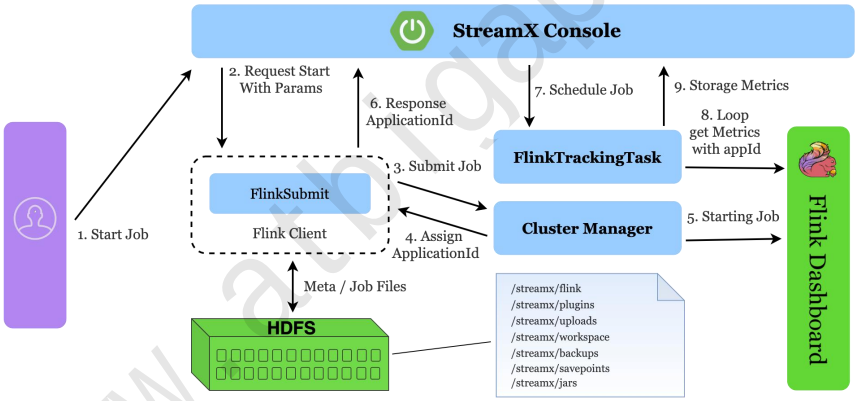
Streampark 使用体验与建议

项目的初衷是 —— 让流处理更简单，使用 StreamPark 开发流处理作业，可以极大降低学习成本和开发门槛，让开发者只用关心最核心的业务。

1. 入门介绍

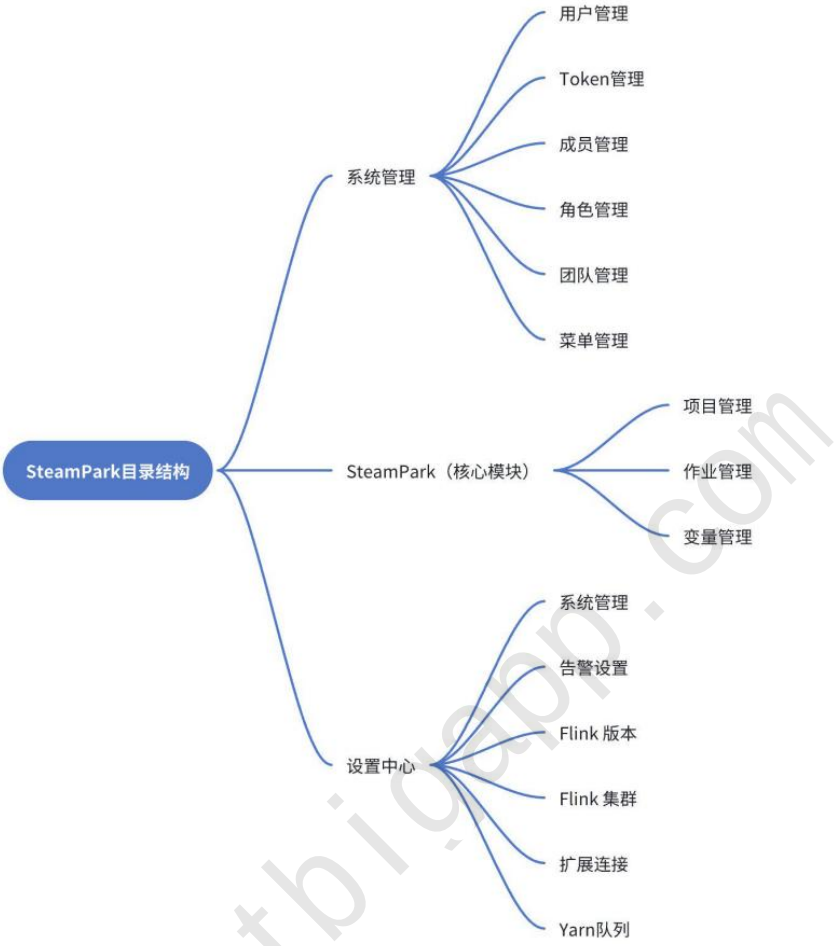
实时即未来，在实时处理流域 Apache Spark 和 Apache Flink 是一个伟大的进步,尤其是 Apache Flink 被普遍认为是下一代大数据流计算引擎，我们在使用 Flink & Spark 时发现从编程模型，启动配置到运维管理都有很多可以抽象共用的地方，我们将一些好的经验固化下来并结合业内的最佳实践，通过不断努力诞生了今天的框架 —— StreamPark。



任务启动流程图

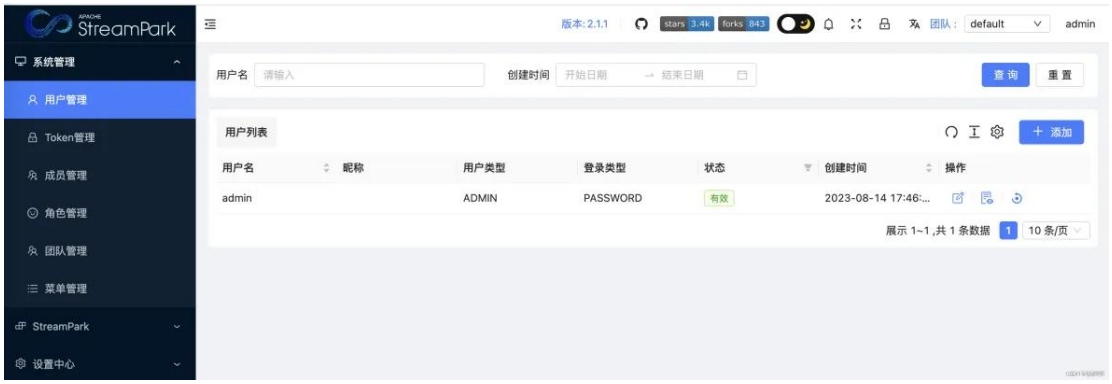
随着大数据时代的发展，流数据处理成为越来越多企业的需求。面对繁重的数据任务，如何有效地管理和处理流数据呢？今天我们将为您介绍一款名为 Streampark 的流处理极速开发框架，它能为您提供一站式的流处理解决方案，满足您的各种需求。

2. 功能界面



2.1. 系统管理页面

系统管理页面似乎和我们普通的管理系统没什么区别，主要就是做隔离和权限控制。这里不再细讲，直接贴图：



2.2.设置中心

2.2.1.系统设置

maven 配置：

系统内置了 maven，通过上传 settings.xml 到指定目录，便可以读取该文件来下载项目的依赖。

Docker 环境配置：

flink on k8s 时使用，需要配置目标 k8s 集群所使用的 Docker 容器服务的连接信息，主要用于做镜像的 pull 和 push。

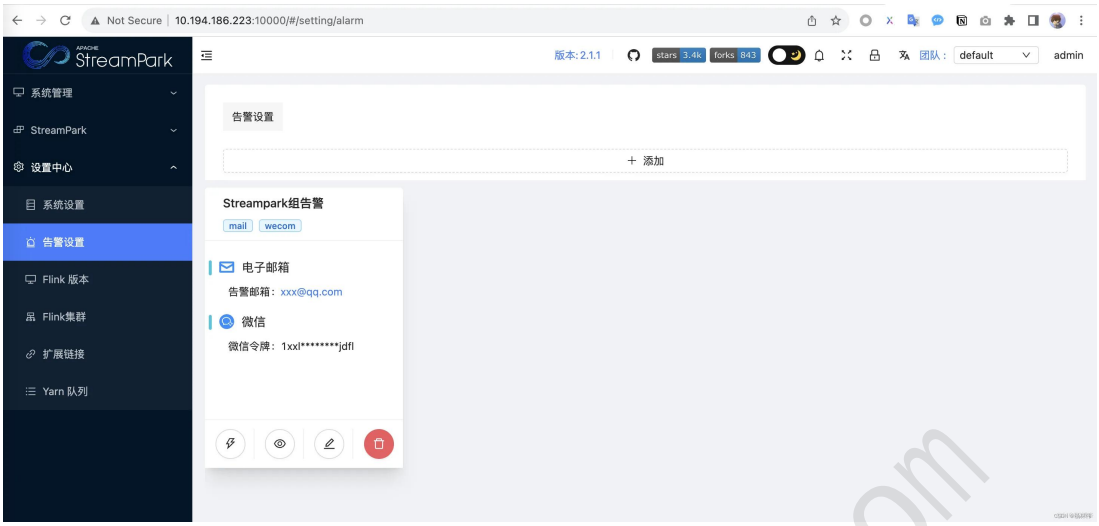
邮箱配置：

主要配置发送人的邮箱账号密码、告警邮箱的 host/port 等，因为推送告警消息需要指定具体的发送人和使用的邮件服务器。

k8s Ingress 配置：

配置 ingress 域名地址。

2.2.2. 告警设置

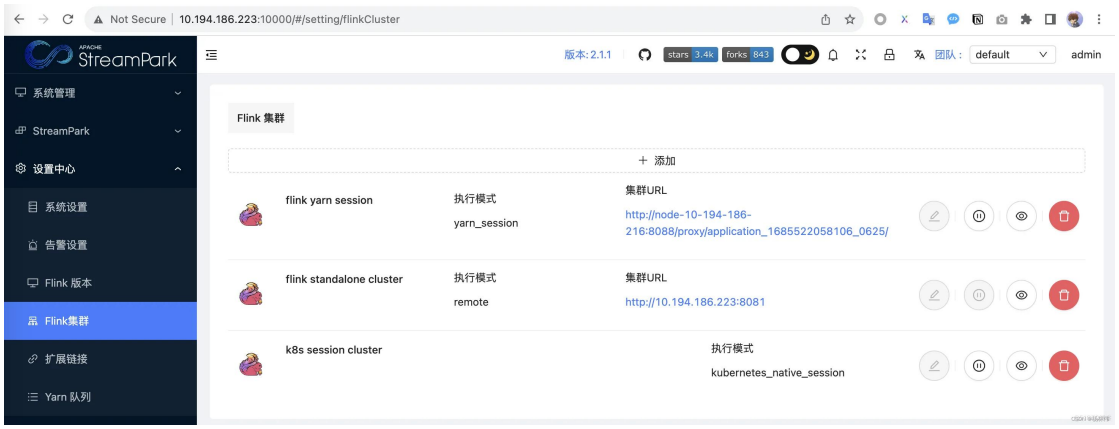


配置告警组，以及接受告警的对象，支持电子邮箱、钉钉、微信、飞书

2.2.3. flink 版本

指的就是 flink sdk 解压后的路径，需要配置其路径

2.2.4. flink 集群



最终都是部署 flink 集群到某个机器，yarn 或 k8s，都会有一个 JobManager 的 url

独立集群：

配置 flink 版本、JobManager URL 即可，容器安装时已经自动也安装好了

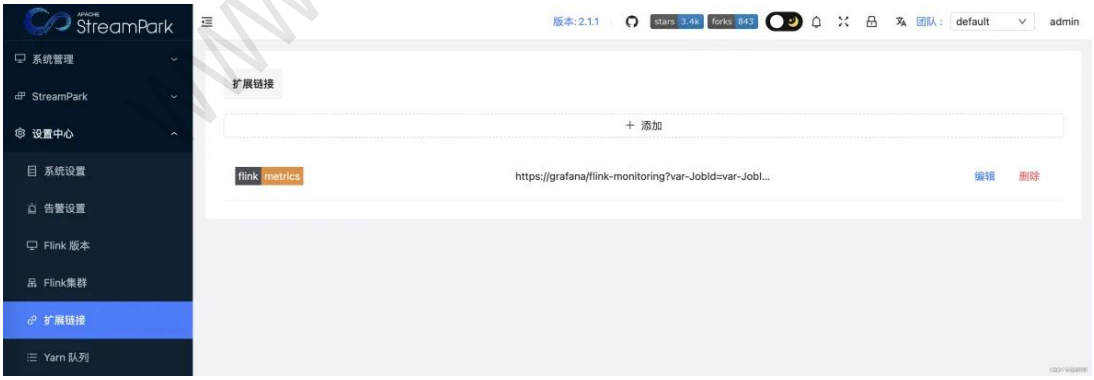
yarn session 集群：

这里应该只能固定一个，写死的 yarn 集群。需要配置 flink 版本、yarn 队列、还有其它的 flink 配置、需要 export hadoop_home

k8s session 集群：

需要指定 kube 文件，使程序有权限部署指定镜像的 flink session 至 k8s。需要配置 flink 版本、flink 基础镜像标签、flink 配置等

2.2.5. 扩展链接



主要就是配置外链地址，点击作业详情，可以通过传入 jobid 来注入进预置的地址，然后就能跳转到指定的外链，如跳转到 grafana 页面查看 flink 的指标详情等

2.2.6. yarn 队列

主要就是配置当前配置 hadoop 集群，能使用的队列名

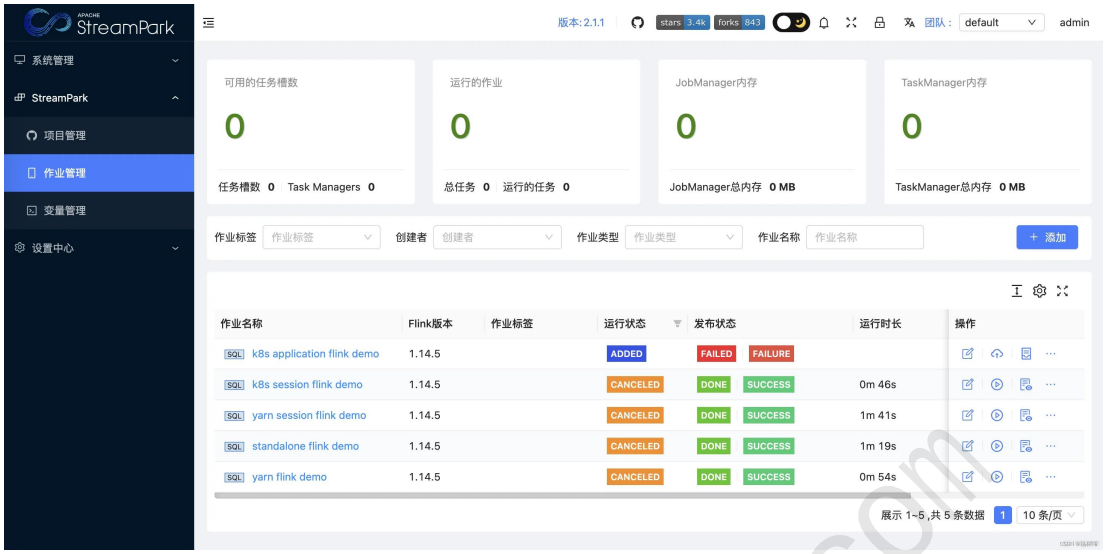
2.3. Streampark（核心）

Streampark 模块是最核心的部分，前面的配置都为它做基础，我们来看看其界面。

2.3.1. 项目管理

可以理解为一个小 jenkins 吧，可以通过指定仓库地址、账号密码以及指定分支来构建相关的项目，打包出来的 jar 直接就可以在作业管理使用。

2.3.2.作业管理



Custom Code：

就是从项目管理编译打包后的文件夹里面获取打包后的 jar，然后通过指定程序的参数、还有配置相关的 flink 参数，提交到前面创建的 flink 集群（或 Application 模式），可以理解为 jar 模式

FlinkSQL：

flink sql 就很容易理解了，就是写 flink sql 来提交作业，应该会有一个特定的主程序去执行的（备注：还没看代码），它还有一个特点，就可以通过写 maven 的依赖或上传 jar 包来对该作业添加依赖，只是不明白为什么 Custom Code 不给它添加依赖的 jar。

2.3.3.变量管理

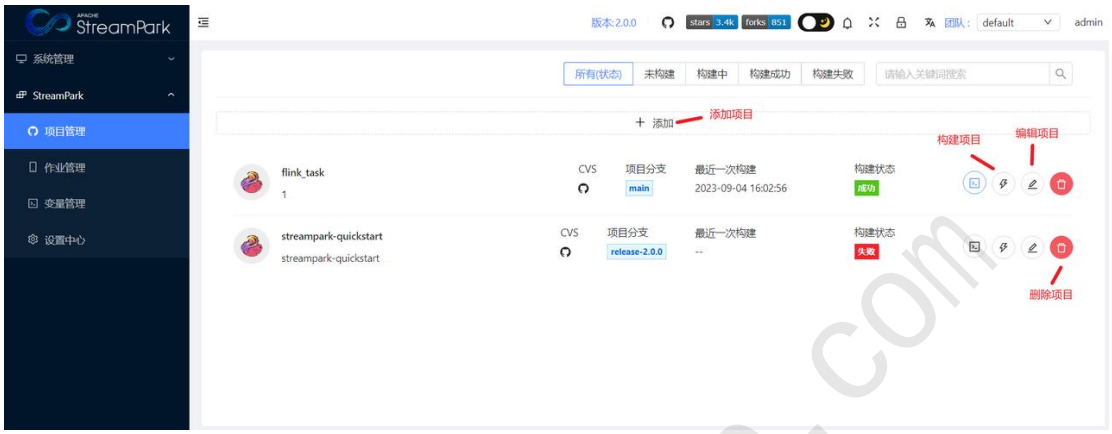
提供一个统一管理程序参数的页面，可以将动态参数包装成变量供作业引用。

3.使用方式

3.1. 项目管理

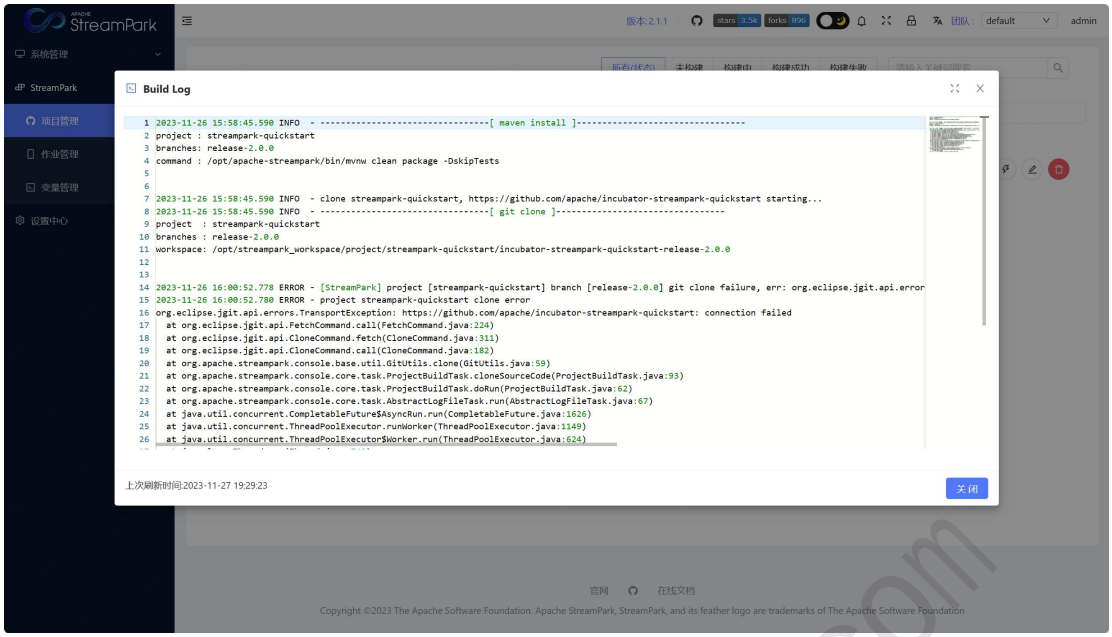
项目管理是 Streampark 管理 Flink 项目的地方，可以添加、查看、编辑、构建、删除等操作。

项目管理中可查看各个状态的项目。



添加项目时，输入项目名称、代码仓库地址、用户名、密码，选择项目分支。

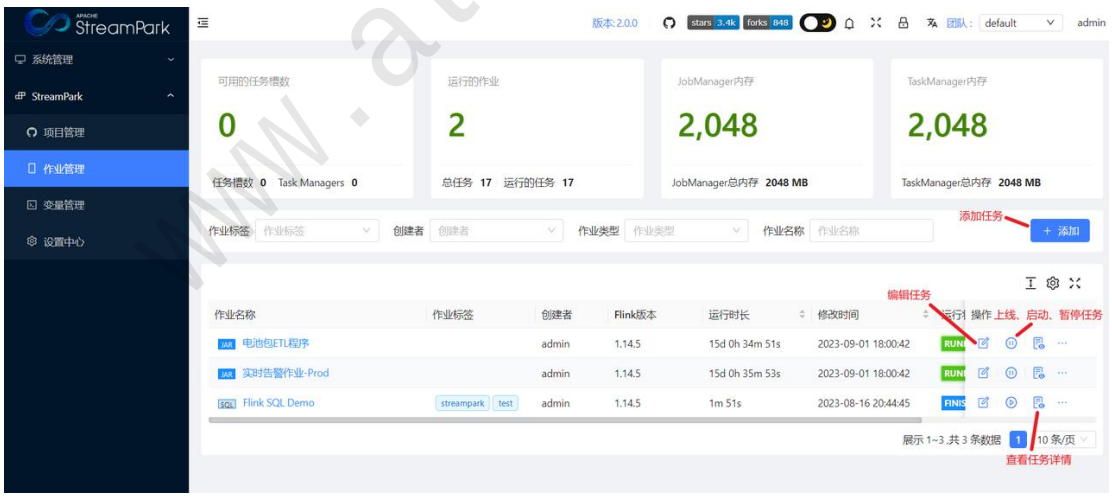
点击项目构建 SteamPark 会从指定的项目仓库中拉取 Flink 作业代码进行打包构建。



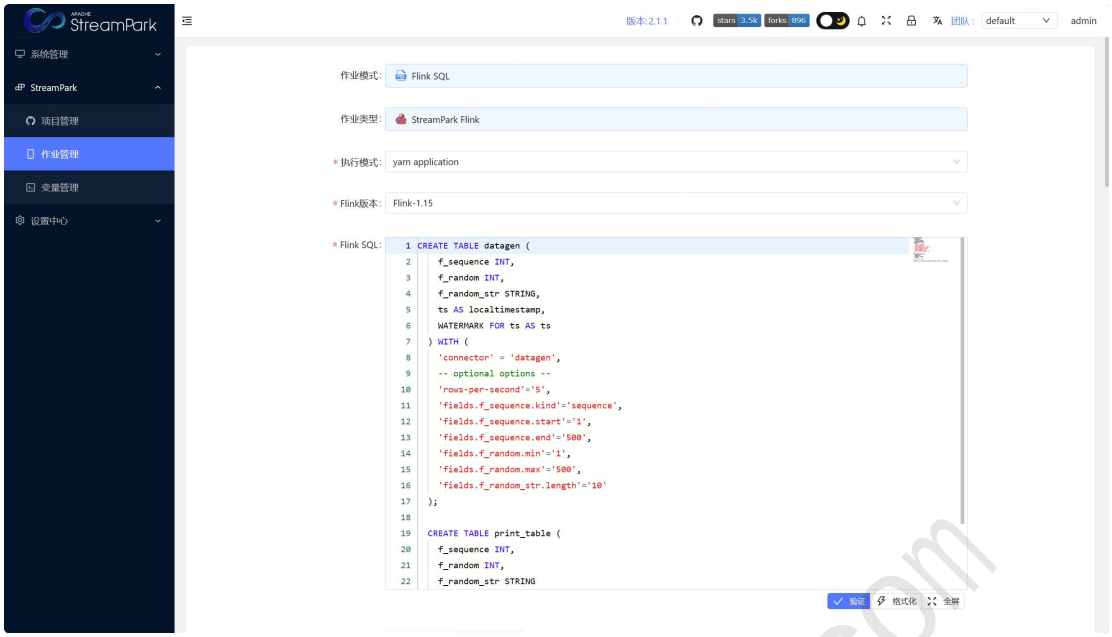
构建好的项目可在作业管理中添加使用，参考作业管理 - 添加作业，选择 Datastream 作业。

3.2. 作业管理

作业管理是 Streampark 添加任务、上线任务、启动任务、暂停任务、查看任务详情的地方。



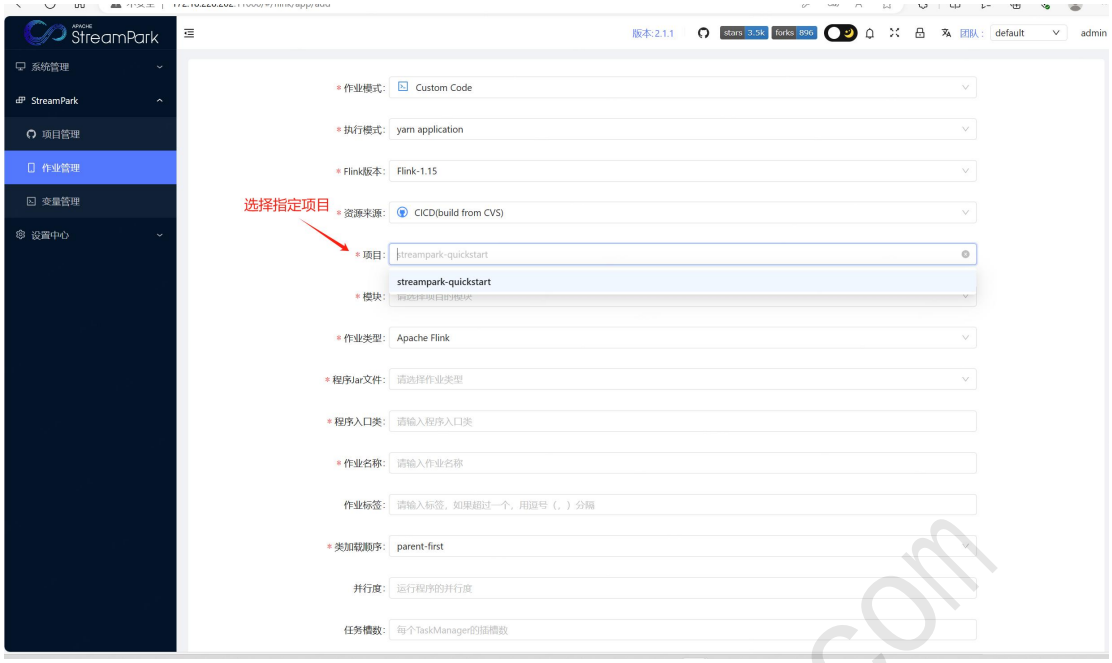
StreamPark 提供 Flink SQL 和 Custom Code 两种模式，Flink SQL 通过 SQL 编辑器进行作业提交，适用于简单的计算场景。



StreamPark 提交 Flink SQL 模式

Custom Code 模式提供手动上传 Jar 包或者通过选择项目管理中的项目进行构建。

Custom Code 手动提交方式



Custom Code 通过选择项目方式

添加任务中各配置的描述：

作业模式：选择作业类型是 DataStream 作业还是 SQL 作业

执行模式：yarn 提交 flink 任务的模式，可选择 yarn application 或 yarn session

Flink 版本：选择配置的 flink 版本，可从多个选择，参考设置中心 - Flink 版本

并行度： Flink 的并行度，相当于 -Dparallelism.default 参数

任务槽数：每个 Taskmanager 分配的 slot 数

重启次数：任务失败重启的次数

CheckPoint 失败策略：checkpoint 失败处理策略，例如：在 5 分钟内（checkpoint 的失败间隔），如果 checkpoint 失败次数超过 10 次（checkpoint 最大失败次数），会触发操作(发送告警或者重启作业)

总内存：设置分配给 Jobmanager 和 TaskManager 的内存，建议不与 flink 内存同时设置

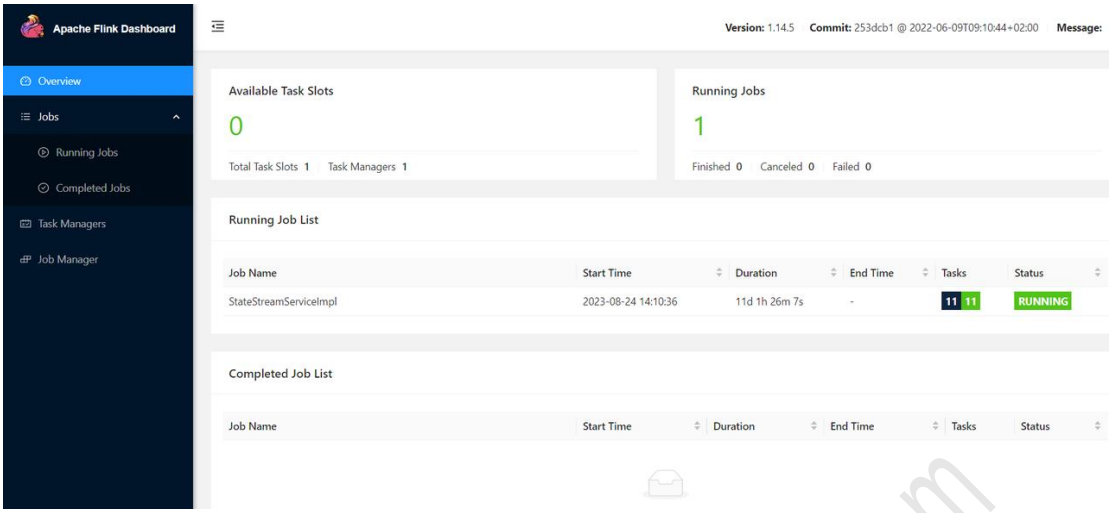
JM 内存：详细的 Jobmanager 内部参数设置

TM 内存：详细的 Taskmanager 内部参数设置

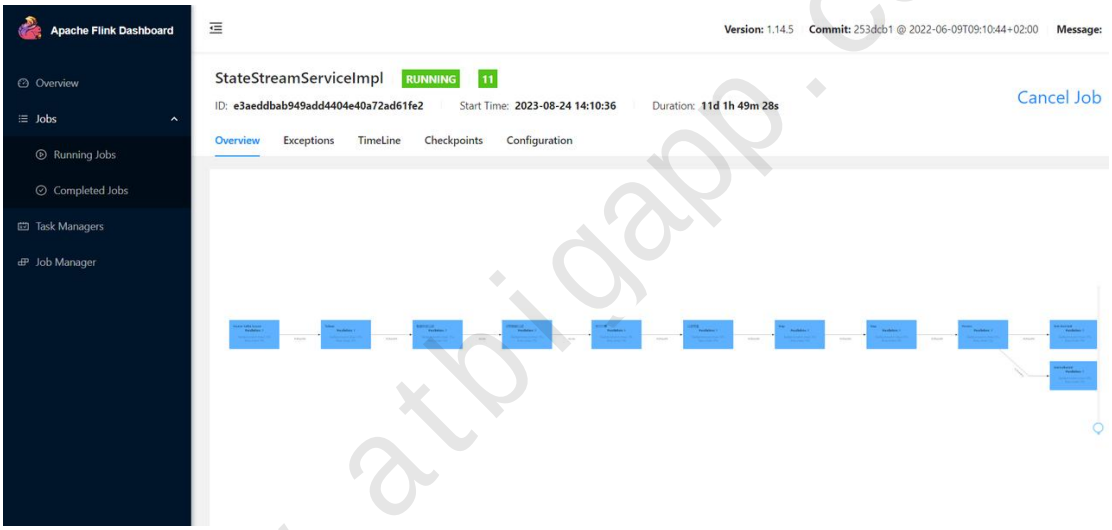
动态参数：程序的其他参数设置，通过 key=value 形式

作业启动后，运行状态显示 RUNNING，可点击任务名称跳转到 Flink 原生界面查看任务运行情况。

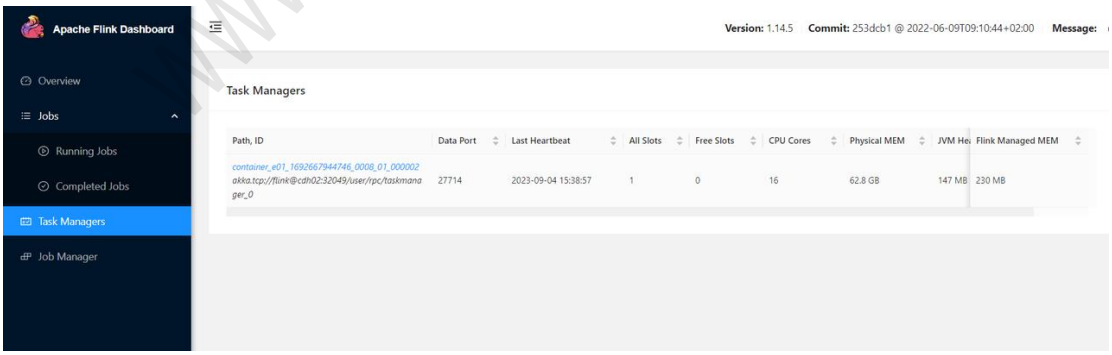
Overview 标签：



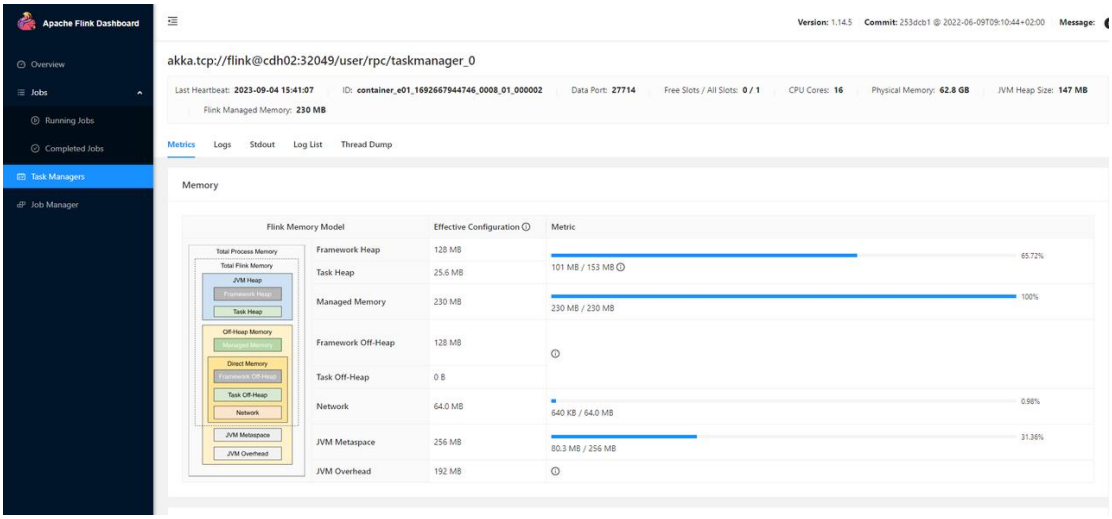
点击 Job Name 可查看任务算子链



Taskmanagers 标签：



点击 container 的 ID 可查看更详细的信息，包括分配的内存使用情况。



3.3. 变量管理

变量管理：Streampark 变量管理，可以供我们进行供我们定义变量，并作用到作业部署上。

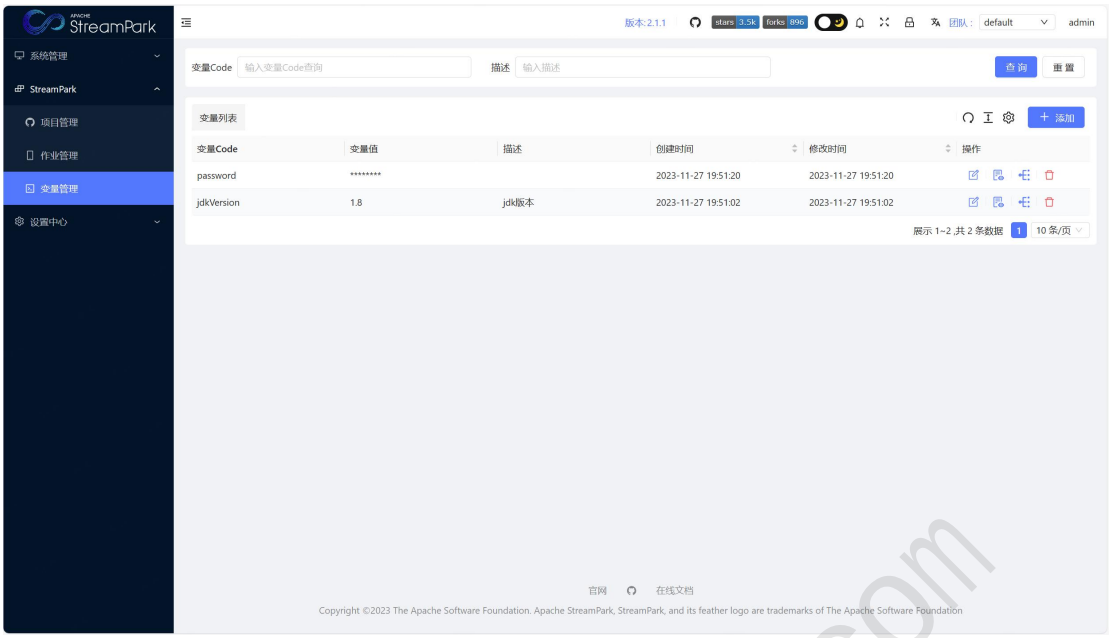
在实际生产环境中，Flink 作业一般很复杂，会依赖多个外部组件，例如，从 Kafka 中消费数据时要从 HBase 或 Redis 中去获取相关数据，然后将关联好的数据写入到外部组件，这样的情况下会导致如下问题：

Flink 作业想要关联这些组件，需要将外部组件的连接信息传递给 Flink 作业，这些连接信息是跟随 StreamPark 的 Application 一起配置的，一旦一些组件的连接信息有变化，依赖这些组件的 Application 都要修改，这会导致大量的操作且成本很高。

团队中一般有很多开发人员，如果对组件连接信息没有统一的传递规范，会导致相同的组件有不同的参数名，这样难以统计外部组件到底被哪些作业依赖。

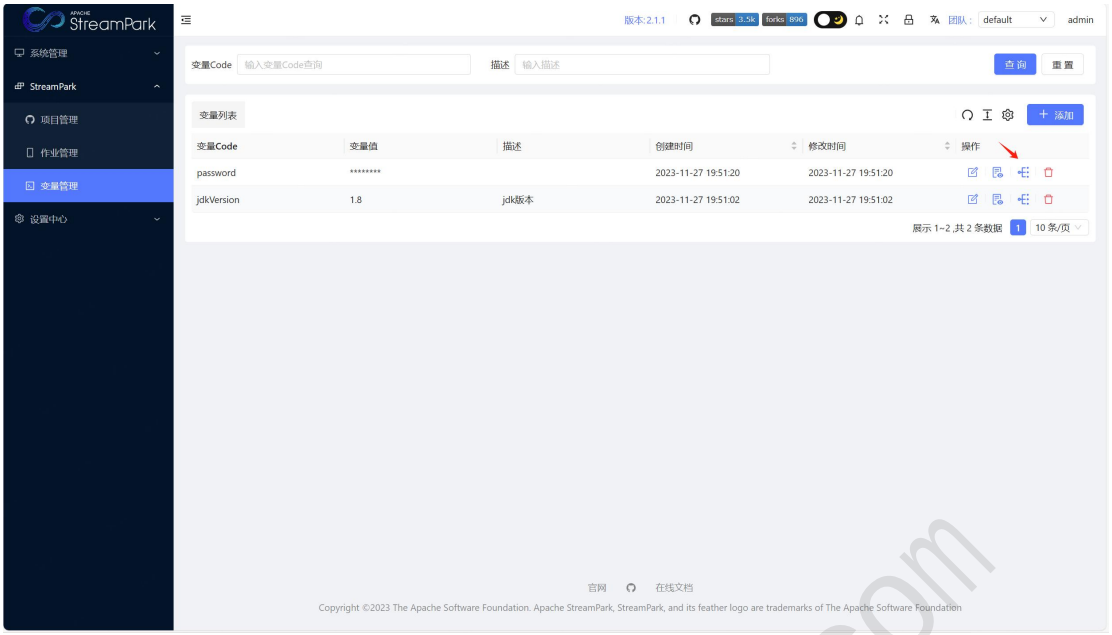
在企业生产中，通常有多个环境，比如有测试环境、生产环境等，很多时候无法通过 IP 和端口来判断属于哪个环境，这样的话本来属于生产环境的 IP 和端口可能配置到了测试环境，导致生产故障。

我们可以通过变量管理界面进行**查看，添加，编辑**以及**查看变量关联了哪些作业**。



点击添加按钮进行变量的添加，通过输入**变量 Code**，**变量值**，**描述**以及是否**启用数据脱敏**进行变量的添加。

点击变量列表中的 **Application 依赖**按钮可以查看变量的引用途径。



从下面可以看出 jdkVersion 的变量正在被 Application 名称为 Flink SQL Demo 的作业进行引用。

我们点击 Flink SQL Demo 作业查看变量是如何被使用的。

可以看到 Flink SQL Demo 的程序参数通过${}表达式引用我们的变量。

至此 StreamPark 的变量管理的功能以及作用就介绍明白了，很多使用程序中的参数可以通过变量的方式进行管理方便统一进行修改。

4. 评价与建议

总体来说，Streampark 功能是挺全的，也使用了不少的技术，它的自我描述如下：

●让流处理更简单，使用 StreamPark 开发;

●可以极大降低学习成本和开发门槛，让开发者只用关心最核心的业务;

●StreamPark 规范了项目的配置,鼓励函数式编程,定义了最佳的编程方式;

●提供了一系列开箱即用的 Connectors;

●标准化了配置、开发、测试、部署、监控、运维的整个过程;

●提供了 scala 和 java 两套 api;

●其最终目的是打造一个一站式大数据平台,流批一体,湖仓一体的解决方案。

个人使用之后，如果能在以下方面能做得更好，就很 nice 了：

●hadoop 集群可支持多个，不要写死，指的是界面可以上传，做集群测试，但是可能要安装多一个 oss 服务;

●flink 版本也支持多个，指的是界面可以上传 sdk，当然也可以指定链接下载解压;

●监控这块不是很明确，目前是通过外链的形式，如果能再集成多一个 Prometheus 这套就更完美了，因为这已经是监控标准了；

●作业管理，Custom Code 模式似乎不支持依赖 jar 的添加，一般打的包都是源码的，如果是 shade 打包方式，类容易出现无法加载的问题；

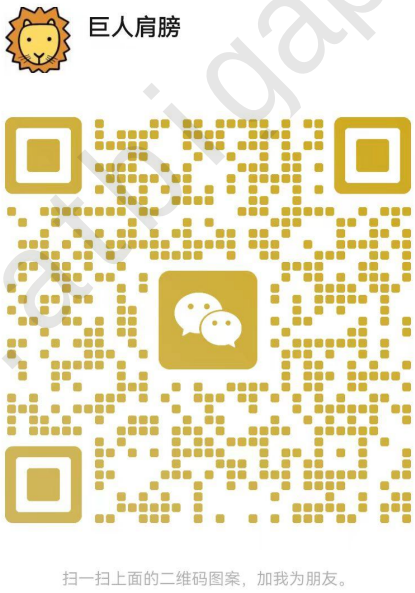
●jar 没有做统一管理（如：connector、format、catalog 等），即使 flnk sql 模式依赖了 jar，写代码时也没有自动补全；

●提交作业到 yarn，kerberos 认证不知道怎么处理，需要配置哪些参数（如：krb5.conf、keytab、principal 等要放到哪）；

●k8s 部署方式，多实例处理；

5.交流讨论

欢迎加入 StreamPark 分享、交流、互助大家庭：



欢迎访问巨人肩膀：www.atbigapp.com

一个致力于大数据技术和应用发展与推广，通过环境及基础能力的提供降低开发者门槛、提高开发者效率的互助互利的大数据开放平台。