bigint	20	0	1	1	主键Id
username	varchar	50	0	1	0	用户名
password	varchar	100	0	0	0	密码
salt	varchar	20	0	0	0	盐
email	varchar	100	0	0	0	邮箱
mobile	varchar	100	0	0	0	手机号
status	tinyint	4	0	0	0	状态 0：禁用 1：正常
create_user_id	bigint	20	0	0	0	创建者ID
create_time	datetime	0	0	0	0	创建时间

4.1.2 token表

表名： sys_user_token

名称	类型	长度	小数点	是否为NULL	主键	注释
user_id	bigint	20	0	1	1	主键Id
token	varchar	100	0	1	0	token
expire_time	datetime	0	0	0	0	过期时间
update_time	datetime	0	0	0	0	更新时间

4.2 业务库（cold）

4.2.1 公司表

表名：cz_company

名称	类型	长度	小数点	是否为NULL	主键	注释
id	varchar	50	0	1	1	主键
company	varchar	100	0	0	0	公司名称
abbreviation	varchar	50	0	0	0	公司简称
companyNumber	varchar	50	0	0	0	企业编号
companyAddress	varchar	100	0	0	0	企业地址
companyPhone	varchar	20	0	0	0	公司电话
managementName	varchar	50	0	0	0	质量管理员
managementPhone	varchar	20	0	0	0	联系电话
leader	varchar	50	0	0	0	负责人姓名
leaderPhone	varchar	20	0	0	0	负责人电话
message	varchar	255	0	0	0	备注
webAddress	varchar	100	0	0	0	网站
companyType	varchar	20	0	0	0	企业类型
createdTime	timestamp	0	0	1	0	创建时间，CURRENT_TIMESTAMP
updateTime	timestamp	0	0	1	0	修改时间，CURRENT_TIMESTAMP，根据当前时间戳更新

4.2.2 仓库表

表名： cz_warehouse

名称	类型	长度	小数点	是否为NULL	主键	注释
id	varchar	50	0	1	1	主键
houseCode	varchar	50	0	1	0	仓库编码
houseName	varchar	100	0	0	0	仓库名称
houseAddress	varchar	255	0	0	0	仓库地址
houseType	int	2	0	0	0	库房类型：1-冷库， 2-恒温库
companyId	varchar	50	0	0	0	所属公司Id
companyName	varchar	100	0	0	0	公司名称
principalName	varchar	100	0	0	0	负责人
principalTel	varchar	20	0	0	0	负责人电话
longitude	decimal	10	6	0	0	经度
latitude	decimal	10	6	0	0	维度
areaSize	double	6	0	0	0	库房面积
houseStatus	int	2	0	0	0	状态：1-正常,0-空库
createdTime	timestamp	0	0	1	0	创建时间，CURRENT_TIMESTAMP
updateTime	timestamp	0	0	1	0	修改时间，CURRENT_TIMESTAMP， 根据当前时间戳更新

4.2.3 主机表

表名： cz_host

名称	类型	长度	小数点	是否为NULL	主键	注释
id	varchar	50	0	1	1	主键
hostCode	varchar	50	0	1	0	主机编码
hostName	varchar	50	0	0	0	主机名称
houseId	varchar	50	0	0	0	仓库Id
houseCode	varchar	50	0	0	0	仓库编码
houseName	varchar	50	0	0	0	仓库名称
hostStatus	int	2	0	0	0	主机状态：1-正常，0-停用
hostModel	varchar	50	0	0	0	设备型号
simCode	varchar	50	0	0	0	sim卡号

名称	类型	长度	小数点	是否为NULL	主键	注释
createdTime	timestamp	0	0	1	0	创建时间，CURRENT_TIMESTAMP
updateTime	timestamp	0	0	1	0	修改时间，CURRENT_TIMESTAMP，根据当前时间戳更新

4.2.4 设备表

表名： cz_monitor

名称	类型	长度	小数点	是否为NULL	主键	注释
id	varchar	50	0	1	1	主键
userName	varchar	50	0	0	0	人员姓名
userPhone	varchar	20	0	0	0	人员电话
monitorState	varchar	20	0	0	0	预警类型，0：温湿度，1：温度，2：湿度
hostId	varchar	50	0	0	0	预警的主机Id
hostCode	varchar	50	0	0	0	预警的主机编码
hostName	varchar	50	0	0	0	预警的主机名称
createdTime	timestamp	0	0	1	0	创建时间,CURRENT_TIMESTAMP
updateTime	timestamp	0	0	1	0	修改时间，CURRENT_TIMESTAMP，根据当前时间戳更新

4.2.5 实时指标表

表名: cz_message_realtime

名称	类型	长度	小数点	是否为NULL	主键	注释
meterCode	varchar	50	0	1	1	设备编码
meterId	varchar	50	0	0	0	设备ID
meterName	varchar	100	0	0	0	设备名称
hostId	varchar	50	0	0	0	主机Id
hostCode	varchar	50	0	0	0	主机编码
hostName	varchar	100	0	0	0	主机名称
houseId	varchar	50	0	0	0	仓库Id
houseCode	varchar	50	0	0	0	仓库编码
houseName	varchar	100	0	0	0	仓库名称
companyId	varchar	50	0	0	0	公司Id
companyName	varchar	100	0	0	0	公司名称
tem	int	5	0	0	0	温度
maxTem	int	5	0	0	0	温度上限
minTem	int	5	0	0	0	温度下限
hum	int	5	0	0	0	湿度
maxHum	int	5	0	0	0	湿度上限
minHum	int	5	0	0	0	湿度下限
temAlert	int	5	0	0	0	温度状况： 1：高温， 0：正常， -1： 低温
humAlert	int	5	0	0	0	湿度状况： 1：高湿， 0：正常， -1： 低湿
state	int	5	0	0	0	设备状态: 1-在用，0-停用，2-异常
lon	varchar	30	0	0	0	经度
lat	varchar	30	0	0	0	维度
curtime	varchar	50	0	0	0	提交时间
createTime	timestamp	0	0	0	0	创建时间,CURRENT_TIMESTAMP

4.3 定时任务库（cold-job）

4.3.1 CRON触发器

表名：QRTZ_CRON_TRIGGERS

作用：存储触发器的cron表达式表

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	1	1	调度名称
TRIGGER_NAME	varchar	200	0	1	1	触发器名称
TRIGGER_GROUP	varchar	200	0	1	1	触发器分组名称
CRON_EXPRESSION	varchar	120	0	1	0	CRON表达式
TIME_ZONE_ID	varchar	80	0	0	0	时区

4.3.2 启动的触发器

表名：QRTZ_FIRED_TRIGGERS

作用： 存储已触发的 Trigger 相关的状态信息，以及相关job的执行信息

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	-1	调度名称
ENTRY_ID	varchar	95	0	0	-1	实例名称
TRIGGER_NAME	varchar	200	0	0	0	触发器名称
TRIGGER_GROUP	varchar	200	0	0	0	触发器分组名称
INSTANCE_NAME	varchar	200	0	0	0	实例名称
FIRED_TIME	bigint	13	0	0	0	启动时间
SCHED_TIME	bigint	13	0	0	0	执行时间
PRIORITY	int	11	0	0	0	优先级
STATE	varchar	16	0	0	0	状态

名称	类型	长度	小数点	是否为NULL	主键	注释
JOB_NAME	varchar	200	0	0	-1	任务名称
JOB_GROUP	varchar	200	0	-1	0	任务分组名称
IS_NONCONCURRENT	varchar	1	0	-1	0	是否立即执行
REQUESTS_RECOVERY	varchar	1	0	-1	0	恢复标识

4.3.3 任务详情

表名： QRTZ_JOB_DETAILS

作用： 存储每一个已配置的 jobDetail 的详细信息

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	1	1	调度名称
JOB_NAME	varchar	200	0	1	1	任务名称
JOB_GROUP	varchar	200	0	1	1	任务分组名称
DESCRIPTION	varchar	250	0	0	0	描述信息
JOB_CLASS_NAME	varchar	250	0	1	0	任务的类名称
IS_DURABLE	varchar	1	0	1	0	是否持久化
IS_NONCONCURRENT	varchar	1	0	1	0	是否并发
IS_UPDATE_DATA	varchar	1	0	1	0	是否更新数据
REQUESTS_RECOVERY	varchar	1	0	1	0	是否接受恢复执行，默认为false， 设置了RequestsRecovery为true，则该job会被重新执行
JOB_DATA	blob	0	0	0	0	存放持久化job对象

4.3.4 锁表

表名： QRTZ_LOCKS

作用：存储程序的悲观锁的信息(假如使用了悲观锁)

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	0	调度名称
LOCK_NAME	varchar	40	0	0	0	悲观锁名称

4.3.5 调度状态表

表名： QRTZ_SCHEDULER_STATE

作用： 存储集群中note实例信息，quartz会定时读取该表的信息判断集群中每个实例的当前状态。

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	1	调度名称
INSTANCE_NAME	varchar	200	0	0	1	实例名称， 配置文件中org.quartz.scheduler.instanceId
LAST_CHECKIN_TIME	bigint	13	0	0	0	上次检查时间
CHECKIN_INTERVAL	bigint	13	0	0	0	检查间隔时间

3.3.6 简单触发器表

表名： QRTZ_SIMPLE_TRIGGERS

作用：存储简单的 Trigger，包括重复次数，间隔，以及已触发的次数。

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	1	调度名称
TRIGGER_NAME	varchar	200	0	0	1	qrtz_triggers表trigger_name的外键
TRIGGER_GROUP	varchar	200	0	0	1	qrtz_triggers表trigger_group的外键
REPEAT_COUNT	bigint	7	0	0	0	重复的次数统计
REPEAT_INTERVAL	bigint	12	0	0	0	重复的间隔时间
TIMES_TRIGGERED	bigint	10	0	0	0	已经触发的次数

4.3.7 触发器表

表名：QRTZ_TRIGGERS

作用： 保存触发器的基本信息

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	-1	调度名称
TRIGGER_NAME	varchar	200	0	0	-1	触发器的名字
TRIGGER_GROUP	varchar	200	0	0	-1	触发器所属组的名字
JOB_NAME	varchar	200	0	0	0	qrtz_job_details表job_name的外键
JOB_GROUP	varchar	200	0	0	0	qrtz_job_details表job_group的外键
DESCRIPTION	varchar	250	0	-1	0	描述信息
NEXT_FIRE_TIME	bigint	13	0	-1	0	下一次触发时间，默认为-1，意味不会自动触发
PREV_FIRE_TIME	bigint	13	0	-1	0	上一次触发时间（毫秒）
PRIORITY	int	11	0	-1	0	优先级
TRIGGER_STATE	varchar	16	0	0	0	当前触发器状态，设置为ACQUIRED,如果设置为WAITING,则job不会触发（ WAITING:等待 PAUSED:暂停ACQUIRED:正常执行 BLOCKED：阻塞 ERROR：错误）
TRIGGER_TYPE	varchar	8	0	0	0	触发器的类型，使用cron表达式
START_TIME	bigint	13	0	0	0	开始时间
END_TIME	bigint	13	0	-1	0	结束时间
CALENDAR_NAME	varchar	200	0	-1	0	日程表名称，表qrtz_calendars的calendar_name字段外键
MISFIRE_INSTR	smallint	2	0	-1	0	措施或者是补偿执行的策略
JOB_DATA	blob	0	0	-1	0	blob字段，存放持久化job对象

4.3.8 暂停的触发器组

表名： QRTZ_PAUSED_TRIGGER_GRPS

作用： 存储已暂停的 Trigger 组的信息

名称	类型	长度	小数点	是否为NULL	主键	注释
SCHED_NAME	varchar	120	0	0	1	调度名称
TRIGGER_GROUP	varchar	200	0	0	0	qrtz_triggers表trigger_group的外键

4.3.9 调度表

表名： schedule_job

作用： 调度任务实例

名称	类型	长度	小数点	非空	主键	注释
job_id	bigint	20	0	1	1	任务id
bean_name	varchar	200	0	0	0	spring bean名称
params	varchar	2000	0	0	0	参数
cron_expression	varchar	100	0	0	0	cron表达式
status	tinyint	4	0	0	0	任务状态 0：正常 1：暂停
remark	varchar	255	0	0	0	备注
create_time	datetime	0	0	0	0	创建时间

4.3.10 调度日志表

表名：schedule_job_log

作用：调度日志

名称	类型	长度	小数点	非空	主键	注释
log_id	bigint	20	0	0	-1	任务日志id
job_id	bigint	20	0	0	0	任务id
bean_name	varchar	200	0	-1	0	spring bean名称
params	varchar	2000	0	-1	0	参数
status	tinyint	4	0	0	0	任务状态 0：成功 1：失败
error	varchar	2000	0	-1	0	失败信息
times	int	11	0	0	0	耗时(单位：毫秒)
create_time	datetime	0	0	-1	0	创建时间

5. 公共组件工程（cold-common）

cold-common工程的内容包括一些被其他系统引用的类：

- DAO
- Entity
- 工具类

5.1 统一返回

```
public class R extends HashMap<String, Object> {
    private static final long serialVersionUID = 1L;

    public R() {
        put("code", 0);
        put("msg", "success");
    }

    public static R error() {
        return error(HttpStatus.SC_INTERNAL_SERVER_ERROR, "未知异常，请联系管理员");
    }

    public static R error(String msg) {
        return error(HttpStatus.SC_INTERNAL_SERVER_ERROR, msg);
    }

    public static R error(int code, String msg) {
        R r = new R();
        r.put("code", code);
        r.put("msg", msg);
        return r;
    }

    public static R ok(String msg) {
        R r = new R();
        r.put("msg", msg);
        return r;
    }

    public static R ok(Map<String, Object> map) {
        R r = new R();
        r.putAll(map);
        return r;
    }

    public static R ok() {
        return new R();
    }

    public R put(String key, Object value) {
        super.put(key, value);
        return this;
    }
}
```

5.2 分页工具

```
@Data
public class PageUtils implements Serializable {
    private static final long serialVersionUID = 1L;
    /**
     * 总记录数
     */
    private int totalCount;
    /**
     * 每页记录数
     */
    private int pageSize;
    /**
     * 总页数
     */
    private int totalPage;
    /**
     * 当前页数
     */
    private int currPage;
    /**
     * 列表数据
     */
    private List<?> list;

    /**
     * 分页
     * @param list        列表数据
     * @param totalCount  总记录数
     * @param pageSize    每页记录数
     * @param currPage    当前页数
     */
    public PageUtils(List<?> list, int totalCount, int pageSize, int currPage) {
        this.list = list;
        this.totalCount = totalCount;
        this.pageSize = pageSize;
        this.currPage = currPage;
        this.totalPage = (int)Math.ceil((double)totalCount/pageSize);
    }

    /**
     * 分页
     */
    public PageUtils(IPage<?> page) {
        this.list = page.getRecords();
        this.totalCount = (int)page.getTotal();
        this.pageSize = (int)page.getSize();
        this.currPage = (int)page.getCurrent();
        this.totalPage = (int)page.getPages();
    }

    public Map getPageMap(){
        Map<String, Object> map = new HashMap<String, Object>();
        map.put("page", getTotalPage());
        map.put("total", this.getTotalCount());
        map.put("items", this.getList());
        return map;
    }
```

```
}
```

5.3 自定义异常

```
@Data
public class MyException extends RuntimeException {
    private static final long serialVersionUID = 1L;

    private String msg;
    private int code = 500;

    public MyException(String msg) {
        super(msg);
        this.msg = msg;
    }

    public MyException(String msg, Throwable e) {
        super(msg, e);
        this.msg = msg;
    }

    public MyException(String msg, int code) {
        super(msg);
        this.msg = msg;
        this.code = code;
    }

    public MyException(String msg, int code, Throwable e) {
        super(msg, e);
        this.msg = msg;
        this.code = code;
    }

    public static void main(String[] args) throws MyException{
        try{
            int i = 1/0;
        }catch (Exception e){
            throw new MyException("除法中分母不能为空");
        }

    }
}
```

6. 总结

通过本章的学习，大家需要了解冷链设备监控的业务场景和项目目标，同时要掌握此类项目的技术框架、数据流程等。