思码逸深度代码分析系统 3.116 版本产品说明手册

思码逸

20221230

目录

1 认识思码逸 **7**

2 技术原理 & 术语解释 **8**

2.1 抽象语法树 Abstract Syntax Tree (AST) · · · · · 8

2.1.1 概述 · · · · · 8

2.1.2 示例 · · · · · 8

2.2 代码当量 Equivalent Lines of Code (ELOC) · · · · · 9

2.2.1 指标概述 · · · · · 9

2.2.2 计算原理 · · · · · 9

2.2.3 实例对比 · · · · · 11

2.3 代码影响力 DevRank · · · · · 16

2.3.1 指标概述 · · · · · 16

2.3.2 指标解读 · · · · · 16

2.3.3 指标导航 · · · · · 16

2.3.4 示例 · · · · · 17

2.4 开发价值 Dev Value · · · · · 18

2.4.1 指标概述 · · · · · 18

2.4.2 指标解读 · · · · · 18

2.4.3 指标导航 · · · · · 18

2.5 代码质量问题 Code Quality Issue · · · · · 18

2.5.1 指标概述 · · · · · 18

2.5.2 指标解读 · · · · · 18

2.5.3 指标导航 · · · · · 19

2.5.4 示例代码 · · · · · 19

2.6 代码不重复度 Dryness · · · · · 19
2.6.1 指标概述 · · · · · 19
2.6.2 指标解读 · · · · · 20
2.6.3 指标导航 · · · · · 20
2.6.4 示例代码 · · · · · 20
2.7 静态测试覆盖度 Static Test Coverage · · · · · 21
2.7.1 指标概述 · · · · · 21
2.7.2 指标解读 · · · · · 21
2.7.3 指标导航 · · · · · 21
2.7.4 示例代码 · · · · · 21
2.8 注释覆盖度 Documentation Coverage · · · · · 22
2.8.1 指标概述 · · · · · 22
2.8.2 指标解读 · · · · · 22
2.8.3 指标导航 · · · · · 22
2.8.4 示例代码 · · · · · 22
2.9 函数圈复杂度 Function Cyclomatic Complexity · · · · · 23
2.9.1 指标概述 · · · · · 23
2.9.2 指标解读 · · · · · 23
2.9.3 指标导航 · · · · · 24
2.9.4 示例代码 · · · · · 24
2.10 模块性 Modularity · · · · · 24
2.10.1 指标概述 · · · · · 24
2.10.2 指标解读 · · · · · 24
2.10.3 指标导航 · · · · · 24

3 快速上手 25

3.1 快速开始！ · · · · · 25
3.2 第一步：完成系统基础设置 · · · · · 25
3.2.1 要点概述 · · · · · 25
3.2.2 入口导航 · · · · · 25
3.2.3 基础设置 · · · · · 26
3.2.4 自定义字段 · · · · · 27
3.2.5 SMTP 设置 · · · · · 32

3.2.6 登录鉴权 · · · · · 34
3.2.7 LDAP 设置 · · · · · 36
3.2.8 示例 · · · · · 37
3.2.9 周日报配置 · · · · · 39
3.2.10 示例 · · · · · 41
3.2.11 专家系统 · · · · · 42
3.2.12 设置提交类别 · · · · · 43
3.2.13 注释覆盖度计算规则 · · · · · 46
3.3 第二步：导入代码库 · · · · · 47
3.3.1 导入或新增代码库 · · · · · 47
3.3.2 从 Git 地址新增代码库 · · · · · 48
3.3.3 从 GitLab 导入代码库 · · · · · 55
3.3.4 表格文件导入代码库 · · · · · 59
3.3.5 系统集成导入代码库 · · · · · 69
3.4 第三步：配置用户账号 · · · · · 74
3.4.1 导入或设置账户 · · · · · 74
3.4.2 团队设置 · · · · · 76
3.4.3 角色设置 · · · · · 85
3.4.4 账户设置 · · · · · 91
3.5 第四步：配置项目 · · · · · 107
3.5.1 导入或配置项目组 · · · · · 107
3.5.2 项目组管理 · · · · · 108
3.5.3 项目组用户 _ 团队管理 · · · · · 113
3.5.4 项目组代码库管理 · · · · · 115

4 功能模块介绍 118

4.1 系统各工作区介绍 · · · · · 118
4.1.1 项目表现 · · · · · 118
4.1.2 团队表现 · · · · · 118
4.1.3 工程师表现 · · · · · 119
4.1.4 迭代表现 · · · · · 119
4.1.5 技术债监控 · · · · · 119
4.1.6 工具箱 · · · · · 119

4.2 项目表现 ····· 120
4.2.1 项目表现 ····· 120
4.2.2 效率报表 ····· 138
4.2.3 质量报表 ····· 151
4.2.4 质量报表 (历史数据) ····· 156
4.2.5 同行对比 ····· 162
4.2.6 开发负载 ····· 166
4.3 团队表现 ····· 170
4.3.1 团队表现 ····· 170
4.3.2 效率报表 ····· 173
4.4 工程师表现 ····· 174
4.4.1 工程师表现 ····· 174
4.4.2 贡献者排行 ····· 175
4.4.3 贡献者列表 ····· 177
4.4.4 提交列表 ····· 179
4.5 迭代表现 ····· 182
4.5.1 迭代表现 ····· 182
4.5.2 迭代表现相关设置 ····· 182
4.5.3 工作面板 ····· 190
4.5.4 工作量 ····· 204
4.5.5 迭代质量 ····· 206
4.5.6 迭代进度 ····· 209
4.6 技术债监控 ····· 210
4.6.1 技术债监控 ····· 210
4.6.2 代码问题 ····· 210
4.6.3 代码重复 ····· 214
4.6.4 单测覆盖度 ····· 215
4.6.5 注释覆盖度 ····· 216
4.6.6 函数复杂度 ····· 217
4.7 工具箱 ····· 218
4.7.1 日志 ····· 218
4.7.2 展示板 ····· 222

4.8 系统设置 · · · · · 224
4.8.1 要点概述 · · · · · 224
4.8.2 账户设置 · · · · · 225
4.8.3 角色设置 · · · · · 225
4.8.4 项目设置 · · · · · 225
4.8.5 团队设置 · · · · · 225
4.8.6 代码库管理 · · · · · 225
4.8.7 代码库分析 · · · · · 225
4.8.8 排行榜设置 · · · · · 225
4.8.9 通知设置 · · · · · 225
4.8.10 规则设置 · · · · · 225
4.8.11 技能标签关联 · · · · · 225
4.8.12 系统集成 · · · · · 226
4.8.13 分析配置 · · · · · 226
4.8.14 系统设置 · · · · · 226
4.8.15 迭代表现 · · · · · 226
4.8.16 OpenAPI · · · · · 226
4.8.17 关于系统 · · · · · 226

5 思码逸产品和指标常见问题回答 FAQ 226

5.1 代码当量 · · · · · 226
5.1.1 Q: 为什么 Playground 里展示的 AST 和我在其他地方看到过的不太一样? · · · · · 226
5.1.2 Q: 项目的初始提交的当量通常比较大，但不一定全是该提交者的工作，如何处理? · · · · · 226
5.1.3 Q: 为什么有些厉害的开发人员他的代码当量并不高? · · · · · 227
5.1.4 Q: 对于自动生成的代码文件，会计算代码当量吗? · · · · · 227
5.1.5 Q: 对于项目中属于第三方库的源代码，会计算代码当量吗? · · · · · 227
5.1.6 Q: 代码当量识别第三方库代码的方法是什么? · · · · · 227
5.1.7 Q: 如何计算不同语言的代码当量? · · · · · 227
5.1.8 Q: Merge Commit 的代码当量是如何处理的? · · · · · 227
5.1.9 Q: 反复修改代码会增加代码当量吗? · · · · · 228
5.1.10 Q: 简洁高效代码的代码当量小于复杂代码的代码当量，对于开发人员来说是不是不公平? · · 228
5.1.11 Q: 一次提交大量的代码是否会计入代码当量? · · · · · 228
5.1.12 Q: 如何提高代码当量? · · · · · 228

5.1.13 Q: 编写新功能和代码重构工作的价值如何横向对比，是否能用代码当量比较？ ····· 228
5.1.14 Q: 代码当量：编辑类型加权、节点类型加权、节点重复加权（重复代码），这 3 者权重各组占比是怎样的？是否有节点类型、节点重复加权对应的算法说明？ ····· 228
5.1.15 Q: 什么是节点类型的加权？它的权重是如何计算的？ ····· 228
5.1.16 Q: 重复惩罚的调整系数是如何计算的？ ····· 229
5.1.17 Q: 假设有一个开发人员写了 1000 行代码，但是他恶意注水，通过复制粘贴把这 1000 行变成了 10000 行，他的代码当量会怎样变化？ ····· 229
5.1.18 Q: 开发人员故意 game 代码当量指标，怎么办？ ····· 229
5.2 代码不重复度 ····· 229
5.2.1 Q: 统计代码不重复度时，函数相似度的最小边际距离是多少？ ····· 229
5.2.2 Q: 不重复度是根据产品里代码重复页面的数据计算的吗？ ····· 229
5.3 测试覆盖度 ····· 229
5.3.1 Q: 什么是动态测试覆盖度？什么是静态测试覆盖度？Merico 的测试覆盖属于哪一种？ ····· 229
5.3.2 Q: 统计测试覆盖度时，如何识别测试函数？ ····· 230
5.3.3 Q: 统计测试覆盖度时，测试框架 JUnit 我们支持吗？ ····· 230
5.3.4 Q: 为什么有的文件里的函数实际没有被测试，但 Merico 的指标却显示它的函数已被测试覆盖？ ····· 230
5.4 注释覆盖度 ····· 230
5.4.1 Q: 注释覆盖率是 sonar 的基本功能的数据加工出来的吗，是别的竞品无法给的数据吗？ ····· 230

6 发版说明 230

6.1 版本更新说明 v3.113.0 ····· 230
6.2 版本更新说明 v3.109.0 ····· 236
6.3 版本更新说明 v3.103.0 ····· 238
6.4 版本更新说明 v3.12.0 ····· 240
6.5 版本更新说明 v3.6.0 ····· 242

1 认识思码逸

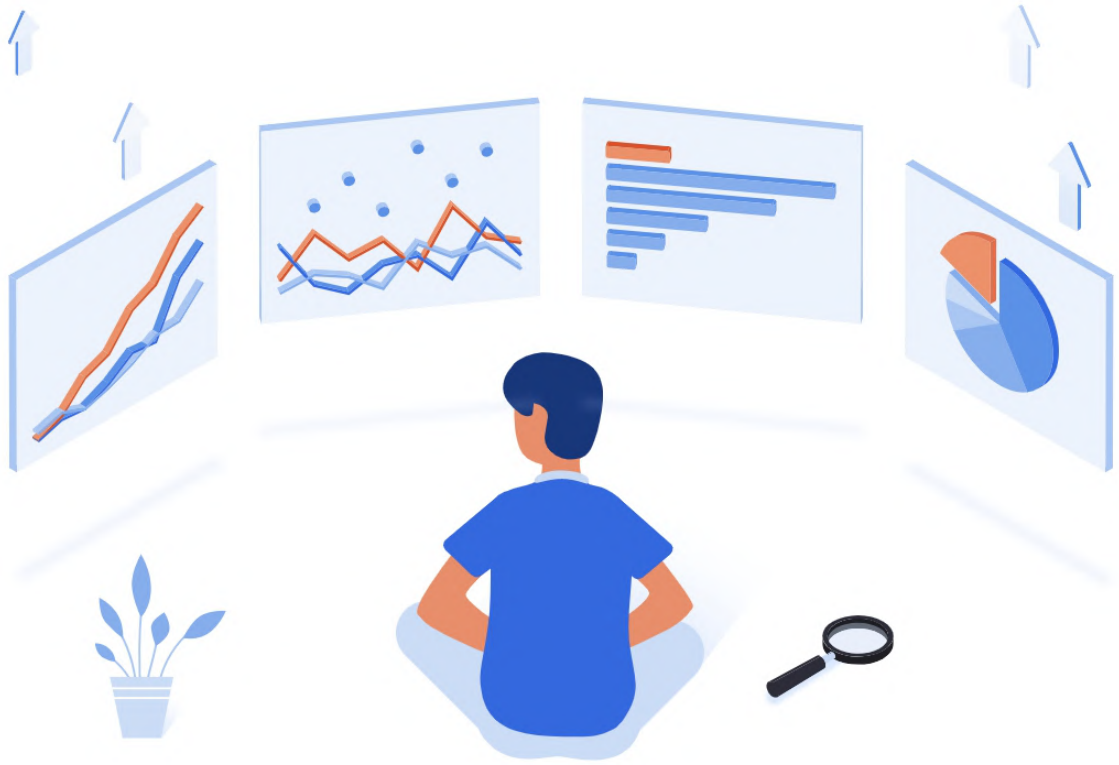

图 1: 欢迎

您好！欢迎使用思码逸，我们很荣幸与您一同开始研发效能提升之旅。

思码逸专为企业研发团队提供一站式研发效能分析平台及配套解决方案。基于深度代码分析技术，思码逸能够从代码库中提取数据，并整合研发全生命周期其他环节数据，提供研发效率、软件工程质量、组织人才发展等不同维度洞见，为企业高管、研发负责人、项目负责人、开发者等团队不同角色提供个性化视图。以更智能的度量驱动研发管理升级，进而推动开发团队与开发者个人的共同提升。

通过量化效能，提高研发过程及结果的可见度，思码逸产品一方面帮助管理者发现效能瓶颈或提效机会，智能辅助研发管理决策；另一方面鼓励团队成员自我驱动与沟通协作。通过调动研发团队整体实践改进，解决效能难以管理、软件工程实践难以落实推动、组织与人才发展缺乏指引等痛点。

快速开始 将在几分钟内帮助你了解如何上手使用思码逸产品。本手册也会随着产品的迭代实时更新，以帮助您最大程度挖掘思码逸产品的价值。

2 技术原理 & 术语解释

2.1 抽象语法树 Abstract Syntax Tree (AST)

2.1.1 概述

抽象语法树是编译过程中构建的一种中间数据结构。编译过程通常经历词法分析、语法分析、语义分析、中间代码生成、机器码生成等多个步骤。其中，抽象语法树是语法分析的结果，语义分析的载体。大部分程序分析和语义理解的技术都是基于对抽象语法树的分析。相比之下，源代码更贴近自然语言的表达，不利于机器理解，而且噪音较多；机器码则面向系统底层运行，高度优化和抽象，已经不能很好地反映人表达的逻辑形式。

各个编程语言会对应不同的抽象语法树定义。我们在各个语言抽象语法树的基础上，进一步提炼出统一的、编程语言无关的通用抽象语法树（universal abstract syntax tree, UAST）。我们在 UAST 上执行统一的算法和模型，而不必为每一门编程语言适配不同的算法和模型。在本文档中，思码逸 Merico 的抽象语法树（AST）通常即指代通用抽象语法树（UAST）。

2.1.2 示例

抽象语法树示例图片：左侧为 commit 提交前的代码抽象语法树，右侧为 commit 提交后的代码抽象语法树

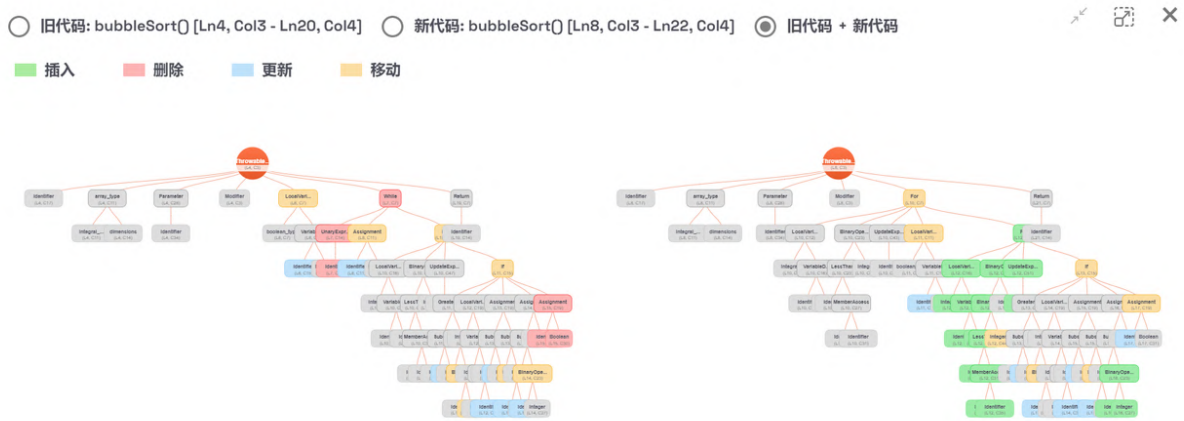

图 2: 抽象语法树示例

2.2 代码当量 Equivalent Lines of Code (ELOC)

2.2.1 指标概述

代码当量是衡量开发者修改代码的工作量的指标。与代码行数（LOC）、提交个数（NOC）等简单指标相比，基于抽象语法树（AST）计算的代码当量能更准确地反应修改代码的工作量。

2.2.1.1 指标优势

代码行数是简单且常用的衡量代码工作量的指标。但是它的缺点很明显，例如：容易受到代码风格、换行习惯、注释、格式化操作等的干扰；无法识别出对代码的实际修改，简单的复制粘贴、移动代码块等会产生大量的行数增删变化。

代码当量很好地解决了这些问题。它将源代码解析成抽象语法树这种更能体现代码语法结构、代码本质的形式，通过比较代码修改前后抽象语法树之间的变化，来计算一次修改行为的工作量。

- 首先，代码被解析为抽象语法树后，消除了代码书写风格、注释格式等与代码逻辑无关因素的干扰。
- 其次，基于树结构的比较，能很好地识别移动代码（Move）、更新代码（Update）等操作，为它们赋予更合理的工作量。同时，在抽象语法树的基础上，代码当量能通过简单的语义分析，区分代码中的“数据”和“逻辑”，弱化非关键的“数据”修改，强化“逻辑”修改。
- 更进一步地，代码当量还有很多智能调节机制来处理实际开发中常见的场景，例如对重复代码的调节、排除由开发工具自动生成的代码、排除第三方库的代码、平衡不同编程语言之间的差异等。

2.2.2 计算原理

代码当量的基础计算过程如下：

1. 分别将修改前的代码和修改后的代码解析为抽象语法树（AST）。
2. 使用 tree diff 算法计算将修改前的 AST 转换成修改后的 AST 的编辑脚本（Edit Script）。编辑脚本里包括四种对树的编辑操作：插入、删除、移动、更新。
3. 对于被编辑的抽象语法树节点，根据它的节点类型和编辑操作类型，分别进行加权计算。
4. 最后，对所有被编辑的节点的加权结果进行求和，即为这次修改的代码当量。
5. 计算过程中如触发各种当量调节规则，如重复代码识别、第三方库识别、自动生成代码识别等，会根据当量调节规则扣减当量。

2.2.2.1 算法图示

下图简单演示了这个过程如何从代码的修改计算出代码当量的数值。

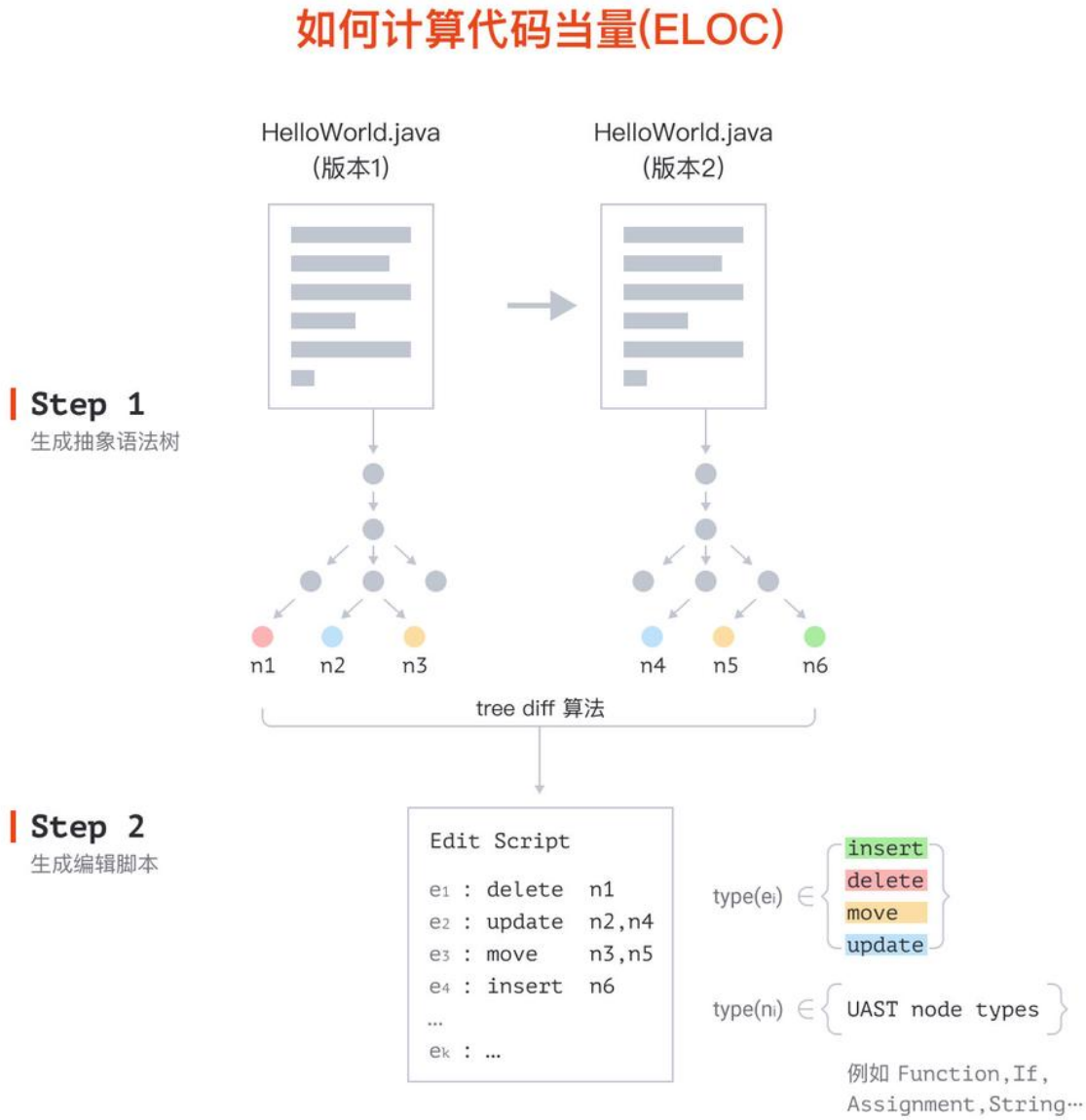
如何计算代码当量(ELOC)
HelloWorld.java
(版本1)
HelloWorld.java
(版本2)
Step 1
生成抽象语法树
n1
n2
n3
n4
n5
n6
tree diff 算法
Step 2
生成编辑脚本
Edit Script
e1 : delete n1
e2 : update n2,n4
e3 : move n3,n5
e4 : insert n6
...
ek : ...
type(ei) ∈
insert
delete
move
update
type(ni) ∈
UAST node types
例如 Function,If,
Assignment,String…
Step 3
计算代码当量
ELOC = Σe We.edit_type · We.node_type
W 是不同的编辑类型和节点类型所对应的权重
* 这里演示的是代码当量（ELOC）的基础原理。在实际运用中，代码当量的计算过程有更多的智能调节机制，例如对重复代码的调节。

图 3: 如何计算代码当量

2.2.3 实例对比

2.2.3.1 例 1

代码行数很容易因为简单的修改而显著增加。比如下面的代码变动，尽管代码修改后本质并没有发生变化，这个修改仍会产生 1 行添加和 4 行删除。

而单纯的格式变化对 AST 没有影响，此段代码修改前后 AST 是相同的，因此其代码当量为 0。

example1.java

```
1  - for(int i = 0; i < 10; i++)
2  - {
3  -     System.out.println("Hello World!");
4  - }
   1 + for(int i = 0; i < 10; i++) {System.out.println("Hello World!");}
```

图 4: 例 1

2.2.3.2 例 2

代码行数不擅长检测代码块的移动。比如下面的代码变动，简单地交换类中函数的顺序会产生 4 行添加和 4 行删除。

但是从抽象语法树的角度，这次修改只是改变了 myMethod() 函数对应节点在其父节点下的顺序，该节点本身未发生任何修改。因此修改 myMethod() 的代码当量为 0。

example2.java

```
1  1  public class MyClass {
2     -   static void myMethod() {
3     -     System.out.println("Hello World!");
4     -   }
5     -
6  2     public static void main(String[] args) {
7  3       myMethod();
8  4     }
   5  +
   6  +   static void myMethod() {
   7  +     System.out.println("Hello World!");
   8  +   }
9  9   }
```

图 5: 例 2

2.2.3.3 例 3

代码行数无法区分不同性质的代码的工作量。考察以下 Python 代码，它的功能是在给定的字典中找到对称对。

测试数据`test_cript`和实际功能函数`find_sym_pairs()`贡献了相等数量的行数（7 行），然而这两段代码所包含的工作量显然不同。

通过识别代码语义，可以区分不同性质的代码内容，进而更合理地评估相关变更所包含的工作量。

example3.py 0 → 100644

```
+ test_dict = {
+     'a' : 1,
+     'b' : 2,
+     'c' : 3,
+     1 : 'a',
+     2 : 'b'
+ }
+
+ def find_sym_pairs(test_dict):
+     for key in test_dict.keys():
+         val = test_dict.get(key)
+
+         if test_dict.get(val) == key:
+             yield key, val
+     return
+
+ print([(k, v) for k, v in find_sym_pairs(test_dict)])
```

图 6: 例 3

2.2.3.4 例 4

以 ELOC Playground 中的 Java 示例代码为例，从旧的代码片段变为新的代码片段，产生的当量为 18.75：

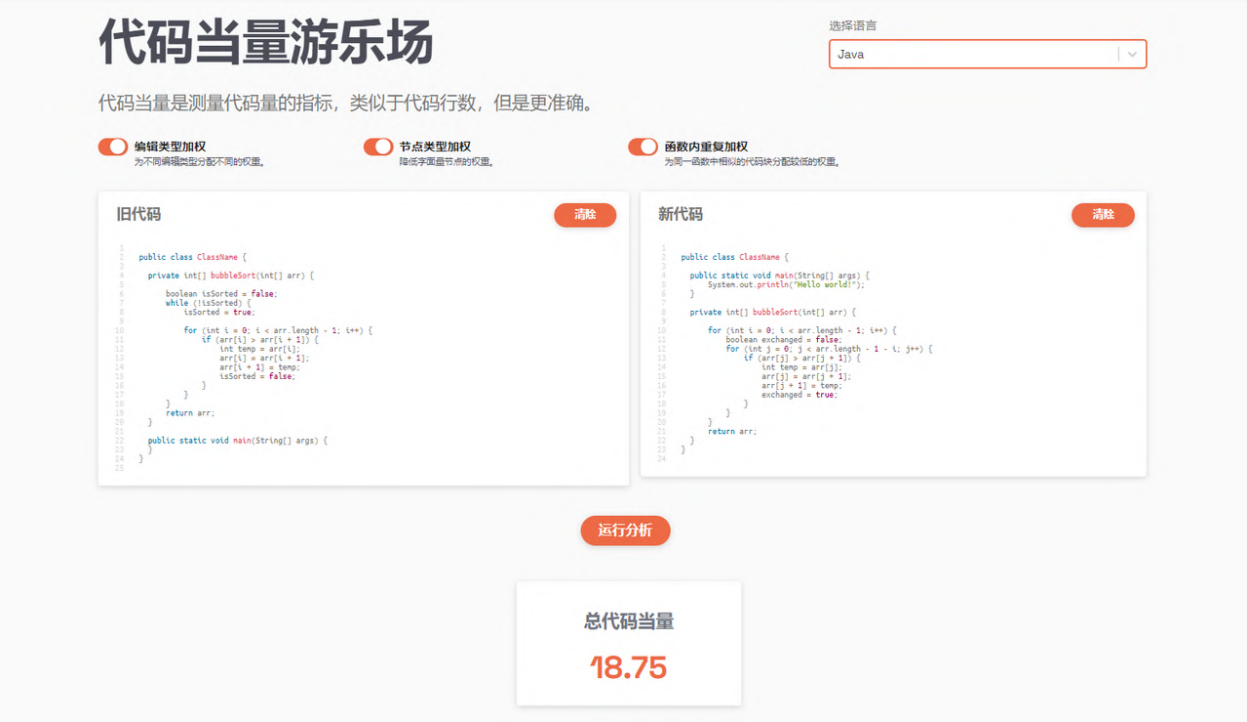

图 7: 例 4

当量分析过程中，识别到如下函数对：

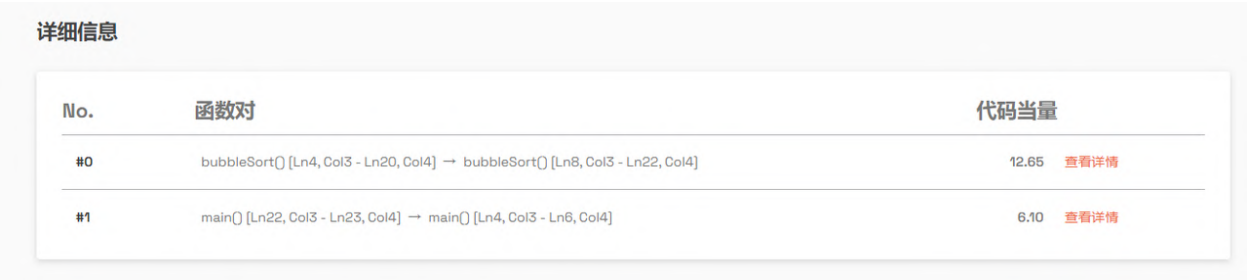

图 8: 当量分析过程中识别到的函数对

可以看到各函数对从旧代码到新代码的变化过程中，产生了代码当量，点击 **查看详情**按钮，可以查看每个函数对的 AST 解析展示：

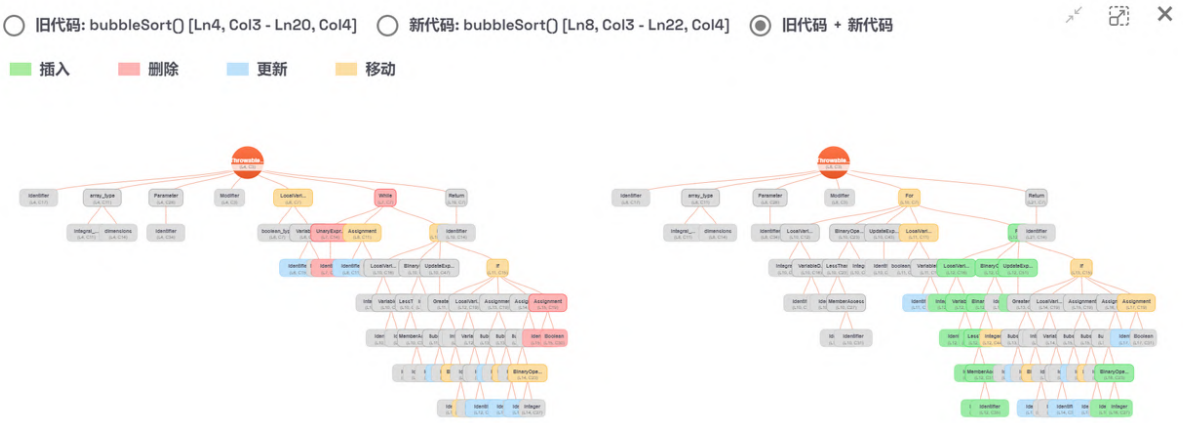

图 9: 每个函数对的 AST 解析展示

可以看到旧代码片段和新代码片段分别被解析为了对应的抽象语法树（AST），抽象语法树中包含不同类型的 AST 节点。并且在从旧树到新树的变化过程中，各个 AST 节点有不同的编辑类型。

在上图中，各 AST 节点上标识了其节点类型，如：`Identifier`，`Parameter`，`Modifier`；并用不同颜色标识出不同的 AST 节点编辑类型，包括：插入，删除，更新，移动。

不同的 **AST 节点类型**各有其当量计算权重，不同 **AST 节点编辑类型**亦各有其当量计算权重。可以通过分别关闭三种加权开关中的一个来了解主要的三类加权对当量分析的影响。

以以上 ELOC PlayGround 里自带的 Java 代码示例为例：

编号	编辑类型加权	节点类型加权	函数内重复加权
1	打开	打开	打开
2	关闭	打开	打开
3	打开	关闭	打开
4	打开	打开	关闭

编辑类型加权：为不同编辑类型分配不同的权重。

节点类型加权：降低字面量节点的权重。

函数内重复加权：为同一函数中相似的代码块分配较低的权重。

在思码逸深度代码分析系统企业版中，代码当量计算过程里除了以上节点级别的当量调节规则外，还包含了函数级别、文件级别、提交级别的当量调节规则，通过抽象语法树的节点变化和各级别当量调节规则的共同作用，最终得到各次提交的代码当量。

2.3 代码影响力 DevRank

2.3.1 指标概述

代码影响力是综合了函数的代码当量和函数调用关系的指标，也就是论文里提到的 DevRank。为便于理解，我们以百分比的形式计算该指数，可以直观理解为贡献比例。

其中调用关系反映函数间相互依赖的关系。越多函数直接或间接地依赖于某个函数，那么该函数的开发影响力就越高；也意味着，如果该函数作出修改，回归测试的范围相应越大，即修改成本较高。通常来说，这类函数的重要性也越高。

函数之间带历史的依赖关系构成的图被称为“**调用-提交关系图（Call Commit Graph）**”。代码影响力即是在这张图上运用类似 PageRank 的图算法计算出的。（公式详见我们在 FSE’18 上发表的论文）。

2.3.2 指标解读

代码影响力越大代表该提交或者该开发者贡献代码对整个项目代码库的影响越广，需要引起更多的关注，如确保函数被有效的测试覆盖、注释覆盖以及代码评审。

2.3.3 指标导航

工程师表现 > 贡献者排行 > 代码影响力榜

2.3.4 示例

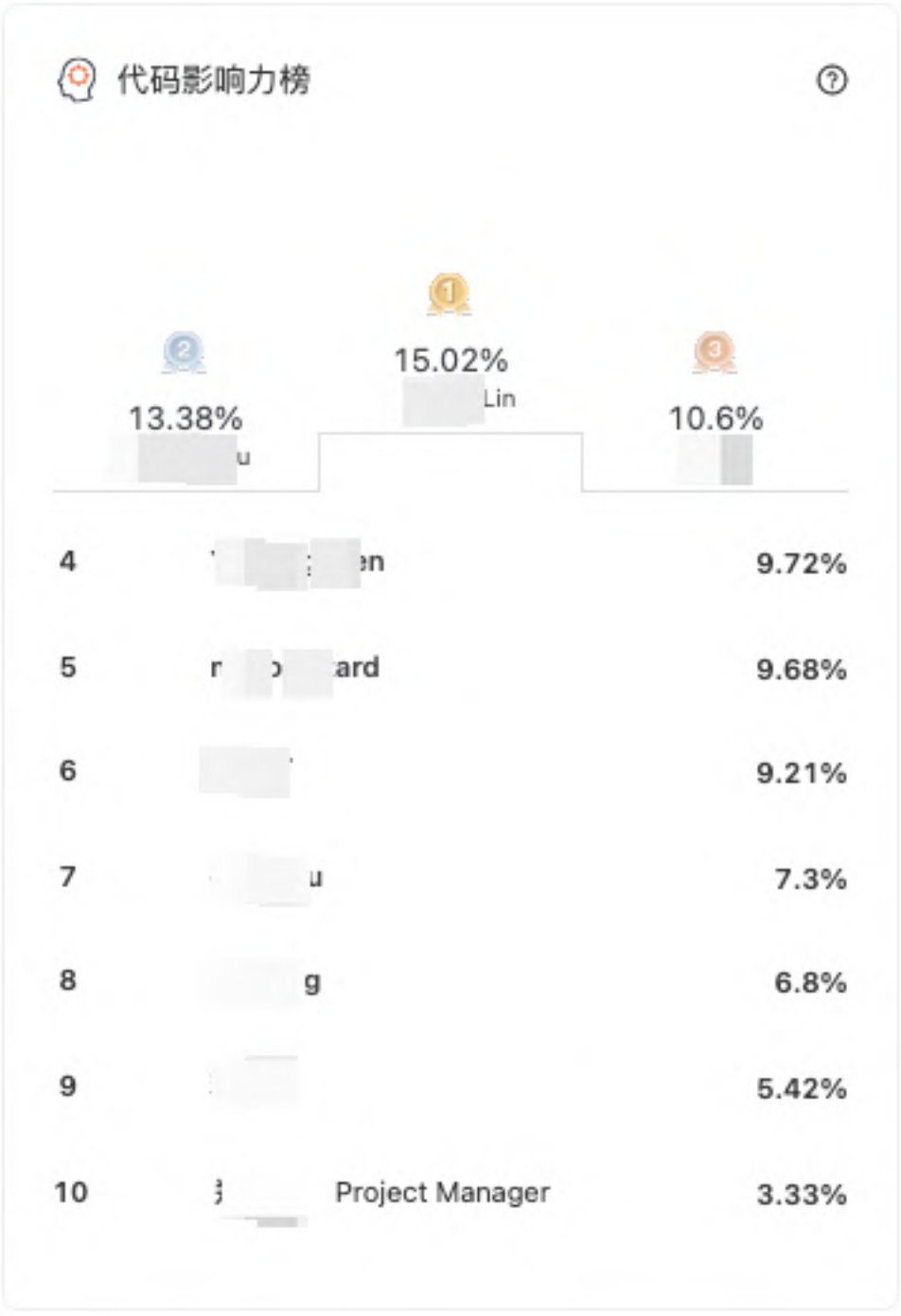

图 10: 代码影响力榜

2.4 开发价值 Dev Value

2.4.1 指标概述

目前的开发价值数值上即为代码影响力。

开发价值的改进方向：开发价值旨在成为综合了代码当量、代码影响力和工程质量的综合指数，更全面的反映开发的价值。其中工程质量指标包括不重复度、测试覆盖度、注释覆盖度、代码问题数、圈复杂度等。

2.4.2 指标解读

开发价值越高意味着代码产出、影响力、质量的综合评分越高。不能一味追求高产出而把代码写的过于复杂、冗余，也不能只注重代码质量而降低开发效率，需要平衡效率和质量达到价值最优。

2.4.3 指标导航

无

2.5 代码质量问题 Code Quality Issue

2.5.1 指标概述

代码质量问题是思码逸通过部分自定义的代码质量规则和集成一些开源的代码质量检查工具检测出来的。目前已集成的开源工具包括 SonarQube、Cppcheck。

2.5.2 指标解读

质量问题按类型分为六类：

1. **错误（Bug）：**即代码中存在的明显错误，破坏了您代码的正常运行，需要立即修复。
2. **漏洞（Vulnerability）：**即代码中容易受到攻击的细节，需要您提起关注。
3. **异味（Code Smell）：**即代码中可能导致深层次问题的症状，例如冗长的参数列表、过长或过短的变量名。如果您扫描出此类问题，说明您的代码有些混乱且难以维护。
4. **安全热点（Security Hotspot）：**存在对安全性敏感的代码。
5. **性能（Performance）：**该问题很可能引起系统运行时的性能问题。
6. **可移植性问题（Portability）：**代码有局限性，不易移植到其他环境中。

质量问题按严重性分为五个级别，从高到低分别是：

1. **阻塞（Blocker）：**很可能会影响生产环境正常运行的错误，例如：内存泄漏，未关闭的 JDBC 链接等等。必须立即修复代码。

2. **严重（Critical）：**可能会影响生产环境正常运行的错误，也可能是代表安全漏洞的问题，例如：空的 catch 块，SQL 注入等等。必须立即检查代码。
3. **主要（Major）：**严重影响开发人员生产力的质量缺陷，例如：无效的代码逻辑，重复的代码，未使用的参数等等。
4. **次要（Minor）：**稍微影响开发人员生产力的质量缺陷，例如：行不应太长，“switch”语句应至少包含 3 种情况等等。
5. **提示（Info）：**仅为提示信息。

查看测出的代码质量问题时，应根据项目特性和实际需求，约定项目的代码规范，关注重点问题，屏蔽轻微问题的干扰。建议根据问题的严重程度，按时间节点或迭代节奏明确相应的解决比例要求。日常开发中，要适时进行 Code Review。外部链接参考：https://docs.sonarqube.org/latest/user-guide/issues

2.5.3 指标导航

技术债监控 > 代码问题

2.5.4 示例代码

```
function seek(input) {
    let target = 32; // 不符合规范的写法
    for (let i of input) {
        // 不符合规范的写法
        if (i == target) {
            return true;
        }
    }
    return false;
}
```

这段代码的第 2、3 行存在严重程度为“主要”的代码异味：Unchanged variables should be marked “const”。在 Javascript 中，对于初始化后其值就不再改变的变量，应该用`const`明确地表示“该变量不会、也不应该被修改”。这样使得代码更加清晰，易于维护。

2.6 代码不重复度 Dryness

2.6.1 指标概述

代码不重复度体现的是项目中函数的不重复程度。

对于当前项目中的函数，我们使用基于 Minhash 和 LSH 的文本查重算法，寻找出相似度较高的函数，认为它们互相之间是重复的函数，并按重复关系分组。不重复度即为：无重复的函数数目占项目中总函数的比例。

2.6.2 指标解读

代码不重复度越高，意味着项目中重复的函数较少，更加符合“Don’t repeat yourself”原则，代码的可维护性较好。

通常建议对于重复或者相似的逻辑，将其提炼成函数或者抽象出可以公用的基类，从而增加可读性，也能减少需要的单元测试数目。但同时也要在不重复度和函数圈复杂度上做好平衡，避免将一段代码封装得过于复杂。

2.6.3 指标导航

技术债监控 > 代码重复

2.6.4 示例代码

```
def hello():
    print('Hello world')

def remove_node_meta_v1(self, node_meta):
    """
    Remove indexed function(node_meta)
    """
    if node_meta not in self._node_meta_mh:
        return

    mh = self._node_meta_mh.pop(node_meta, None)

    self._lsh.remove(node_meta)

def remove_node_meta_v2(self, node_meta):
    """
    Remove indexed function(node_meta)
    """
    if node_meta not in self._node_meta_mh:
        return

    mh = self._node_meta_mh.pop(node_meta, None)
    if not mh:
        return

    self._lsh.remove(node_meta)
```

本例中有三个函数 `hello()`，`remove_node_meta_v1()`，`remove_node_meta_v2()`。

其中`remove_node_meta_v1()`和`remove_node_meta_v2()`非常相似，他们会被判定为重复函数；而`hello()`则没有与之相似的函数。

因此本例的代码不重复度为 1 / 3 = 0.33。

2.7 静态测试覆盖度 Static Test Coverage

2.7.1 指标概述

被测试函数覆盖的函数占项目中非测试函数总数比例。

对于各个编程语言，我们收集了一些主流测试框架的路径要求或者命名规范，根据文件路径判断某文件是否是测试文件。如果该文件是测试文件，那么里面定义的函数就归到测试函数的类别。

然后根据静态分析，找到测试函数中所调用到的函数，认为这些函数即为被测试覆盖。这里的调用关系具有传递性。即，如果函数 A 调用了函数 B，而测试函数 Test 里调用了 A，那么 A 和 B 都视为被 Test 所覆盖。

静态测试覆盖度 = 被测试覆盖的函数数目 / 项目中非测试函数的总数 = 被测试覆盖的函数数目 / (被测试覆盖的函数数目 + 未被测试覆盖的函数数目)

2.7.2 指标解读

测试覆盖度数据越高，意味着项目中被测试覆盖的函数比例越多，代码的可靠性较好。

从长远来说，提升测试覆盖度，可以提高代码质量，减少维护成本，降低重构难度。但是测试覆盖度并非越高越好，它会加大开发者的工作量，需要考虑投入产出比。

建议对于核心函数、复杂函数增加单元测试覆盖，同时应按照测试框架的要求规范测试文件、测试函数的命名。而对于项目整体，需关联效率和其他质量指标，综合分析，找到平衡点。

2.7.3 指标导航

技术债监控 > 单测覆盖度

2.7.4 示例代码

```python
# my_functions.py

def my_add(a, b):
    return a + b

def my_sub(a, b):
    return a - b
```

```python
# test_my_functions.py

def test_my_add():
    assert my_add(1, 2) == 3
```

本例中共有两个文件共三个函数。其中test_my_functions.py会被识别出为测试文件，test_my_add()即为测试函数。test_my_add()中调用了my_add()，my_add()即为被测试所覆盖的函数。my_sub()未被测试函数调用，它未被测试覆盖。

本例的测试覆盖度为 1/2 = 0.5

2.8 注释覆盖度 Documentation Coverage

2.8.1 指标概述

有注释的函数占项目中总函数个数的比例。

函数的注释包括以下几种情况:

1. 位于函数内部的注释
2. 位于函数上方，紧挨着函数的注释
3. Python 的文档字符串（Docstring）

2.8.2 指标解读

注释覆盖度数值越高，意味着项目中有注释的函数数目越多，代码的可读性较好。

通常建议结合业务逻辑和函数复杂度，梳理项目的注释规范，结合业务特征明确注释覆盖度标准，为复杂的函数适当添加注释。

2.8.3 指标导航

技术债监控 > 注释覆盖度

2.8.4 示例代码

```
class MyClass:
    def __init__(self, name):
        """
        Init with a name
        """
        self.name = name

    def greet(self):
        message = 'Hello, ' + self.name
        # print a greeting message
        print(message)

    # This method with return the name
    def get_name(self):
        return self.name
```

```
    def get_upper_name(self):
        return self.name.upper()

    # get name with lower case

    def get_lower_name(self):
        return self.name.lower()
```

在这个代码片段中共有 5 个函数 `__init__()`, `greet()`, `get_name()`, `get_upper_name()`, `get_lower_name()`。

其中有注释的函数为：

- `__init__()`: 有符合 Python 定义的文档字符串
- `greet()`: 函数内部有注释
- `get_name()`: 函数上方有紧挨着的注释

没有注释的函数为：

- `get_upper_name()`: 没有任何注释
- `get_lower_name()`: 上方的注释与函数分离，不符合格式要求

因此这个片段的注释覆盖度为 3/5 = 0.6

2.9 函数圈复杂度 Function Cyclomatic Complexity

2.9.1 指标概述

圈复杂度也称为条件复杂度或者循环复杂度，是一种衡量代码复杂度的标准。函数的圈复杂度数值上为函数中线性独立路径的个数。计算方式为函数控制流程图中条件判定节点的数量加一。

2.9.2 指标解读

当函数中的条件判断语句越多，意味着函数的逻辑越复杂，出错风险和维护难度便会增加。圈复杂度较高的函数需要引起更多的关注，如确保函数被有效的测试覆盖，或者适时重构以降低圈复杂度。通常建议圈复杂度保持在 10 以下。

降低圈复杂度的技巧通常有：

1. 将部分逻辑拆分成独立的函数
2. 优化算法
3. 简化、合并条件表达式
4. 用合理的数据结构（如键值对）替代连续的`if-else`或者`switch`语句

2.9.3 指标导航

技术债监控 > 选择具体代码库 > 函数复杂度 > 圈复杂度

2.9.4 示例代码

```
def hello(name):
    print('Hello,' + name)

def max2(a, b):
    if a > b:
        return a
    return b
```

在函数hello()中，条件判定节点个数为 0，因此它的圈复杂度为 1。在函数max2()中，有一个条件判定节点 If，因此它的圈复杂度为 2。

2.10 模块性 Modularity

2.10.1 指标概述

是衡量当前项目中函数调用关系的模块化程度的指标。

通过分析项目当前状态中函数间的调用关系图（Call Graph），基于图算法找出调用关系图的最优划分，计算出在这个最优划分下图的模块度（详见https://en.wikipedia.org/wiki/Modularity_(networks)），即为项目的模块性。具体公式为 https://en.wikipedia.org/wiki/Modularity_(networks)#math_3。

2.10.2 指标解读

模块性反映了项目函数间调用关系的紧密程度。数值越高，即调用关系越紧密，项目内聚程度越高。

模块性分数的高低通常仅作为参考，需要结合用户项目自身的业务逻辑来看，有的项目或功能就是高内聚的，模块性就高，有的项目作为公共库主要被其他项目使用，模块性可能就相对偏低。

目前的模块性反映问题不够直观，思码逸将对本指标进行改进。

2.10.3 指标导航

项目表现 > 质量报表 > 模块性

项目表现 > 同行对比 > 质量 > 模块性

3 快速上手

3.1 快速开始!

只需完成如下简单四步，您就可以使用思码逸系统来进行代码分析啦!

Step 1：完成系统基础配置

Step 2：导入或新增代码库

Step 3：导入或设置账户

Step 4：导入或配置项目组

对接外部系统需要的权限如下：

对接系统	功能模块	需要权限	为什么需要此权限?
GitLab	直接新增代码库	read（reviewer）	需要读取代码库进行分析
GitLab	GitLab 系统集成	maintainer+admi	需要配置 repo 的 deployKey
GitLab	GitLab 导入代码库	developer 开发者	使用 token 完成 repo 的读取，但只能读取有 developer 以上权限的 repo；新的方式不设置 deployKey
GitLab	GitLab 导入账户	admin 管理员	需要读取用户的 email，只有 admin 能完成。
JIRA	JIRA 集成	maintainer/admin	如果需要 jira 和当量交叉分析，那么需要 admin 权限导入 email；否则只需要对应 project 的维护者

完成！开始您的思码逸之旅吧!

3.2 第一步：完成系统基础设置

3.2.1 要点概述

- 完成基础的系统设置后，即可新增或导入代码库来进行数据分析。
- 系统设置中，包含：基础设置、自定义字段设置、SMTP 设置、登录鉴权设置、LDAP 设置、周日报配置、专家系统设置、设置提交类别、注释覆盖度计算规则。

3.2.2 入口导航

设置 > 系统设置

图 11: 设置

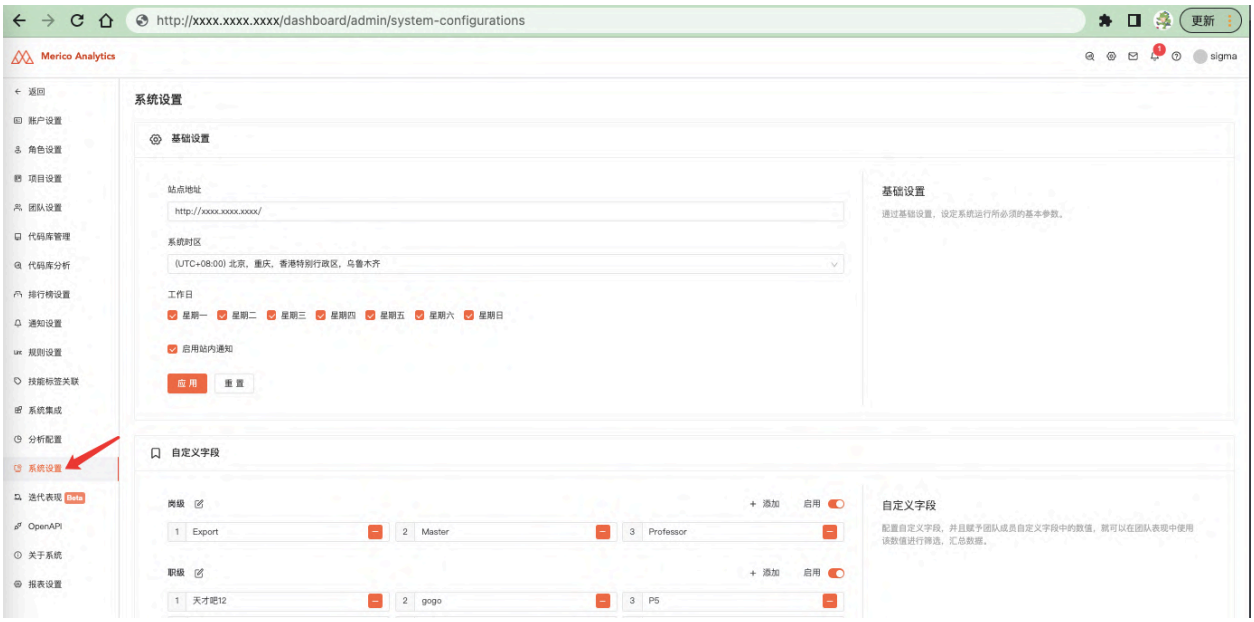

图 12: 系统设置

3.2.3 基础设置

通过基础设置，设定系统运行所必须的基本参数。点击 **基础设置**版块处的 **应用**按钮，保存基础设置信息。

图 13: 基础设置

3.2.3.1 站点地址

当前系统站点地址，以 “/” 结尾，与当前访问的地址一致。如果两者不一致，可能会出现问题：

图 14: 站点地址

3.2.3.2 系统时区

中国大陆一般设置为当前所在时区 UTC+8，周报/日报邮件发送时间和系统设置时区一致。

3.2.3.3 工作日

周报中的统计数据，只统计工作日相关数据，不包含非工作日数据。

3.2.3.4 启用站内通知

勾选此选项时，会按照【通知设置】中的设置规则给相关账户发送站内通知。

3.2.4 自定义字段

3.2.4.1 概述

配置自定义字段，并且给团队成员账户赋予自定义字段中的值，即可在【团队表现】中使用该自定义字段值进行筛选，汇总数据。

如：添加自定义字段“岗级”，并设置该自定义字段的三个可选值：Export，Master，Professor，即可在【账户设置】处给账户配置此三个岗级。

完成自定义字段设置后，点击此版块的**应用**按钮，保存此处设置。

3.2.4.2 示例

3.2.4.2.1 配置自定义字段

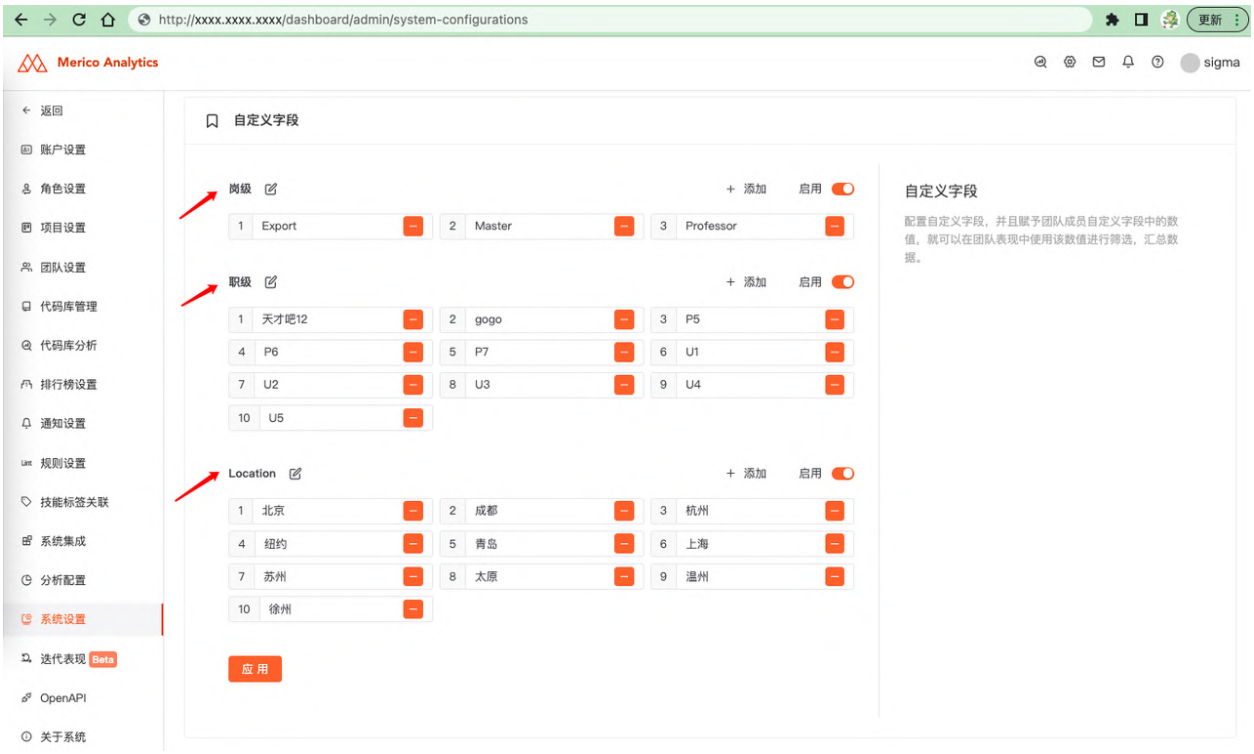

图 15: 配置自定义字段

3.2.4.2.2 账户设置中应用

自定义字段，亦即支持客户定制自己想要的字段，因此示例内容仅供参考

自定义字段-岗级

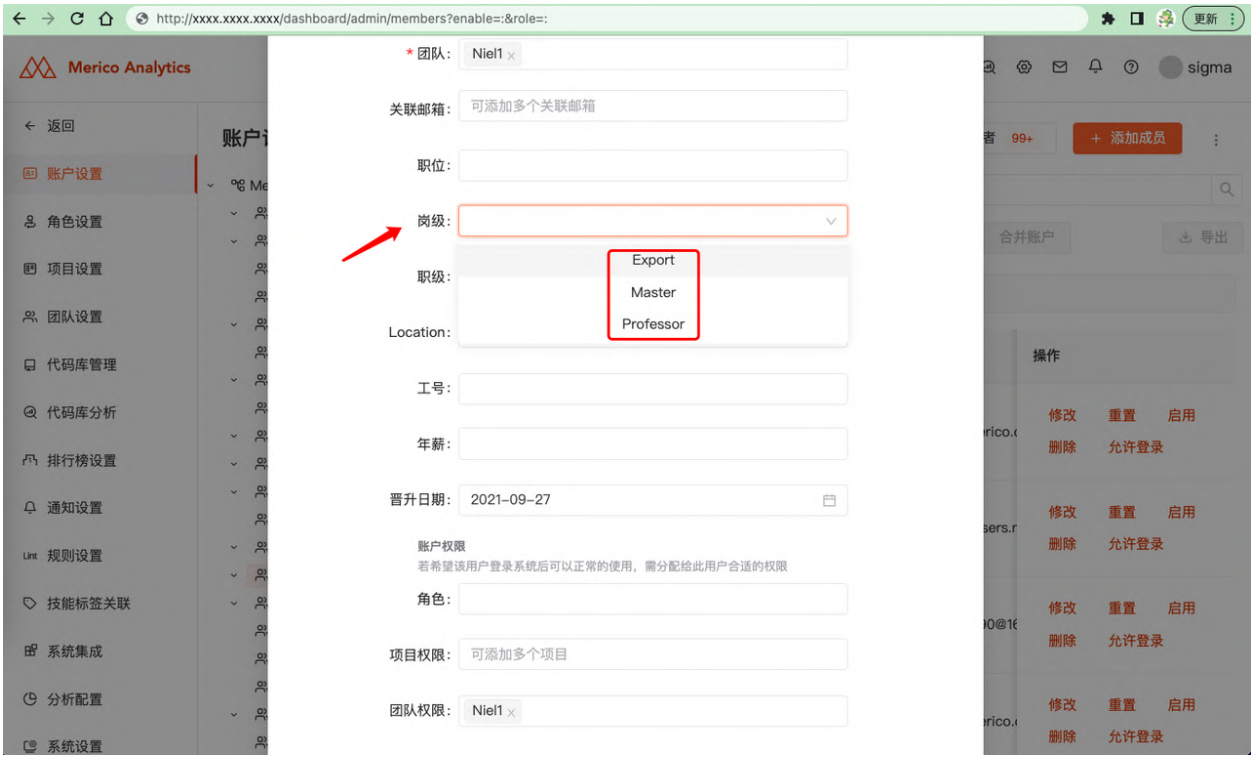

图 16: 自定义字段 - 岗级

自定义字段-职级

图 17: 自定义字段-职级

自定义字段-Location

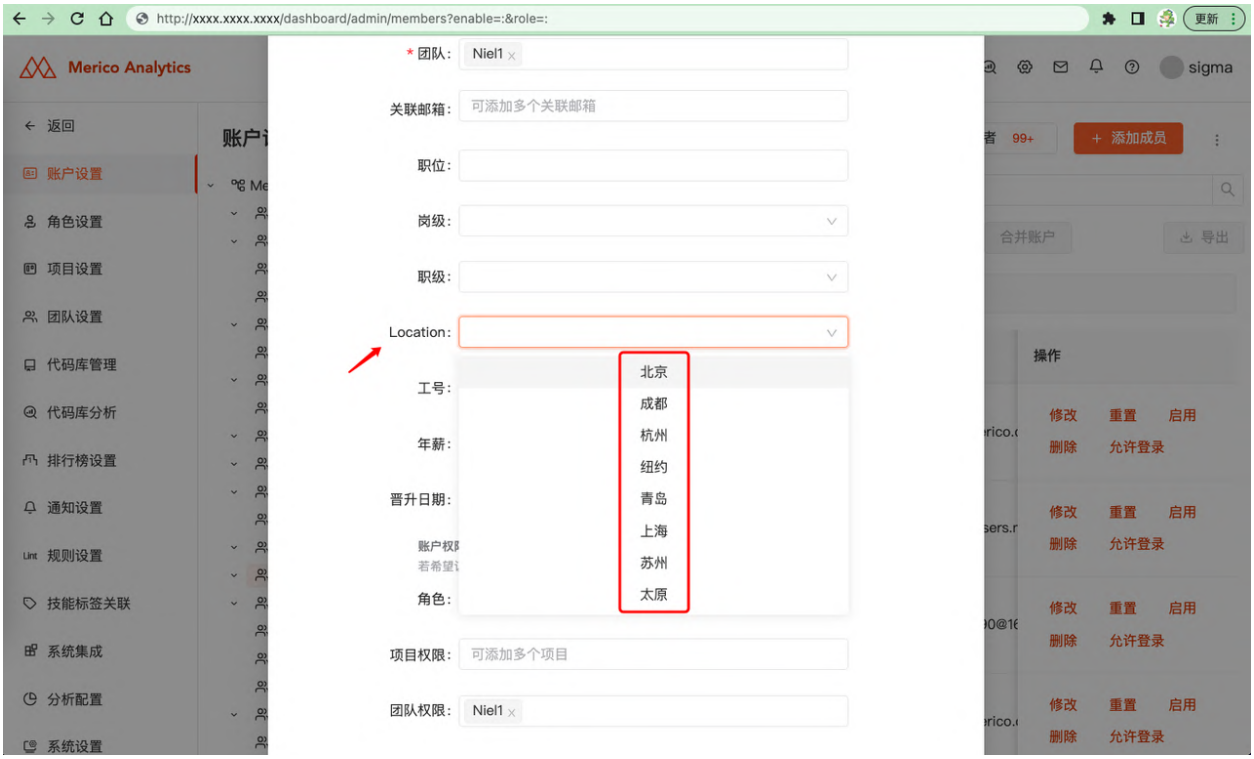

图 18: 自定义字段-Location

3.2.4.2.3 团队表现中应用

【团队表现】>【效率报表】

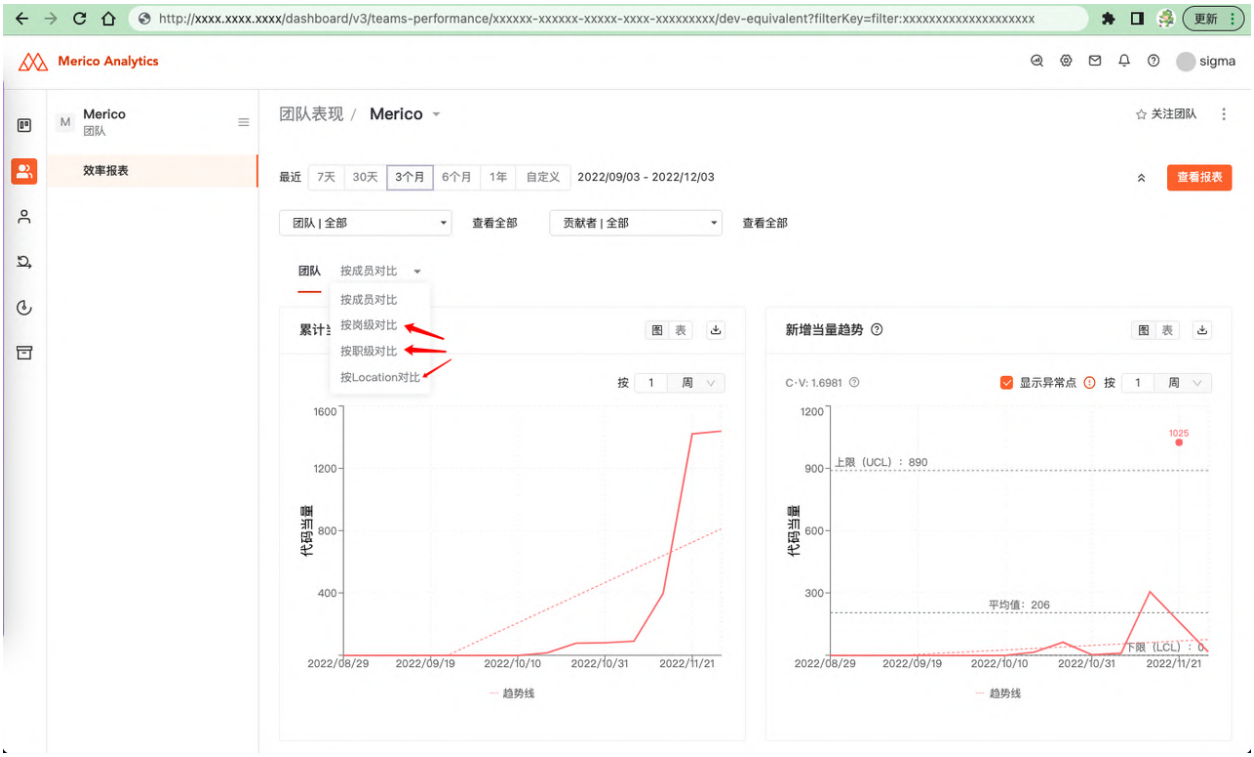

图 19: 【团队表现】>【效率报表】应用自定义字段

3.2.5 SMTP 设置

3.2.5.1 概述

配置系统发件邮箱、发件邮箱密码、SMTP 服务器地址信息，通过 SMTP 服务器发送邮件，如成员账户信息、修改密码信息、周报/日报邮件。

如果没有配置 SMTP 发件邮箱，那么启用新账户的密码和给已有账户重置密码需要在【账户管理】界面上直接输入需要的密码，而不是通过邮件发送密码。

点击 **SMTP 设置**版块处的 **应用**按钮，保存发件设置信息。

3.2.5.2 示例

例如，我们设置一个 163 邮箱，作为发件邮箱

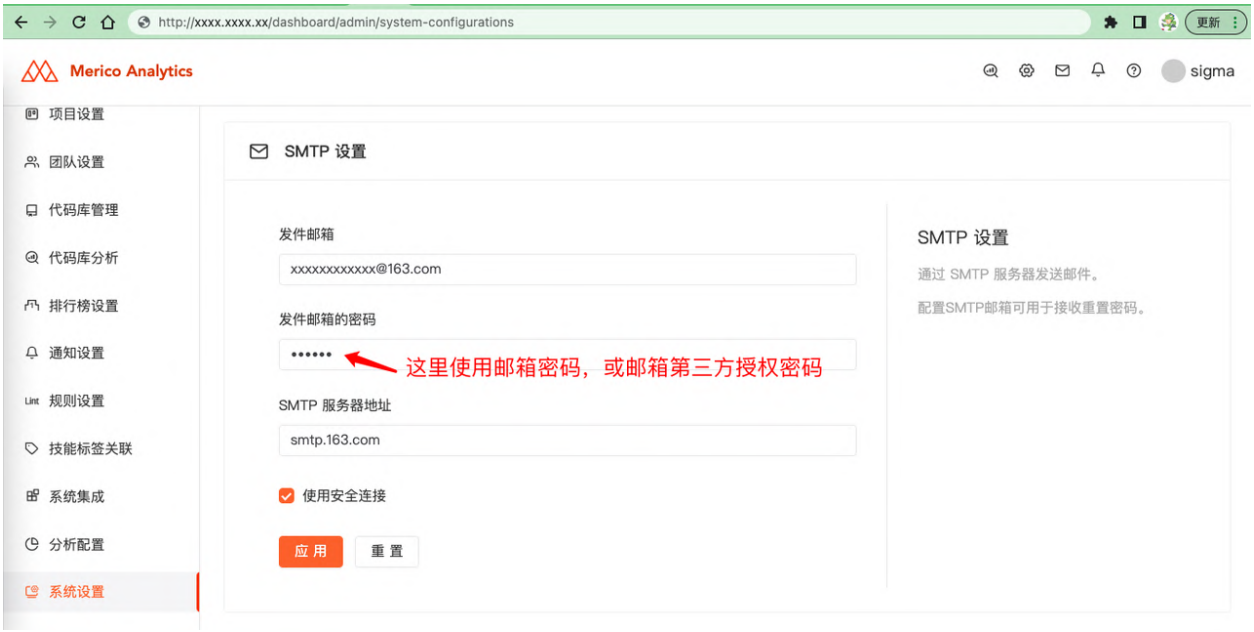

图 20: SMTP 设置示例

163 邮箱设置示例

图 21: 163 邮箱设置示例

3.2.5.3 备注

1. 其他公网邮箱可以参考 163 邮箱配置
2. 客户私有部署的邮箱服务，请联系您部署邮箱服务的运维同事确认 SMTP 服务器地址及使用邮箱密码还是第三方授权密码

3.2.6 登录鉴权

3.2.6.1 概述

通过配置基于 OAuth2.0 的参数项，用户可以凭借在其他平台的账号，通过授权的方式一键登录思码逸系统。

点击 **登录鉴权**版块处的 **应用**按钮，保存登录鉴权信息。

登录鉴权分为简易模式和完整模式

简易模式：

1、默认为简易模式，填写三个必填项，然后可以点击应用生效。

完整模式

1、点击下箭头可以进入完整配置模式。再次点击可收起。

3.2.6.2 示例

简易模式

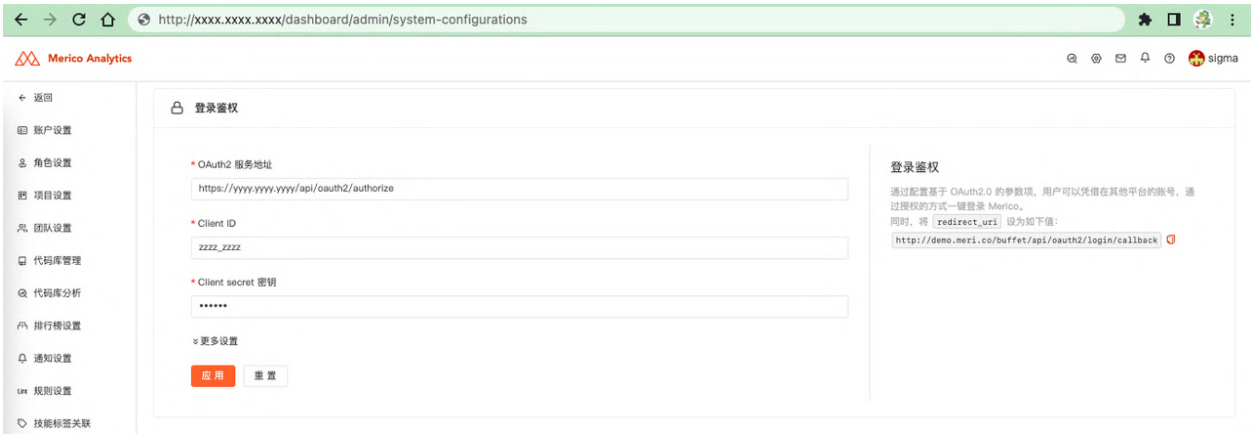

图 22: 简易模式

完整模式

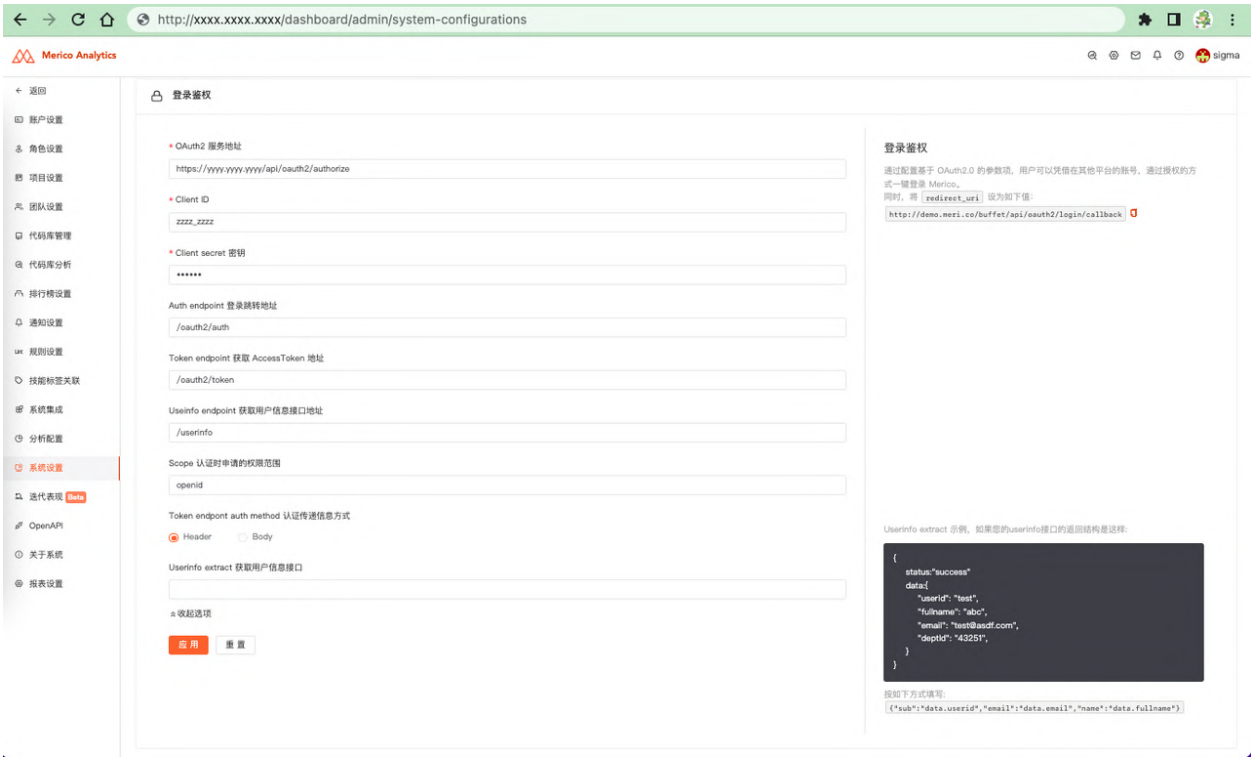

图 23: 完整模式

3.2.7 LDAP 设置

3.2.7.1 概述

通过设置 LDAP 参数，可以接入企业内部的 LDAP 服务器，实现单点登录。如需了解 LDAP 参数含义，请参照如下 RFC：

https://tools.ietf.org/html/rfc4512#section-2.3.2

https://tools.ietf.org/html/rfc4511#section-4.5.1

点击 **LDAP 设置**版块处的 **应用**按钮，保存 LDAP 设置信息。

LDAP 有两种模式，默认为管理员认证模式，所有必填字段有值就可以进行应用或测试

1）管理员认证模式：此时有 5 个必填字段（页面标 * 的为必填）

2）非管理员认证模式：此时有 2 个必填字段（页面标 * 的为必填）

3.2.7.2 LDAP 服务地址

LDAP 服务器地址

3.2.7.3 Bind DN 连接 LDAP 服务的用户

管理员认证模式专有

绑定 DN 信息：用户名，域名

3.2.7.4 Bind Credentials 连接 LDAP 服务的密码

绑定密码

3.2.7.5 Search Base 搜索登录用户的根路径

LDAP 服务器域名信息

3.2.7.6 Search Filter 匹配登录用户的搜索条件

LDAP 服务器用户名查找定位

集成 Windows AD 的登录时，设置为：(&(objectCategory=Person)(sAMAccountName={{username}}))

3.2.7.7 CA Certificate 验证 LDAP 服务期的证书

CA 认证信息

3.2.7.8 User DN Pattern 连接 LDAP 服务的用户

非管理员认证模式专有

3.2.8 示例

管理员认证模式:

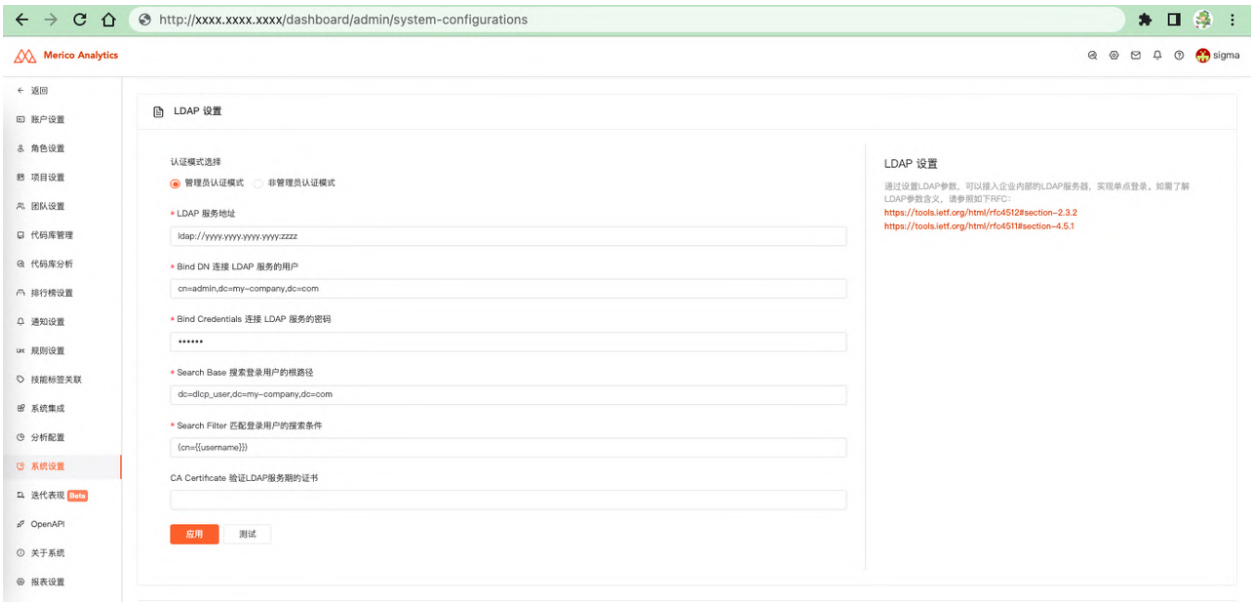

图 24: 管理员认证模式

非管理员认证模式:

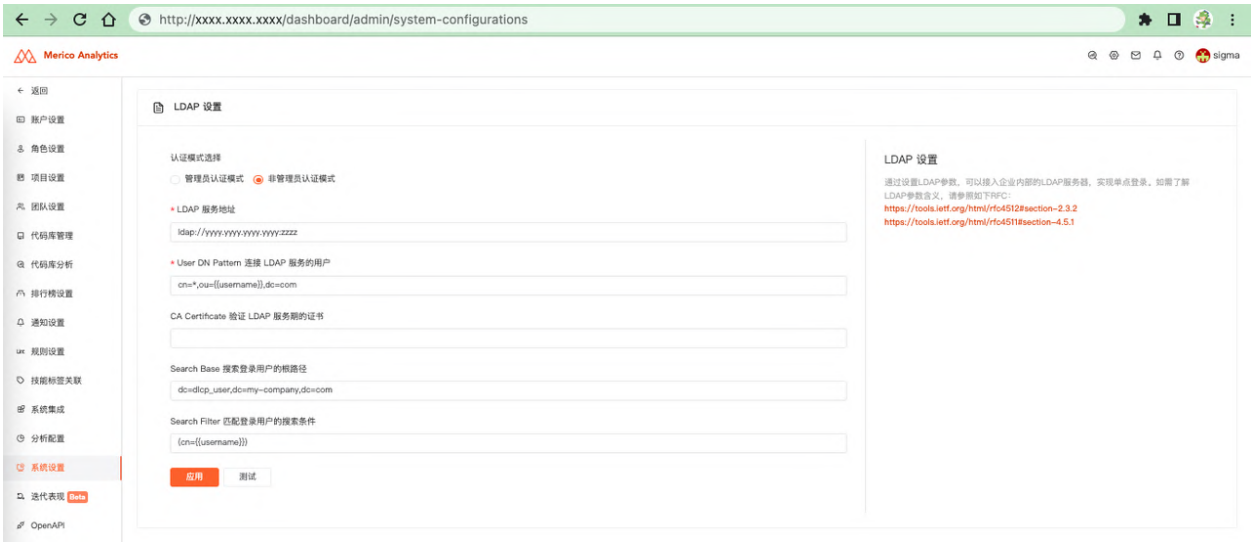

图 25: 非管理员认证模式

3.2.8.1 测试模式

点击 **应用**按钮旁边的 **测试**按钮，可以进入测试模式

图 26: 测试通过示例

图 27: 测试不通过示例

测试条件：用户需要填写所有必填项后才可以进行测试

测试界面：点击测试后弹出测试框。需要用户用户填入真实登录账号进行测试。左侧为输入区，右侧为信息区。

测试结果：登录成功返回测试通过。登录不成功返回测试不通过，同时返回错误信息。

3.2.9 周日报配置

3.2.9.1 概述

可以设置全部代码库定时自动分析时间，分析数据可以以周报或日报的的形式发送，报告类型、接收范围、发送时间都可以灵活配置。

点击 **周日报配置**版块处的 **应用**按钮，保存周日报配置信息。

3.2.9.2 周日报说明

系统支持把分析的周期性结果通过邮件发送，配置接收类型，范围以及接收时间后生效。

3.2.9.3 管理者报告

管理者报告内容主要为研发情况数据汇总和概况总览。为了保证信息安全，报告接收者只会收到其有权限访问的项目数据汇总。

可以通过勾选 **日报**和/或 **周报**，配置 **接收范围**指定某些角色的账户可以收到管理者日报和/或周报。

请注意，选择日报时，分析引擎会每日启动分析，需要确认资源足够支持在当天完成分析。

周/日报的报告都可以 **预览**，点击新打开一个窗口，可以打开报告的设计稿。

3.2.9.4 开发者报告

开发者报告内容主要为程序员个人工作汇总。

可以通过勾选 **日报**和/或 **周报**，配置 **接收范围**指定某些角色的账户可以收到开发者日报和/或周报。

请注意，选择日报时，分析引擎会每日启动分析，需要确认资源足够支持在当天完成分析。

周/日报的报告都可以 **预览**，点击新打开一个窗口，可以打开报告的设计稿。

3.2.9.5 报告发送时间

报告发送时间可以选择 **快速配置**和 **高级配置**两种模式进行配置。

选择 **快速配置**模式时，仅需选择周报和日报发送时间，系统会根据报告发送时间相应调整全部代码库定时启动分析时间。快速配置规则下：

1. 只配置发送时间，其中日报要遵循工作日规则，非工作日不发送。
2. 周报的运行时间为发送时间的前 24 小时，内容为发送时间往前的 7 个工作日。
3. 日报的运行时间为发送时间的前 12 个小时，内容为发送时间往前的 1 个工作日。
4. 发送时间按照系统时间统一发送

选择 **高级配置**模式时，可以使用 **Cron 表达式**精细配置不同报告发送时间和全部代码库定时启动分析时间。

系统维护窗口：系统维护将在指定的时间窗口期每日自动进行，清理过期报告和已删除的代码库数据。维护完成后将显著缩短数据加载时间。整个维护过程大约持续 2 小时，在维护窗口期内系统响应速度可能会有所下降，请根据实际情况设置合理的系统维护时间。

重新计算时间：当调整了问题忽略规则设置后，需要重新统计相关质量问题数据。此处设置用于设置在每天系统维护时（上一条系统维护窗口设置时间），花多长时间重新生成质量数据统计。为 0 时关闭本功能，为 5 小时时，每天花 5 小时重新生成质量数据统计。

3.2.9.6 配置说明

1. 在【**快速配置**】模式下，可以配置每周周报发送时间和日报发送时间。
2. 如配置了周报发送时间，会在此时间前 24 小时触发全部代码库重新分析；
3. 如配置了日报发送时间，会在此时间前 12 小时触发全部代码库重新分析。
4. 周报内容为发送时间前 7 天（不包含发送当天）内的代码分析结果；
5. 日报内容为发送时间前一个工作日内的代码分析结果。
6. 如果周/日报发送时还有代码库分析未完成，会造成周日报数据不准确。
7. 如需自定义代码库定时启动分析时间和周/日报发送时间，请切换至【高级配置】模式进行配置。

3.2.9.7 启用在职人员白名单

勾选白名单选项，报告数据中仅包含在职人员，不包含已离职人员数据。

3.2.10 示例

3.2.10.1 快速配置

下图示例中，周报每周日 0 点发送

下图示例中，日报每个工作日 12 点发送

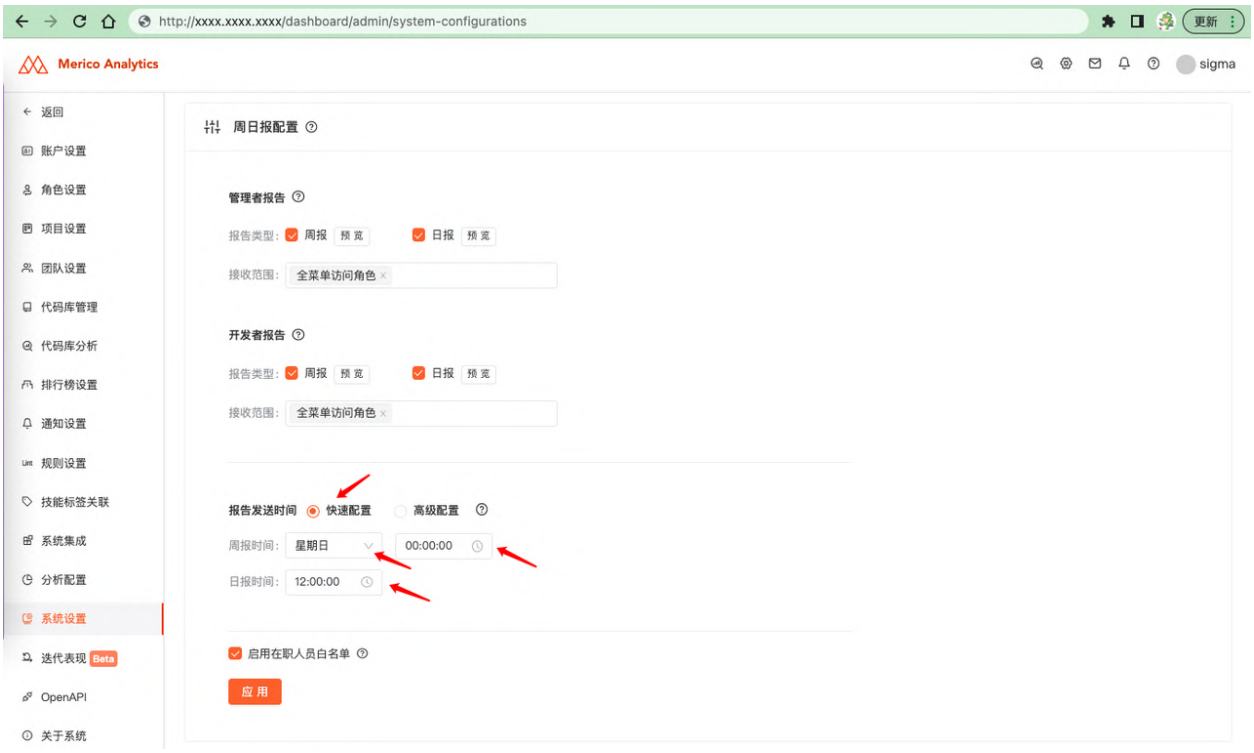

图 28: 快速配置示例

3.2.10.2 高级配置

下图示例中，周报在每个周日 0 点发送

周报涉及代码库定时分析，每个周六 0 点启动

下图示例中，日报每个工作日（周一到周五）12 点发送

日报涉及代码库定时分析

周一 0 点启动，周一 12 点发送周五的工作日报告

周二 0 点启动，周二 12 点发送周一的工作日报告

周三 0 点启动，周三 12 点发送周二的工作日报告

周四 0 点启动，周四 12 点发送周三的工作日报告

周五 0 点启动，周五 12 点发送周四的工作日报告

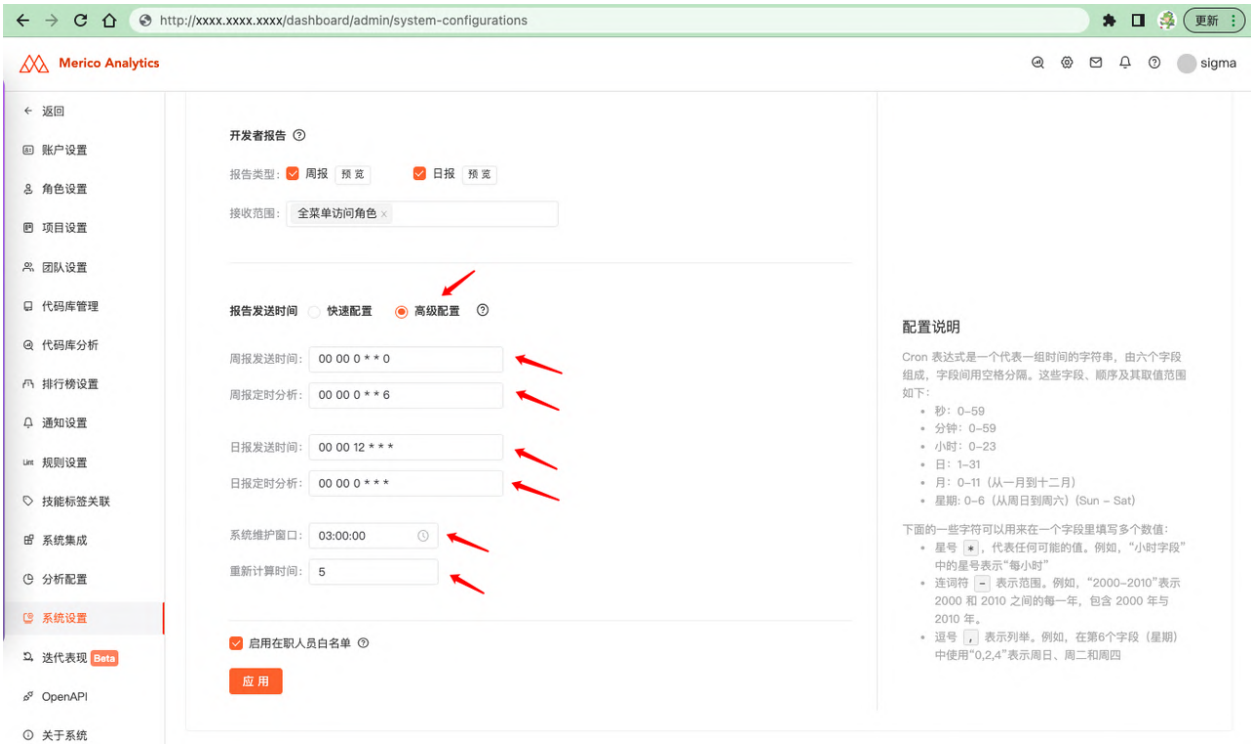

图 29: 高级配置示例

3.2.11 专家系统

3.2.11.1 概述

勾选是否打开专家系统功能，打开此功能开关后，在 **项目表现**的 **效率报表、质量报表、开发负载**等模块会对数据表现给出分析和建议。

3.2.11.2 示例

3.2.11.2.1 设置示例

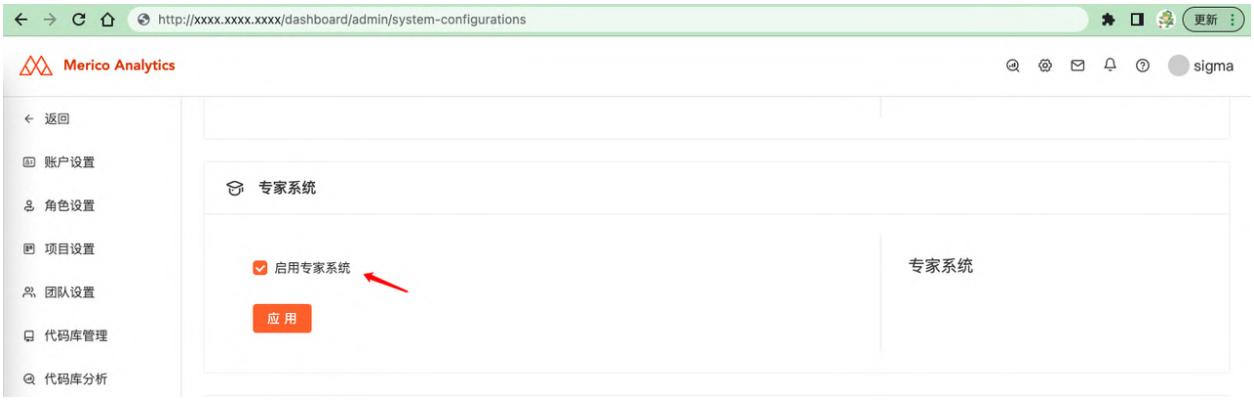

图 30: 专家系统设置示例

3.2.11.2.2 专家系统应用

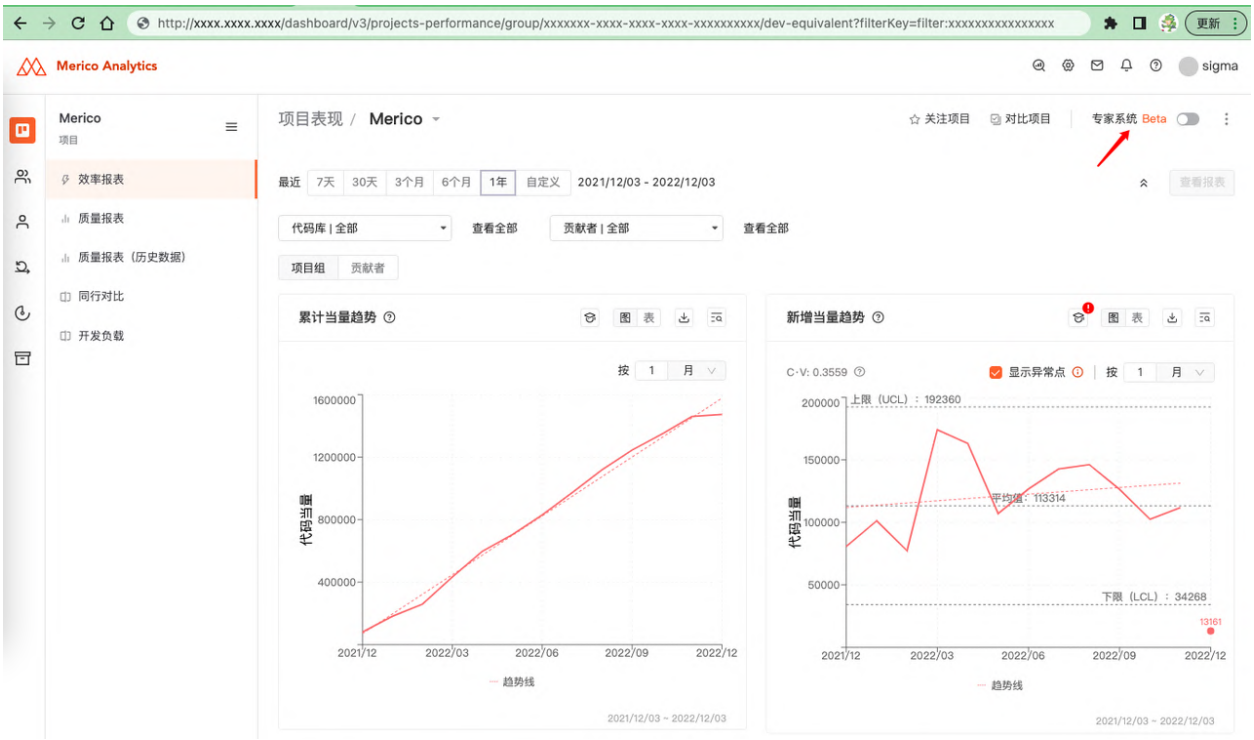

图 31: 专家系统应用示例

3.2.12 设置提交类别

3.2.12.1 概述

在代码提交规范化的前提下，设置提交类别，可查看不同提交类别对应的当量数据。

设置导航：**设置-系统设置-设置提交类别**

3.2.12.2 设置示例

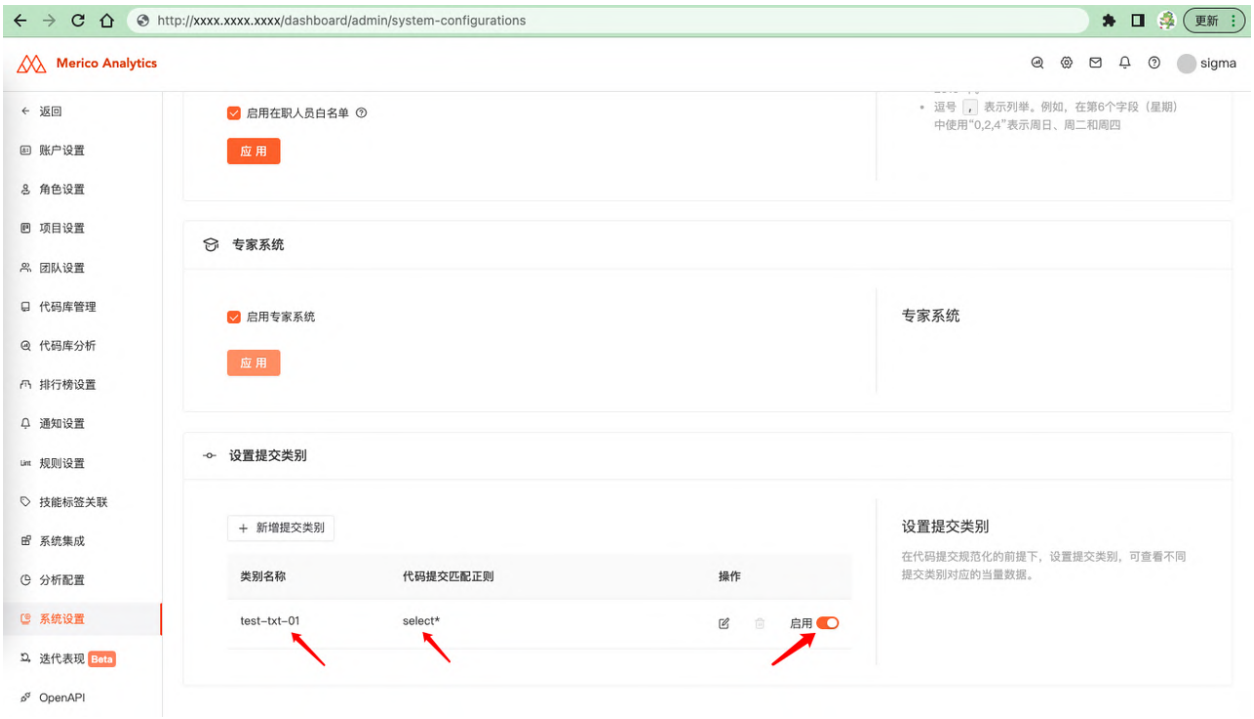

图 32: 设置提交类别示例

3.2.12.3 使用示例

在 **项目表现-> 效率报表**界面，可以点击选择-> **按提交类别查看**

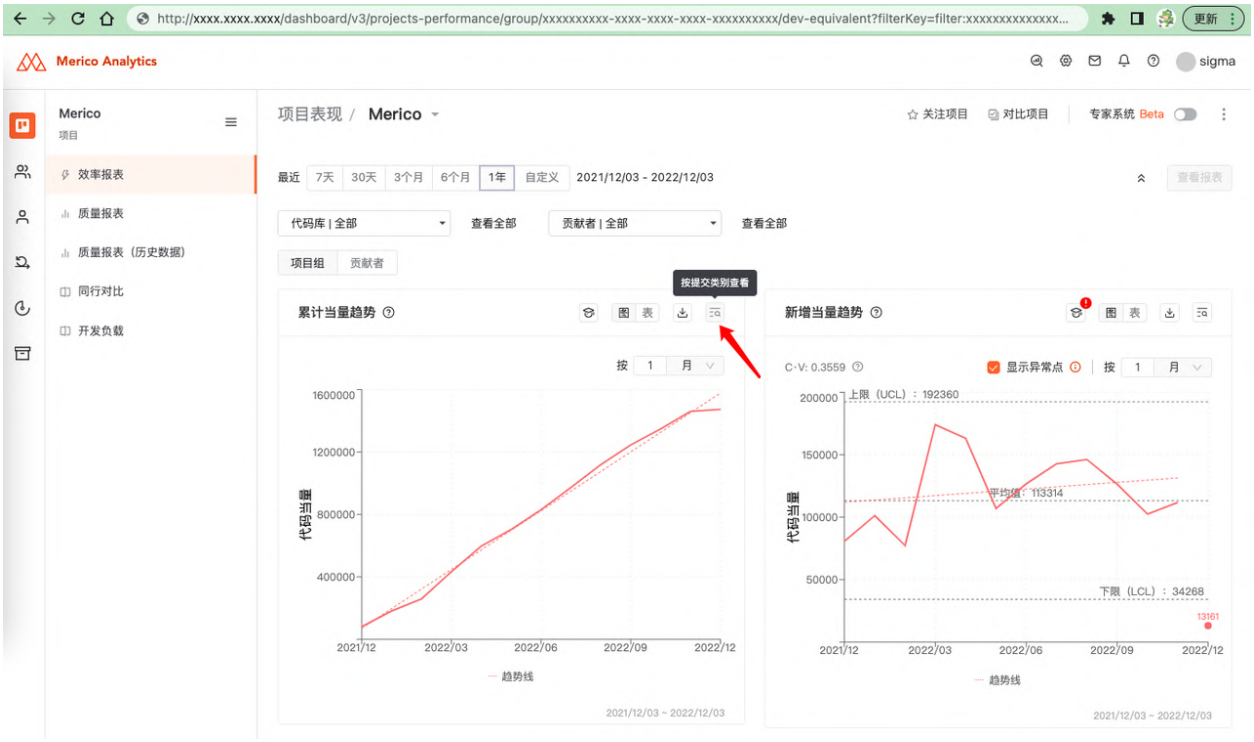

图 33: 提交类别使用示例

按提交类别查看到的页面及数据

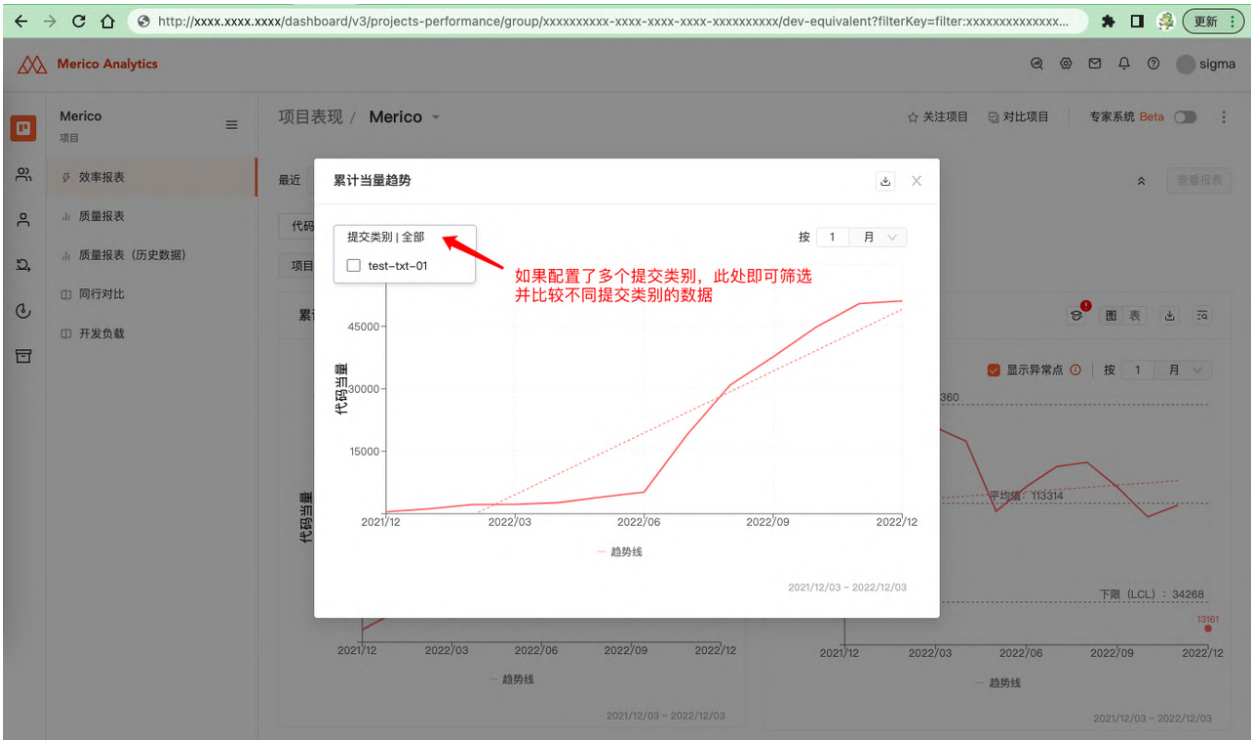

图 34: 按提交类别查看到的页面及数据

3.2.13 注释覆盖度计算规则

3.2.13.1 概述

计算注释覆盖度时，是否需要计算匿名函数

计算注释覆盖度时，包含匿名函数

计算注释覆盖度时，排除匿名函数

设置导航：**设置-系统设置-注释覆盖度计算规则**

3.2.13.2 设置示例

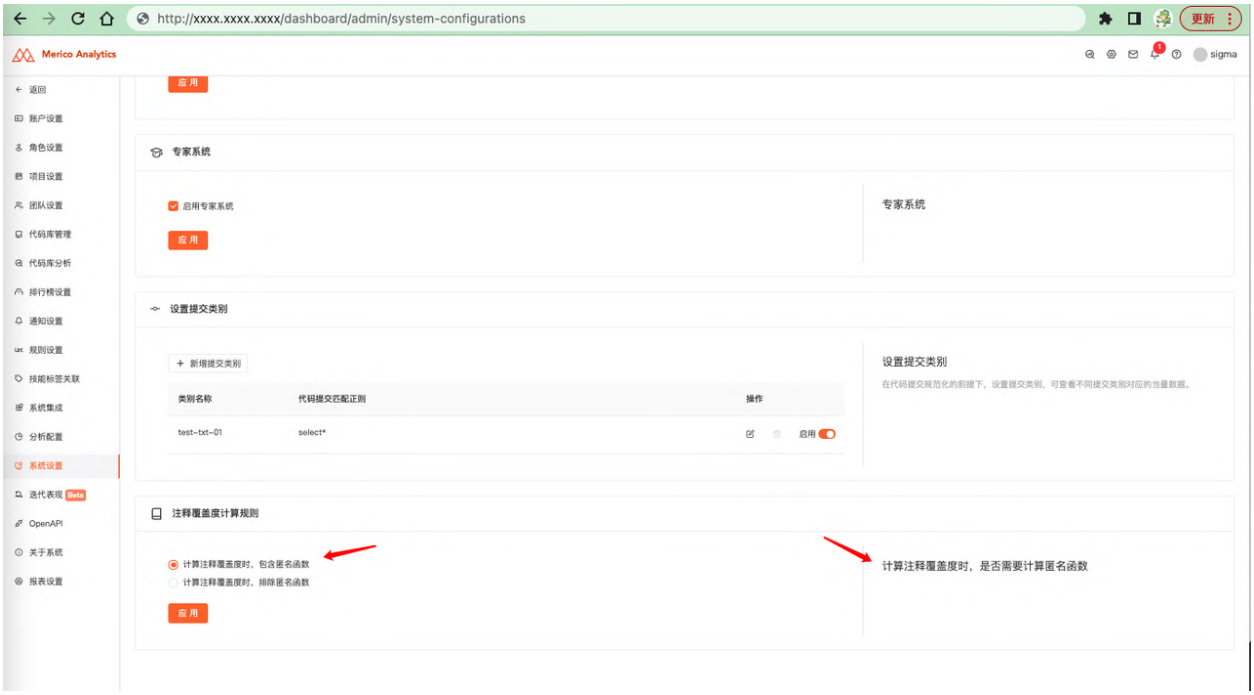

图 35: 注释覆盖度计算规则设置示例

3.3 第二步：导入代码库

3.3.1 导入或新增代码库

3.3.1.1 导入/新增代码库的四种方式

在思码逸系统中，您可以通过如下四种方式导入或新增代码库：

- 方式一：使用 Git 地址新增代码库
- 方式二：从 GitLab 导入代码库
- 方式三：表格文件导入代码库
- 方式四：GitLab 系统集成

3.3.1.2 需满足条件

导入/新增代码库方式	需满足条件
使用 Git 地址新增代码库	- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限 - 操作账户需要有思码逸系统中对应项目组的访问权限

导入/新增代码库方式	需满足条件
从 GitLab 导入代码库	- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限 - 操作账户需要有思码逸系统中对应项目组的访问权限 - 需要 **GitLab 维护者角色**的账户 - 思码逸系统 **系统集成参数**已设置完成
表格文件导入代码库	- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限 - 操作账户需要有思码逸系统中对应项目组的访问权限
GitLab 系统集成	- 操作账户需要思码逸系统 **设置**里的 **系统集成**界面的访问权限 - 需要 **GitLab 管理员**的账户 - 导入 GitLab 账户将附带导入 GitLab 的项目和群组

3.3.2 从 Git 地址新增代码库

3.3.2.1 要点概述

- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限。
- 操作账户需要有思码逸系统中对应项目组的访问权限。
- 此种代码库导入方式支持一次性导入最多 20 个代码库。

3.3.2.2 导航入口

3.3.2.2.1 设置 - 代码库设置 - 新增代码库

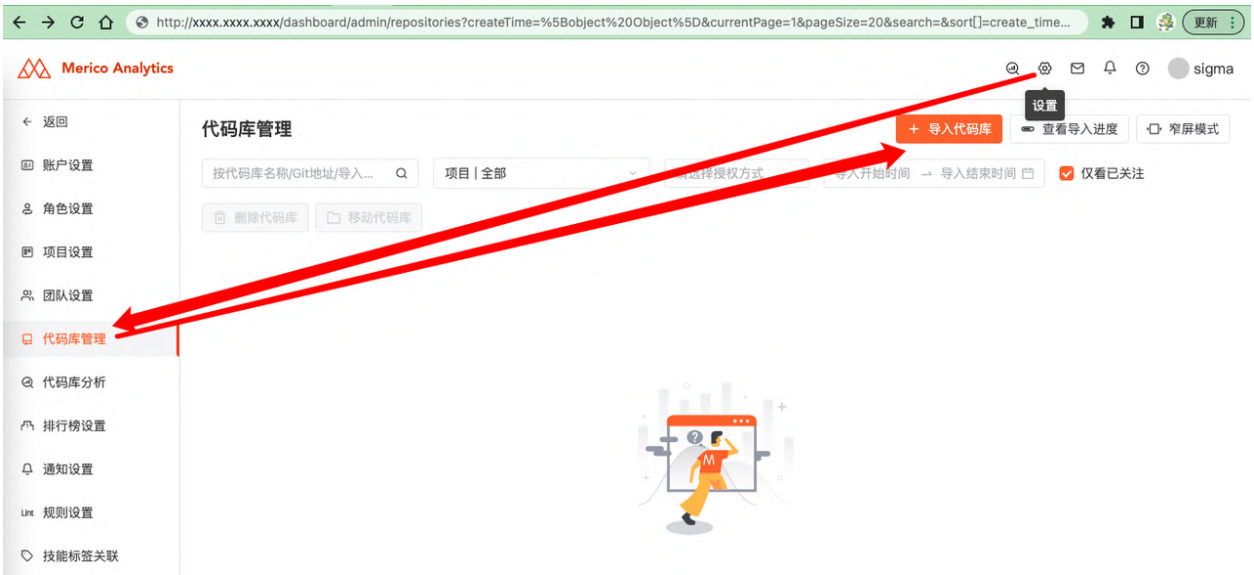

图 36: 设置 - 代码库设置 - 新增代码库

3.3.2.2.2 代码库分析 - 代码库管理 - 导入代码库

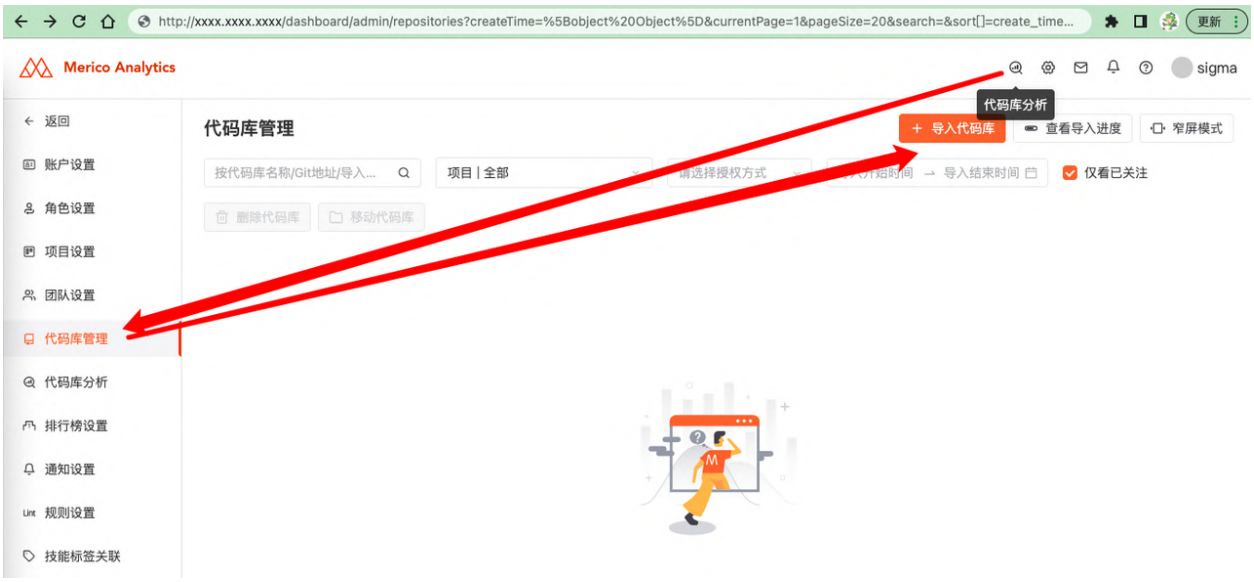

图 37: 代码库分析 - 代码库管理 - 导入代码库

3.3.2.3 设置步骤

3.3.2.3.1 点击导入代码库

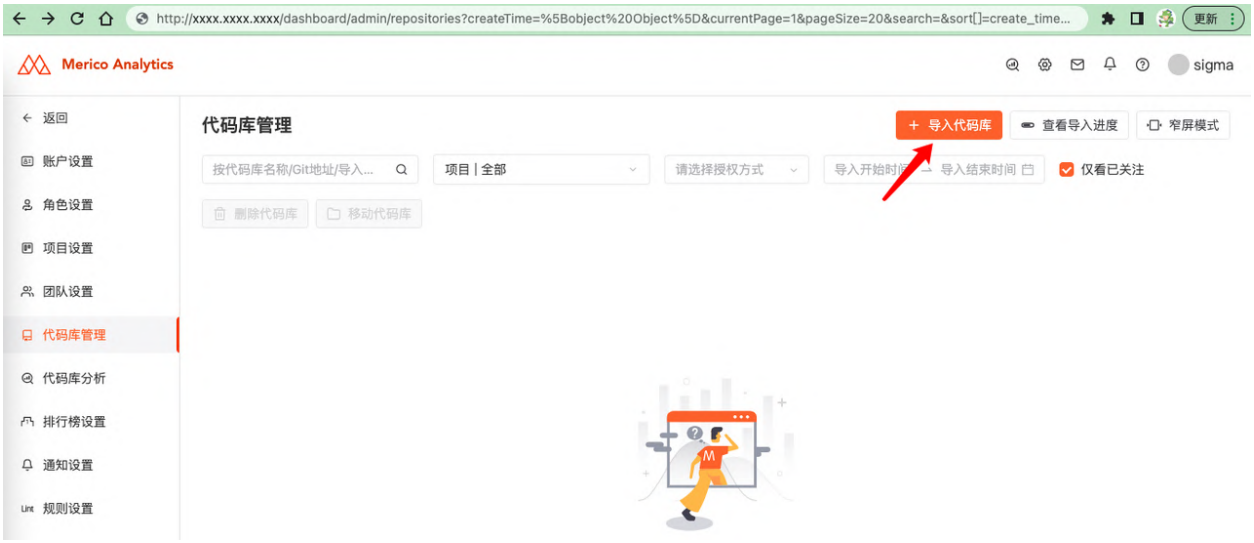

图 38: 点击导入代码库

3.3.2.3.2 点击 Git 代码库导入

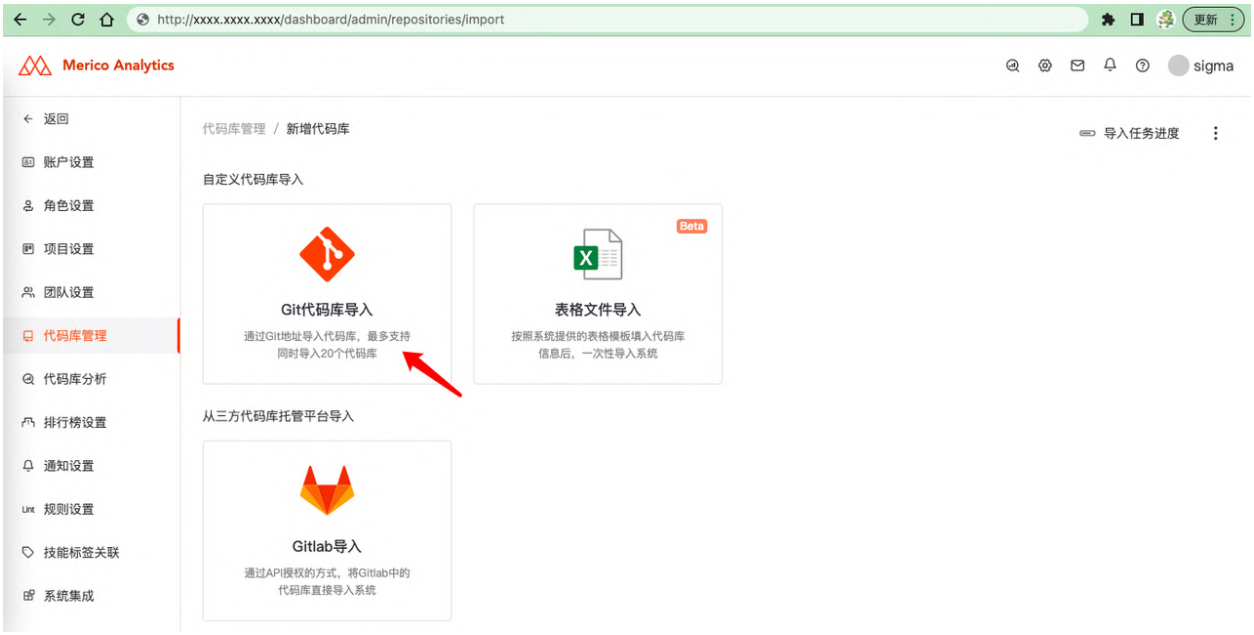

图 39: 点击 Git 代码库导入

3.3.2.3.3 填入 Git 信息

填入 Git 地址，可以同时填入最多 20 条使用相同授权方式的 Git 地址进行批量新增，自动匹配授权方式：

- HTTP/HTTPS 协议只支持直接访问或通过账号/密码访问
- Git/SSH 协议只支持通过公钥访问

点击下一步

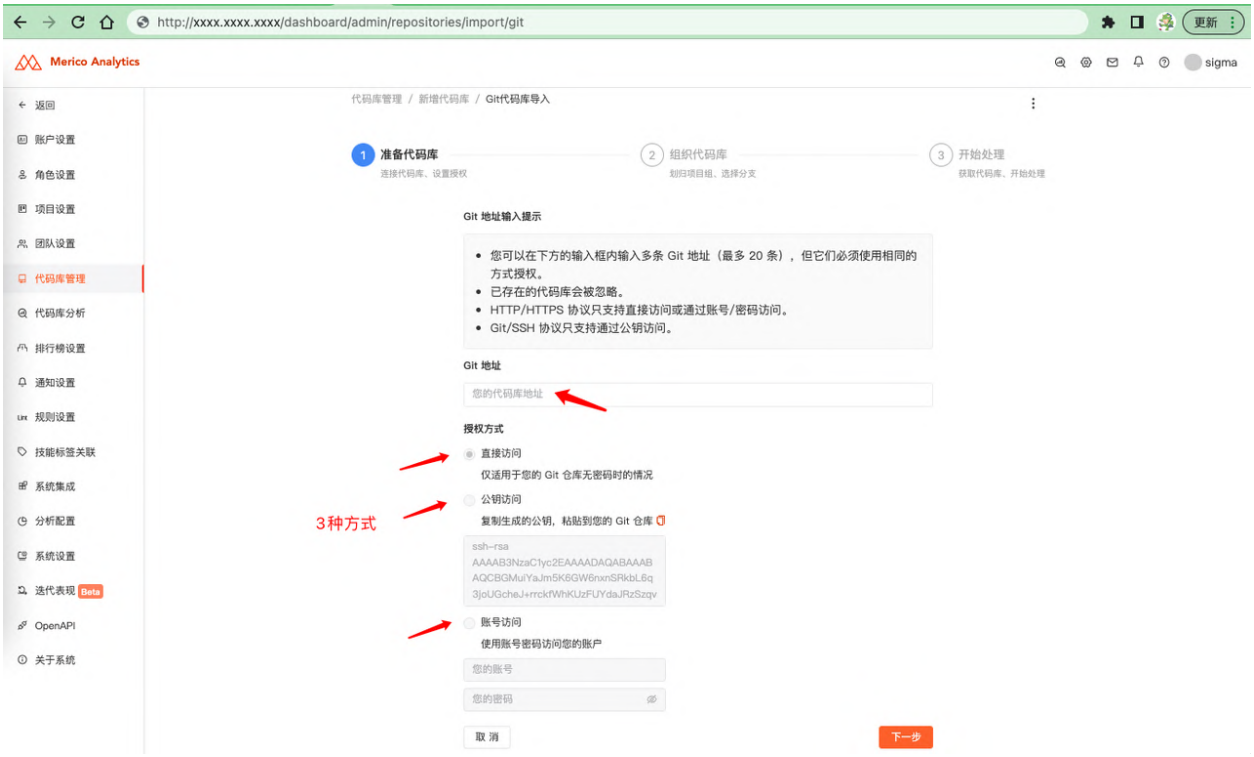

图 40: 准备代码库导入信息

3.3.2.3.4 通过账号密码方式导入示例

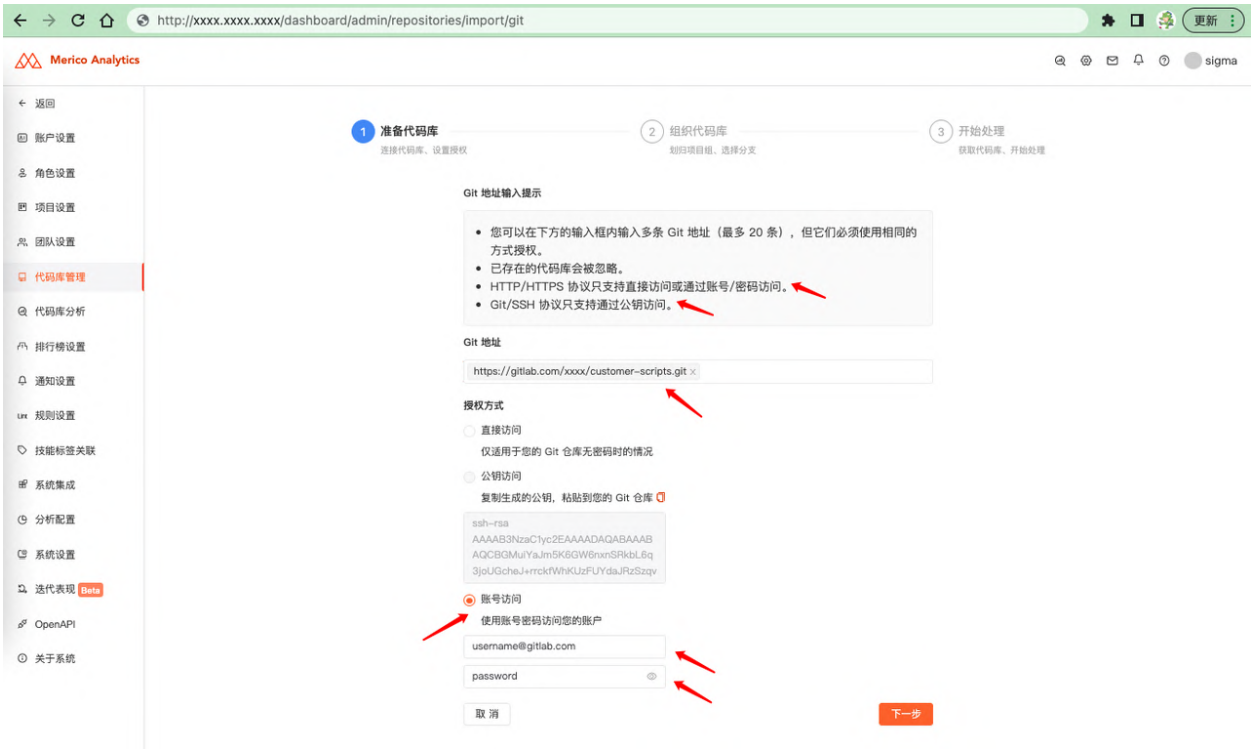

图 41: 通过账号密码方式导入示例

3.3.2.3.5 账号密码方式导入注意

如果您的 Gitlab 账户登录开启了双因子认证，那么通过账号密码方式导入代码库将失败

可能的报错：

分支获取失败: [ERROR_REPO_AUTH_FAIL: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password.

解决方法两种：

1. 关闭双因子认证后，重新添加代码库
2. 改用公钥访问方式，重新添加代码库

3.3.2.3.6 通过公钥访问方式导入示例

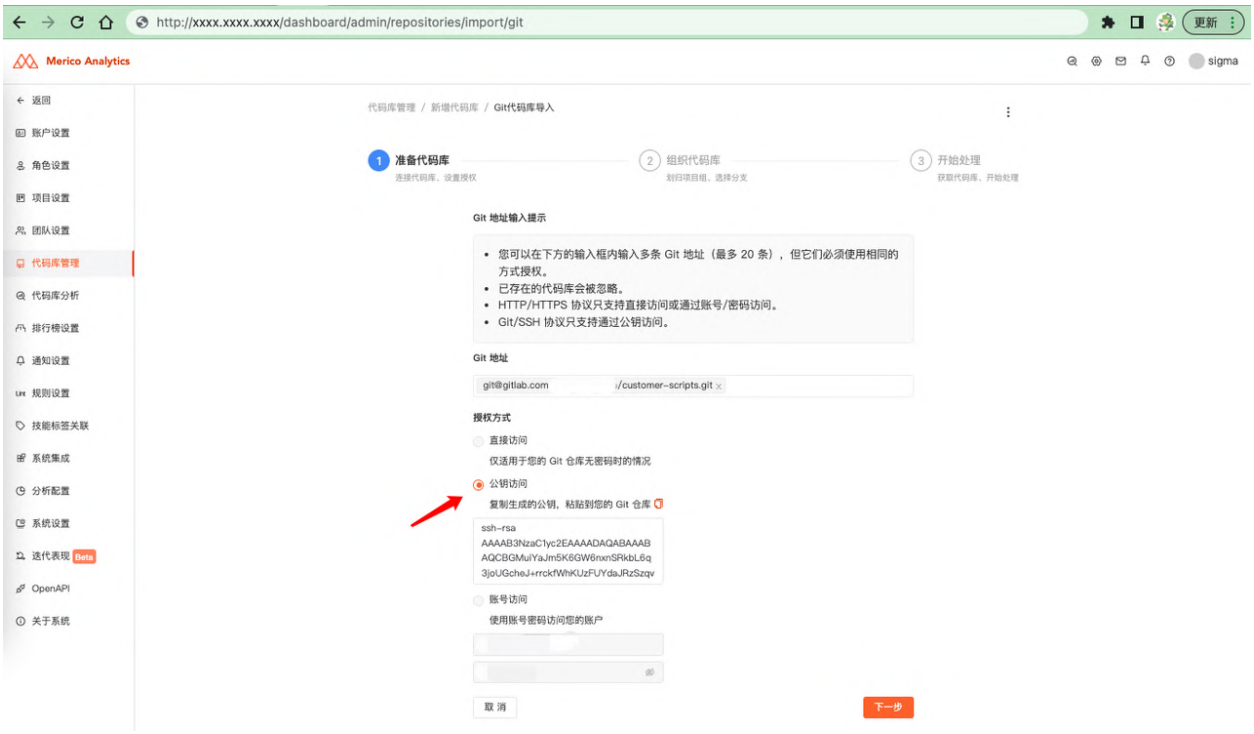

图 42: 通过公钥访问方式导入示例

3.3.2.3.7 组织代码库

选择项目组 (此处也可新建项目组)、分析分支 (用于质量分析)，然后点击 **开始分析**

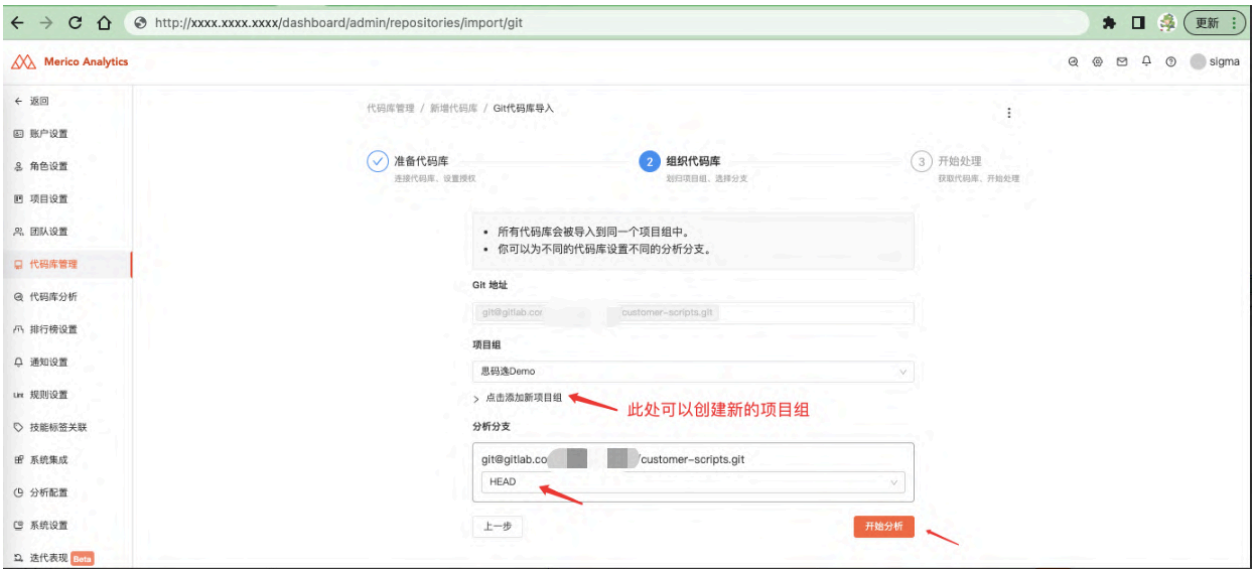

图 43: 组织代码库示例

3.3.2.3.8 成功获取代码库并开始处理

出现成功提示，点击 **完成**按钮完成代码库新增

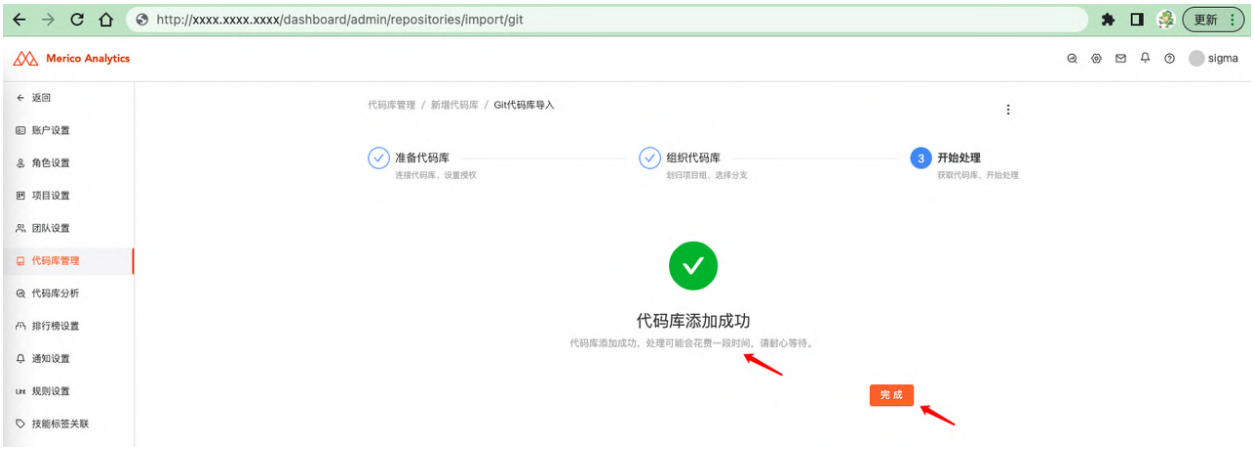

图 44: 代码库添加成功示例

3.3.2.3.9 支持中途退出

如果在通过 Git 地址新增代码库时，发现问题，需要退出导入过程点击页面右上角的 ⋮，可以退出此次导入过程

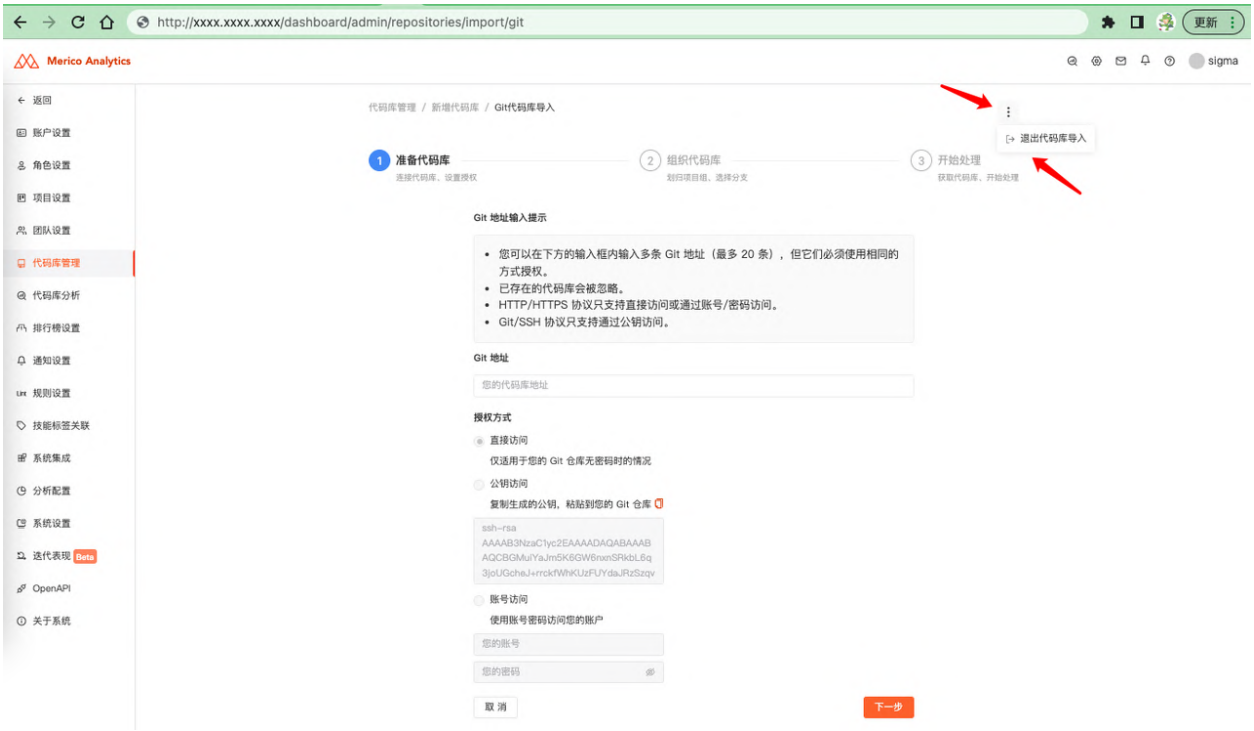

图 45: 支持中途退出

3.3.3 从 GitLab 导入代码库

3.3.3.1 要点概述

- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限。
- 操作账户需要有思码逸系统中对应项目组的访问权限。
- 需要 GitLab **维护者角色**的账户。
- 思码逸系统 **系统集成参数**已设置完成。

3.3.3.2 导航入口

3.3.3.2.1 设置 - 代码库设置 - 新增代码库

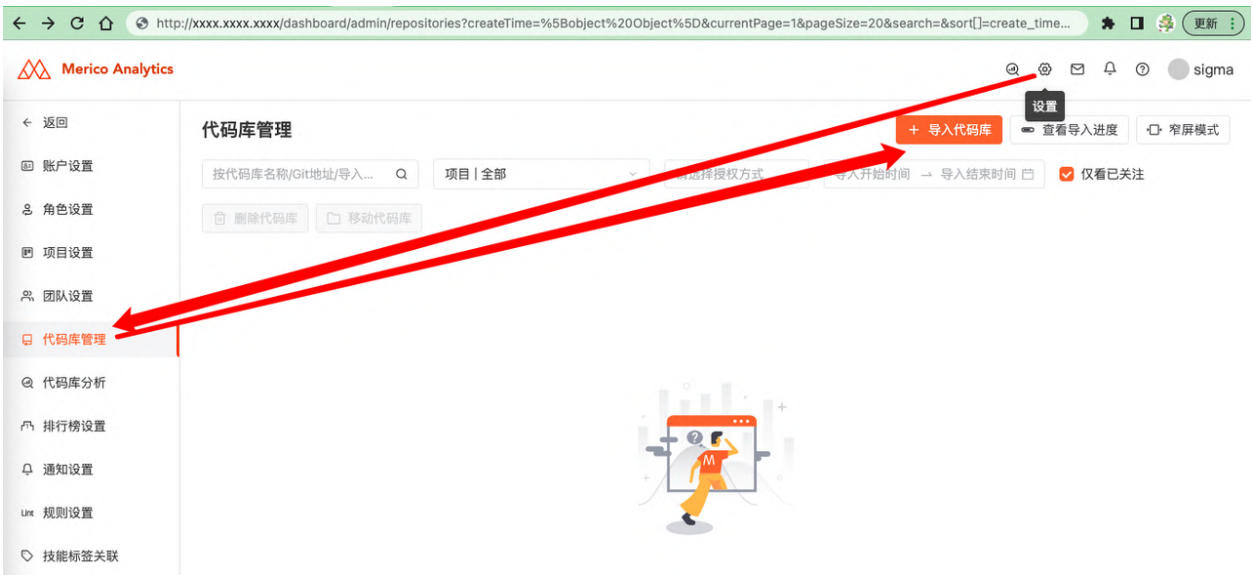

图 46: 设置 - 代码库设置 - 新增代码库

3.3.3.2.2 代码库分析 - 代码库管理 - 导入代码库

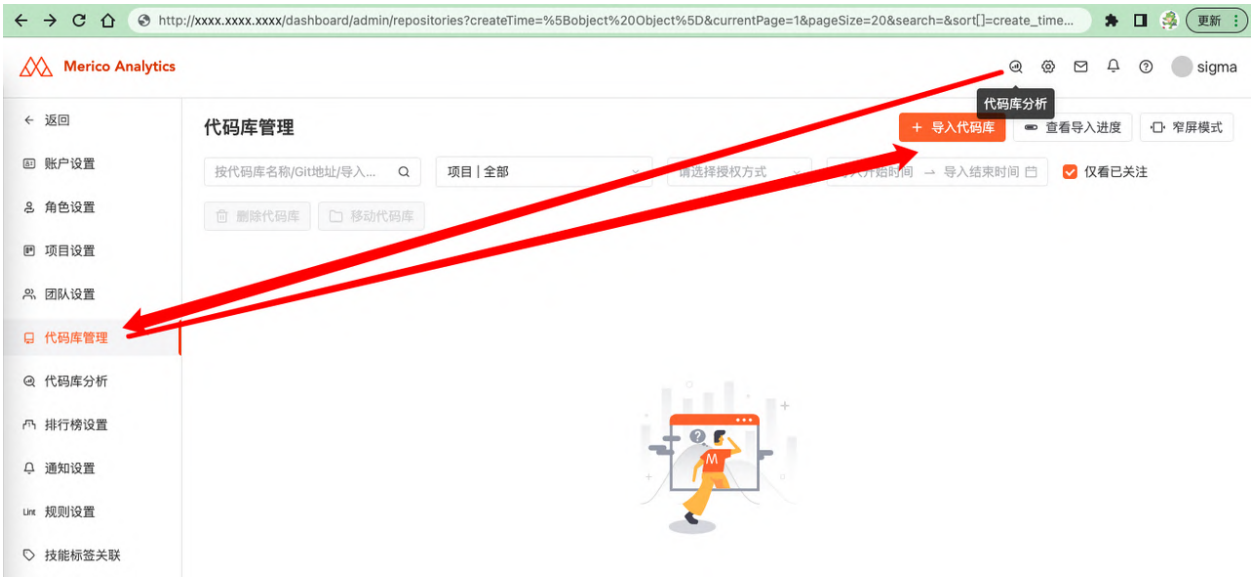

图 47: 代码库分析 - 代码库管理 - 导入代码库

3.3.3.2.3 代码库设置 - 新增代码库

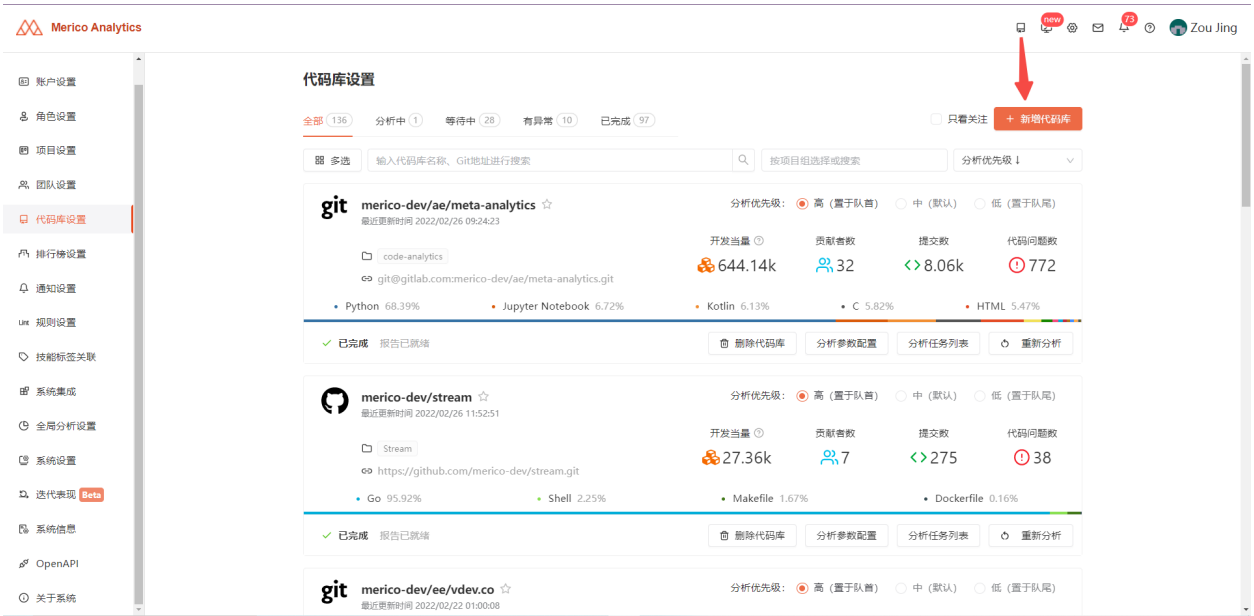

图 48: 代码库设置 - 新增代码库

3.3.3.3 设置步骤

3.3.3.3.1 点击导入代码库

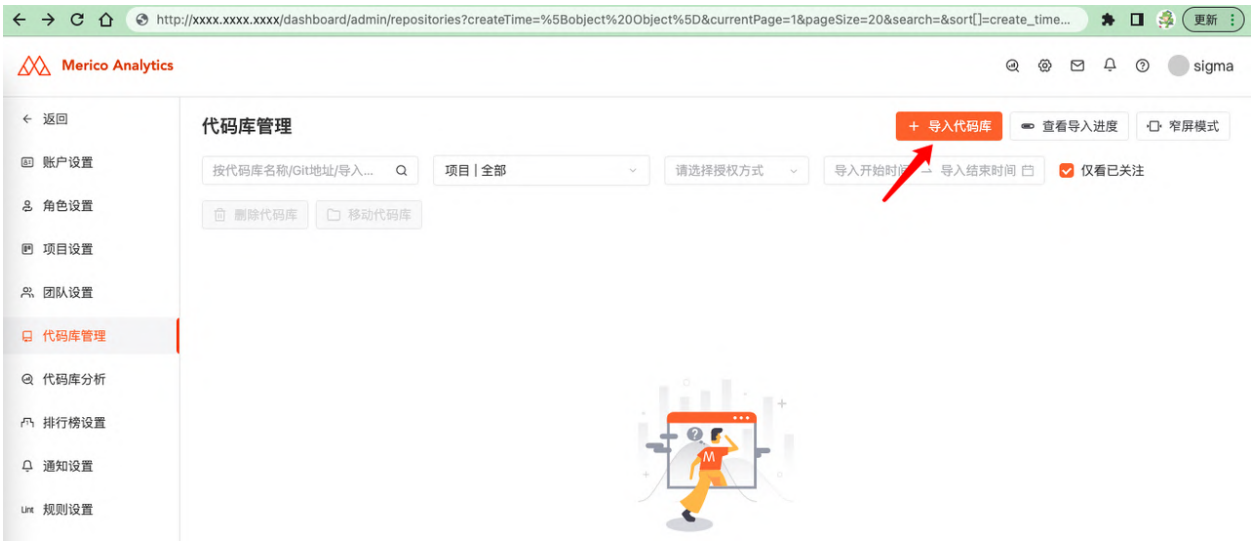

图 49: 点击导入代码库

3.3.3.3.2 点击 Gitlab 导入

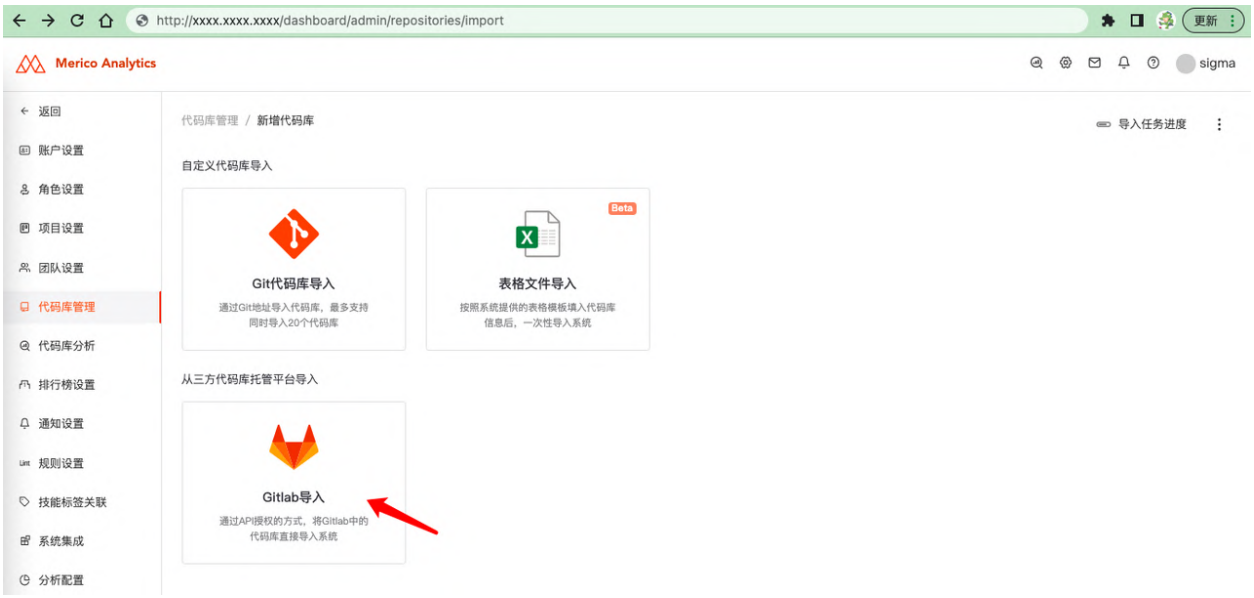

图 50: 点击 Gitlab 导入

3.3.3.3.3 点击下一步

图 51: 获取授权示例

3.3.3.3.4 弹出 GitLab 登录窗口

在此页面，输入用户名、密码，点击 **Sign In** 按钮登录，使用 GitLab 账户 **登录**和 **授权**。

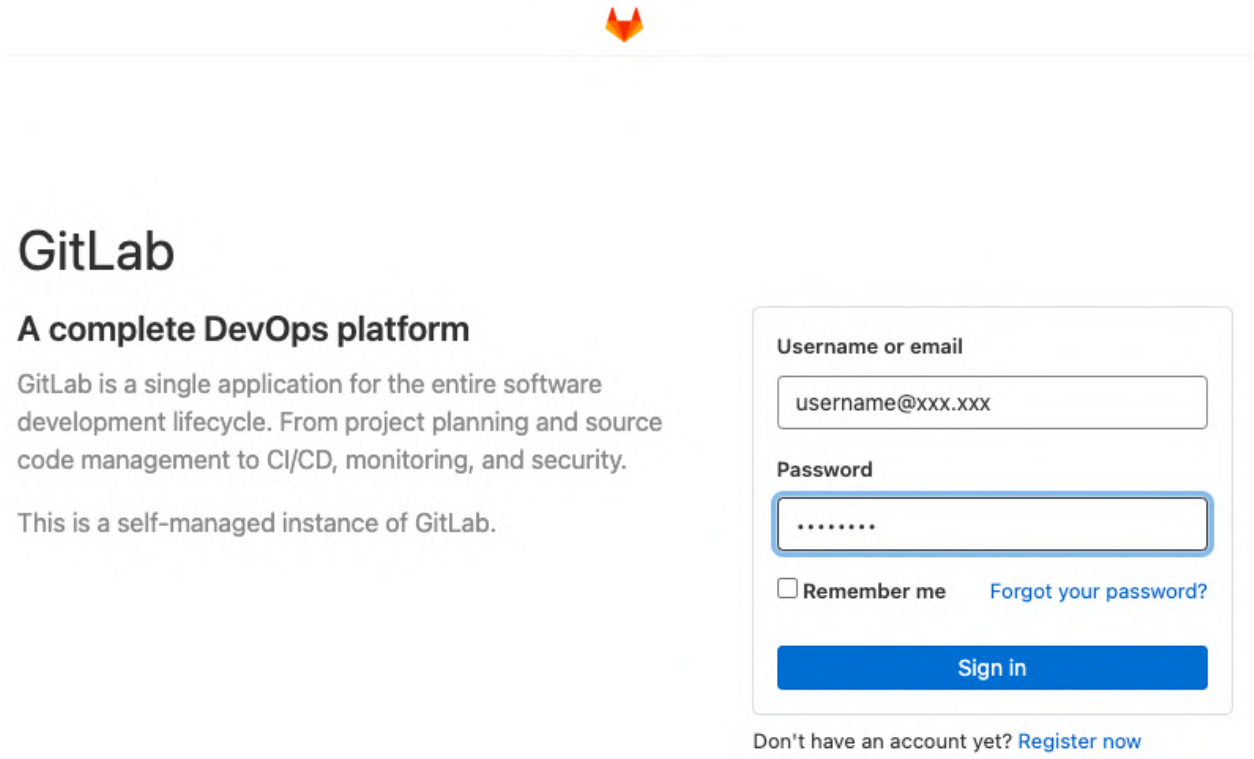

图 52: GitLab 登录窗口示例

3.3.3.3.5 选择需要导入的代码库

选择需要导入的代码库，再点击下一步

3.3.3.3.6 导入成功

导入成功，点击 **完成**按钮结束

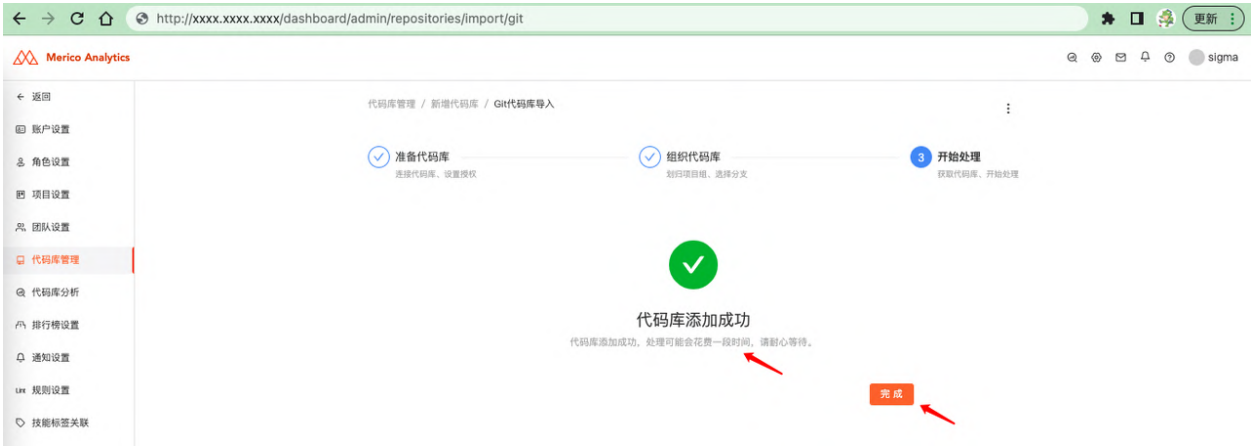

图 53: 代码库添加成功示例

3.3.3.3.7 支持中途退出

如果在通过 Gitlab 导入代码库时，发现问题，需要退出导入过程

点击页面右上角的 ⋮，可以退出此次导入过程

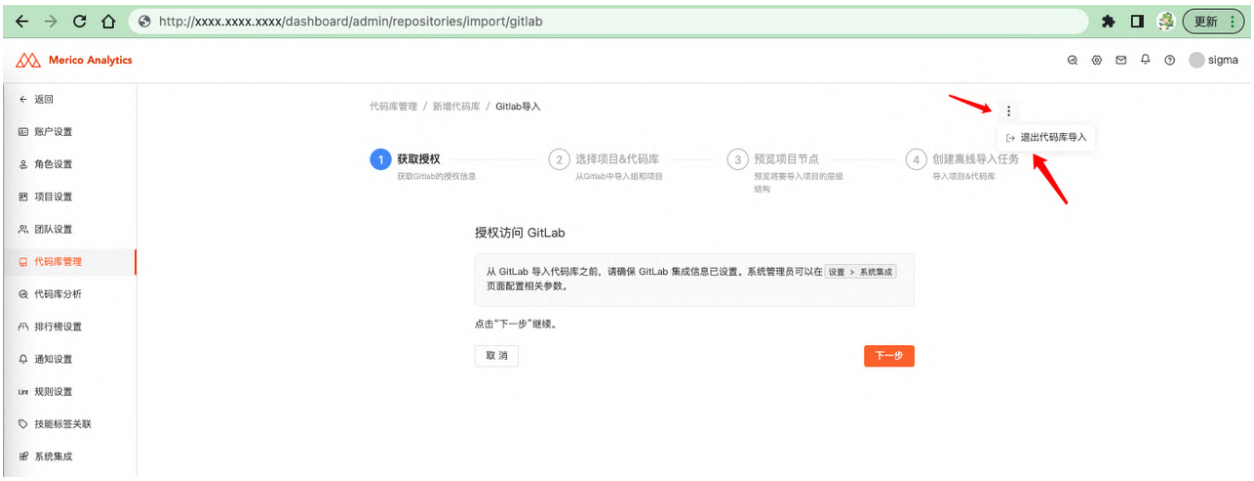

图 54: 支持中途退出

3.3.4 表格文件导入代码库

3.3.4.1 要点概述

- 操作账户需要思码逸系统 **设置**里的 **代码库设置**界面的访问权限。
- 操作账户需要有思码逸系统中对应项目组的访问权限。
- 支持上传文件限制：最大 20MB 或最多 2000 条数据。

3.3.4.2 导航入口

3.3.4.2.1 设置 - 代码库设置 - 新增代码库

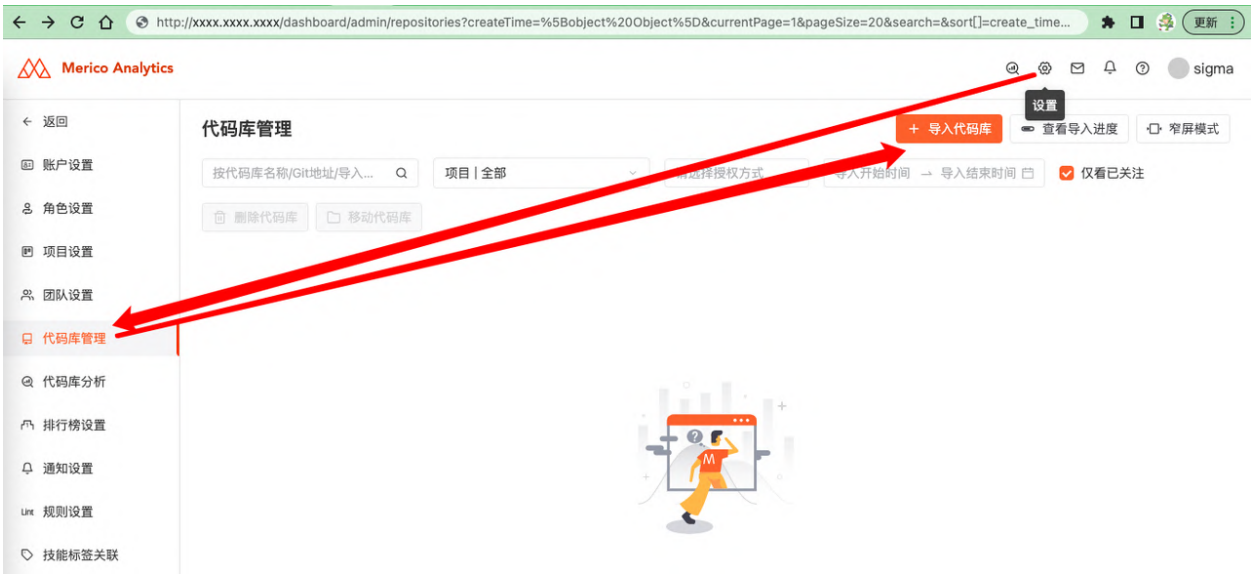

图 55: 设置 - 代码库设置 - 新增代码库

3.3.4.2.2 代码库分析 - 代码库管理 - 导入代码库

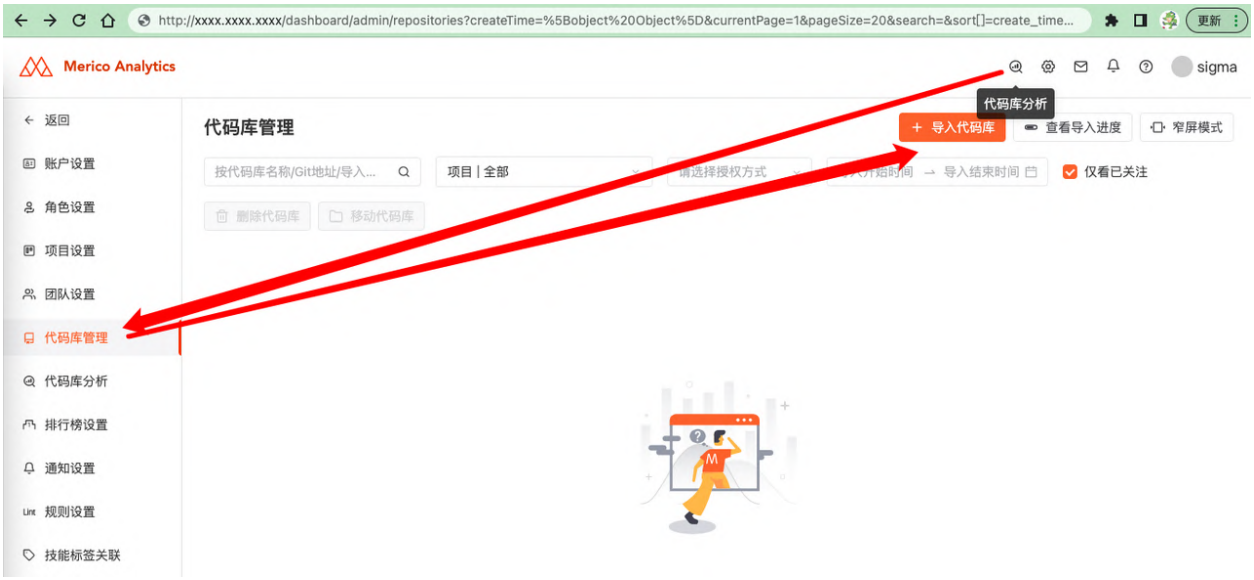

图 56: 代码库分析 - 代码库管理 - 导入代码库

3.3.4.3 设置步骤

3.3.4.3.1 点击导入代码库

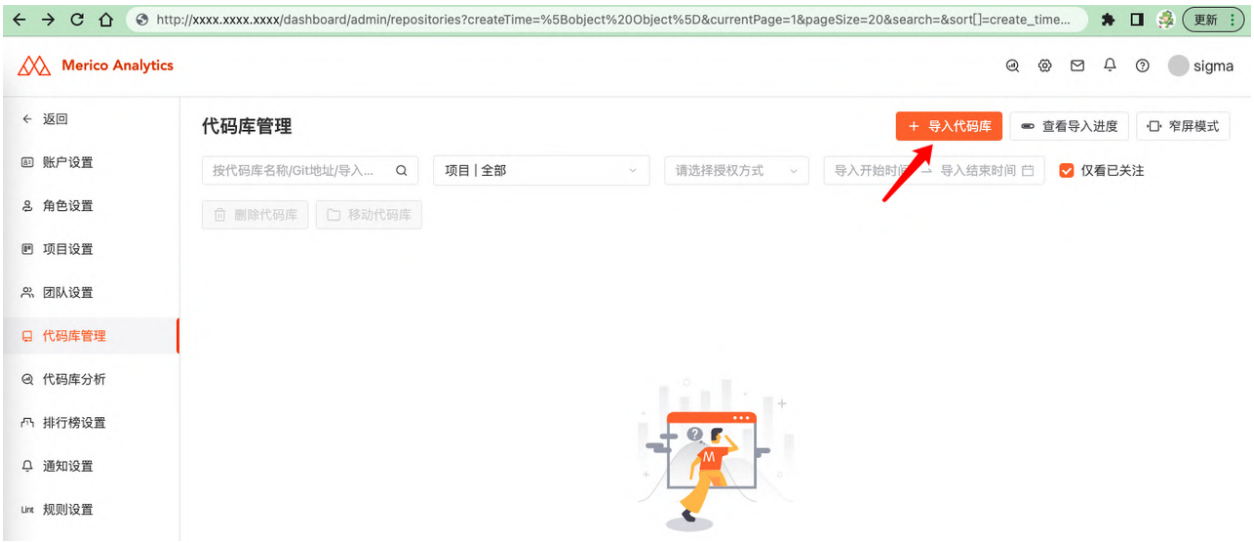

图 57: 点击导入代码库

3.3.4.3.2 进入模版导入

点击 **表格文件导入**

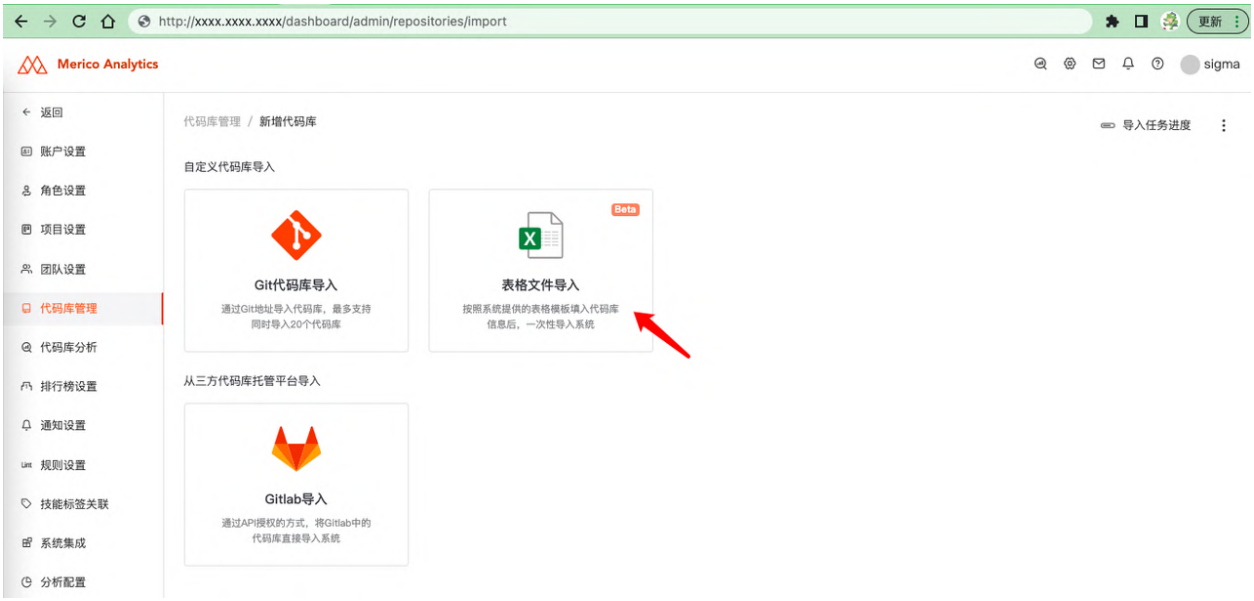

图 58: 点击表格文件导入

3.3.4.3.3 下载模板

点击 **下载表格模版**，可以将模版文件下载到本地

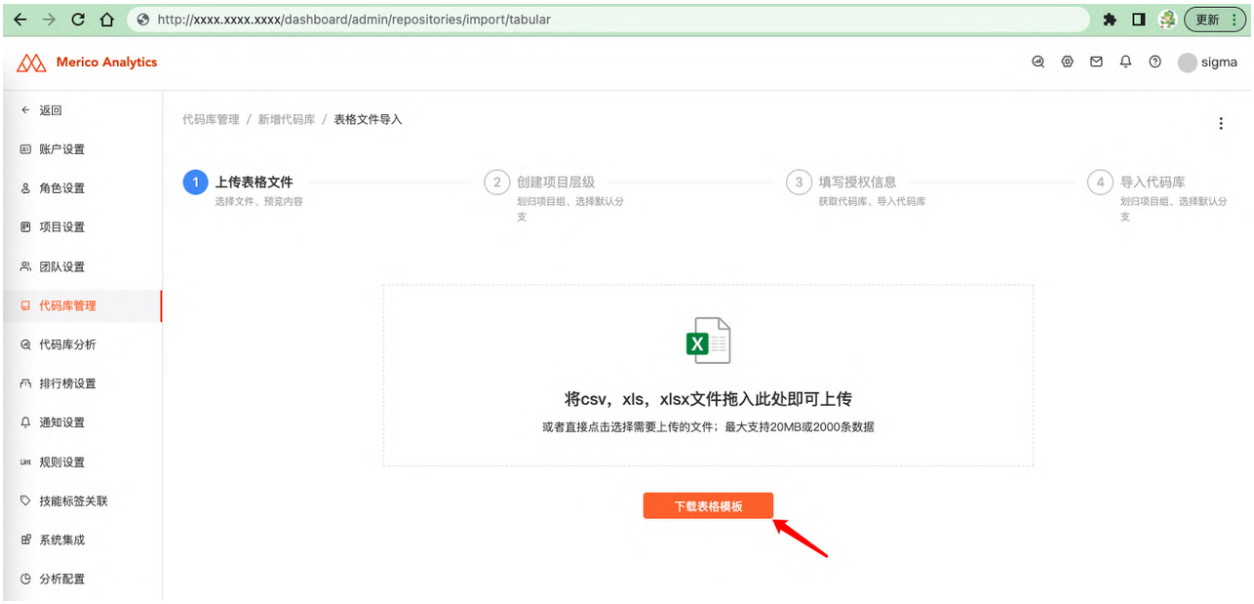

图 59: 下载表格模版

3.3.4.3.4 填写表格

使用 Excel 或 WPS 打开模板文件，填写信息完毕后保存

模板文件填写时有几点注意

1. Git 仓库名称可以直接复用 Git 仓库的名字
2. Git 仓库地址采用 https 的格式：

- 必须要以 “https://” 开头
- 必须要以 “.git” 结尾

3. 项目层级说明

- 根据具体的项目层级自上向下填写；其中「项目层级 1」为最顶层项目，「项目层级 6」为最底层项目。其中，顶级项目在同一文件中都应该是一致的。如果不同，应新建一个文件记录
- 代码仓库所属的项目层级，与实际层级一致即可，不需要保证全部六个层级都填入内容（注意：只能省略后面的层级，前面的层级是必填的，例如如果填写了「项目层级 3」，则必须填写「项目层级 1」和「项目层级 2」）
- 如果项目名称包含如下特殊字符（空格、英文半角逗号、英文半角正斜杠、英文半角反斜杠），请使用 “-” （短横杠）代替。

4. 表格填写示例

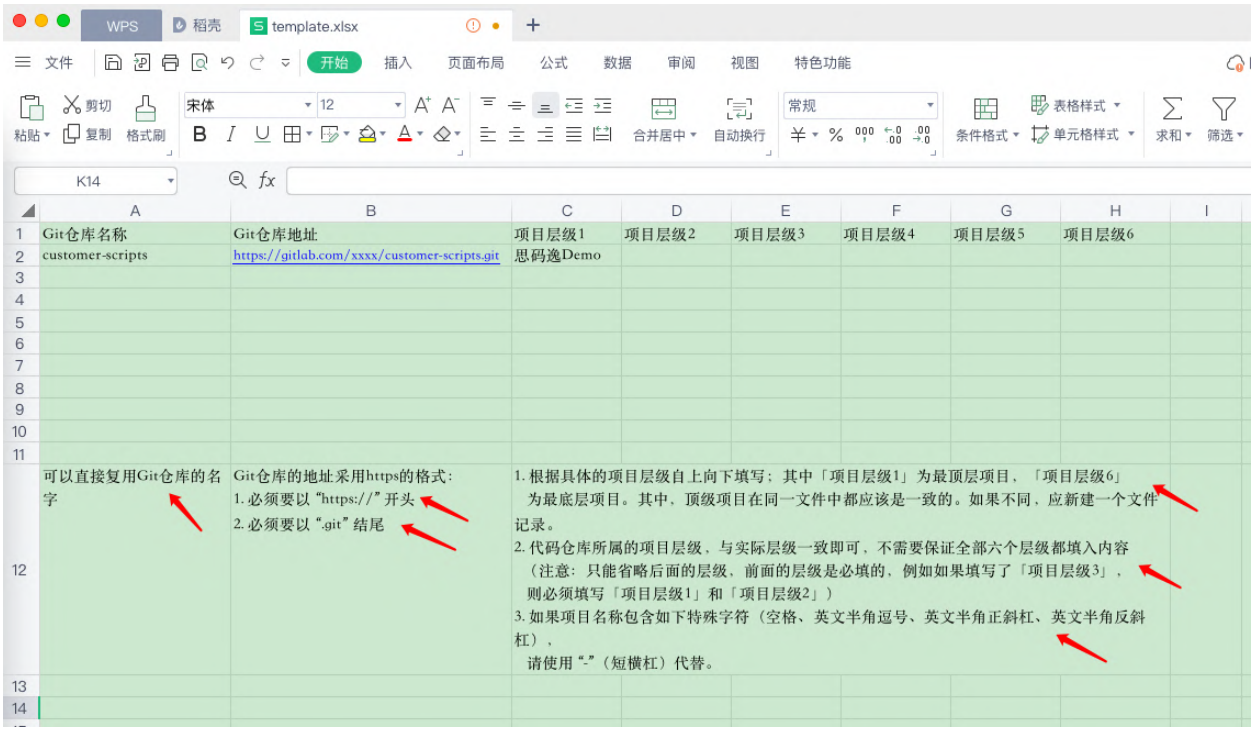

图 60: 表格填写示例

3.3.4.3.5 上传表格

点击 **文件上传**区域，上传填写好的表格

数据解析中...

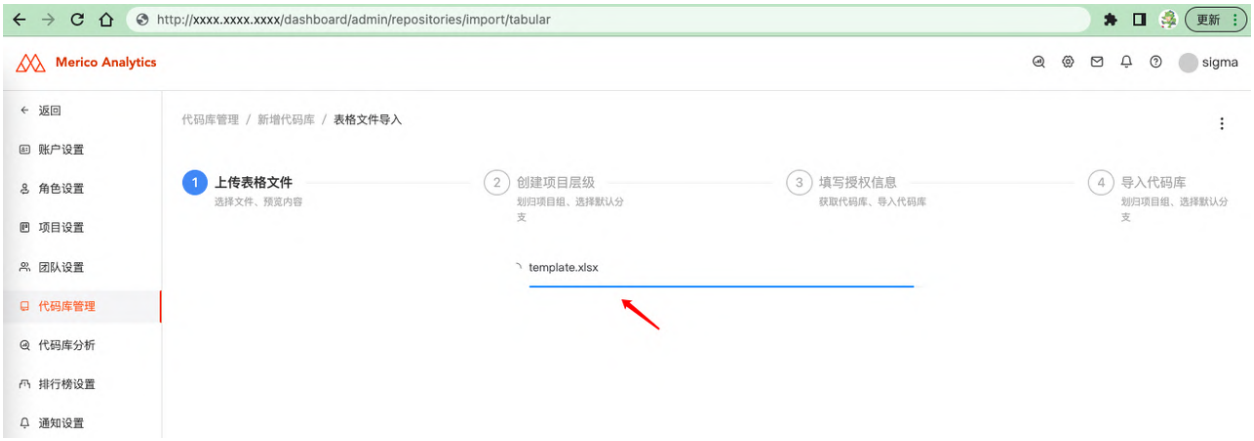

图 61: 数据解析中...

数据解析完成

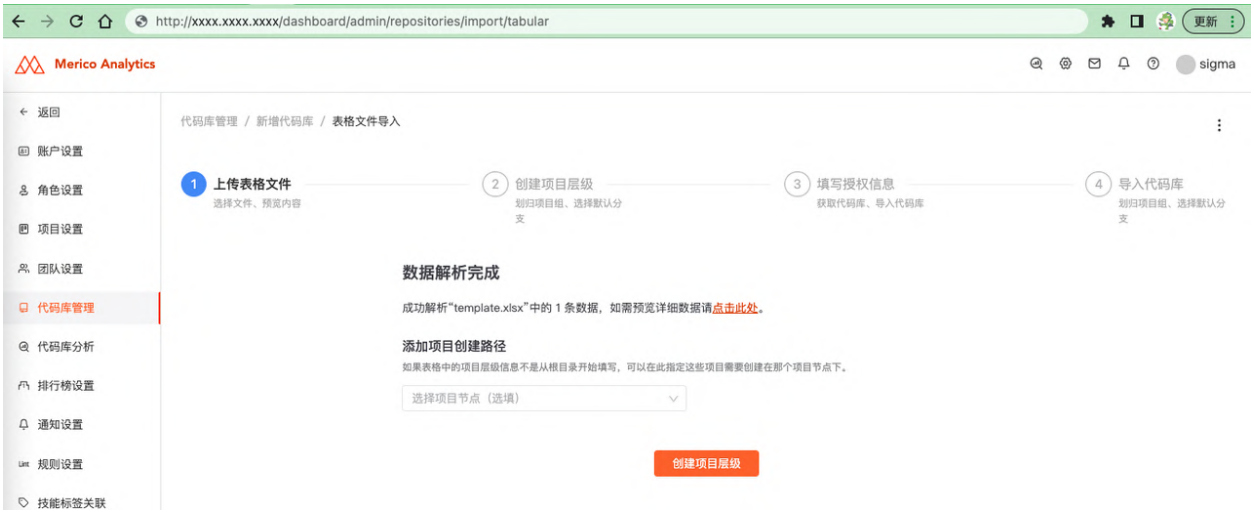

图 62: 数据解析完成

在数据解析页面可以预览数据

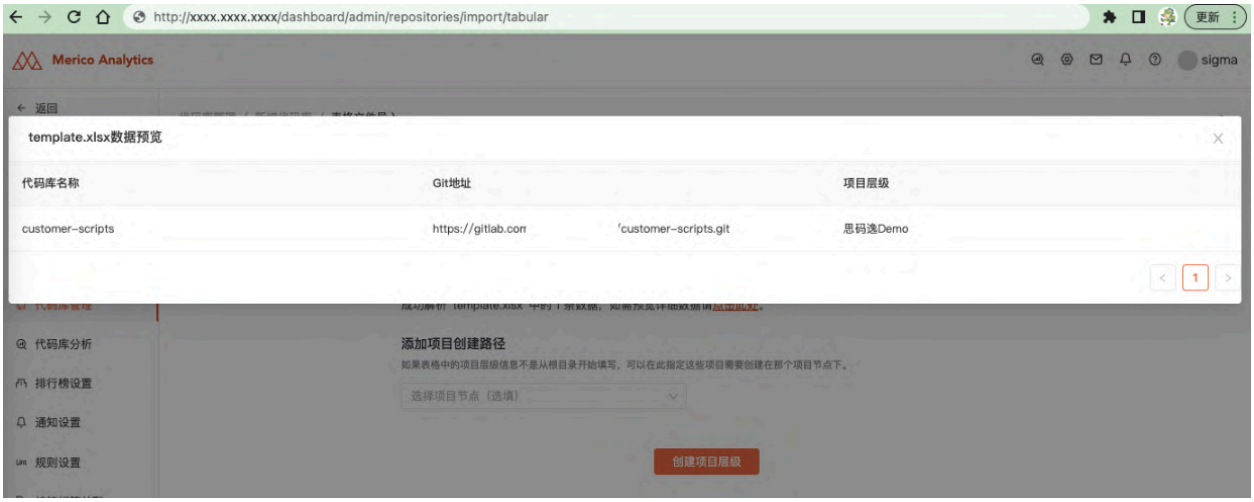

图 63: 预览数据

3.3.4.3.6 创建项目层级 (可选操作)

如果表格中的项目层级信息是从根目录项目组开始，忽略此步骤即可

如果表格中的项目层级信息不是从根目录项目组开始填写，可以在此指定这些或这个项目需要创建在那个项目节点下。选择表格中新增的代码库需要归属的项目层级，点击 **创建项目层级**按钮进入下一步

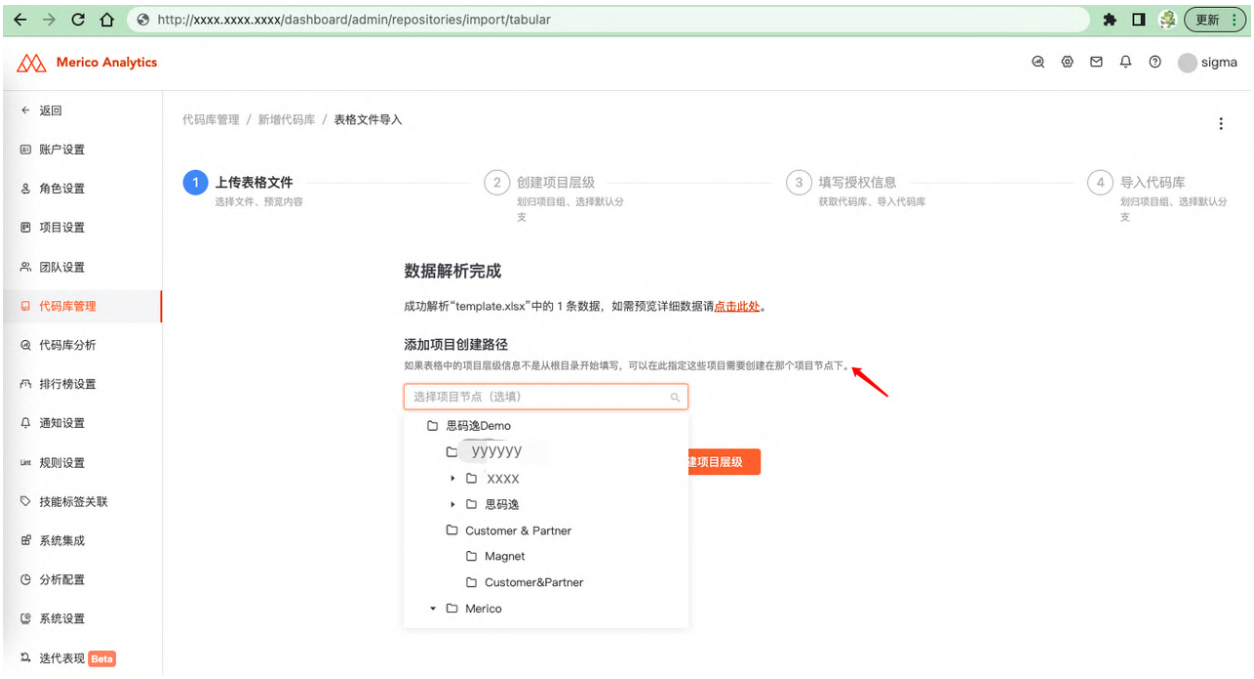

图 64: 支持创建项目层级

3.3.4.3.7 项目层级创建成功

点击 **下一步**按钮进入下一步，此处也可再次预览详细数据

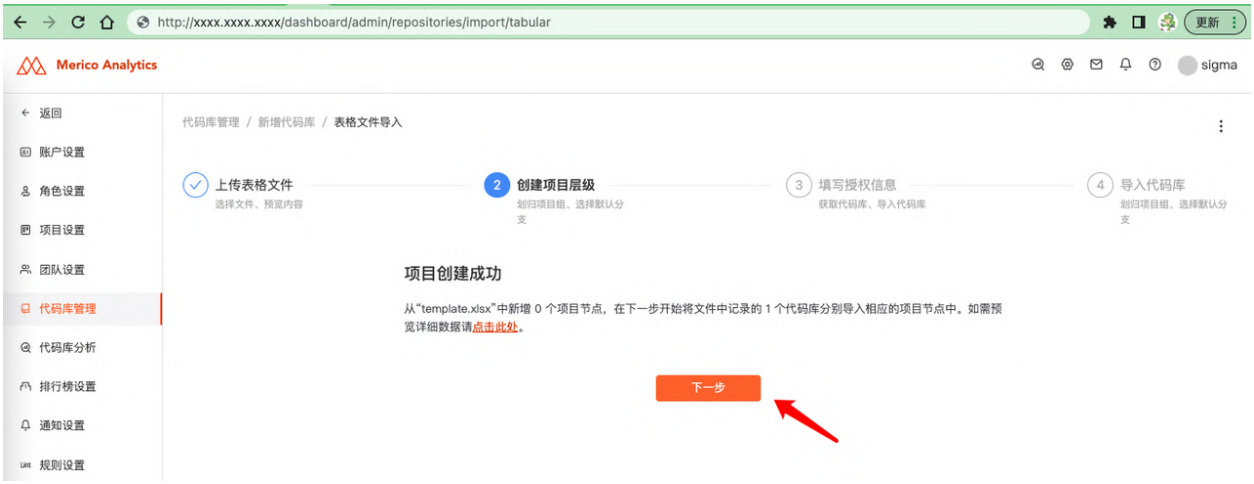

图 65: 项目层级创建成功

再次预览数据

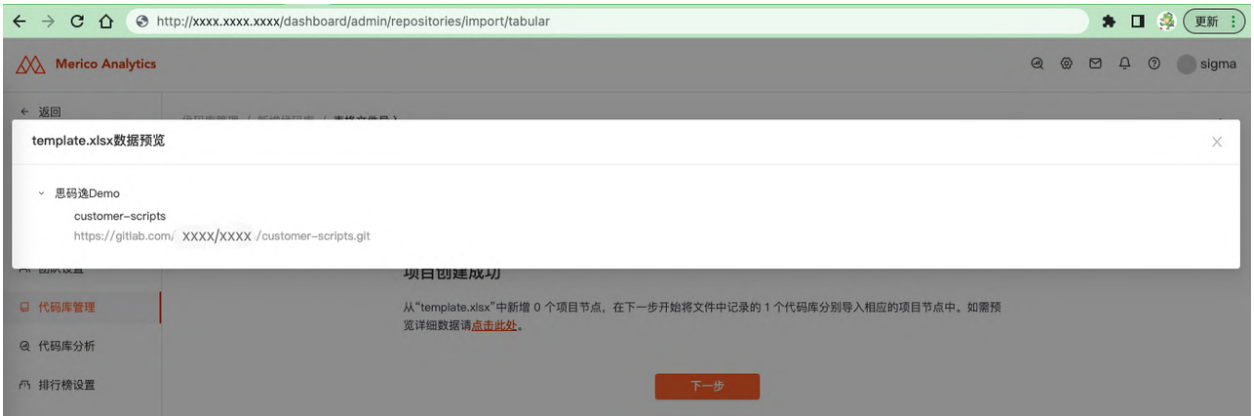

图 66: 预览项目层级数据

3.3.4.3.8 填写授权信息

1. 账号访问方式

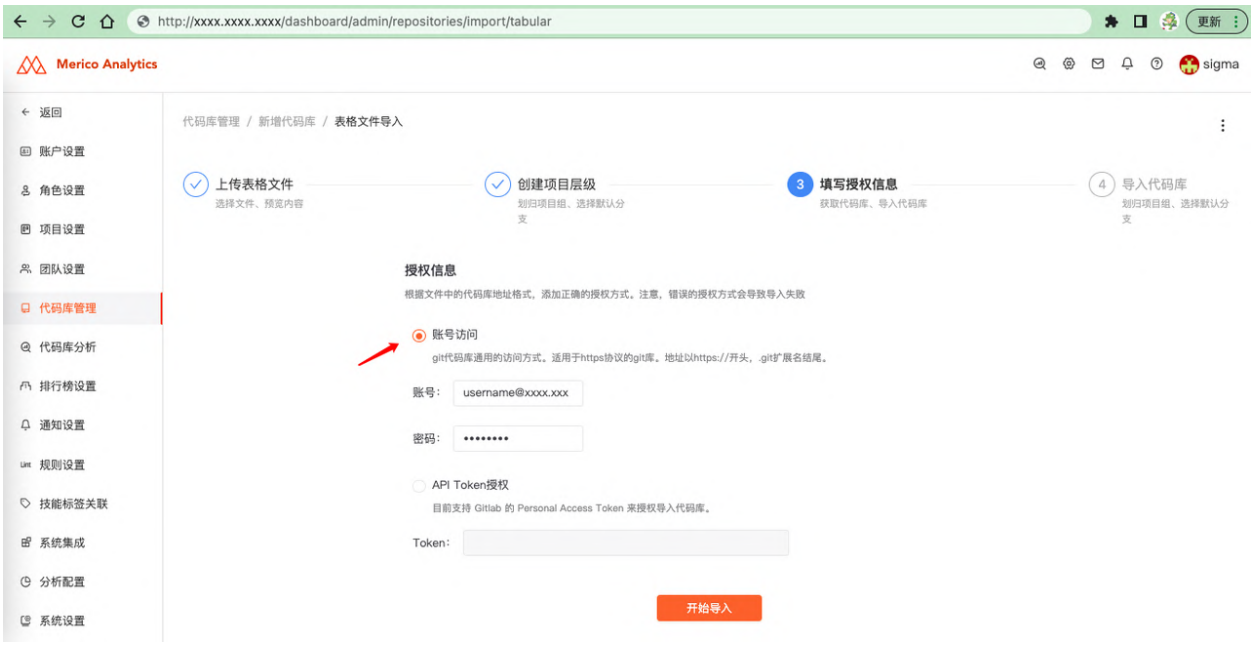

图 67: 表格导入账号信息填写页面

2. API Token 授权方式

Gitlab API Token 生成方式参考 url：https://gitlab.com/-/profile/personal_access_tokens

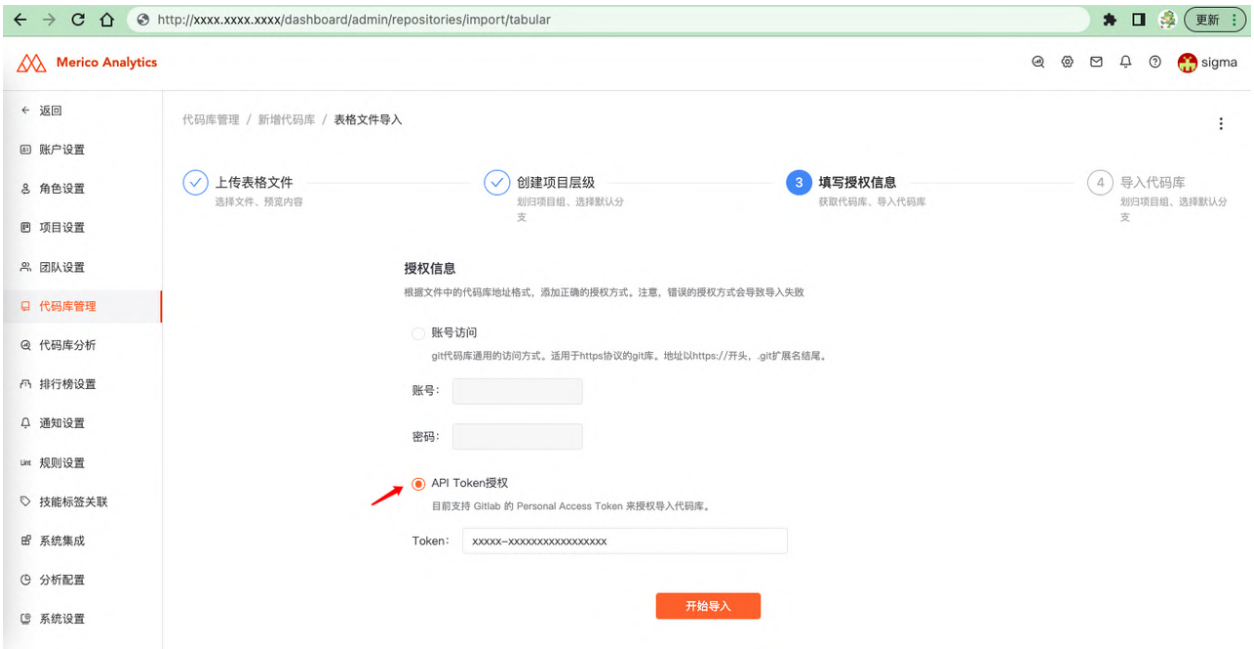

图 68: API Token 授权填写示例

3. 最后点击开始导入按钮

3.3.4.3.9 操作完毕

来到 **代码库导入中...** 界面，代码库导入为独立进程，可退出此界面进行其他操作。

点击 **代码库导入列表**链接可进入 **代码库导入任务进度**界面，查看代码库导入任务的进度

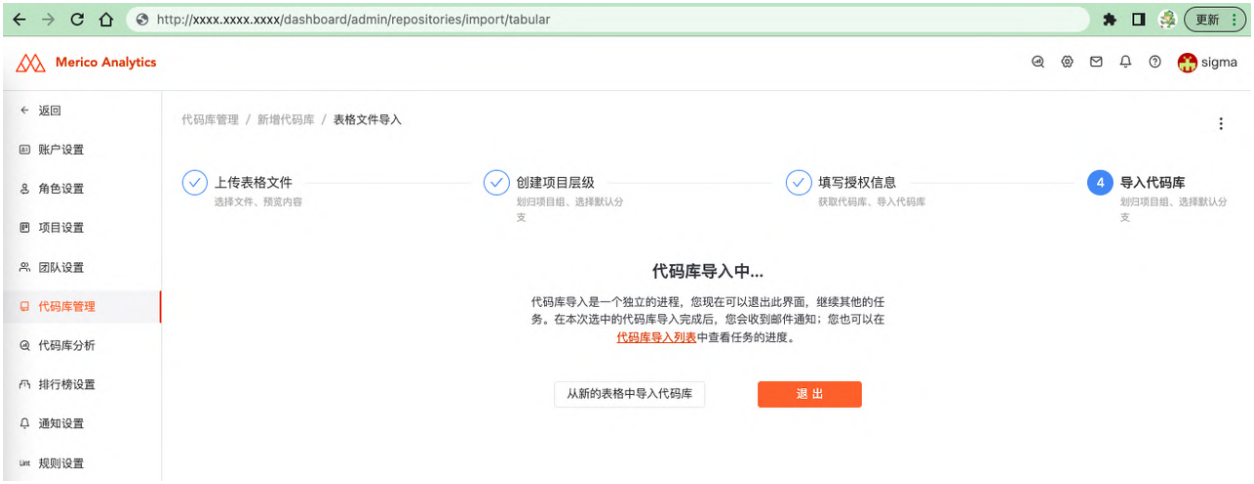

图 69: 代码库导入中...

3.3.4.3.10 查看导入进度

可以在 **导入任务进度**界面查看代码库导入进度和导入失败详细信息

图 70: 导入任务进度示例

3.3.4.3.11 支持中途退出

如果在通过模板导入代码库时，发现问题，需要退出导入过程

点击页面右上角的 ⋮，可以退出此次导入过程

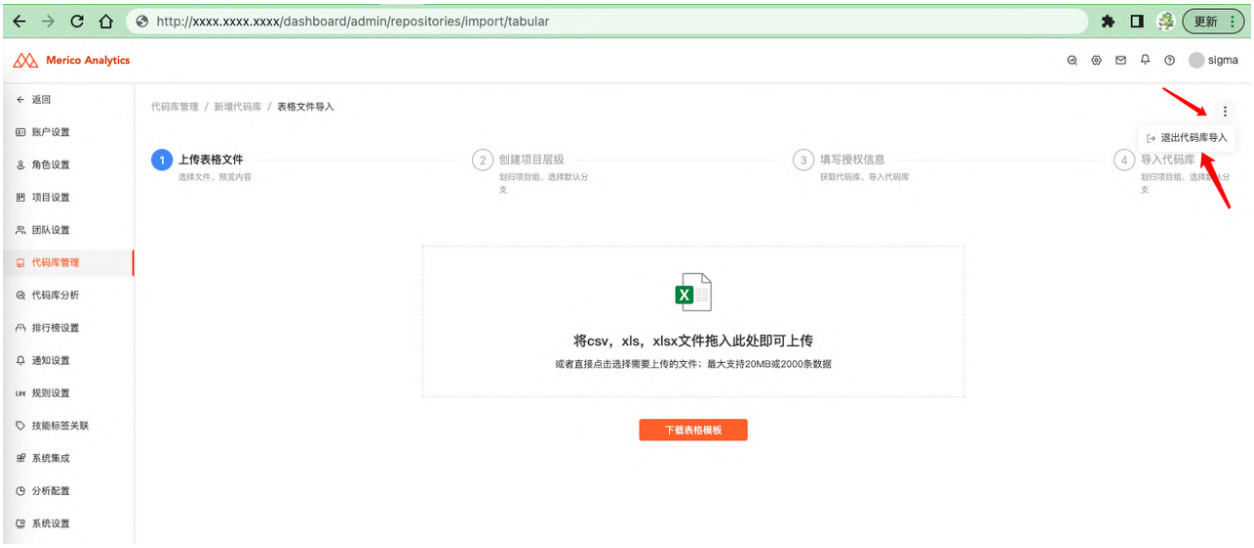

图 71: 支持中途退出

3.3.5 系统集成导入代码库

3.3.5.1 注意事项

- 操作账户需要思码逸系统 **设置-系统集成**界面的访问权限
- 需要 **GitLab 管理员**账户
- 导入 GitLab 账户将附带导入 GitLab 的项目和群组
- 如果没有创建 GitLab 应用，请参考 GitLab 创建应用的说明文档。

3.3.5.2 集成步骤

1. 选择 **设置** -> **系统集成** -> **集成 GitLab**，按照页面提示完成系统集成配置，点击 **保存信息**按钮保存系统集成信息。点击 **开始集成**按钮启动集成。

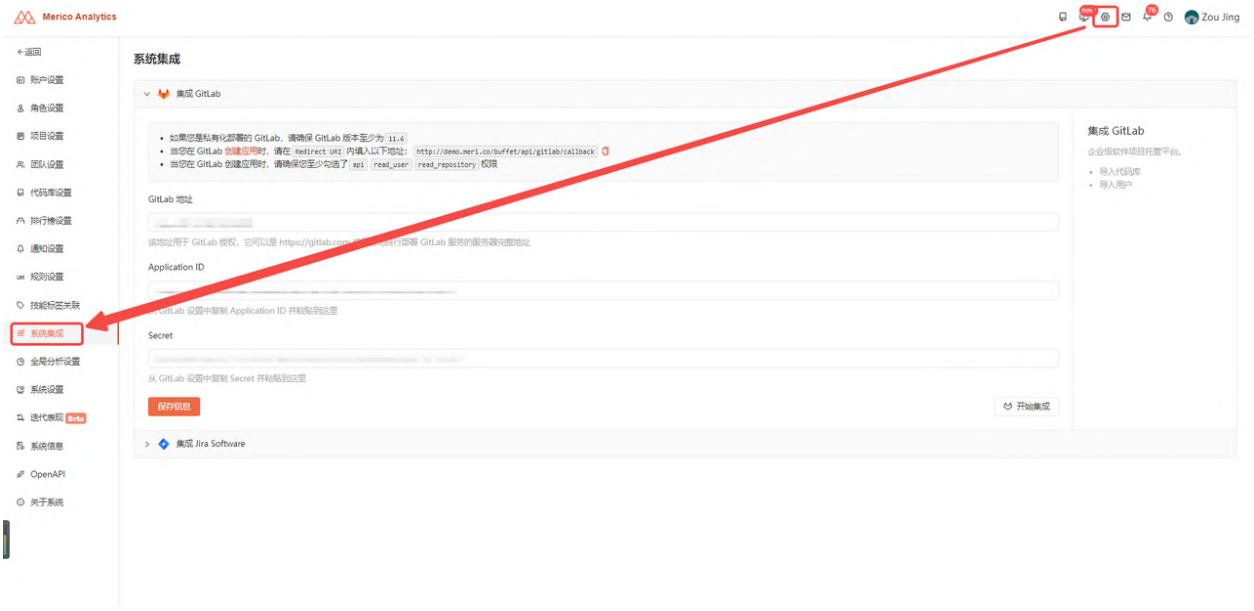

图 72: 系统集成

2. 弹出界面上点击 **下一步**：

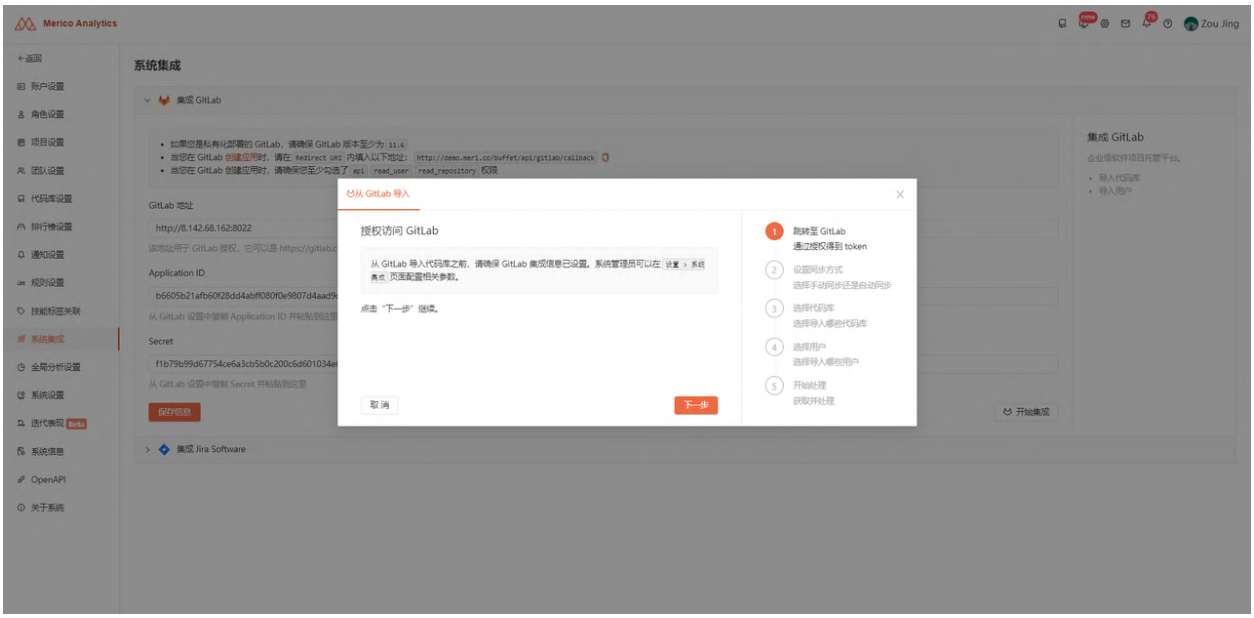

图 73: 授权访问 Gitlab

3. 弹出界面输入 GitLab 用户名和密码，点击 **Sign In**，使用 GitLab 账户 **登录**和 **授权**。

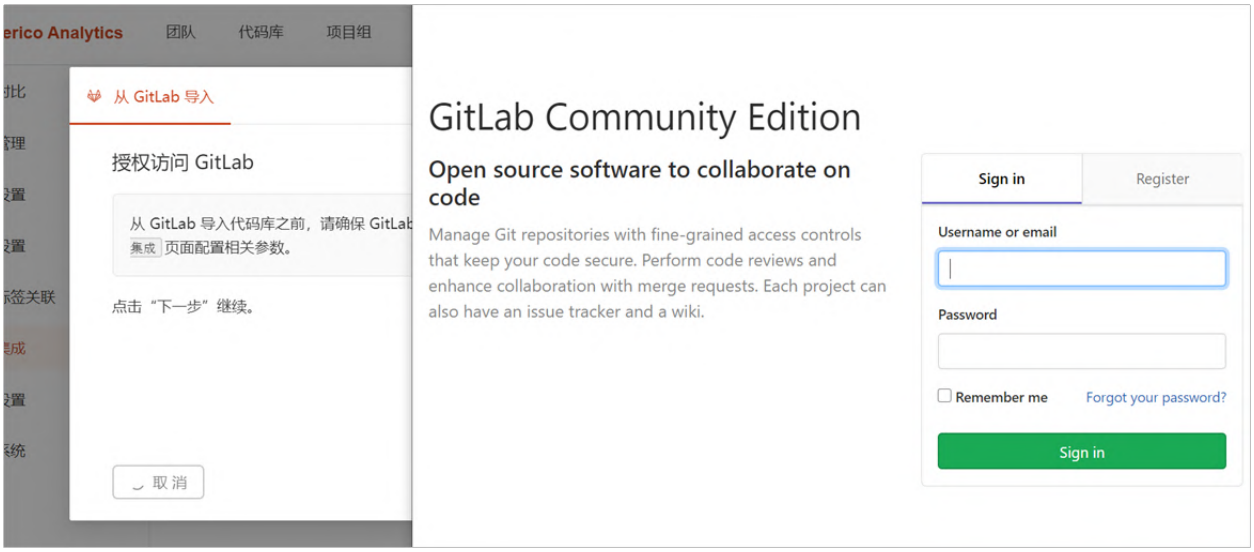

图 74: Gitlab 账号密码填写

4. 可选 **手动同步**和 **自动同步**两种同步方式，这里选择 **手动同步** -> 点击 **下一步**按钮。若选择 **自动同步**的集成方式，会在每天 0 点自动同步 GitLab，如 GitLab 端有新增代码库将会同步新增代码库并分析。

图 75: 同步方式选择

5. 选择需要导入的代码库，点击 **下一步**。

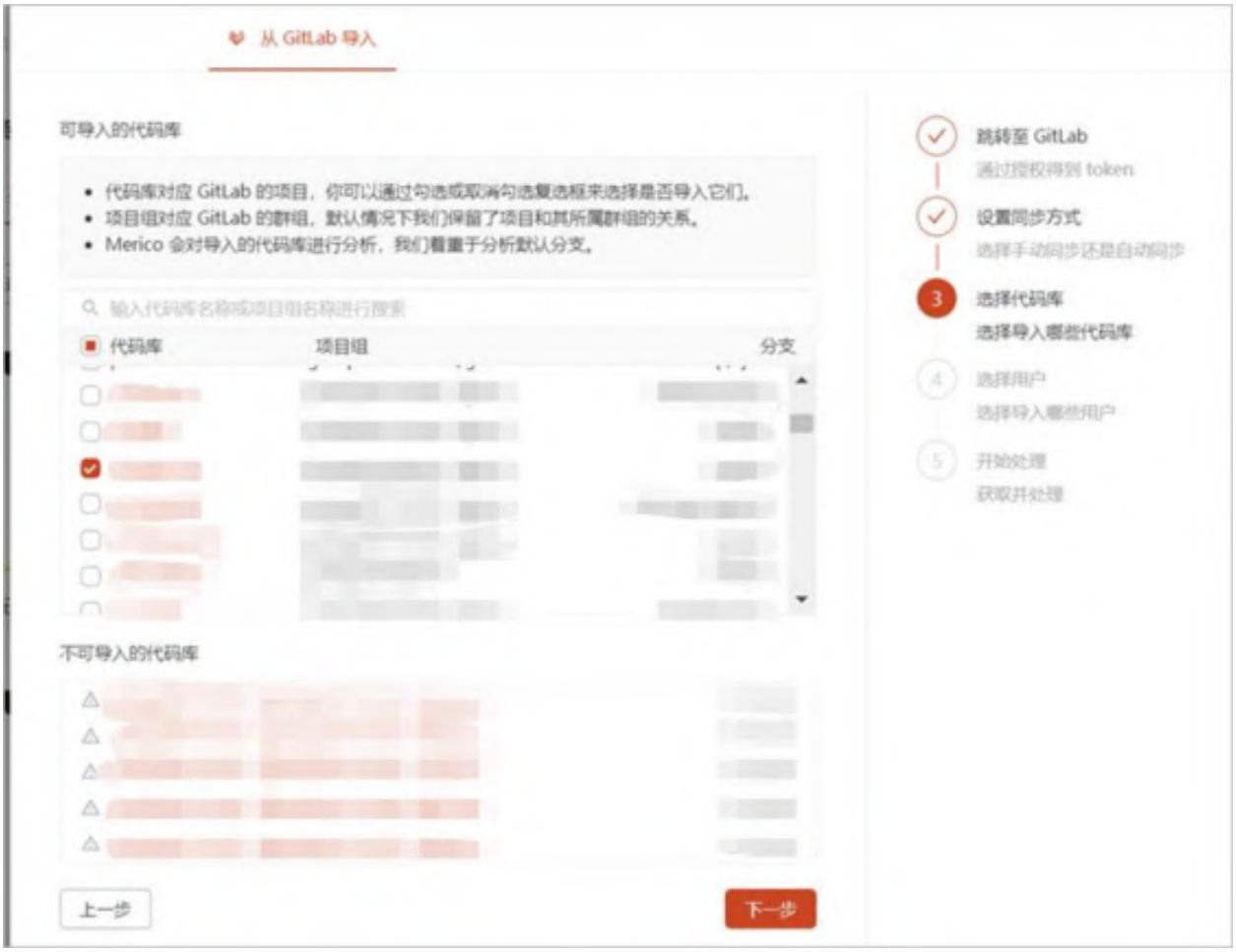

图 76: 选择可导入的代码库

6. 选择需要导入的用户 -> 选择 **启用/禁用**导入的用户 -> 点击 **导入**。

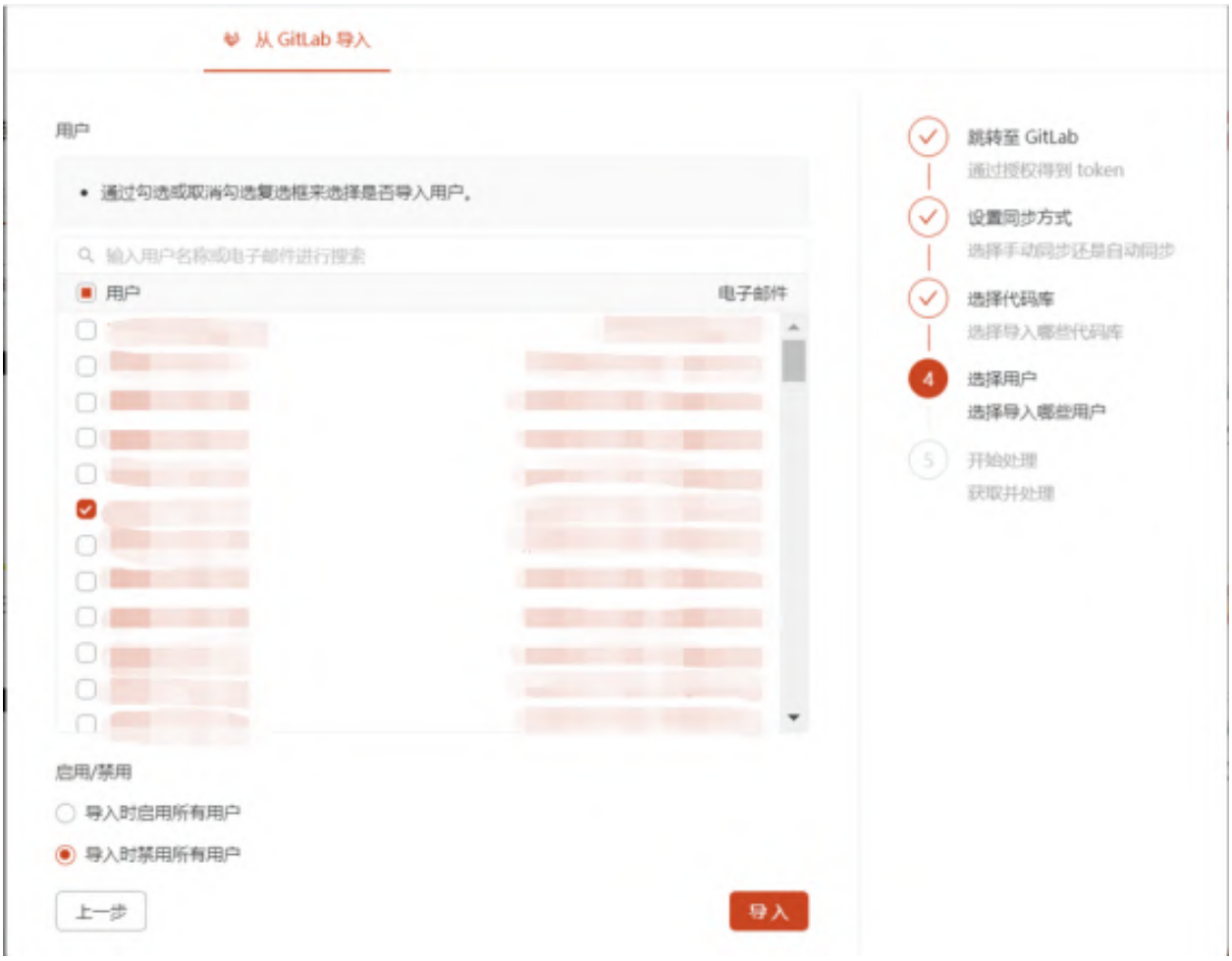

图 77: 从 Gitlab 导入用户

7. 导入成功，点击 **完成** 按钮结束。

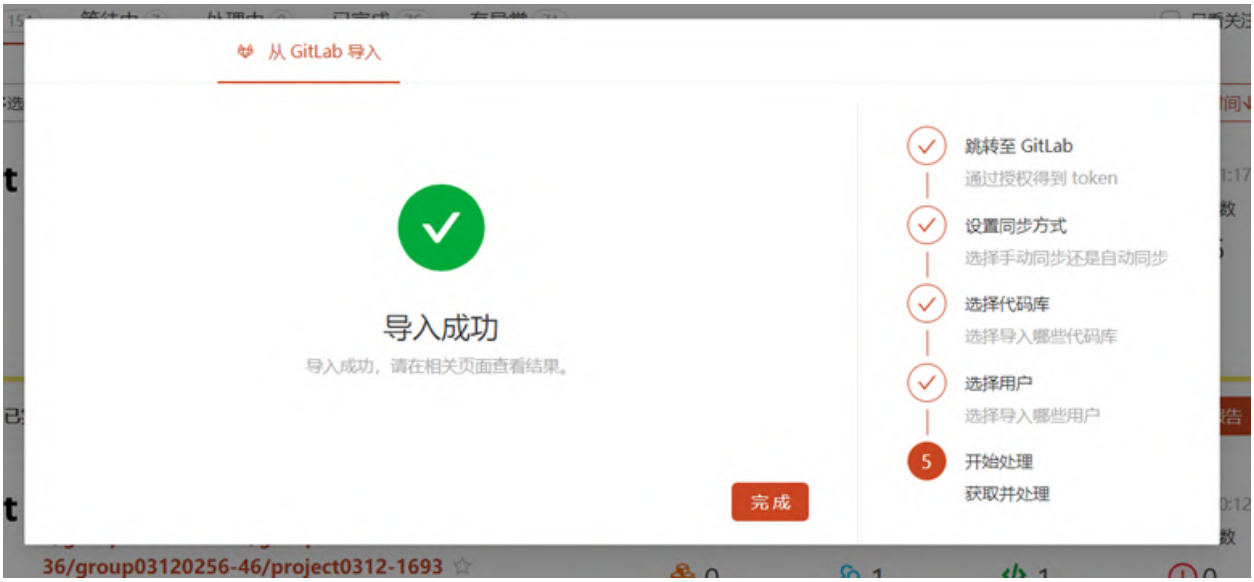

图 78: 导入成功

3.4 第三步：配置用户账号

3.4.1 导入或设置账户

3.4.1.1 账户管理结构

- **团队：**树状结构，顶级父团队只有一个，下属各级子团队可有多个。每个成员账户可归属多个团队。
- **角色：**给不同账户赋予不同的角色，在 **角色设置**中限定不同角色可以访问的不同工作区，也即是可以打开的系统界面。
- **项目权限：**树状结构，顶级父项目组只有一个，下属各级子项目组和项目可有多个。可以为一个账户或一个团队设置可访问项目（组）权限。
- **团队权限：**可以为一个账户或一个团队设置可访问团队数据的权限。某个账户的归属团队和可访问数据的团队**可不一致**。

3.4.1.2 注意如下要点

- 可以给某个项目（组）添加/删除可访问用户账户，也可给某个项目（组）添加/删除可访问的团队，此团队里的全部成员账户都可访问此项目组。
- == **某账户想访问项目表现、团队表现、工程师表现、技术债监控、迭代表现等与代码库分析数据相关的工作区界面之前，都需要先给其分配相应的项目组访问权限。**==

3.4.1.3 团队相关操作

1. 新建团队
2. 编辑团队
3. 删除团队
4. 编辑数据权限
5. 编辑团队成员

3.4.1.4 账户相关操作

1. 添加账户
2. 批量添加账户 - 从 CSV 导入
3. 批量添加账户 - 从 GitLab 导入
4. 批量添加账户 - 表格文件导入
5. 直接添加账户
6. 修改账户信息
7. 重置账户密码
8. 合并账户
9. 禁用/启用账户
10. 允许登录/禁止登录
11. 删除账户
12. 设置贡献者黑名单

3.4.1.5 角色相关操作

1. 创建角色
2. 编辑角色
3. 选择用户
4. 查看用户
5. 删除角色

3.4.1.6 项目相关操作

1. 新建项目
2. 编辑项目
3. 删除项目
4. 选择用户
5. 选择团队

3.4.2 团队设置

3.4.2.1 要点概述

- **团队**: 对应公司组织架构。为树状结构，顶级父团队只有一个，下属各级子团队可有多个。每个成员账户可关联多个团队。
- **团队设置操作：**新建团队 / 编辑团队 / 删除团队 / 编辑数据权限 / 编辑团队成员。
- 只有授权给某账户某团队或其父团队数据查看权限，此账户才可在 **团队表现**处查看此团队的数据。
- 只有 **归属**于某团队的账户的指标数据才会被纳入此团队的汇总数据中。
- 账户的 **团队归属**和 **团队数据查看权限**可不一致。

3.4.2.2 设置导航

菜单 **设置** -> 左侧菜单栏 **团队设置**

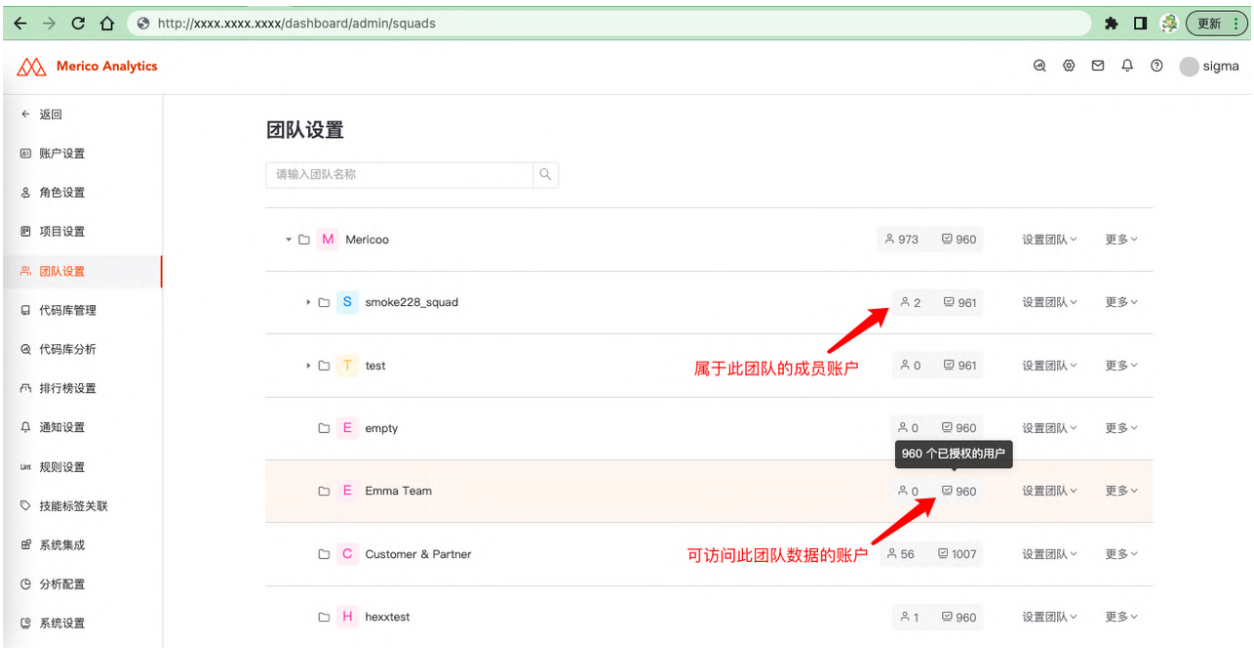

图 79: 团队设置

3.4.2.3 团队设置操作

3.4.2.3.1 新建团队

Step 1：设置团队 -> 创建新团队。注意：不可删除顶层父级团队。

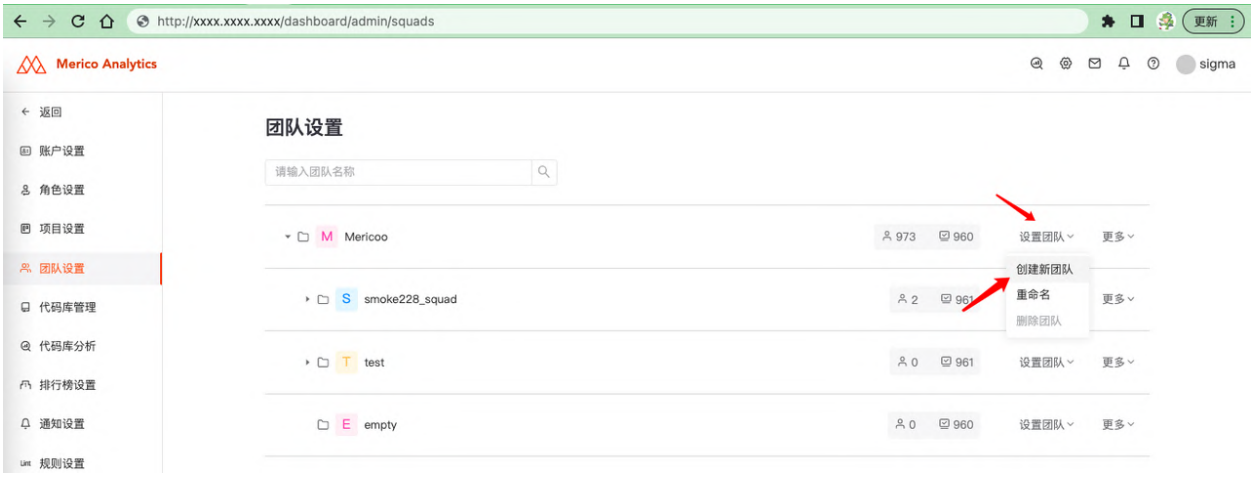

图 80: 创建新团队

Step 2：输入团队信息 -> 点击 **确定**按钮。

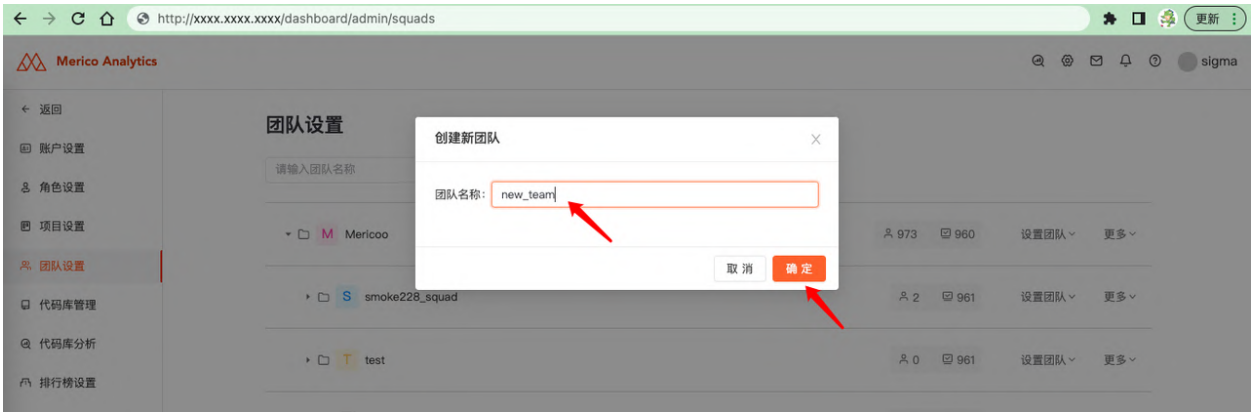

图 81: 填写团队名称

Step 3：查看新的团队创建成功。

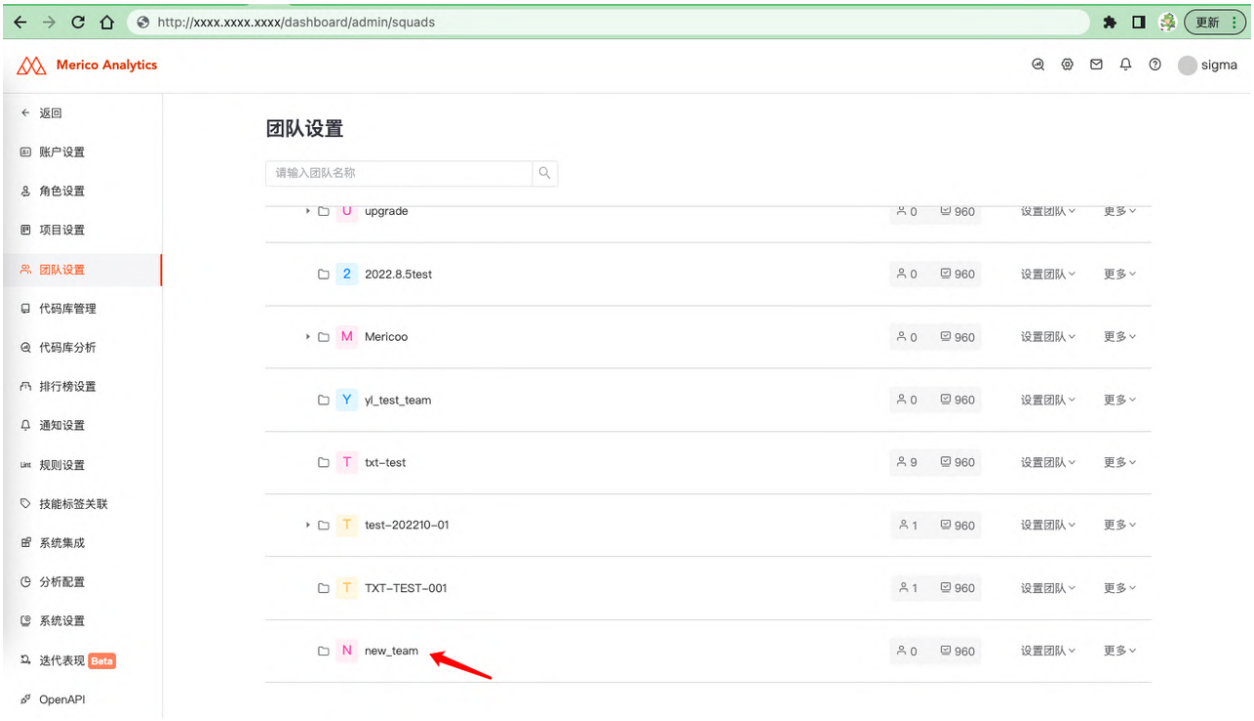

图 82: 创建新团队成功

3.4.2.3.2 编辑团队

在 **设置团队**的下拉菜单里，选择 **重命名**可以重命名该团队，选择 **移动团队**可以移动该团队在组织中的相对位置。

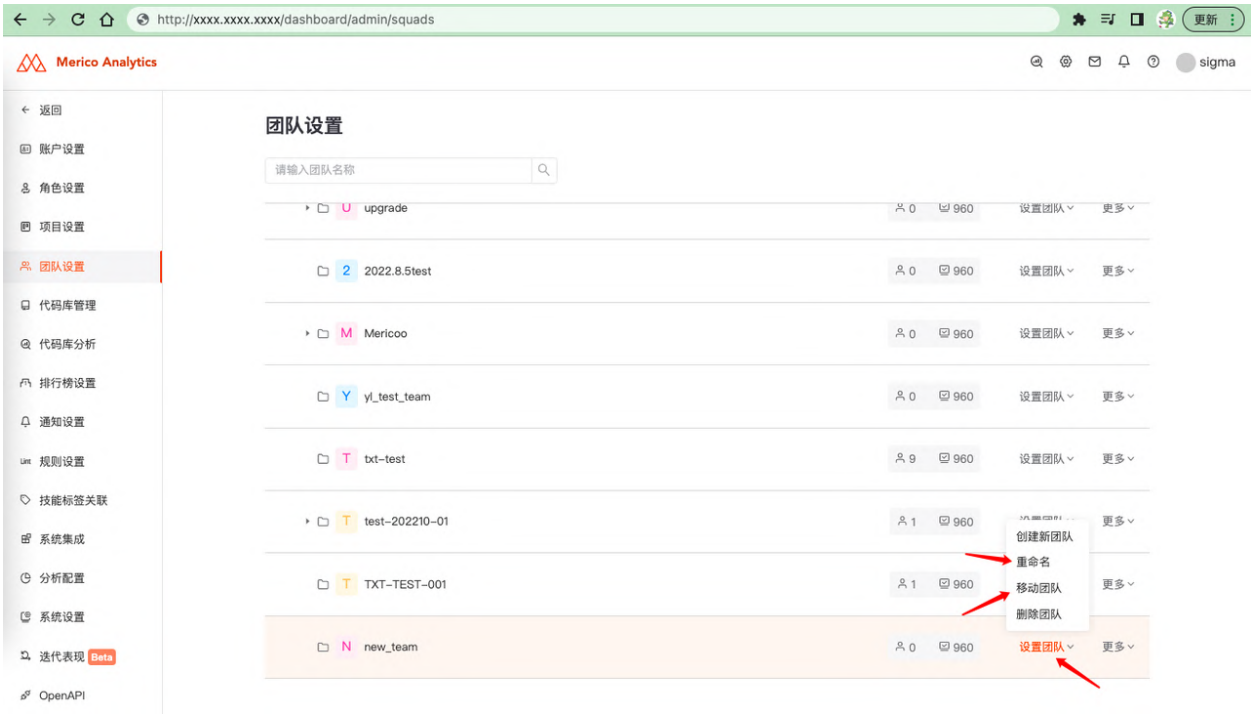

图 83: 重命名和移动团队

重命名团队

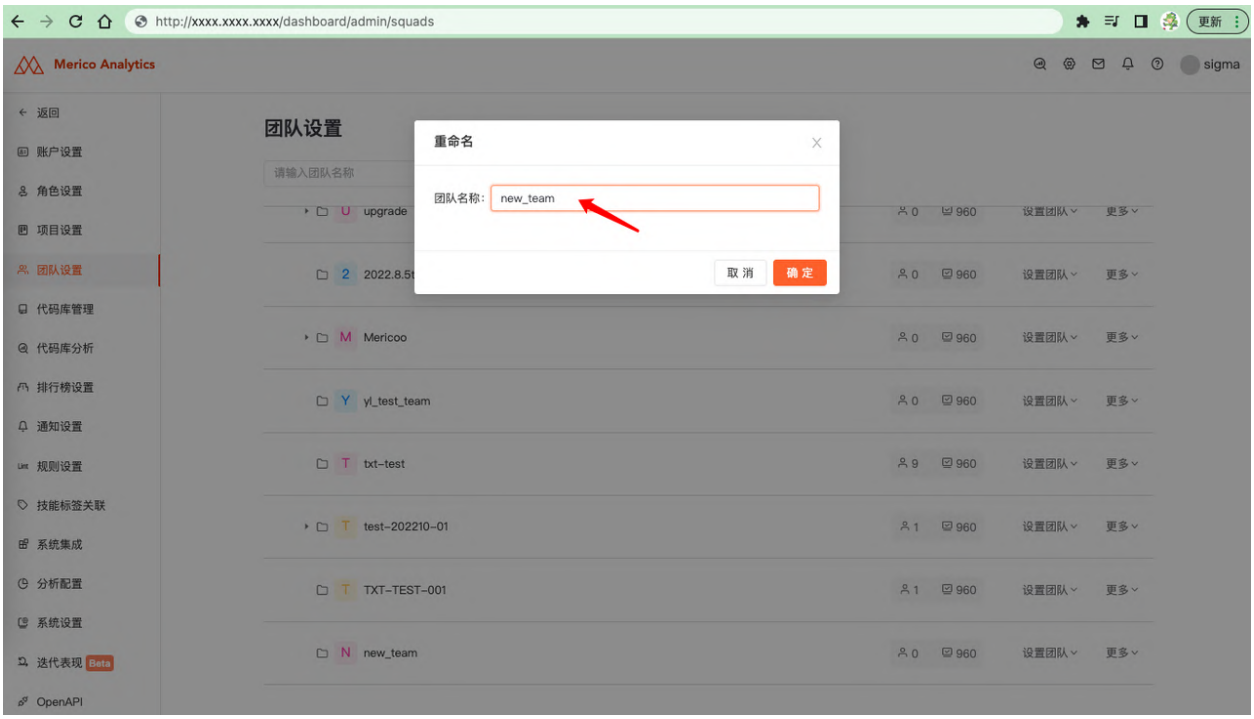

图 84: 重命名团队

移动团队

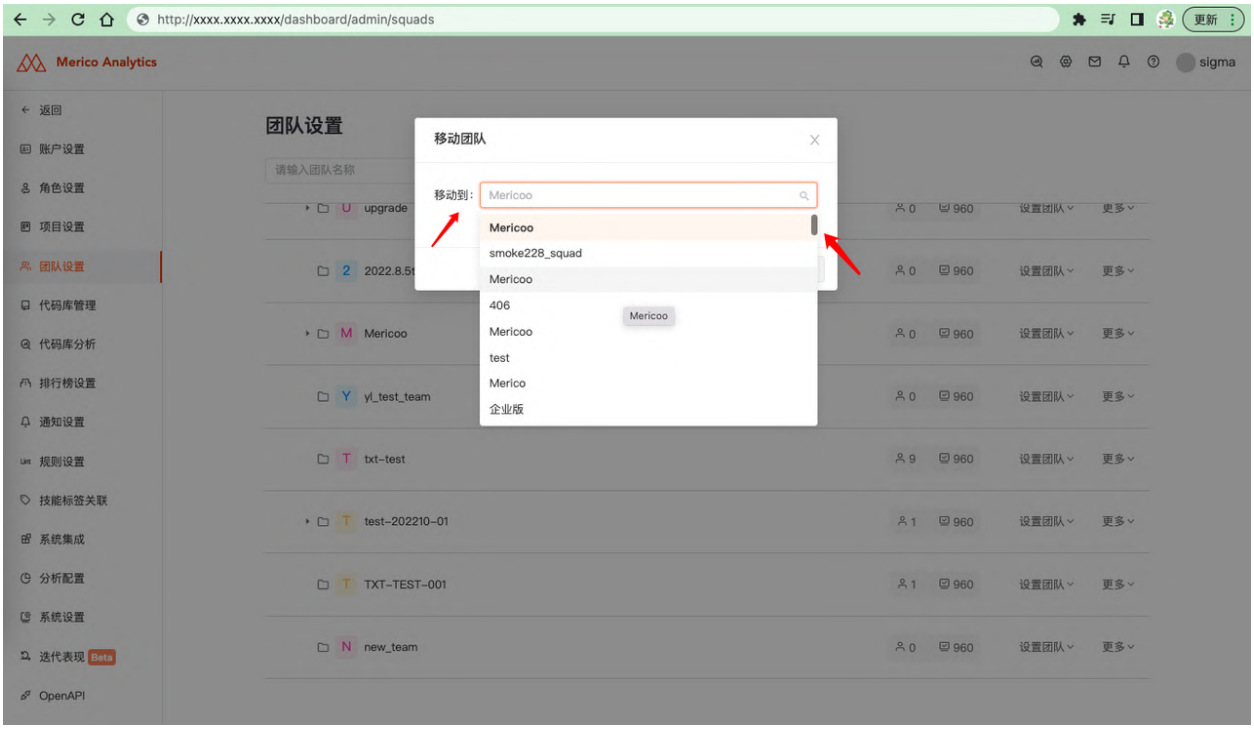

图 85: 移动团队

3.4.2.3.3 删除团队

在 **设置团队**的下拉菜单里，选择 **删除团队**可以删除该团队。

注意：删除团队之前，需要把此团队及其下属团队内的成员全部移除，此团队的全部下属团队也都需要移除。

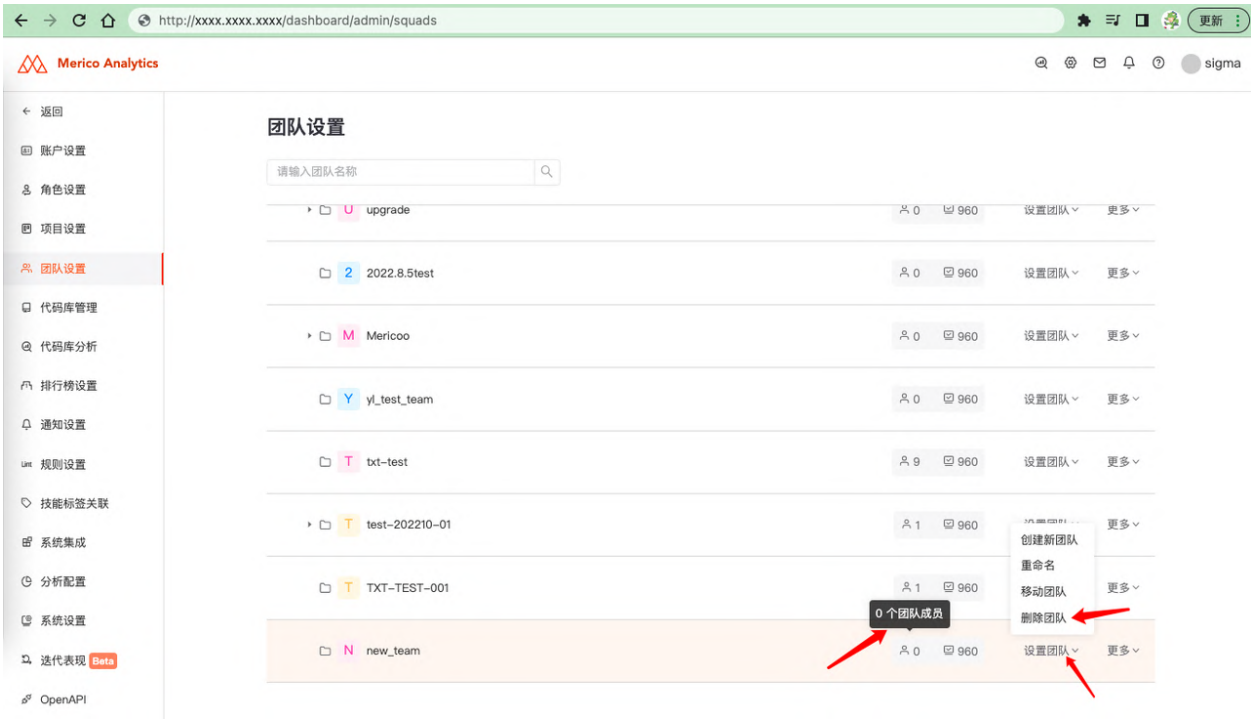

图 86: 删除团队

3.4.2.4 账户团队变更

3.4.2.4.1 编辑数据权限

Step 1：更多 -> 编辑数据权限

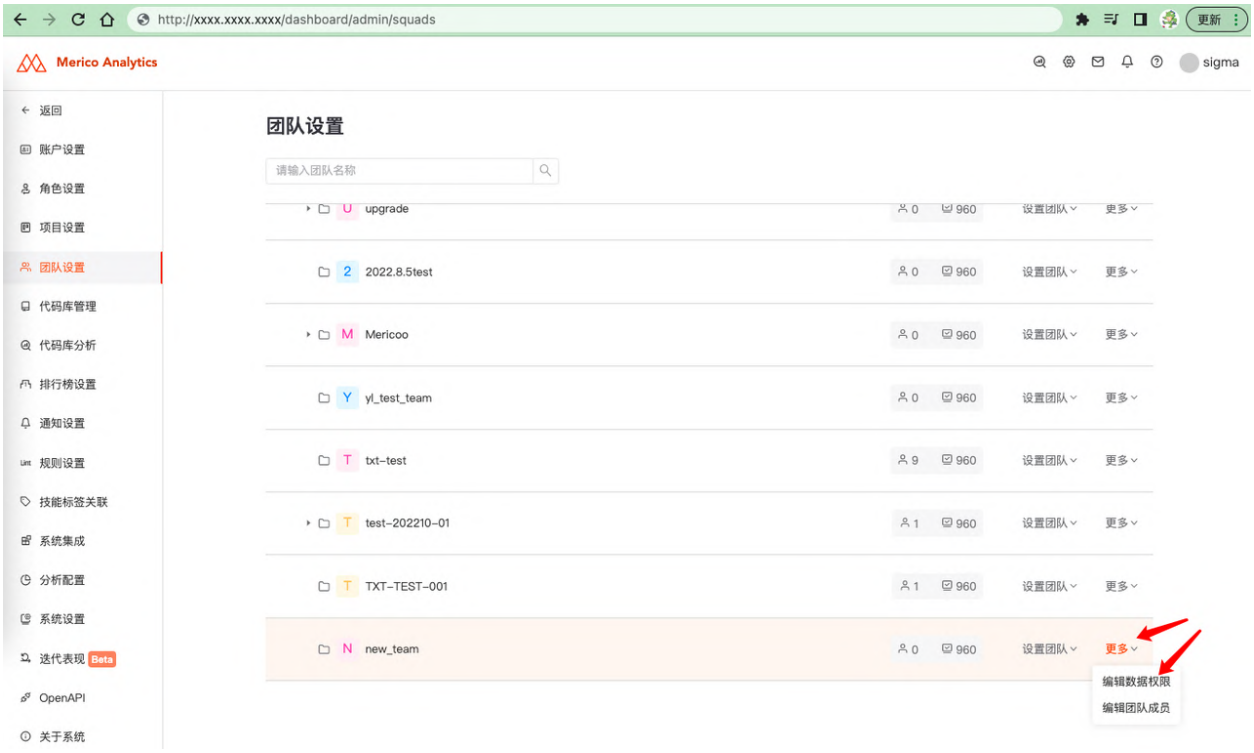

图 87: 编辑团队数据权限

Step 2：勾选或取消勾选账户，以设置某账户**可以/不可以访问此团队的数据**。点击 ** 确定 ** 按钮保存设置。

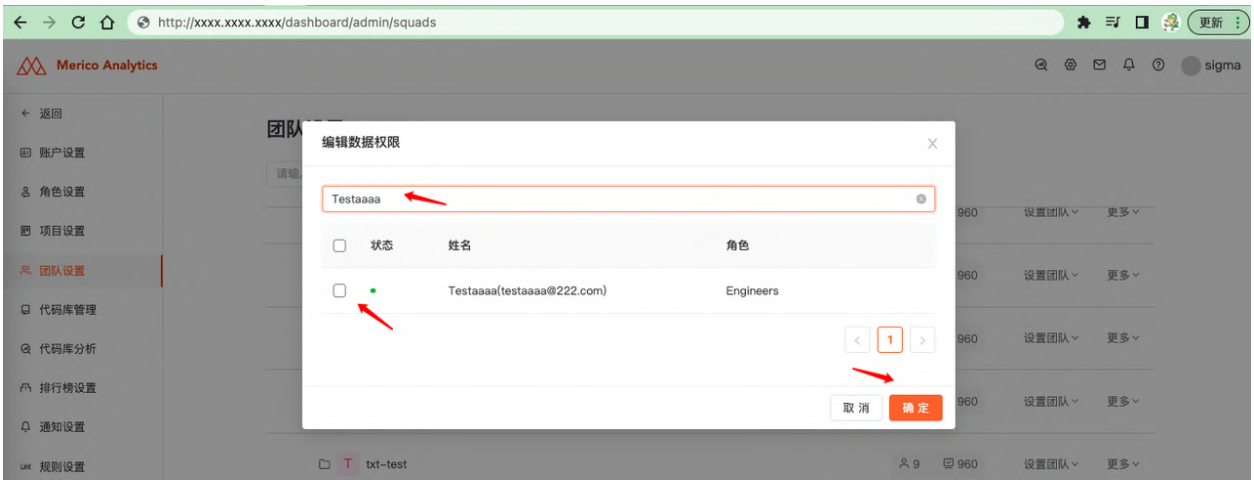

图 88: 勾选或取消勾选账户

设置成功后效果，该团队的已授权用户增加或者减少，图例中是 +1

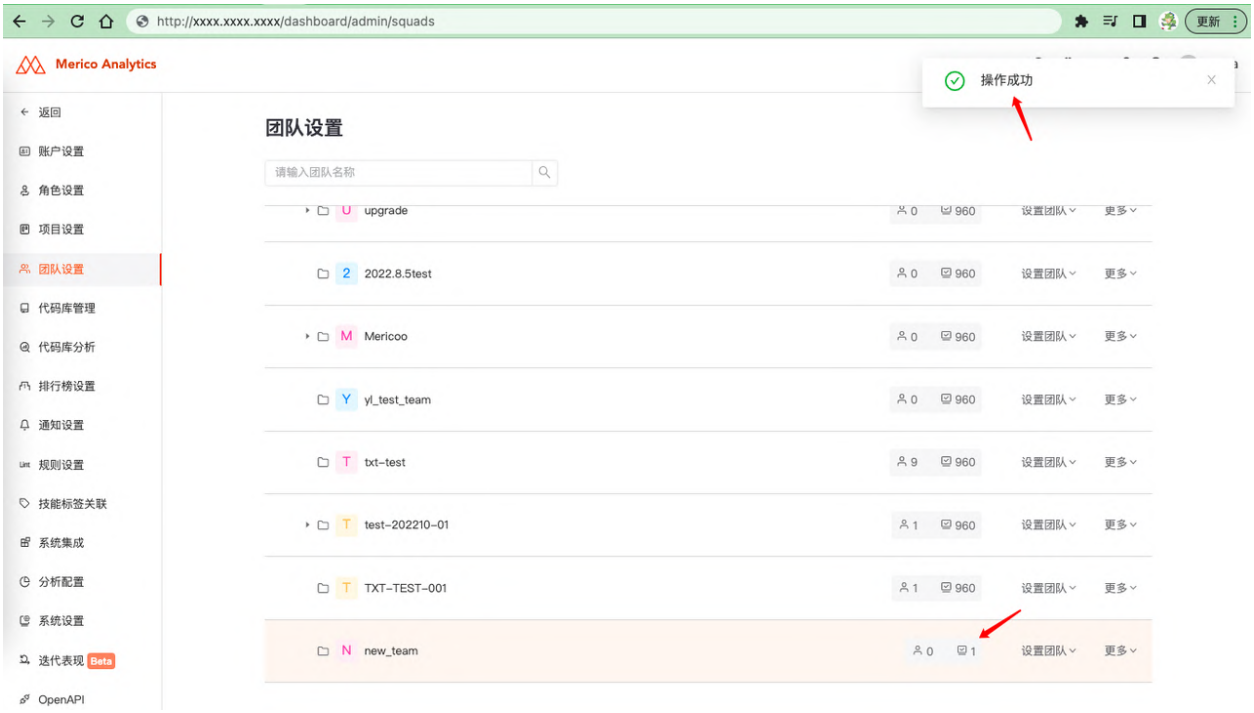

图 89: 设置成功示例

3.4.2.4.2 编辑团队成员

Step 1：更多 -> 编辑团队成员

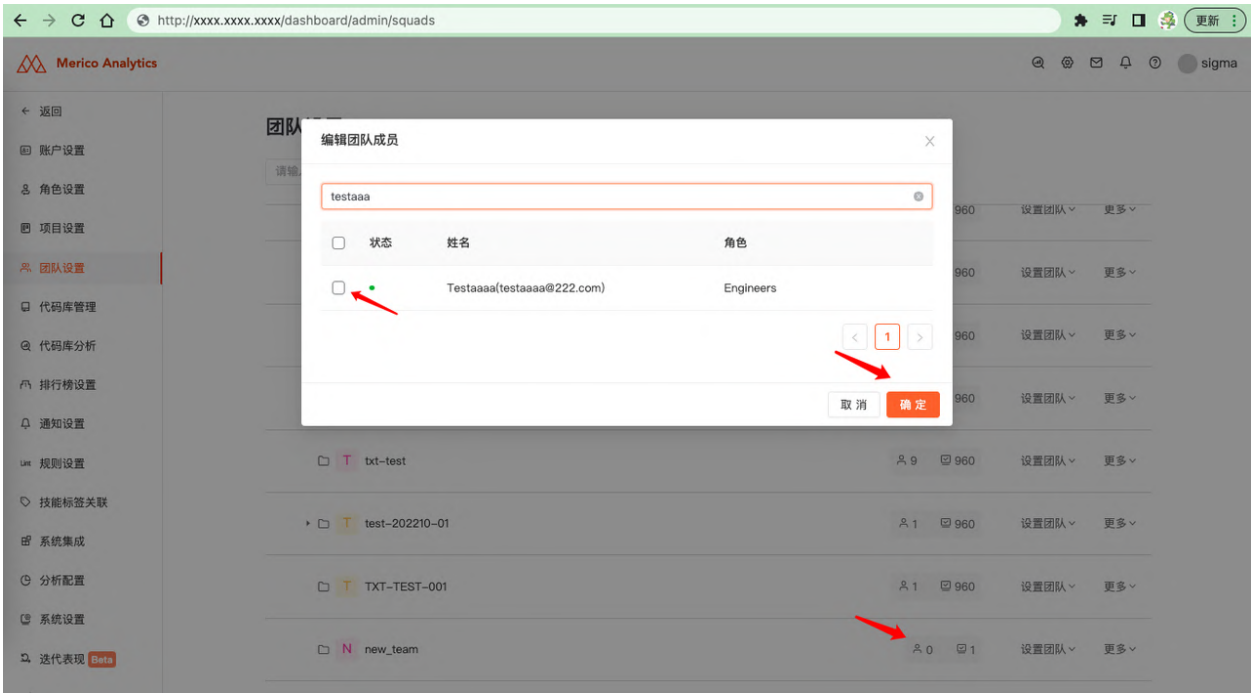

图 90: 编辑团队成员

Step 2：勾选或取消勾选账户，以设置某账户**属于/不属于访问此团队**。点击 **确定**按钮保存设置。

== 属于某团队的账户的指标数据才会被纳入此团队的汇总数据中。==

设置成功后效果，该团队的成员数量增加或者减少，图例中是 +1

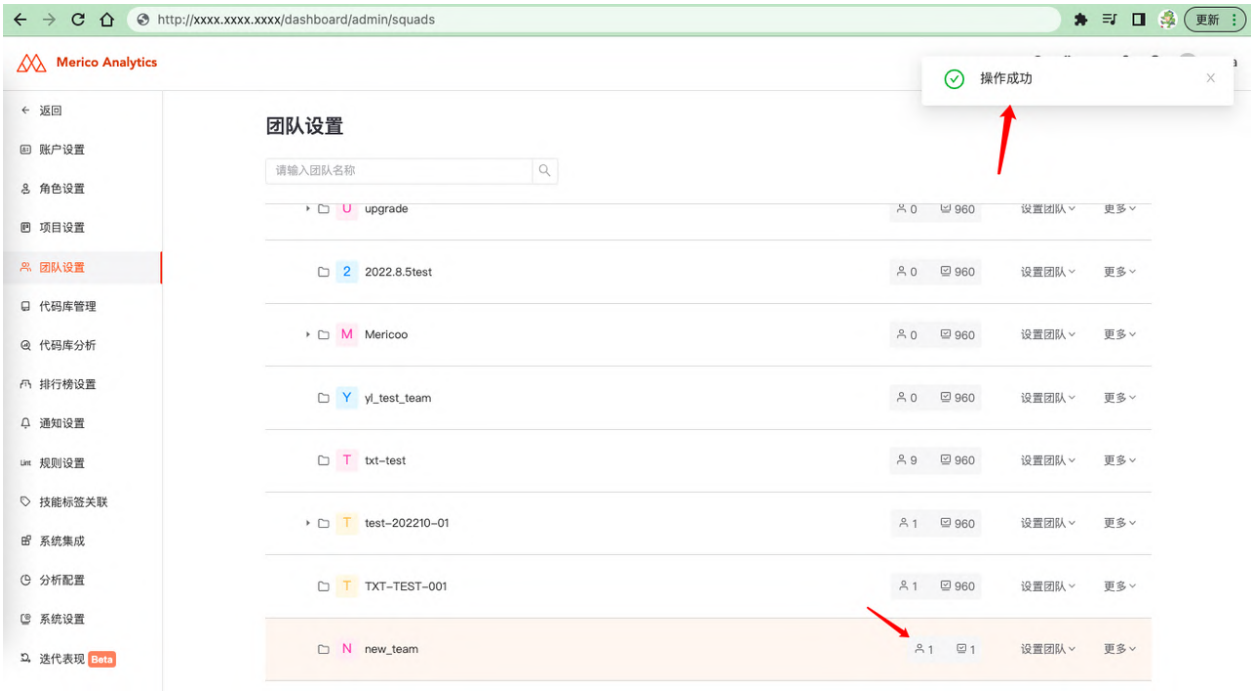

图 91: 勾选或取消勾选账户成功示例

3.4.3 角色设置

3.4.3.1 要点概述

- **角色设置**限定了属于某角色的账户可以访问的不同工作区，也即是可以打开的系统界面（菜单）。
- 可以给一个账户设置多个角色。
- 可以设置某个角色是否可以接收管理者周/日报。此设置在 **设置 - 系统设置 - 周日报配置**处配置。
- 系统默认创建了四个角色：Engineers，Tech Leaders，Project Leaders，Executives，这四个系统默认角色可以被编辑，但是不可被删除。
- 只有没有关联成员账户的自定义角色可以被删除。
- == **给某账户设置了相应的角色后，还需要给在【设置 - 项目设置】处，设置此账户或此账户所在团队可访问的项目组，此账户可访问的工作区（左侧菜单栏列表）才会正常显示。** ==

3.4.3.2 设置导航

菜单 **设置** -> 左侧菜单栏 **角色设置**

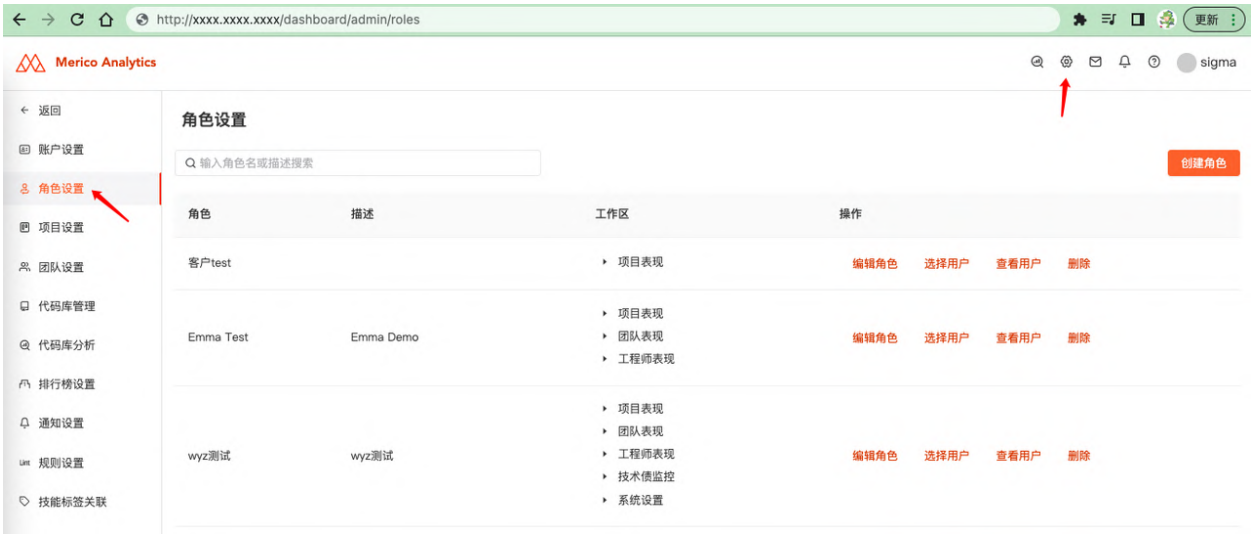

图 92: 角色设置

3.4.3.3 角色设置操作

3.4.3.3.1 创建角色

Step 1：点击 **创建角色**按钮。

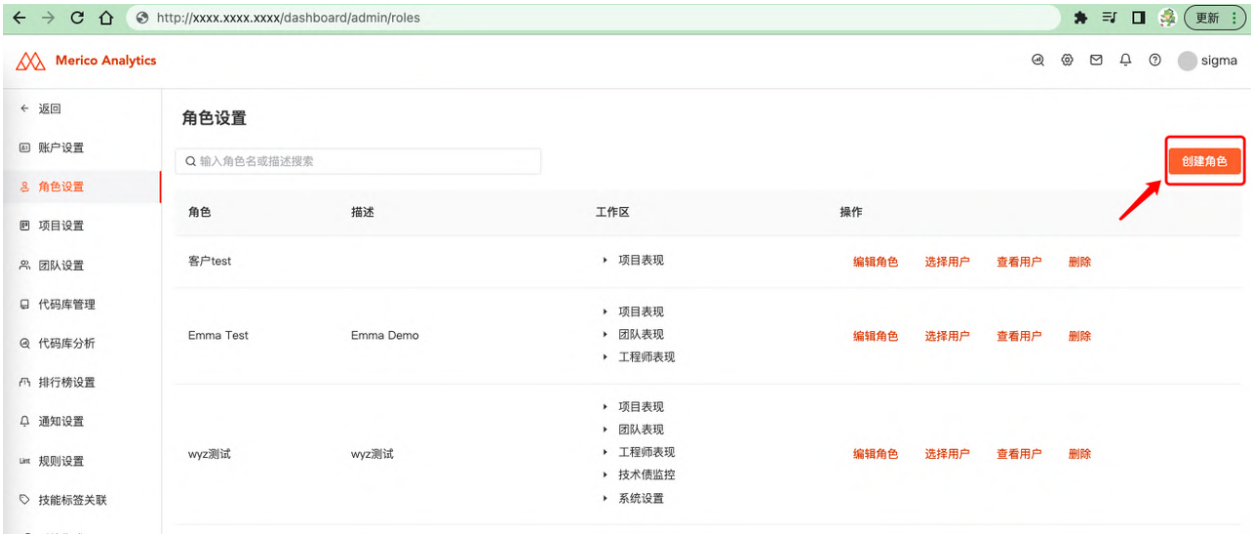

图 93: 创建角色

Step 2：输入角色名称和描述，选择此角色的工作区，即可访问的界面。点击 **确定**按钮完成角色创建。

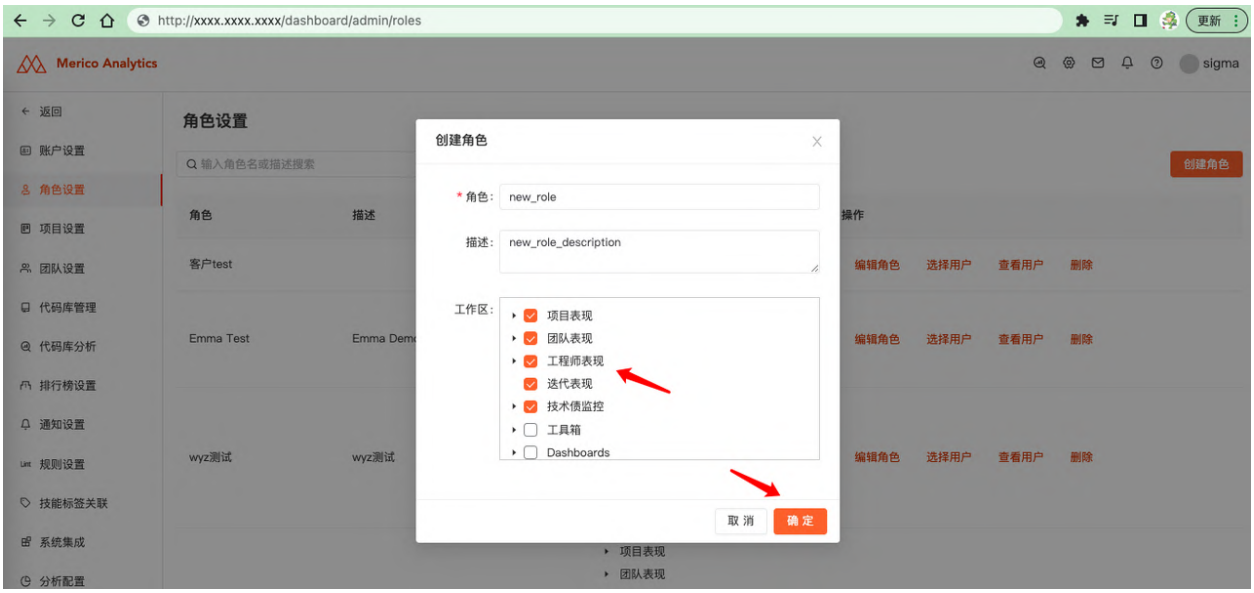

图 94: 配置角色工作区

Step 3：角色创建成功。

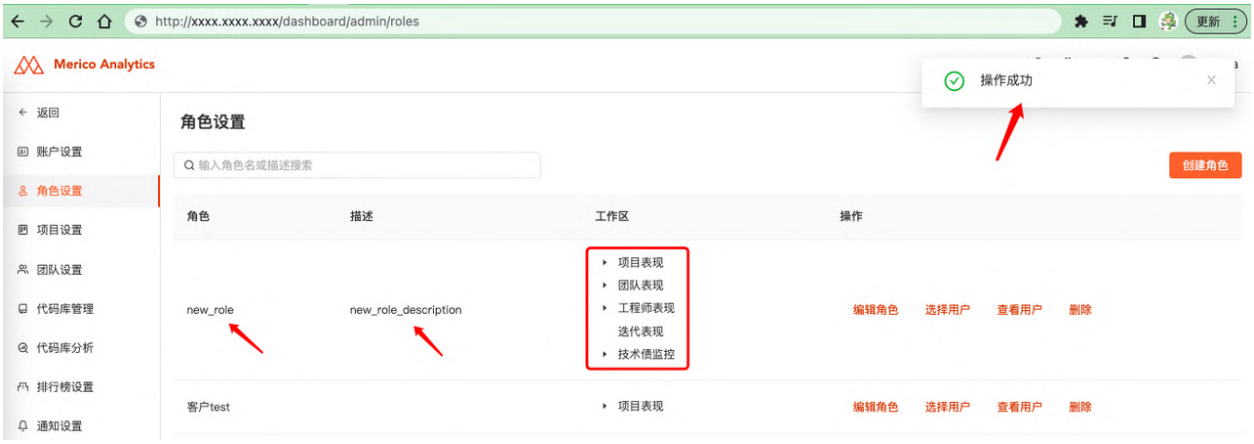

图 95: 角色创建成功

3.4.3.3.2 编辑角色

Step 1：点击某角色的 **编辑角色**按钮。

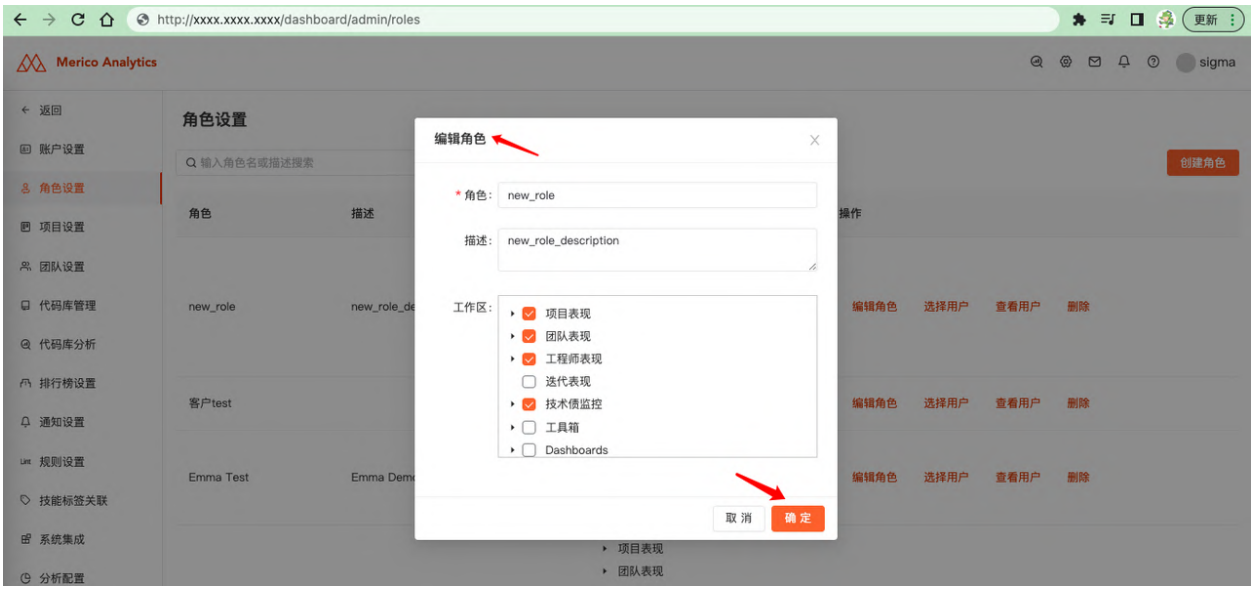

图 96: 编辑角色及其工作区

Step 2：编辑此角色。编辑完成后，点击 **确定**按钮保存编辑结果。

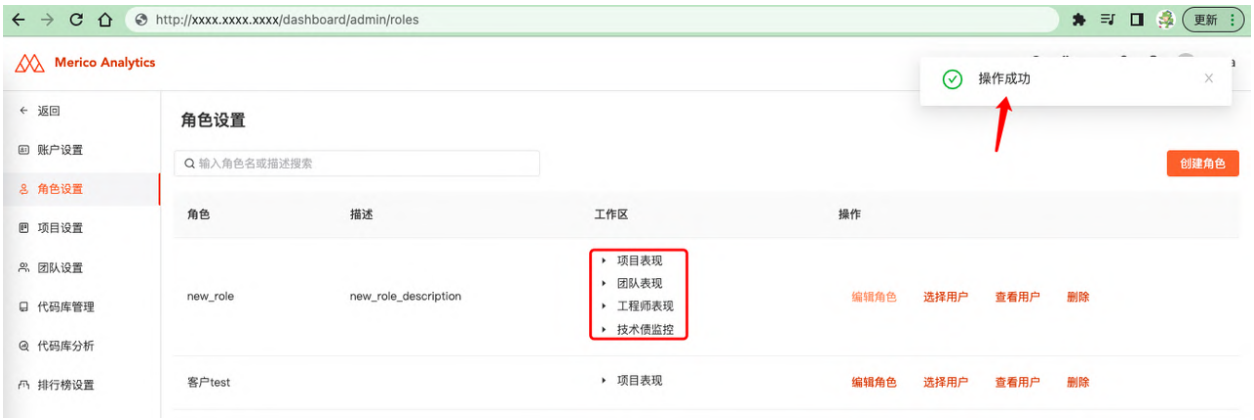

图 97: 编辑角色成功

3.4.3.3.3 选择用户

Step 1：点击某角色的 **选择用户**按钮。

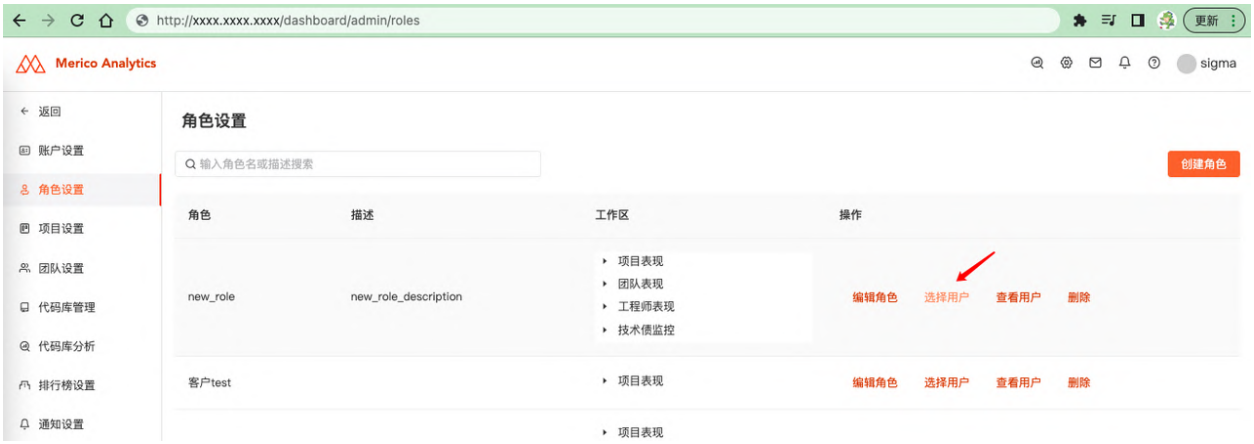

图 98: 角色设置点击选择用户

Step 2：勾选需要选择用户。选择完成后，点击 **确定**按钮保存选择结果。

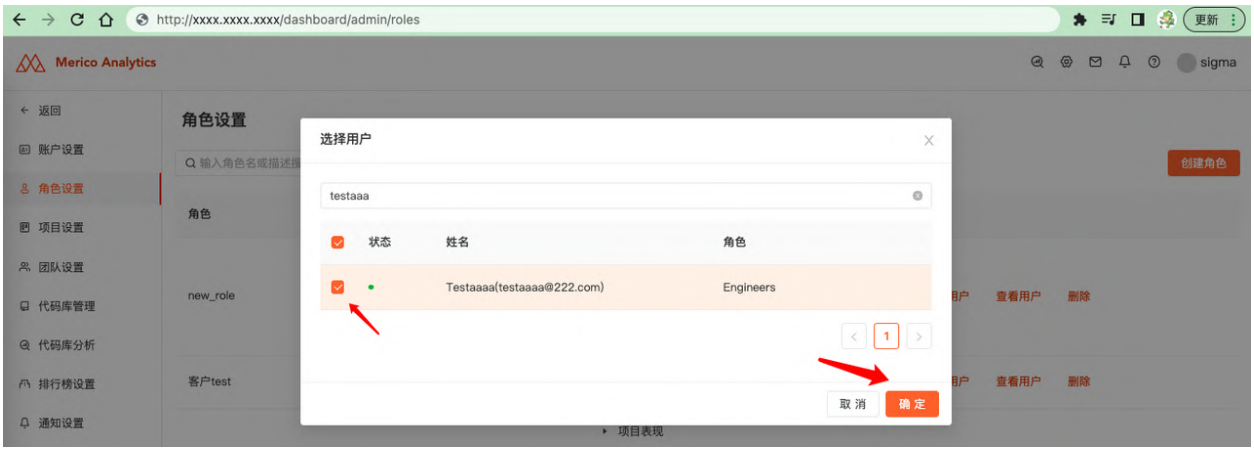

图 99: 选择用户

点击 **确定**后，回到角色设置页面，点击查看用户，可以看到刚刚添加的用户，已在角色用户中

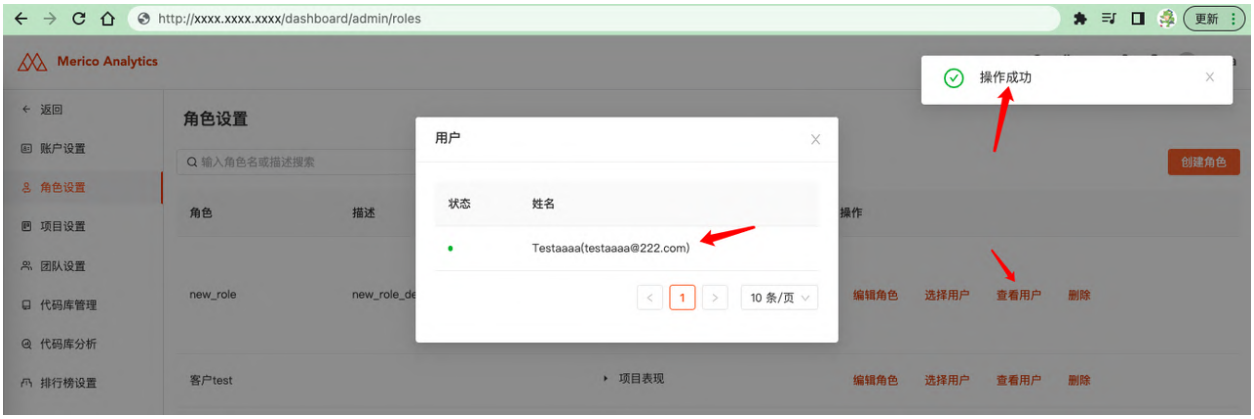

图 100: 操作成功

3.4.3.3.4 删除角色

当某个角色关联了账户时，此角色无法被删除，鼠标放到此角色的 **删除**按钮上时提示 **删除前需要先与关联账户解除关联**。

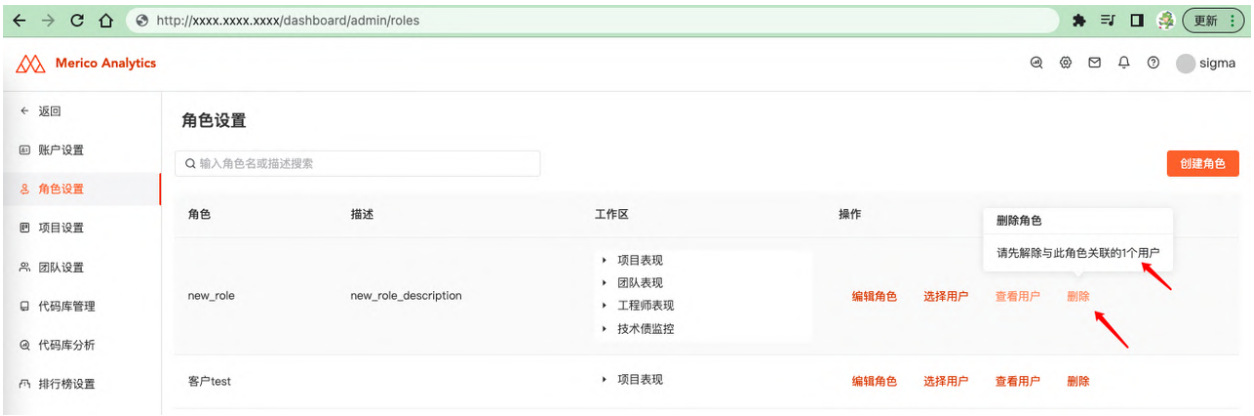

图 101: 删除角色

查看用户处可以查看此角色授予了哪些账户。

选择用户处可以解除被授予此角色的账户与此角色的关联。

解除掉此角色与所有账户的关联后，点击 **确定删除这个角色**按钮，即可删除此角色。

要删除角色，就要在选择用户中，解除这个角色与用户的关联

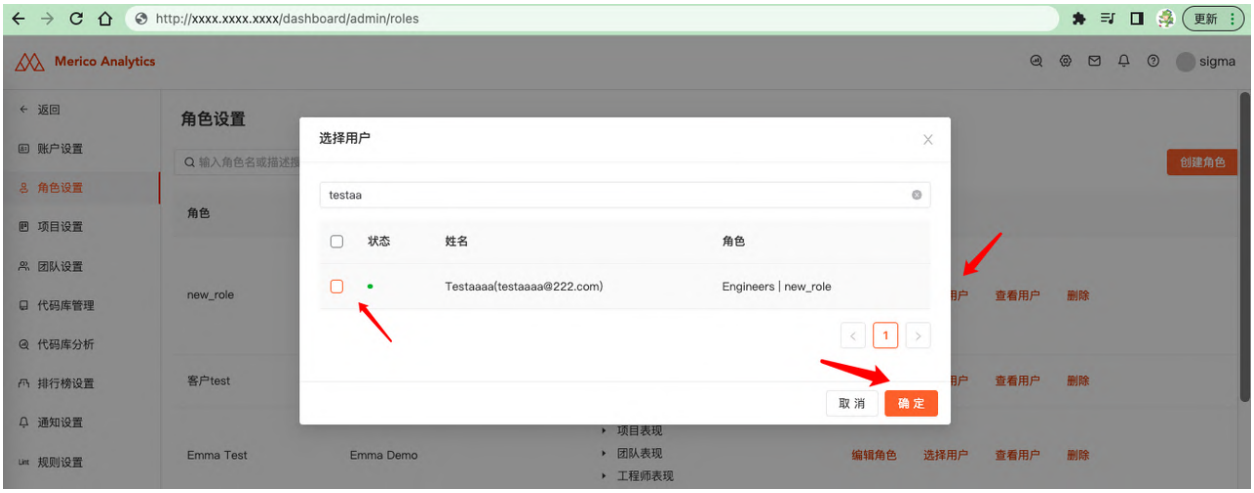

图 102: 选择用户解除关联角色

在保证此角色与所有账户的关联后，就可以删除角色了

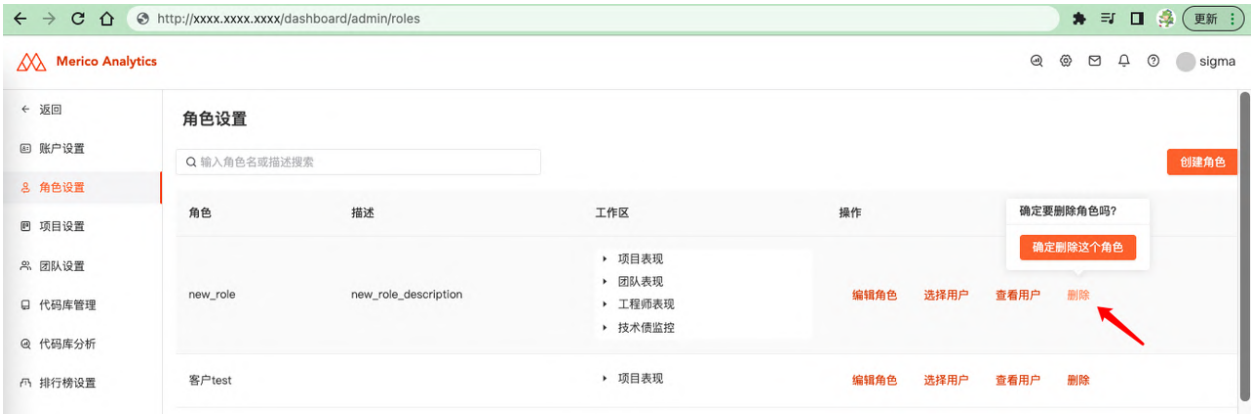

图 103: 可以删除角色

3.4.4 账户设置

3.4.4.1 要点概述

- 需要被分配了 **系统设置**下的 **账户设置**工作区访问权限的角色的账户才能进行账户设置。
- 系统中有一个初始超级管理员账户，为创建 license 时的账户，超级管理员账户拥有所有工作区界面访问权限。
- 一个账户可以归属于**一个或多个**团队，也可以拥有访问**一个或多个**团队/项目数据的权限。
- 账户的 **团队归属**和 **团队数据查看权限**可不一致。
- 一个账户可以设置 **0 个或多个**角色。

- 每个账户都有 **启用/禁用**状态的设置。可以给**启用**了的账户可以设置系统进入权限：**允许登录 / 禁止登录**。
- **账户设置操作**：添加账户 / 修改账户 / 重置账户密码 / 合并账户 / 禁用或启用账户 / 允许登录或禁止登录账户 / 删除账户 / 设置贡献者黑名单。
- **添加成员账户 3 种方式**：1. 直接添加；2. CSV 导入；3. 表格文件导入。
- 可以给一个成员账户设置多个邮箱，此账户的**主邮箱**，用于登录思码逸系统，其他**关联邮箱**产生的分析数据（代码当量，质量数据）都会归于这个成员账户名下。
- == **给某账户设置了相应的角色后，还需要给在【设置-项目设置】处，设置此账户或此账户所在团队可访问的项目（组），此账户可访问的工作区（左侧菜单栏列表）才会正常显示。**==

3.4.4.2 设置导航

菜单 **设置** -> 左侧菜单栏 **账户设置**

图 104: 账户设置

3.4.4.3 账户设置操作

3.4.4.3.1 直接添加账户

Step 1：点击 **添加成员**按钮。

图 105: 添加成员按钮

Step 2：填入此成员账户信息，**姓名，团队和主邮箱必填**。可以给定义了的自定义字段赋值，（归属）团队、角色、项目权限、团队权限都可以选择多个。信息输入完毕后，点击 **确定**按钮保存信息。

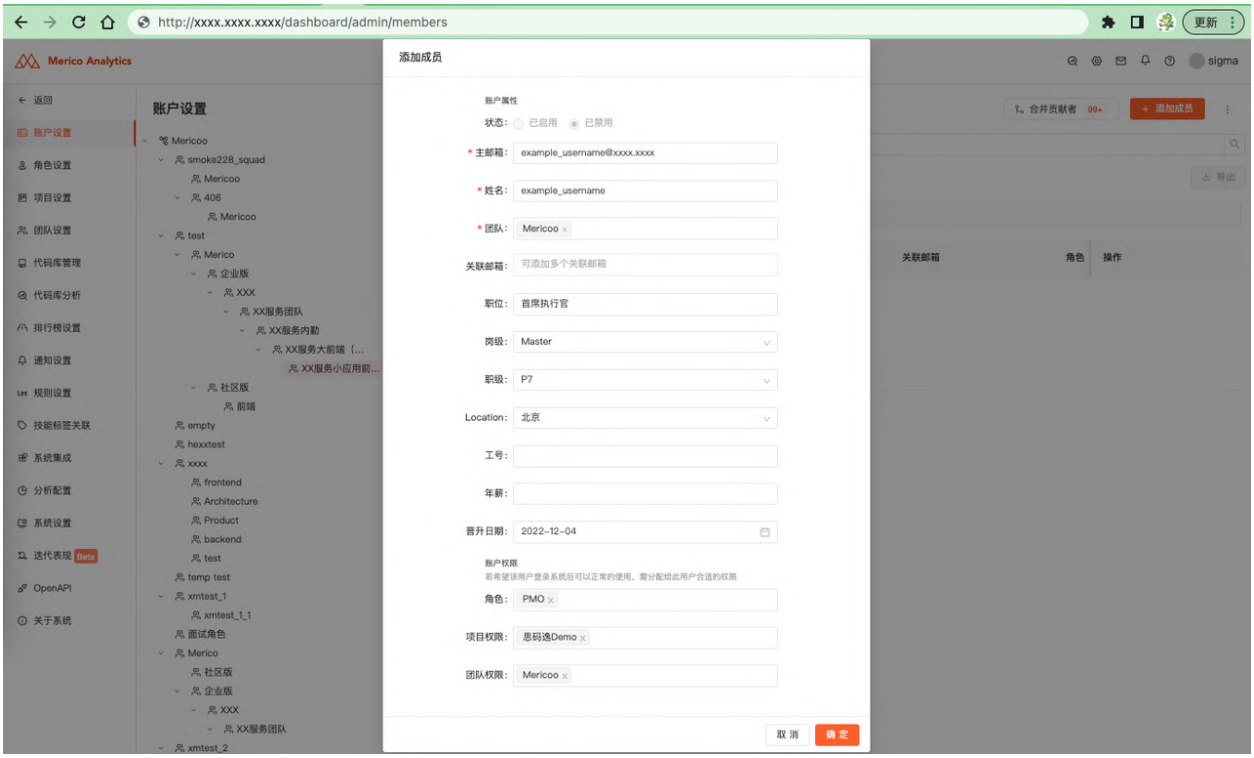

图 106: 添加成员页面

Step3：查看新成员账户已成功创建，初始默认状态为 **已禁用**，需要启用此账户才能登录思码逸系统。

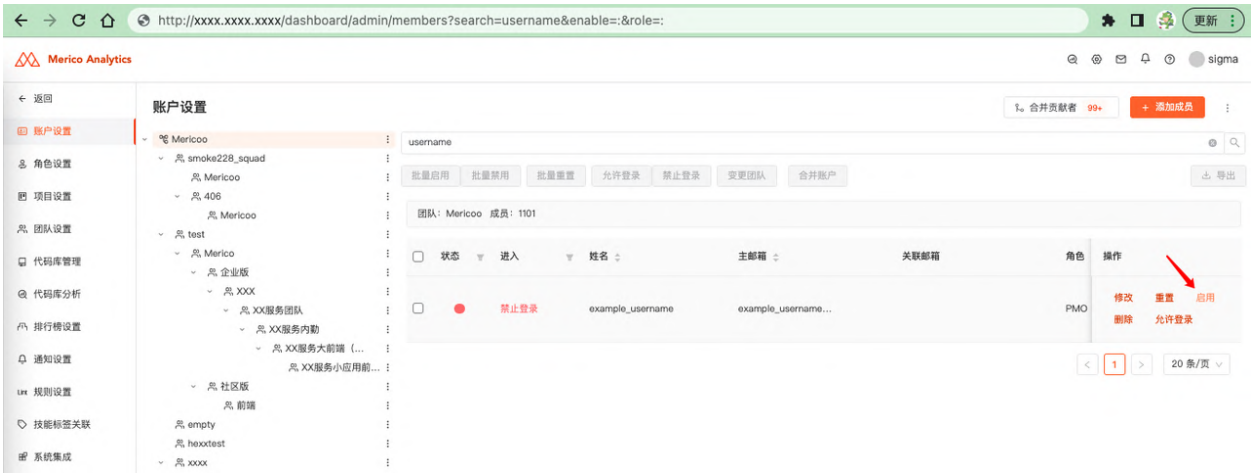

图 107: 启用账户

3.4.4.3.2 批量添加账户

支持两种批量添加账户的方式：

1. CSV 导入
2. 表格文件导入

Step 0： 点击 **添加成员**按钮旁边的 **...** 按钮 -> 下拉菜单中选择 **批量添加成员**。

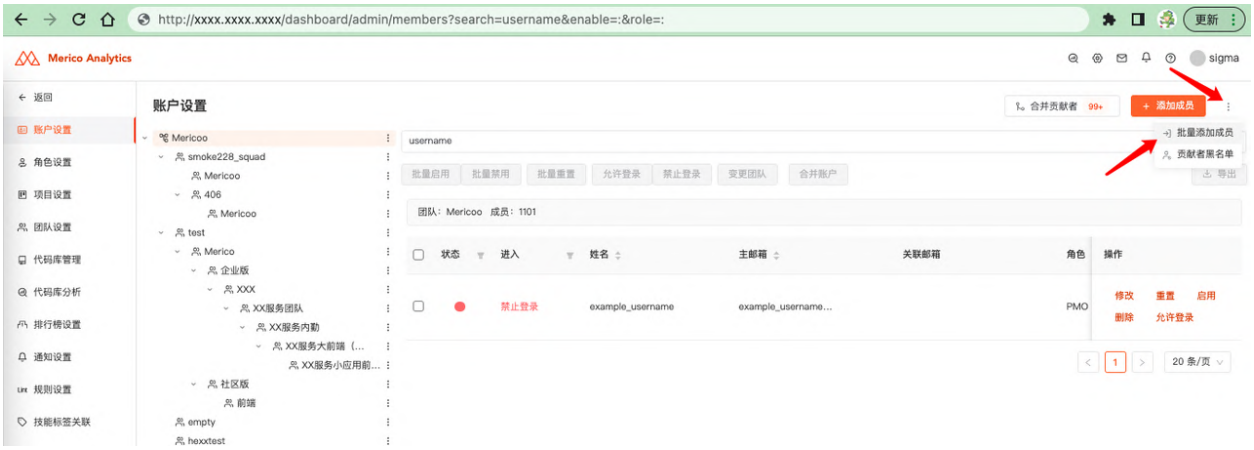

图 108: 点击批量添加成员

从 CSV 导入

Step 1： 点击 **批量添加成员**。

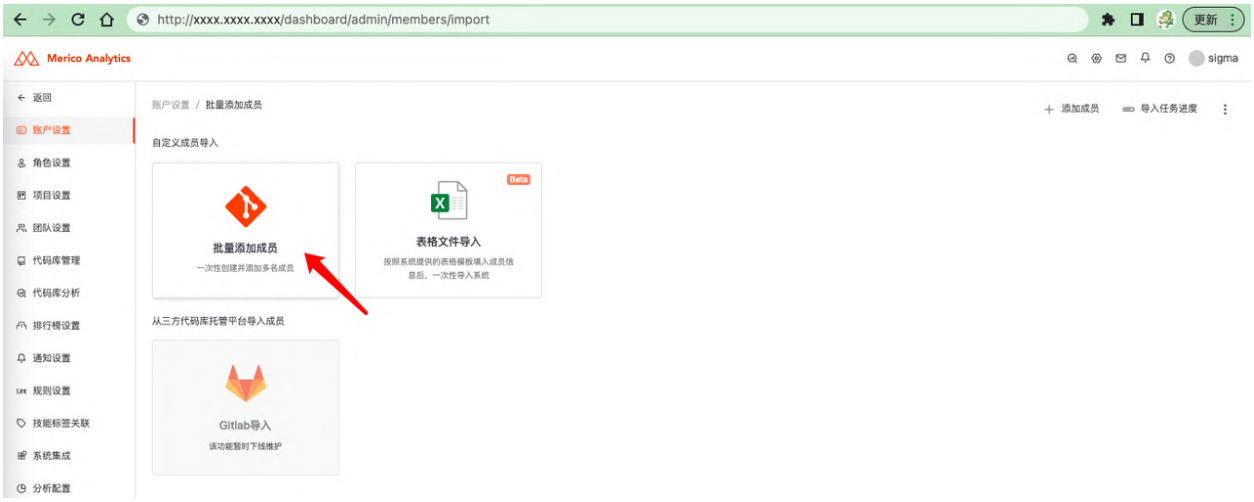

图 109: 点击批量添加成员

Step 2：弹出界面上可以按指引新增成员账户或者按 csv 模板填写信息后上传 csv 文件来添加账户。

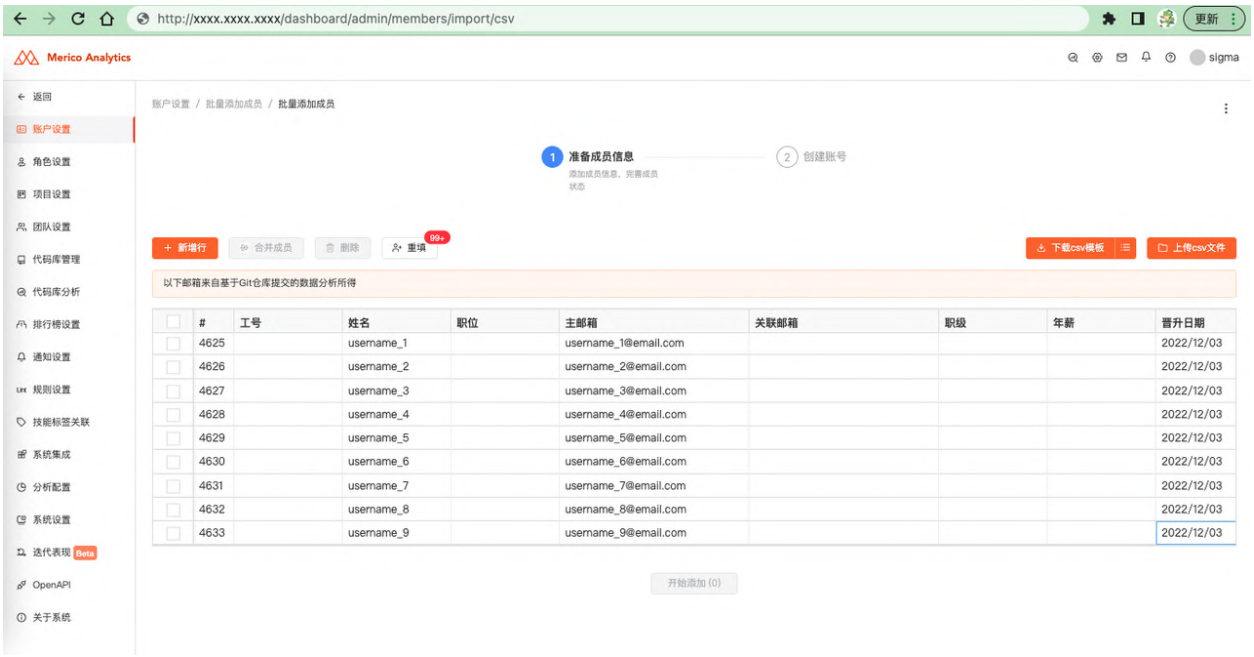

图 110: 准备成员信息

表格文件导入

Step 1：点击 **表格文件导入**。

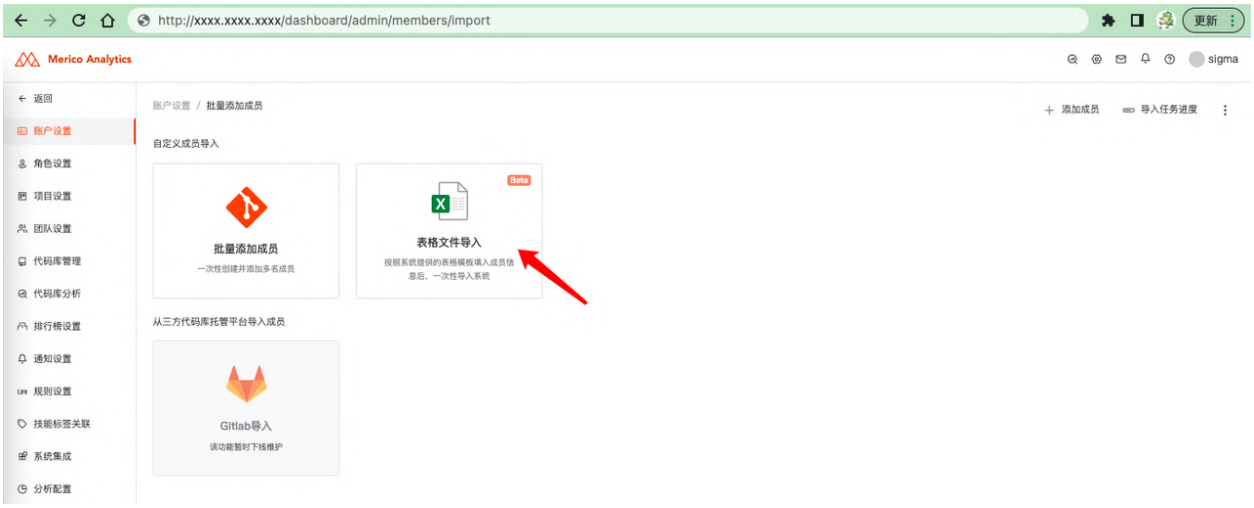

图 111: 点击表格文件导入

Step 2：弹出界面上按指引使用模板表格导入用户信息。

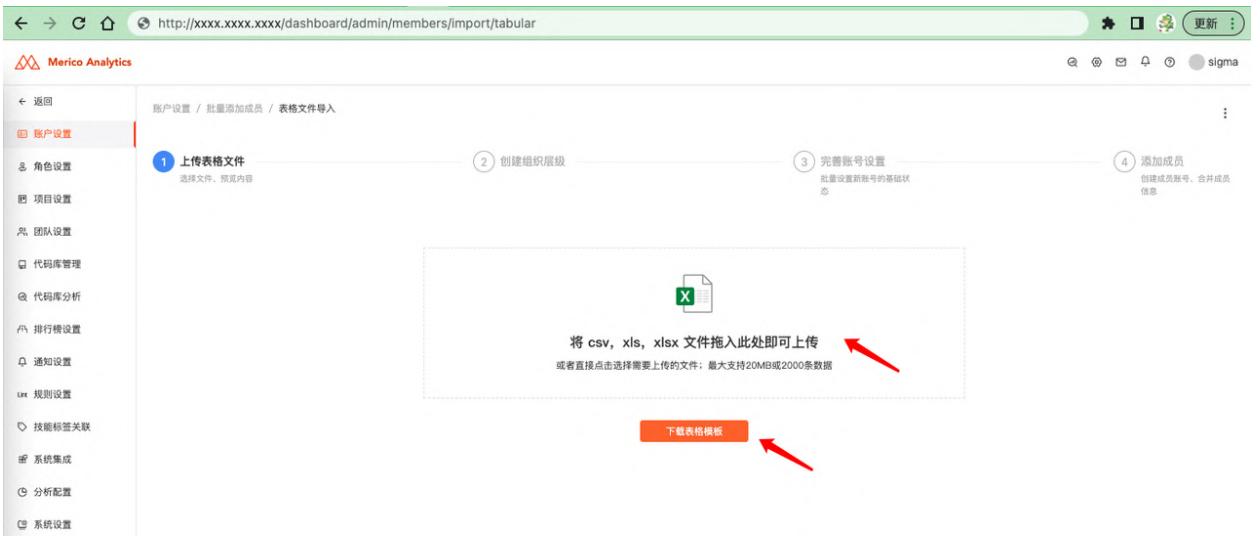

图 112: 下载表格模版

模版内容示例

	A	B	C	D	E	F	G	H	I	J	K
1	姓名	主邮箱	其他邮箱	自定义字段	组织架构层级1	组织架构层级2	组织架构层级3	组织架构层级4	组织架构层级5	组织架构层级6	组织架构层级7
2	XXX	XXX@merico.dev	xxx@qq.com;xxx23@meri.		思码逸	企业版	XXX	XX服务团队	XX服务内勤	XX服务大前端（内勤）	XX服务小应用前端（内勤）
3											
4											
5		登录用邮箱	提交代码时使用的其他邮箱，用英文分号;隔开。	可以根据需要，在这里添加所需的字段。包含多个值时，请用英文分号;隔开。	1.根据具体的组织架构层级自上向下填写；其中「组织架构层级1」为最顶层组织，「组织架构层级7」为最底层组织。其中，顶级团队在同一文件中各行都应该是一致的。如果不同，应新建一个文件记录。 2.成员所属的团队，与实际层级一致即可，不需要保证全部七个层级都填入内容（注意：只能省略后面的层级，前面的层级是必填的，例如如果填写了「组织架构层级3」，则必须填写「组织架构层级1」和「组织架构层级2」）。 3. 如果项目名称包含如下特殊字符（空格、英文半角逗号、英文半角正斜杠、英文半角反斜杠），请使用“-”（短横杠）代替。				
6											

图 113: 模版内容示例

支持中途退出

如果在通过批量导入/csv 导入或表格文件导入时，发现问题，需要退出导入过程

点击页面右上角的 ⋮，可以退出此次导入过程

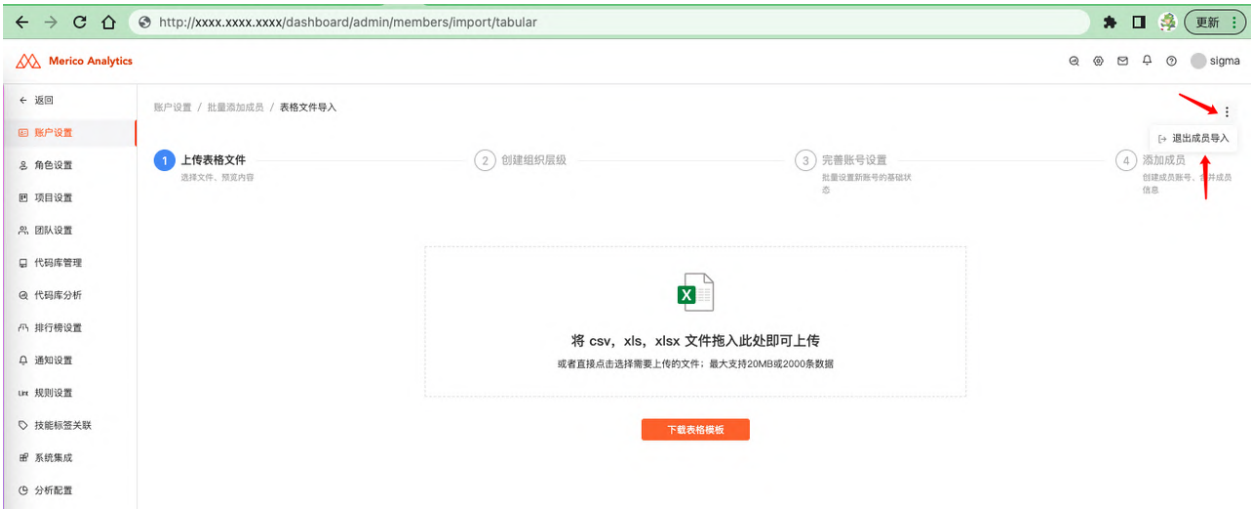

图 114: 支持中途退出

3.4.4.3.3 修改账户信息

Step 1：点击需要修改的账户右侧操作区的 **修改**按钮。

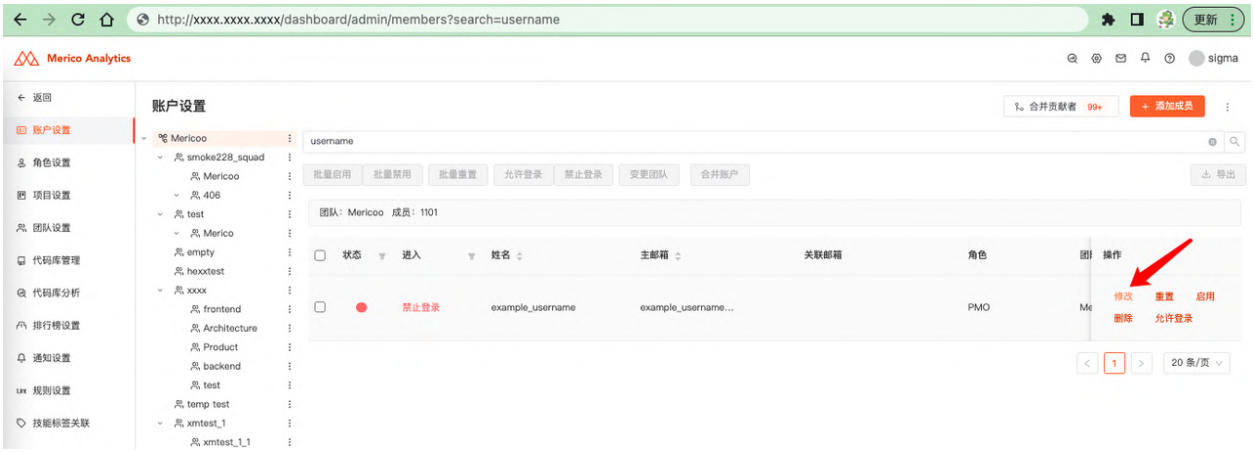

图 115: 修改账户信息

Step 2：修改账户信息，点击 **确定**按钮保存修改信息。

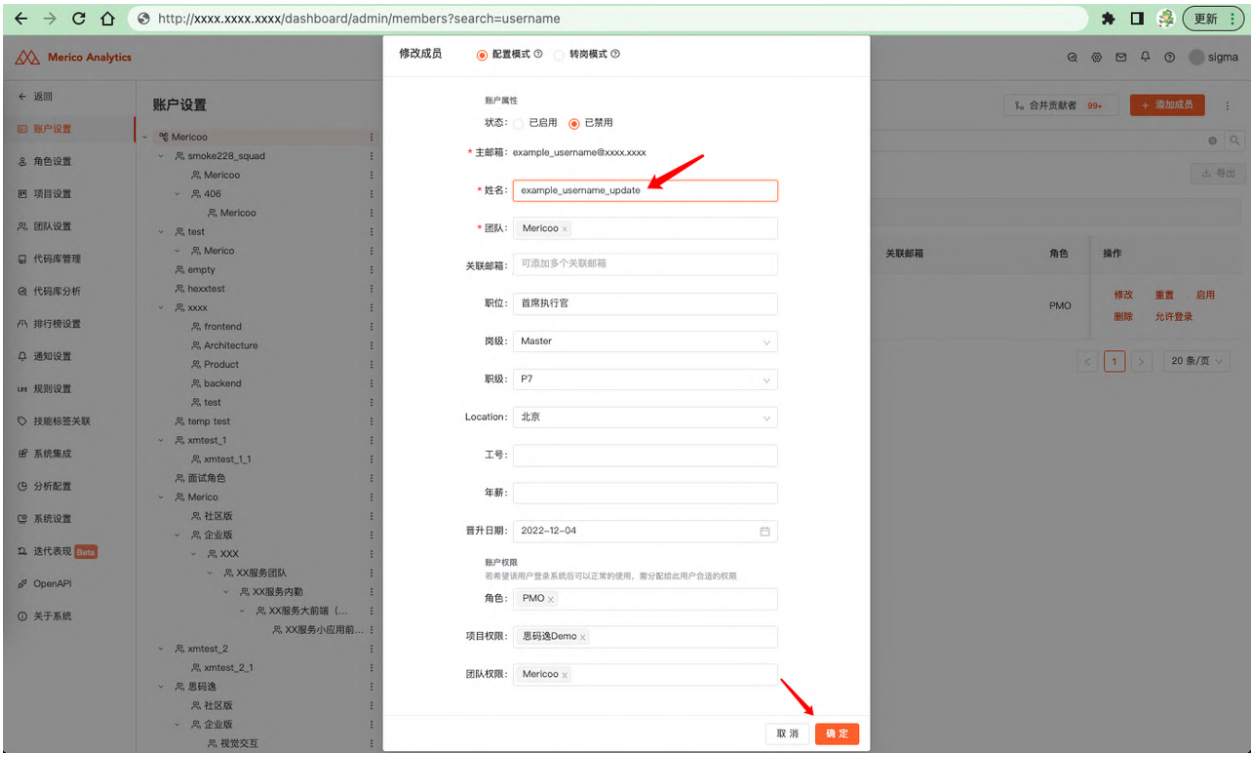

图 116: 修改账户页面

3.4.4.3.4 重置账户密码

如果系统中配置了 **SMTP 发件箱**，新创建的账户密码和重置后的账户密码都会发送至此账户主邮箱即登录邮箱中，不会在系统中明文显示。如果没有配置 SMTP 发件箱，需要在 **账户设置**处设置/修改密码。

配置 SMTP 发件箱时

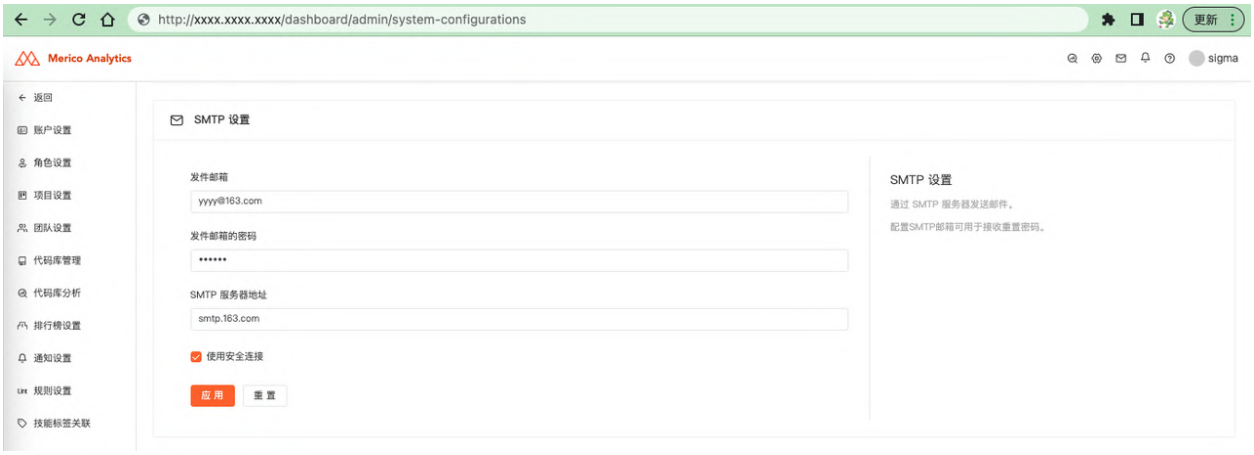

图 117: 配置 SMTP 发件箱时

Step 1：点击需要修改的账户右侧操作区的 **重置**按钮。

Step 2：此示例环境配置了 SMTP 发件箱，在弹出的对话框：确认重置账户的密码吗？中点击 **是**重置后的账户密码会自动发送至此账户主邮箱即登录邮箱中。

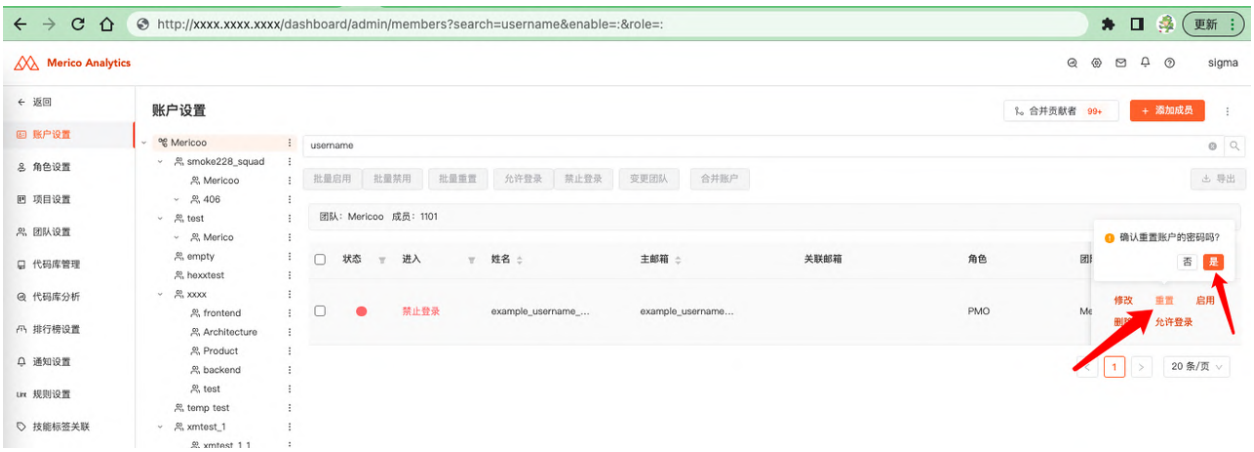

图 118: 重置密码后自动发送邮件

未配置 SMTP 发件箱时

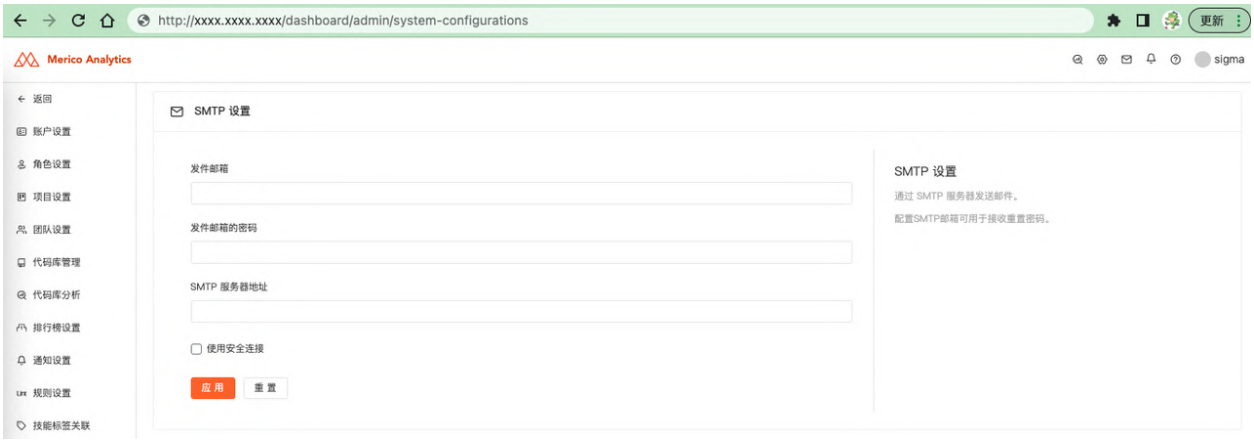

图 119: 未配置 SMTP 发件箱时

Step 1：点击需要修改的账户右侧操作区的 **重置**按钮。

Step 2：此示例环境没有配置 SMTP 发件箱，需要直接设置重置密码，设置好新密码，点击 **确认**按钮保存新密码。

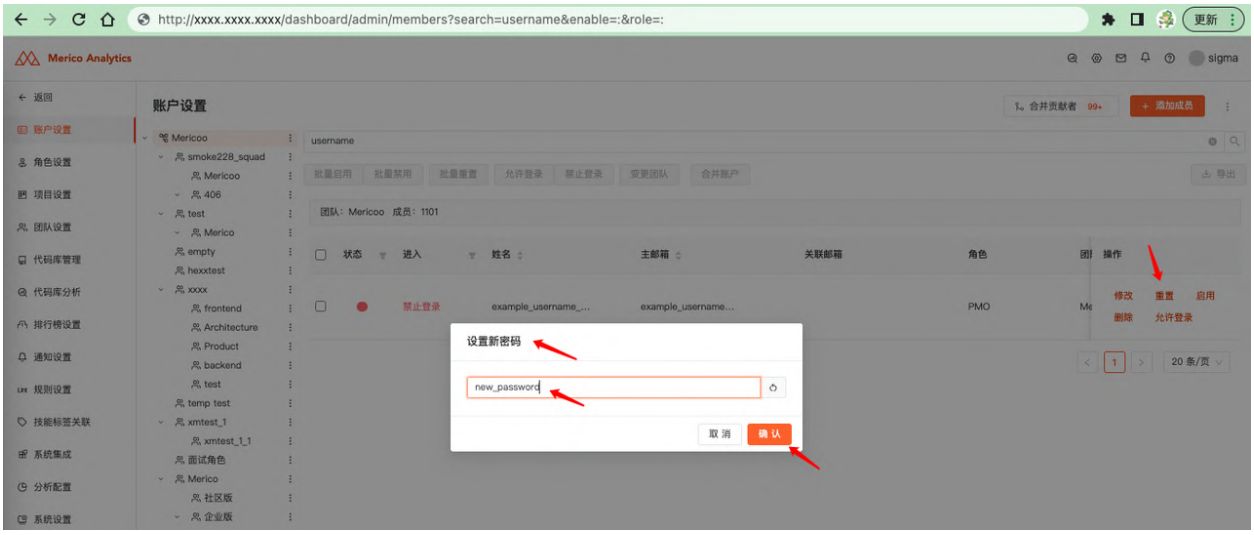

图 120: 重置密码时页面设置新密码

3.4.4.3.5 合并账户

Step 1：勾选需要合并的账户，点击 **合并账户**按钮。

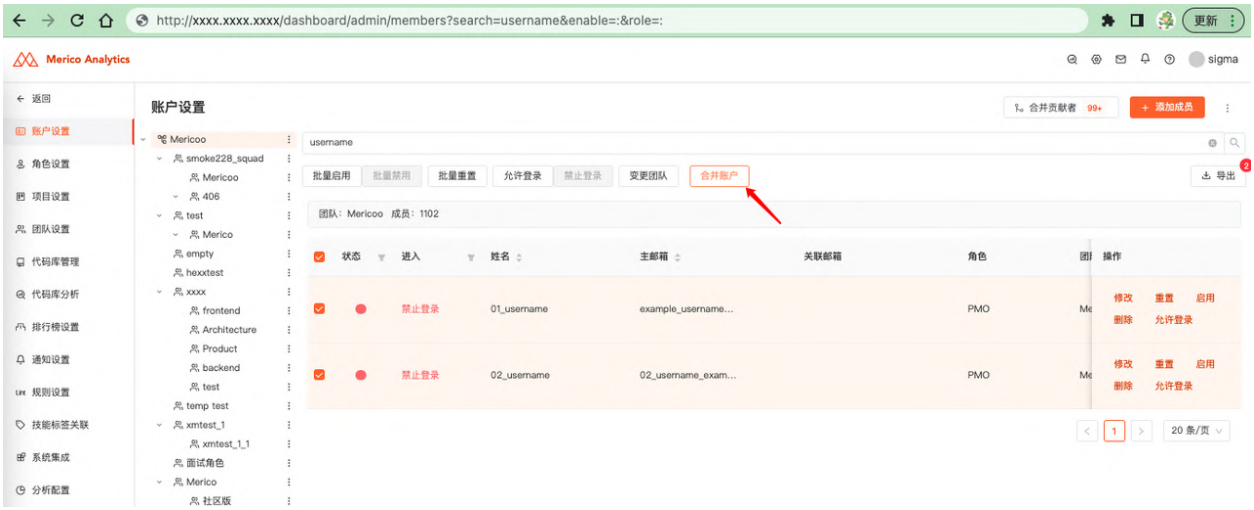

图 121: 合并账户

Step 2：选择主账户作为之后的登录账户，点击 **确定**按钮确认合并。

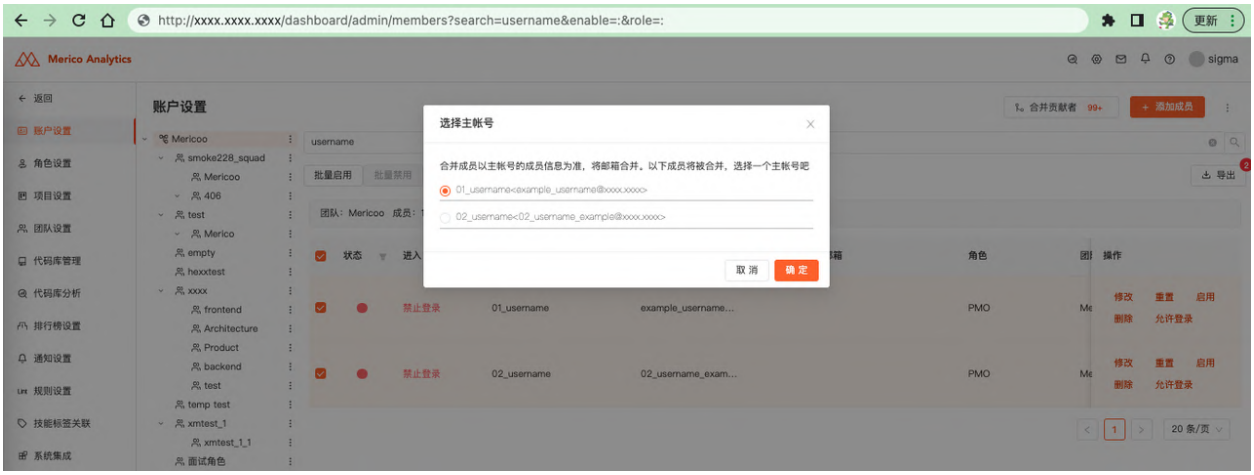

图 122: 选择主账号

合并账户成功

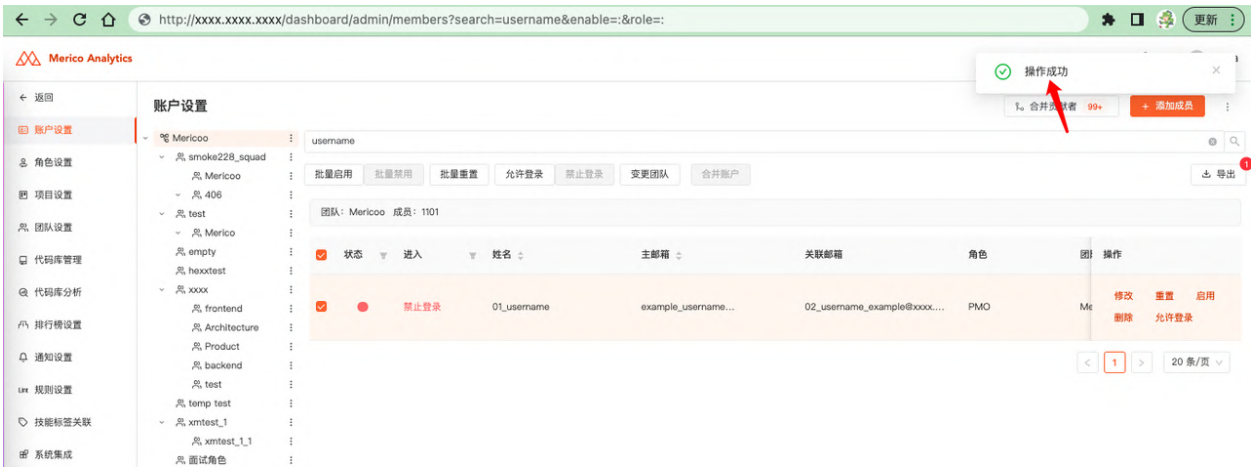

图 123: 合并账户成功

在 **合并贡献者**界面，列出了根据提交信息识别的可能为同一个贡献者的提交邮箱，可以在此界面快速合并贡献者提交邮箱。

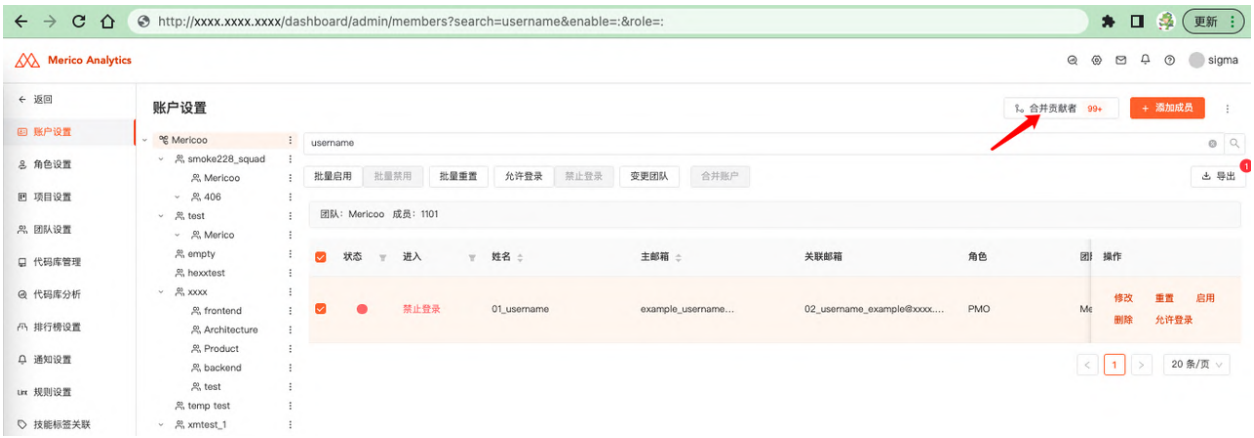

图 124: 合并贡献者

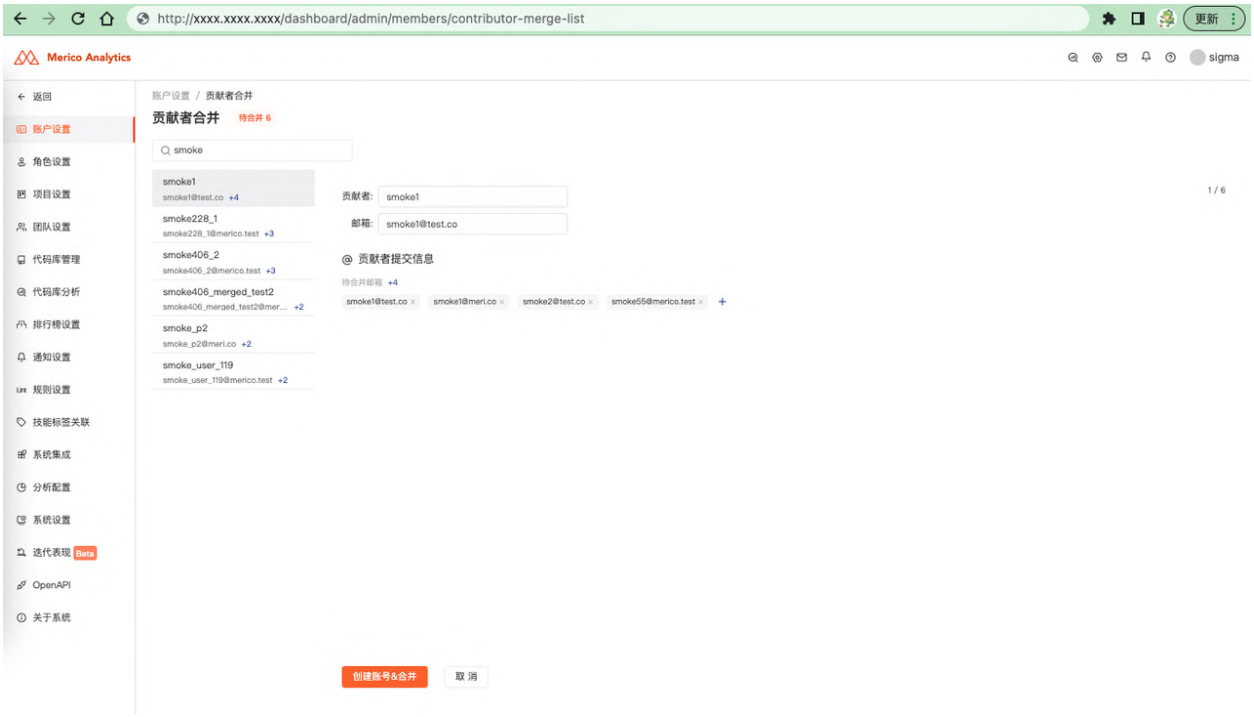

图 125: 贡献者合并页面

3.4.4.3.6 禁用/启用账户

- **启用**：点击 **启用**按钮，启用 **已禁用**状态的成员账户，启用后，此成员账户状态变为 **已启用**，账户状态栏显示为绿色圆球，账户启用后方可设置是否允许登录思码逸系统。
- **禁用**：点击 **禁用**按钮，禁用 **已启用**状态的成员账户，禁用后，此成员账户状态变为 **已禁用**，账户状态栏显示为红色圆球，**不可**登录思码逸系统。

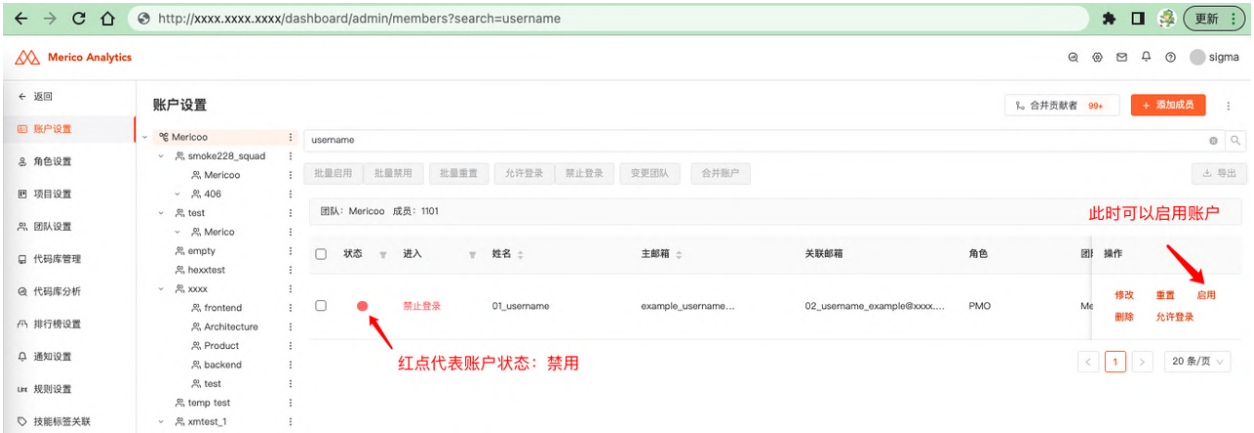

图 126: 启用账户

图 127: 禁用账户

3.4.4.3.7 允许登录/禁止登录

- 点击账户的 **允许登录**或 **禁止登录**按钮来设置此账户是否可登录。

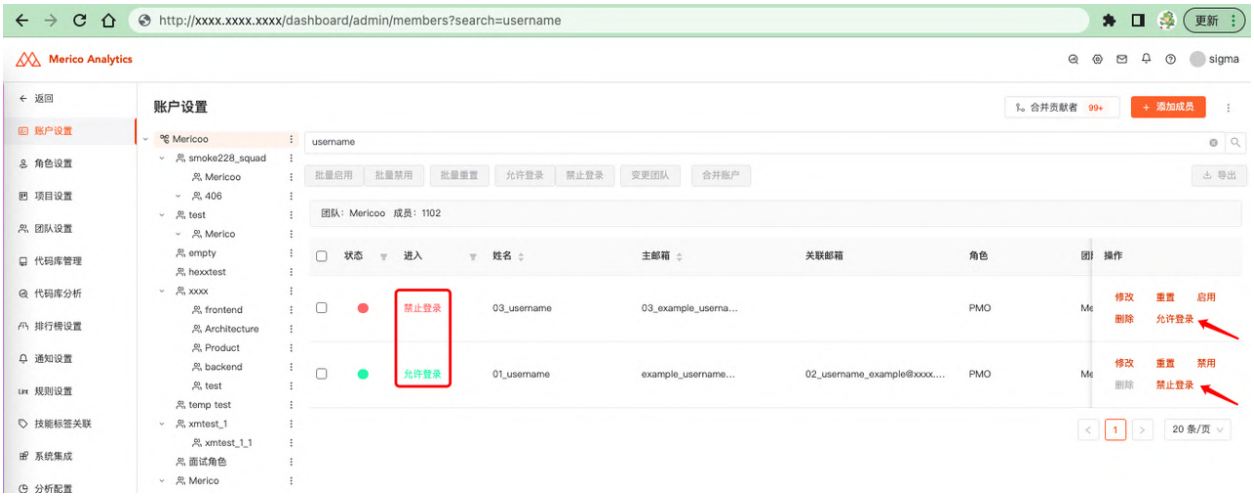

图 128: 允许登录与禁止登录

- 对于启用状态的账户，也可以点击 **修改**按钮，进入账户修改界面，设置此账户是否可登录。

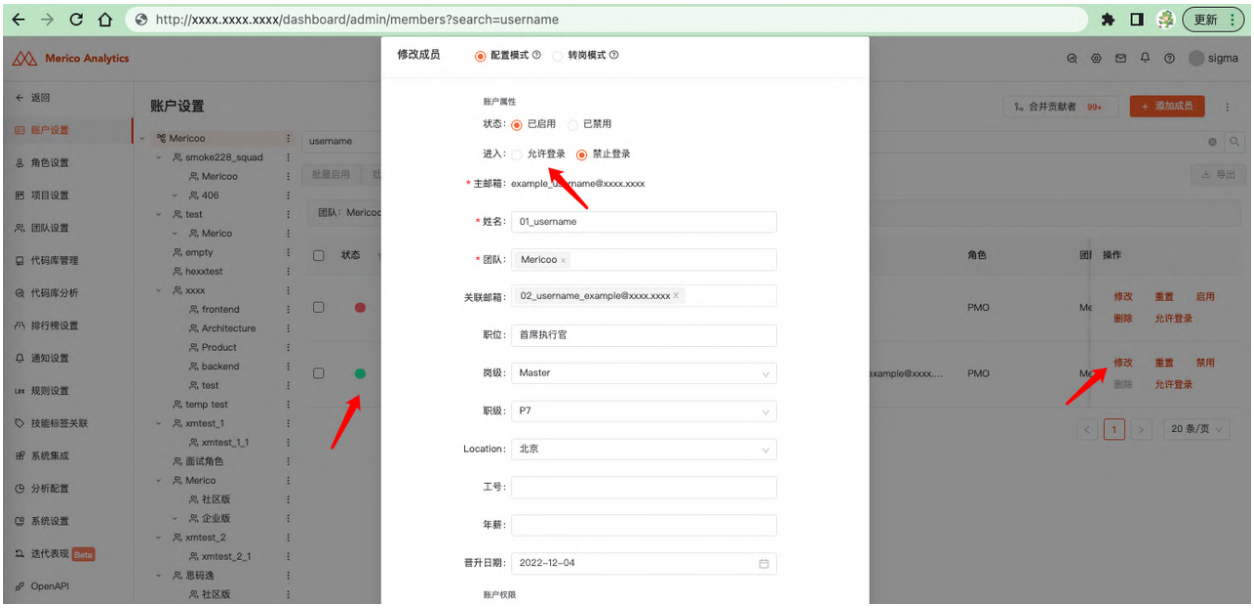

图 129: 页面设置是否可登录

- 可以多选多个账户，**批量**设置多个账户允许登录或者禁止登录。

批量设置多个账户允许登录

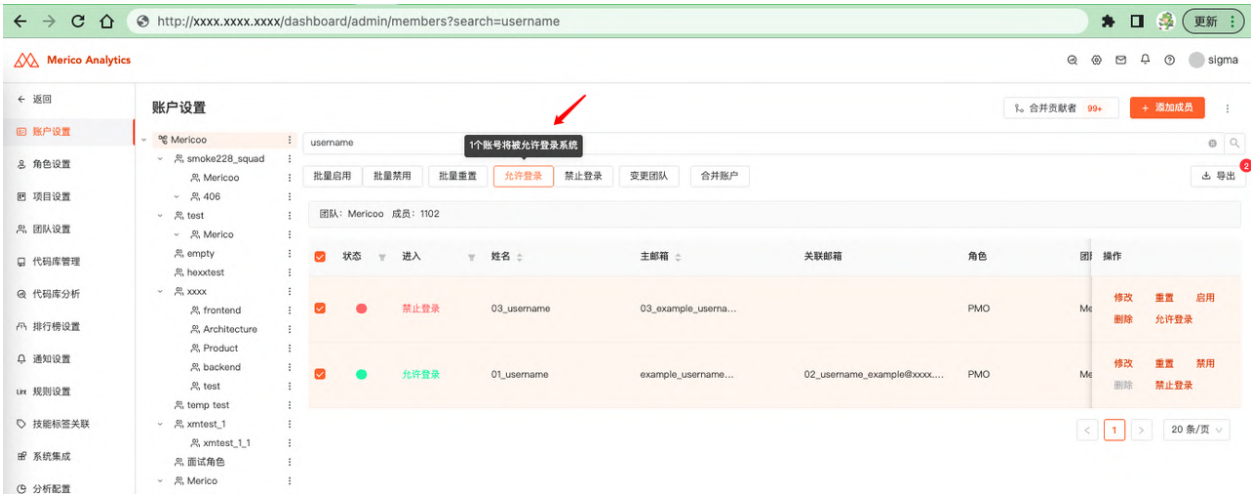

图 130: 批量设置多个账户允许登录

批量设置多个账户禁止登录

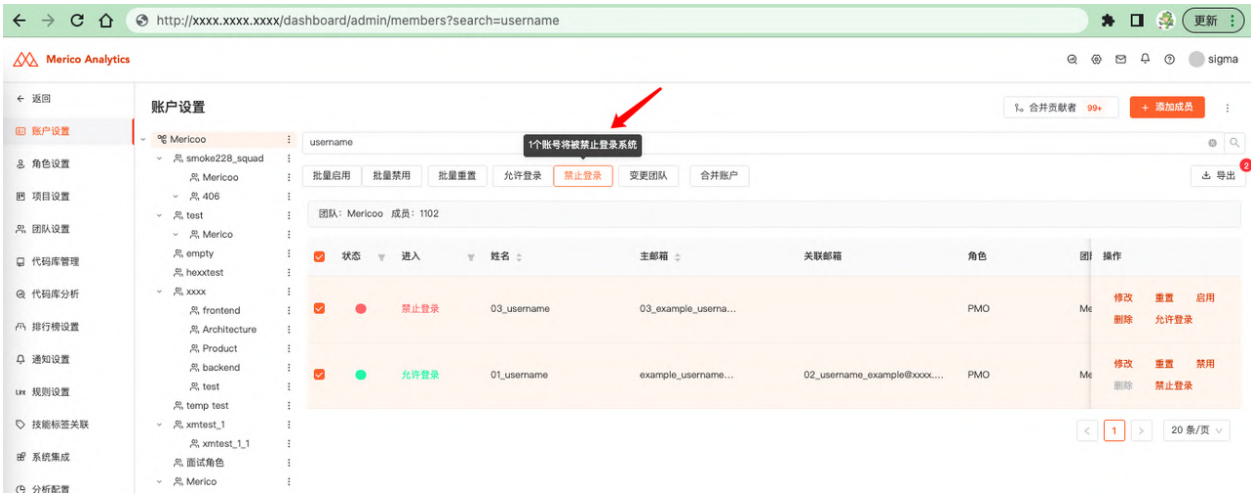

图 131: 批量设置多个账户禁止登录

3.4.4.3.8 删除账户

点击 **删除**按钮删除账户。只有 **已禁用**状态的账户可以被删除。

3.4.4.3.9 贡献者黑名单

将某些邮箱加入贡献者黑名单，来自黑名单中的邮箱的提交跳过分析。

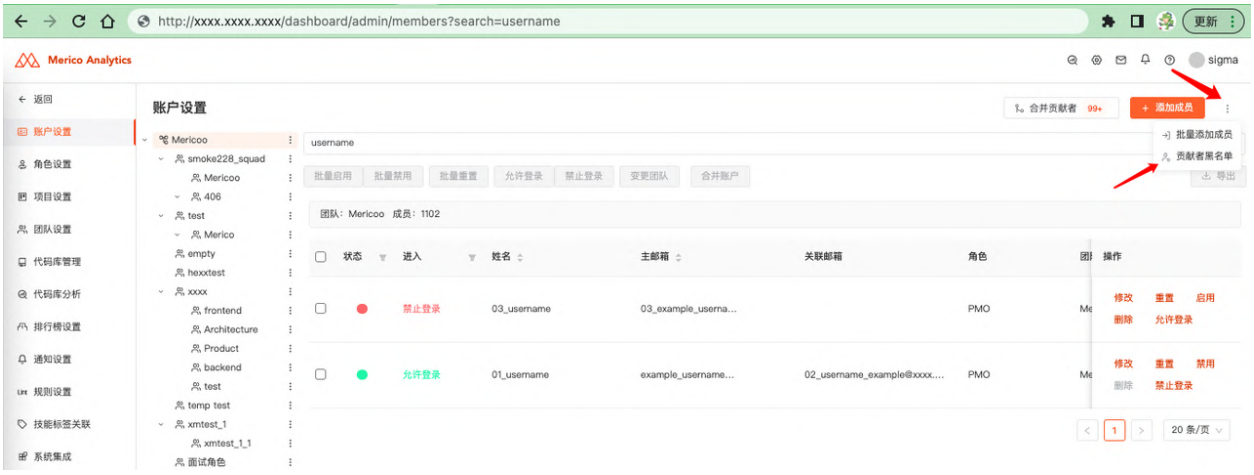

图 132: 进入贡献者黑名单

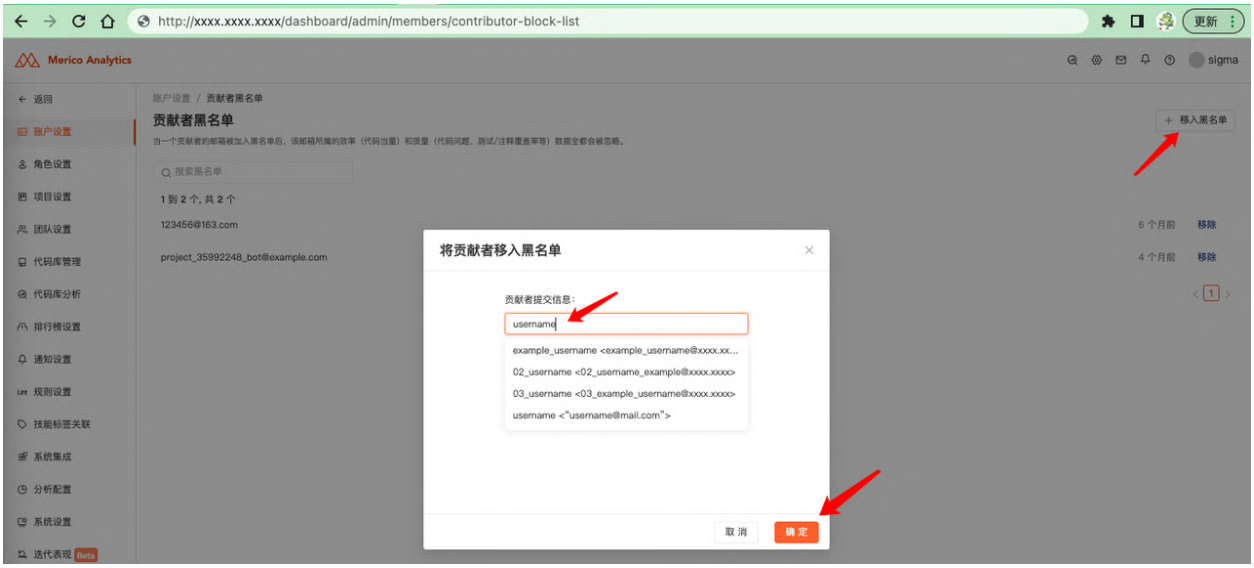

图 133: 设置贡献者黑名单

3.5 第四步：配置项目

3.5.1 导入或配置项目组

3.5.1.1 项目组设置包含三部分

3.5.1.1.1 项目组管理

- 项目组呈树形结构，顶层项目组只有一个。
- 可进行操作：新增（子）项目组 / 编辑项目组 / 移动项目组 / 删除项目组。
- 可以通过 **新增子项目组、从 GitLab 导入项目组、Excel 批量新增代码库时创建项目组**三种方式来新增子项目组。
- 在 **设置 - 项目设置**界面对项目组进行各种设置操作。

3.5.1.1.2 项目组成员管理

- 可以给某个项目组添加/删除可访问用户账户，也可给某个项目组添加/删除可访问的团队，此团队里的全部成员账户都可访问此项目组。
- == **某账户想访问项目表现、团队表现、工程师表现、技术债监控、迭代表现等与代码库分析数据相关的工作区（界面）之前，都需要先给其分配相应的项目组访问权限。**==
- 可进行操作：选择用户 / 选择团队。

3.5.1.1.3 项目组代码库管理

- 一个项目组可包含多个代码库（项目）。
- 可进行操作：新增代码库 / 删除代码库 / 移动代码库。

3.5.2 项目组管理

3.5.2.1 注意事项

- **操作需要权限：**操作账户有以下两种权限皆可：
  1. 思码逸系统 **超级管理员**权限。
  2. 有进入 **项目设置**界面的角色权限，并拥有此项目组或更高层级项目组的访问权限。
- **可进行操作**：新增子项目组 / 编辑项目组 / 移动项目组 / 删除项目组 / 移动代码库 / 删除代码库 / 选择用户 / 选择团队。
- 可以通过 **新增子项目组、从 GitLab 导入项目组、Excel 批量新增代码库时创建项目组**三种方式来新增子项目组。通过 GitLab 导入项目组操作请参考：**系统集成方式导入代码库**，通过 Excel 批量新增代码库时创建项目组请参考: **表格文件导入代码库**。
- 项目组呈树形结构，顶层项目组**只有一个**。唯一顶级父项目组在系统第一次部署时即创建。

3.5.2.2 设置导航

菜单 **设置** -> 左侧菜单栏 **项目设置**

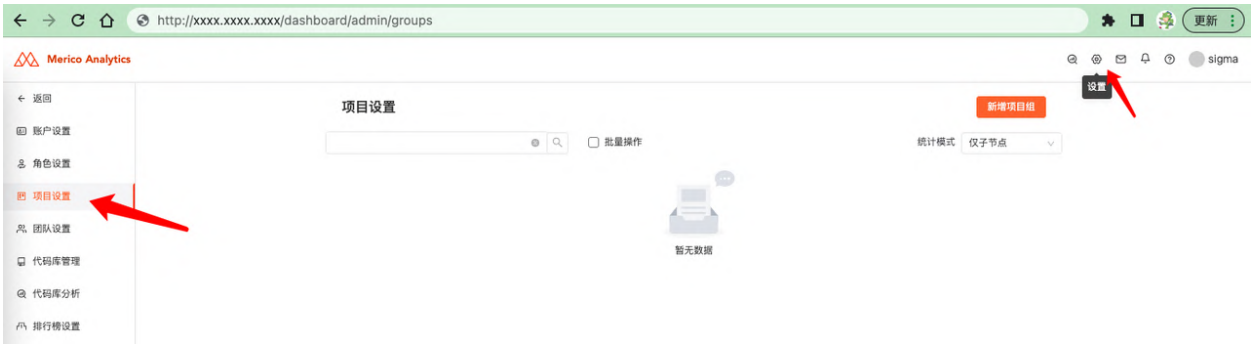

图 134: 项目设置

3.5.2.3 项目组管理操作

Step 0：点击某一项目组右侧 **更多**下拉菜单。

图 135: 项目组管理操作

3.5.2.3.1 新增子项目组

- 点击 **更多**下拉菜单中的 **新增子项目组**，进入 **添加项目**界面。输入新增子项目组名称、描述，点击 **确定**按钮。弹出 **操作成功**提示，在该父级项目组下找到此新增子项目组。
- 也可通过点击右上角 **新增项目组**按钮，直接快速在顶级父项目组下新增子项目组。

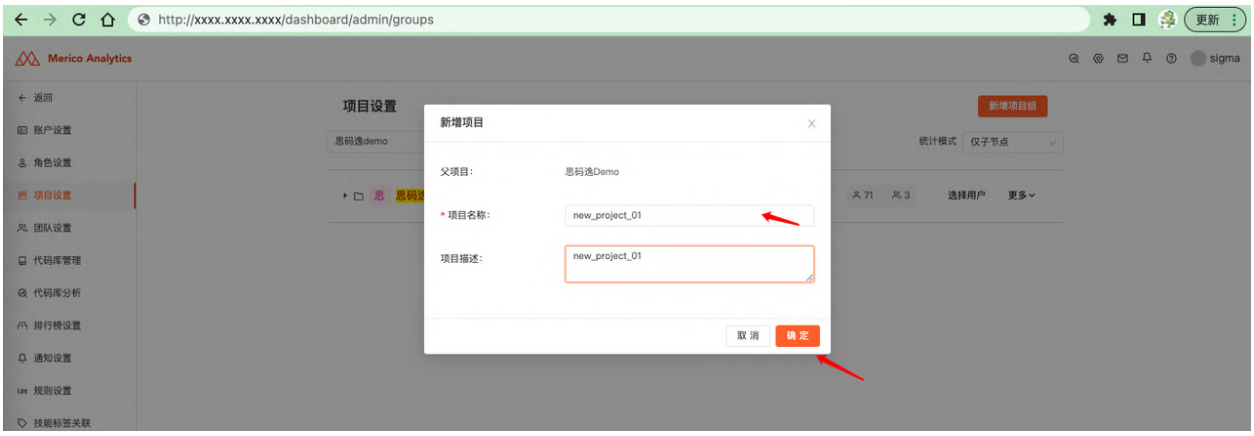

图 136: 新增子项目组

3.5.2.3.2 编辑项目组

点击 **更多**下拉菜单中的 **编辑项目组**，进入 **修改项目**界面，修改项目名称、描述，点击 **确定**按钮。

弹出 **操作成功**提示，发现项目组名称和描述修改成功。

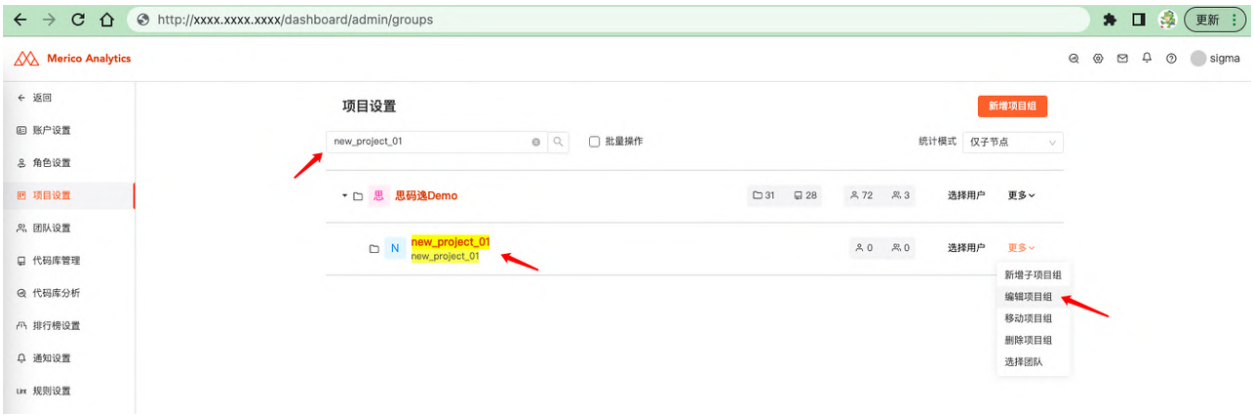

图 137: 进入编辑项目组

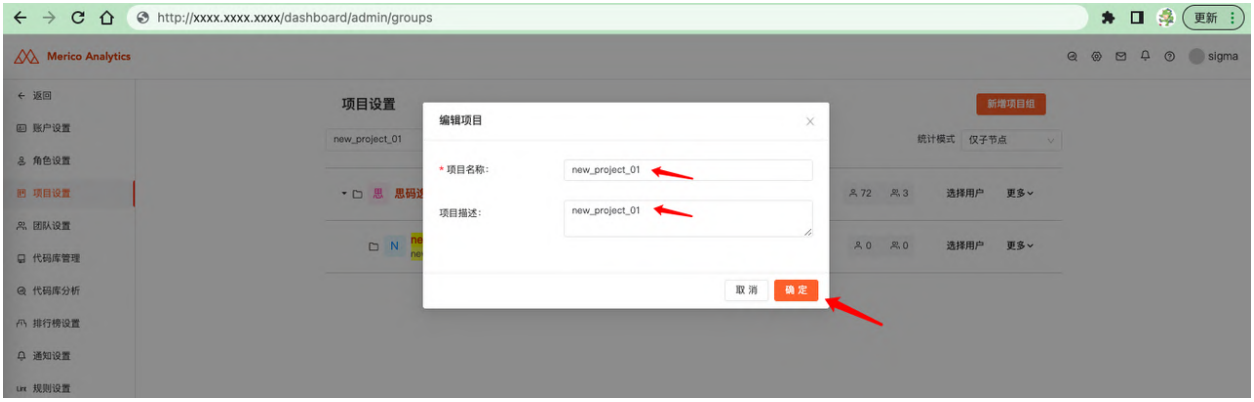

图 138: 编辑项目组

3.5.2.3.3 移动项目组

点击 **更多**下拉菜单中的 **移动项目组**，弹出 **移动代码库和项目组**界面，选择要移动至的项目组，点击 **确定**按钮。

弹出 **操作成功**提示，发现项目组移动成功。

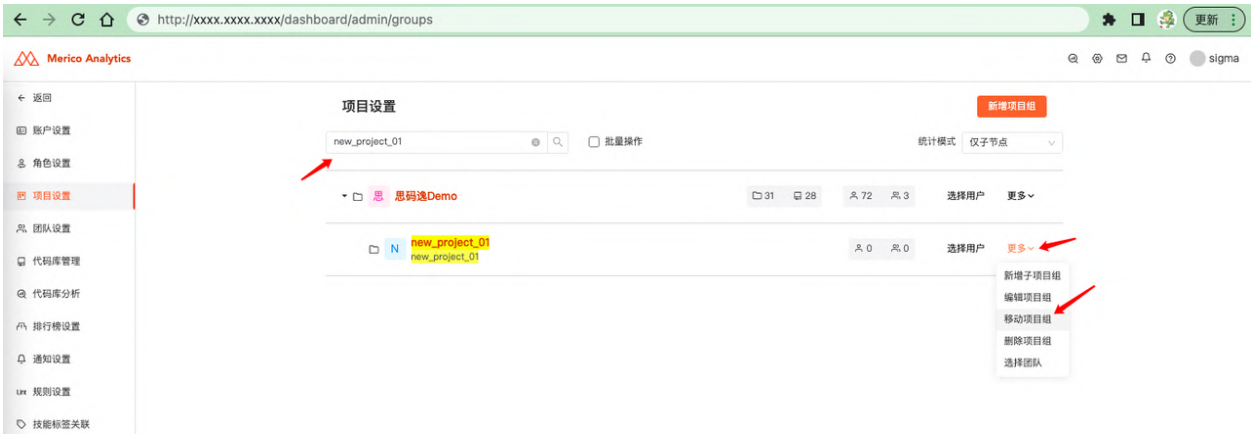

图 139: 进入移动项目组

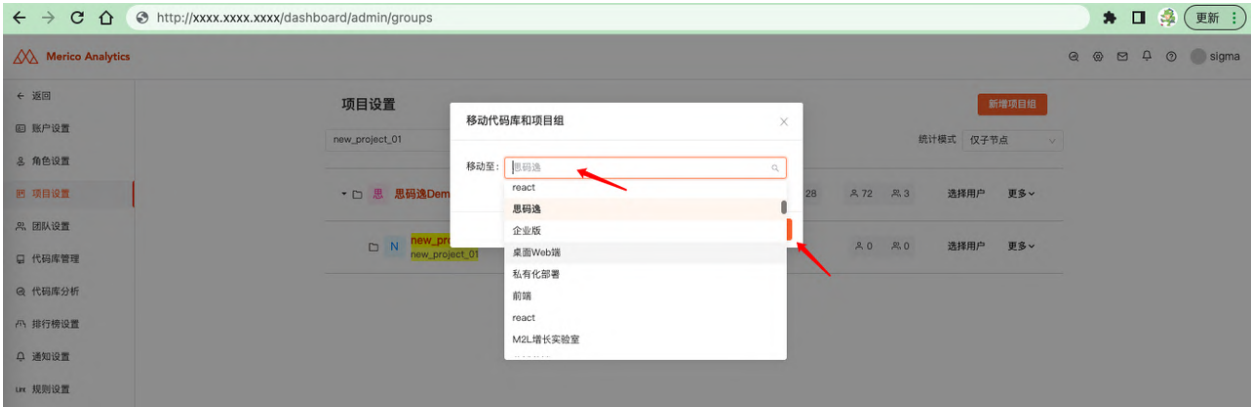

图 140: 移动项目组

3.5.2.3.4 删除项目组

点击 **更多**下拉菜单中的 **删除项目组**，弹出 **确认删除项目组或代码库**界面，选择要删除的项目组，输入项目组名进行确认，点击 **确定**按钮。

弹出 **操作成功**提示，发现项目组删除成功。

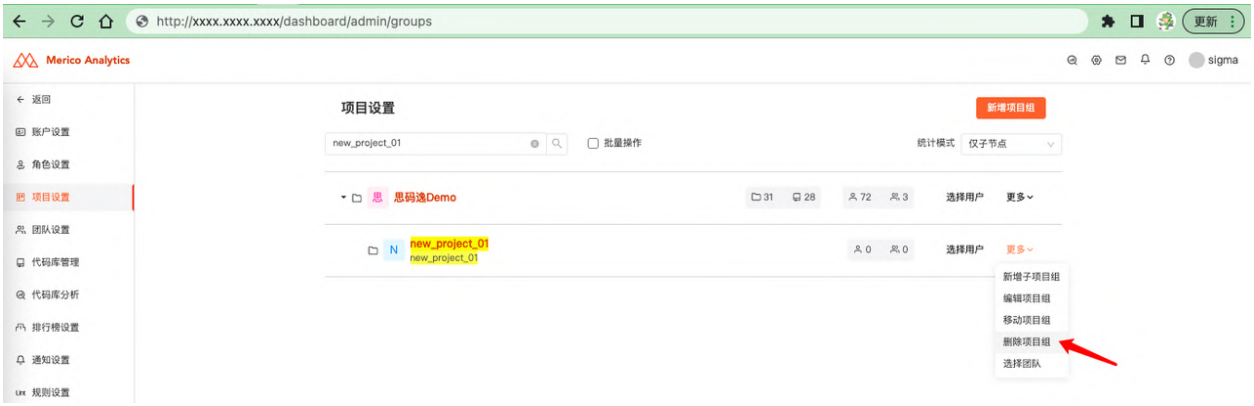

图 141: 进入删除项目组

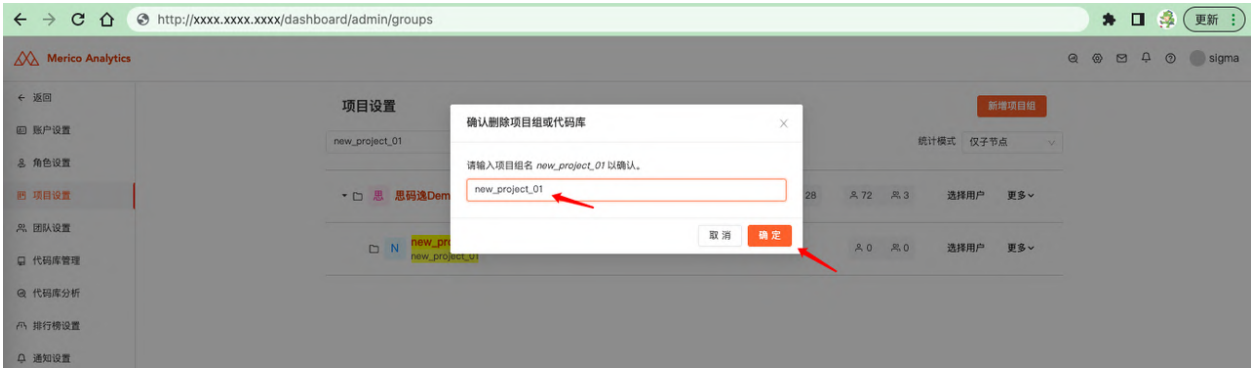

图 142: 删除项目组

3.5.2.3.5 批量操作-移动/删除

勾选 **批量操作**，多选数个 **项目组/代码库**，可批量进行 **移动/删除**操作。

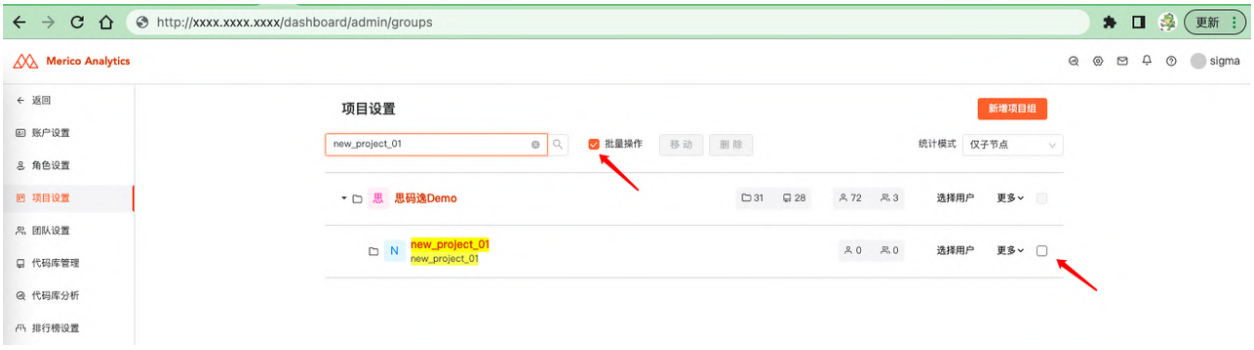

图 143: 点击批量操作

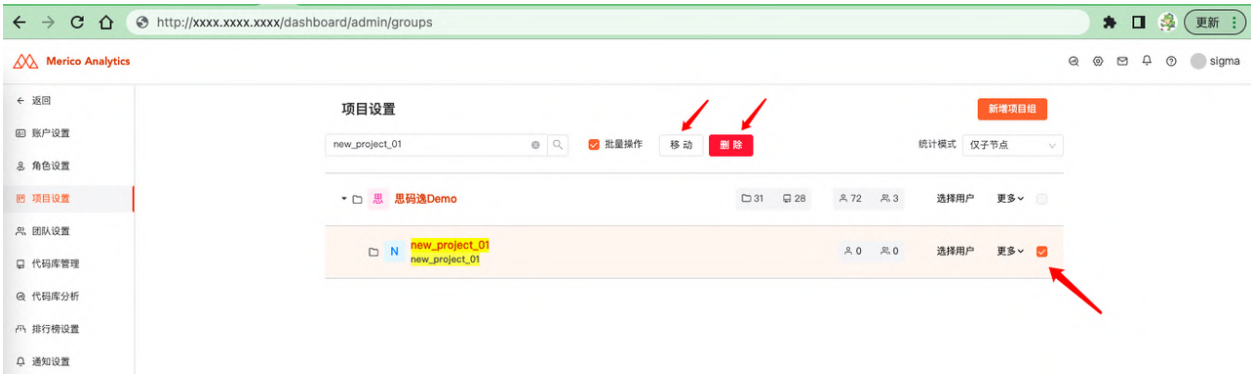

图 144: 批量操作项目

3.5.3 项目组用户 _ 团队管理

3.5.3.1 注意事项

- **操作需要权限：**操作账户有以下两种权限皆可：
  1. 思码逸系统 **超级管理员**权限。
  2. 有进入 **项目设置**界面的角色权限，并拥有此项目组或更高层级项目组的访问权限。
- 添加项目组用户/团队之前需要先给相应成员创建系统账号或者创建相应的团队。
- 可进行操作：选择用户 / 选择团队。
- == **被添加为某项目组用户后，该账号才有此项目组相关数据的访问权限。某账户想访问项目表现、团队表现、工程师表现、技术债监控、迭代表现等与代码库分析数据相关的工作区（界面）之前，都需要先给其分配相应的项目组访问权限。** ==

3.5.3.2 设置导航

菜单 **设置** -> 左侧菜单栏 **项目设置**

图 145: 项目设置

3.5.3.3 项目组用户/团队管理操作

3.5.3.3.1 添加项目组用户

点击 **选择用户**链接，搜索需要选择的人员，此人员需在账户设置里已完成相关设置并启用，勾选并点击 **确定**按钮保存设置。

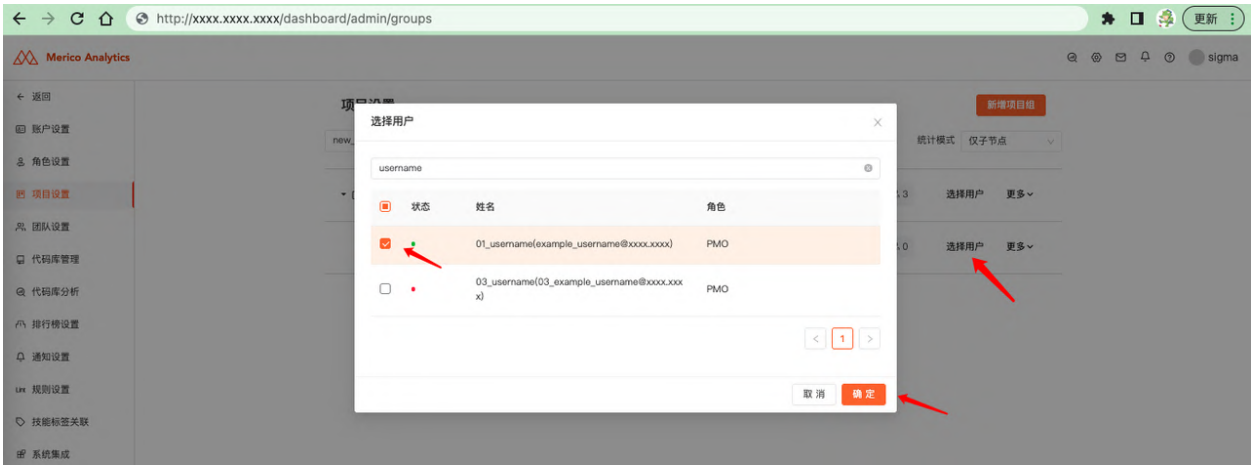

图 146: 选择用户

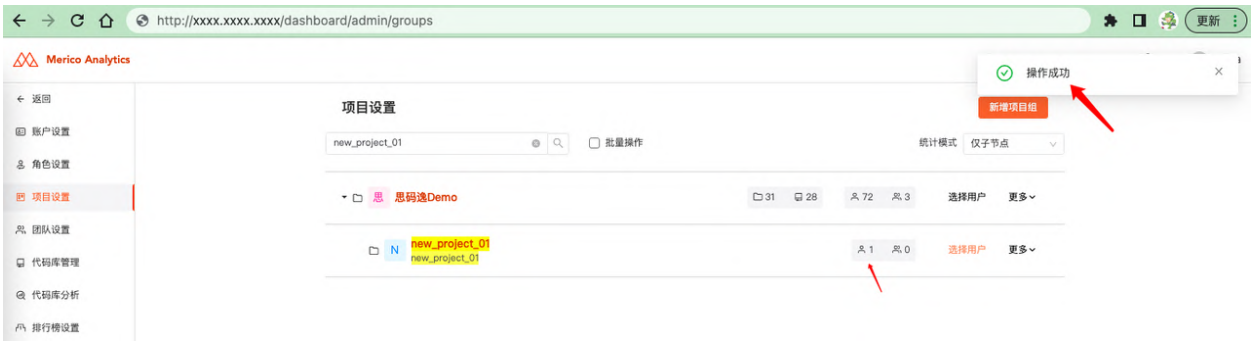

图 147: 添加项目组用户成功

3.5.3.3.2 移除项目组用户

去掉上面选择用户的勾选后并保存，即为将该用户从此项目组移除。

3.5.3.3.3 添加项目组团队

在需要添加团队的项目组右侧 **更多**下拉菜单中点击 **选择团队**，弹出的团队列表中勾选需要添加的团队，点击 **确认**按钮保存。

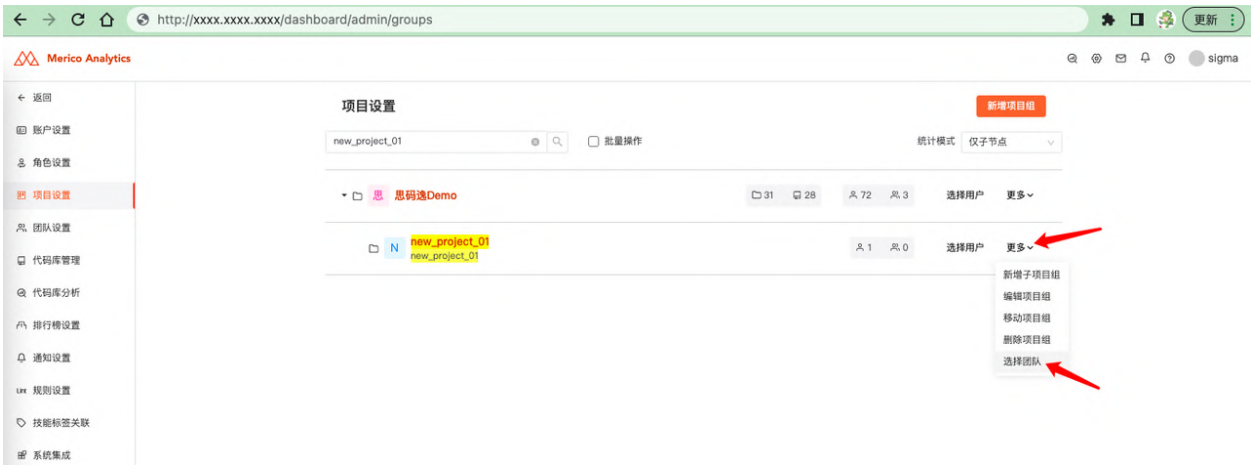

图 148: 添加项目组团队

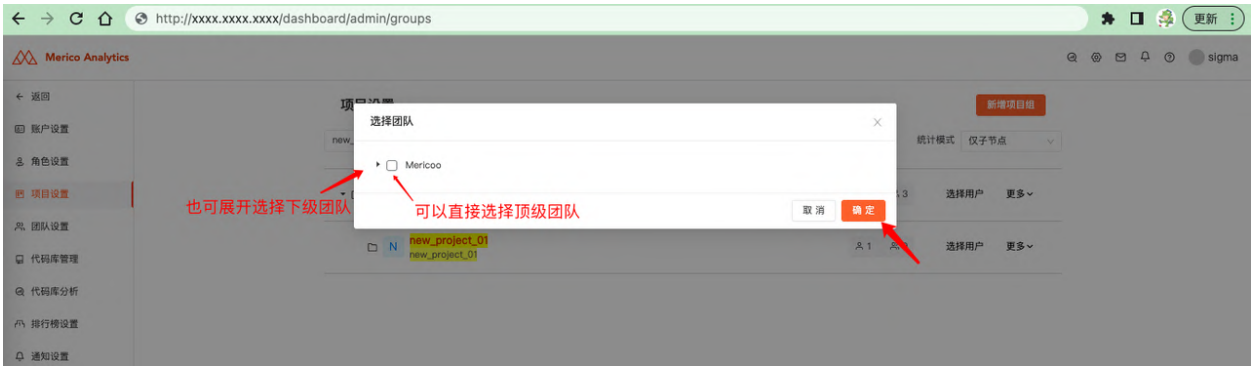

图 149: 选择团队

3.5.3.3.4 移除项目组团队

去掉上面选择团队的勾选后并点击 **确认**按钮保存，即为将该团队从此项目组移除。

3.5.4 项目组代码库管理

3.5.4.1 注意事项

- **操作需要权限：**操作账户有以下两种权限皆可：
    1. 思码逸系统 **超级管理员**权限。
    2. 有进入 **项目设置**界面的角色权限，并拥有此项目组或更高层级项目组的访问权限。
- 可进行操作：新增代码库 / 删除代码库 / 移动代码库。

3.5.4.2 设置导航

菜单 **设置** -> 左侧菜单栏 **项目设置**

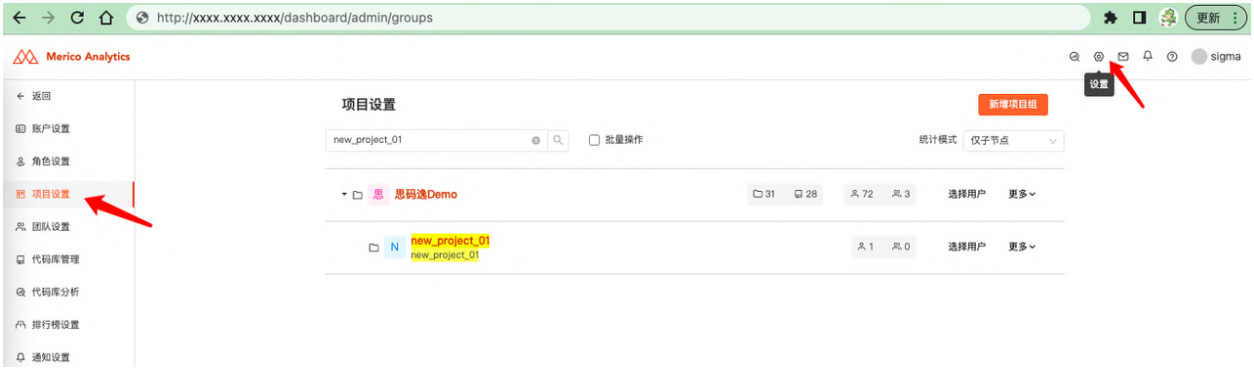

图 150: 项目设置

3.5.4.3 项目组代码库管理操作

3.5.4.3.1 新增代码库

参考：==**从 Git 地址新增代码库**，**从 GitLab 导入代码库**，**表格文件导入代码库**==。

3.5.4.3.2 删除代码库

- 方法一：**设置 - 项目设置 - 删除代码库**
- 方法二：**设置 - 代码库设置 - 删除代码库**

方法一示例：

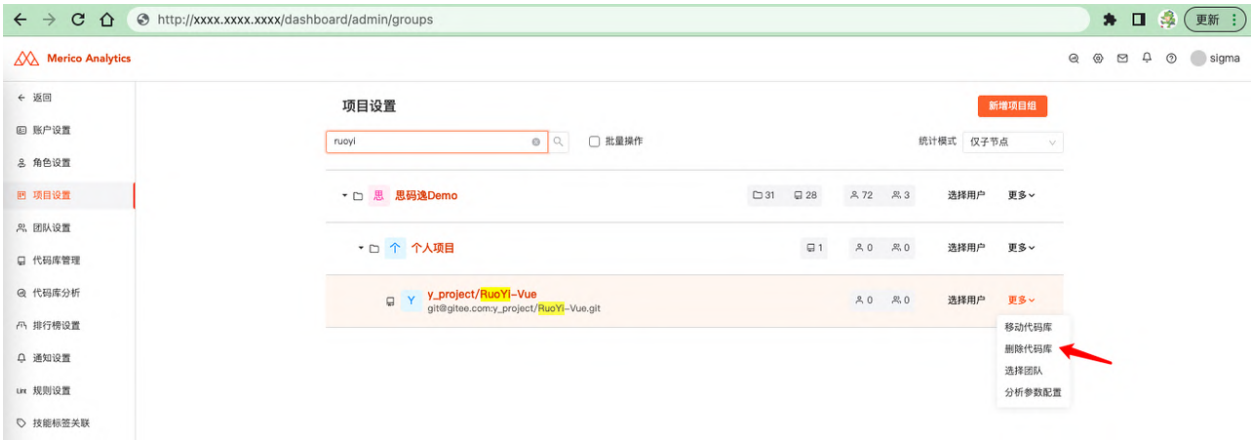

图 151: 设置 - 项目设置 - 删除代码库

方法二示例：

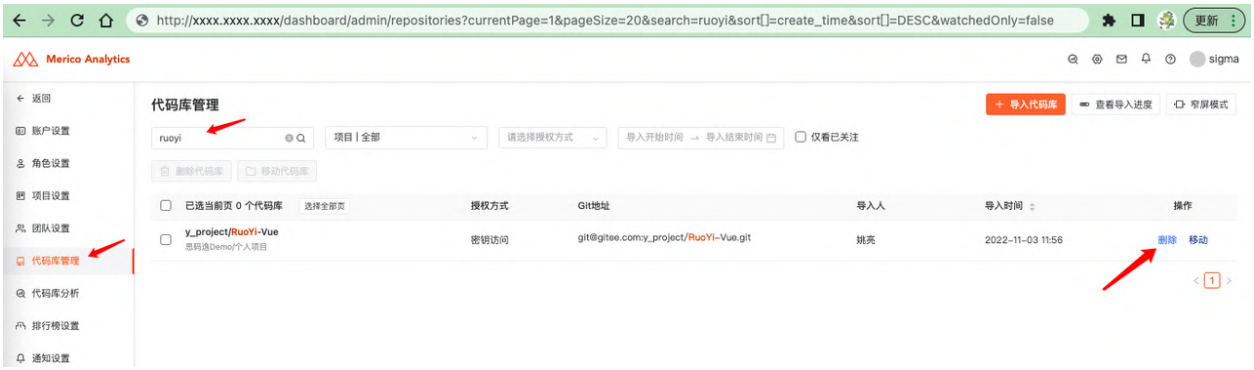

图 152: 设置 - 代码库设置 - 删除代码库

3.5.4.3.3 移动代码库

- 方法一：**设置 - 项目设置 - 移动代码库**
- 方法二：**设置 - 代码库设置 - 移动代码库**

方法一示例：

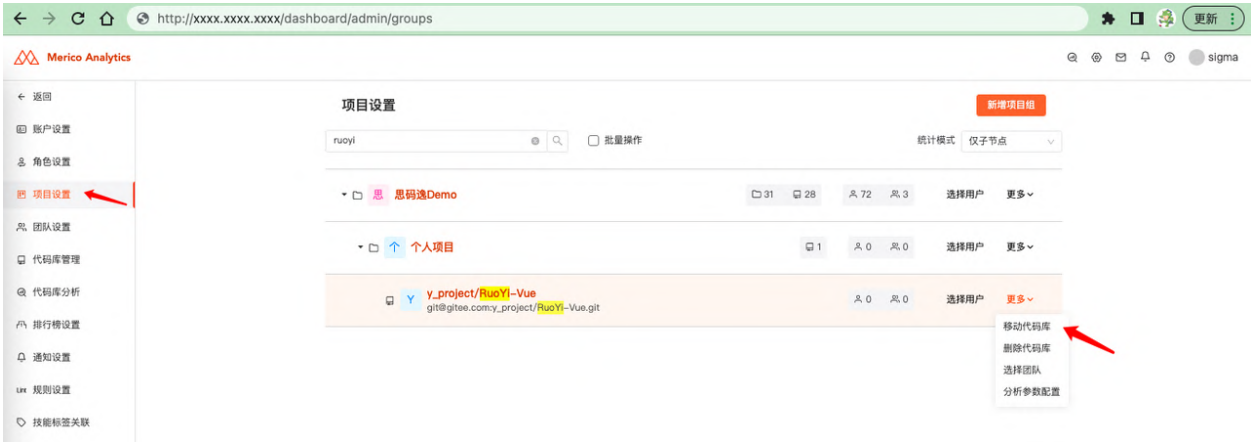

图 153: 设置 - 项目设置 - 移动代码库

方法二示例：

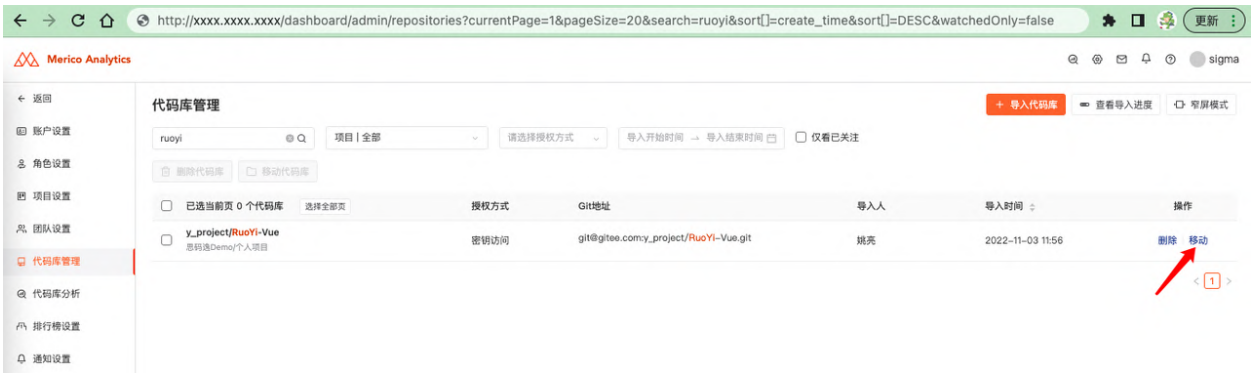

图 154: 设置 - 代码库设置 - 移动代码库

4 功能模块介绍

4.1 系统各工作区介绍

思码逸深度代码分析系统对各代码库进行分析后，将得到的效能数据、质量数据，在如下工作区界面进行分别展示：

4.1.1 项目表现

项目表现版块里，指标数据以项目维度组织，在此版块可以查看如下与项目表现相关的指标和数据：

- **效率报表：**以代码当量为基础，在此界面展示了代码库/项目组/贡献者维度的研发效能指标图表。
- **质量报表：**在此界面展示了代码库/项目组/贡献者维度的如下质量指标图表：单元测试覆盖度、注释覆盖度、代码不重复度、模块性、代码问题数、重点问题密度等数据，支持筛选指定开始日期 (一年内)。
- **质量报表 (历史数据)：**数据信息同质量报表，但增加了支持筛选指定开始日期和结束日期。
- **同行对比：**在此界面展示了用户项目（组）与开源项目的效率指标和质量指标的对比。
- **开发负载：**在此界面展示了人均生产率趋势、贡献者帕累托、开发活动热力图，用以判断人均开发工作量的负荷程度，贡献是否均衡，以及判断资源的利用是否合理。

4.1.2 团队表现

团队表现版块里，指标数据以团队维度组织，在此版块可以查看如下与团队表现相关的指标和数据：

- **效率报表：**以代码当量为基础，在此界面展示了团队/自定义标签分类/团队成员维度的开发效能指标图表。

其他更多团队表现相关报表即将上线，敬请期待。

4.1.3 工程师表现

在工程师表现版块下，可以查看如下与工程师（贡献者）表现相关的指标和数据：

- **贡献者排行：**在此界面展示工程师（贡献者）在效率质量综合、代码当量、代码影响力、代码问题密度评分、单测覆盖度、注释覆盖度等指标上的排行信息。
- **贡献者列表：**在此界面可查看各工程师（贡献者）的包括代码当量、代码影响力、提交数、行数变化等效能指标相关数据。
- **提交列表：**在此界面可查看各提交的具体效能数据和提交信息。

4.1.4 迭代表现

在迭代表现版块下，可以查看与迭代表现相关的指标和数据：迭代进度，迭代的事务/故事点燃尽图，累计代码当量趋势，每日新增代码当量，人均流负载，人均工作负荷，项目 bug 数据，项目组成员工作质量数据。

4.1.5 技术债监控

在技术债监控版块下，可以查看如下与技术债相关的指标和数据：

- **代码问题：**分析得到的代码中的各种问题，在此界面上可看到不同种类、不同严重程度的问题的具体信息，包含此问题在代码中的具体定位和归属者，问题的描述和建议修改方案等。
- **代码重复：**在此界面上展示了分析识别出的重复代码的详细定位和对比。
- **单测覆盖度：**被测试用例覆盖的函数占总函数个数的比例。在此界面上展示了单测覆盖度的评分，未被测试覆盖的函数列表和各文件中各函数被测试覆盖的情况。
- **注释覆盖度：**被注释覆盖的函数占总函数个数的比例。在此界面上展示了注释覆盖度的评分，未被注释覆盖的函数列表和各文件中各函数被注释覆盖的情况。
- **函数复杂度 (仅项目选择代码库时显示)：**展示了代码库中各函数的复杂度分析信息：圈复杂度、函数重复数、函数的出度和入度。

4.1.6 工具箱

包含日志和展示板两类工具：

- **日志工具**：包含登录历史、操作历史、下载历史和系统信息（下载日志）工具
- **展示板工具**：快速打开自己创建或者他人创建并共享的大屏图表数据

4.2 项目表现

4.2.1 项目表现

4.2.1.1 要点概述

在项目表现版块下，可以查看如下与项目表现相关的指标和数据：

- **效率报表：**以代码当量为基础，在此界面展示了代码库/项目组/贡献者维度的研发效能指标图表。
- **质量报表：**在此界面展示了代码库/项目组/贡献者维度的如下质量指标图表：单元测试覆盖度、注释覆盖度、代码不重复度、模块性、代码问题数、重点问题密度等数据，支持筛选指定开始日期 (一年内)。
- **质量报表 (历史数据)：**数据信息同质量报表，但增加了支持筛选指定开始日期和结束日期。
- **同行对比：**在此界面展示了用户项目（组）与开源项目的效率指标和质量指标的对比。
- **开发负载：**在此界面展示了人均生产率趋势、贡献者帕累托、开发活动热力图，用以判断人均开发工作量的负荷程度，贡献是否均衡，以及判断资源的利用是否合理。

4.2.1.2 导航说明

4.2.1.2.1 基础导航

- 左侧菜单栏 **项目表现** -> 点击某个项目组或代码库 -> 进入项目表现详情界面。

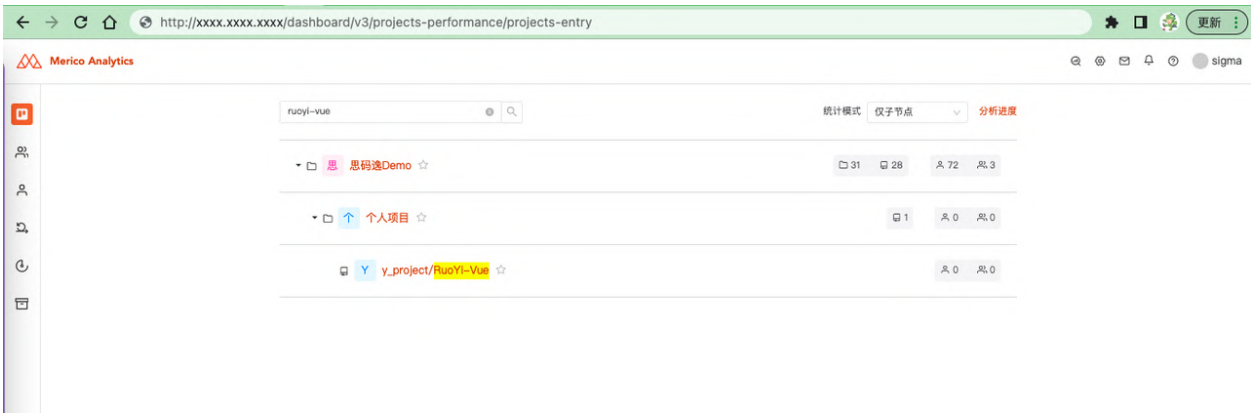

图 155: 进入项目表现详情界面

项目表现详情界面

图 156: 项目表现详情界面

4.2.1.2.2 单选项目（组）查看数据

- 可以单选某个项目（组），来查看此项目（组）的指标数据。
- 点击选择项目图标 ≡
- 在项目列表里点击需要查看数据的项目（组）即可。

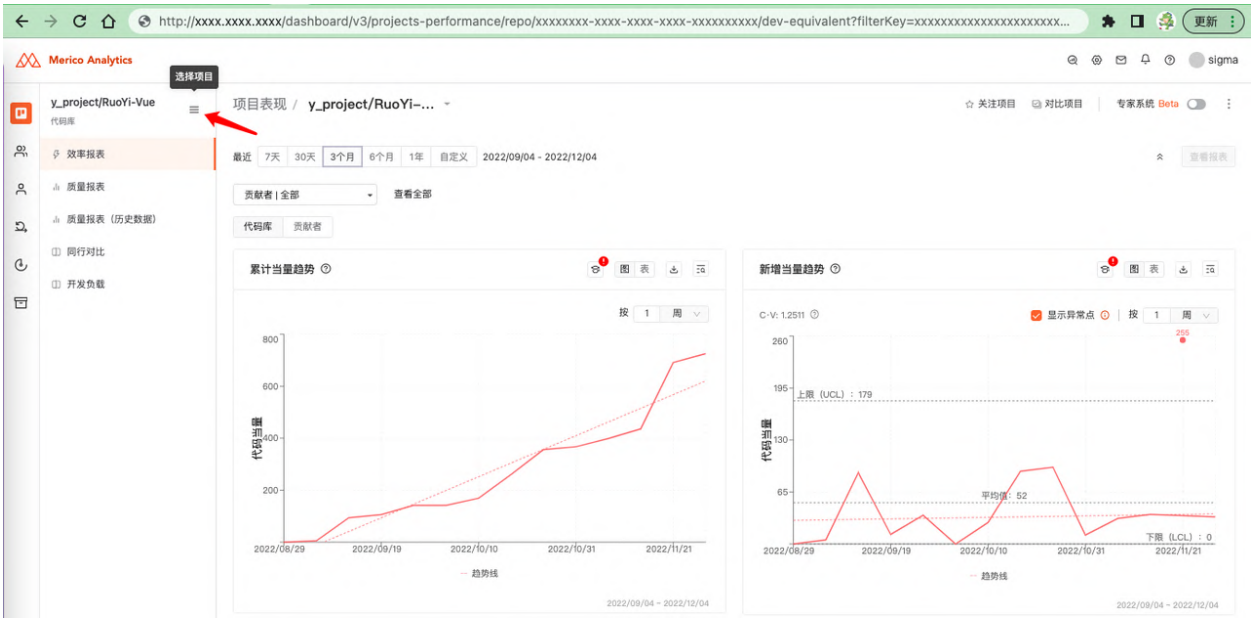

图 157: 进入选择项目侧边栏

图 158: 选择项目

4.2.1.2.3 多选项目（组）查看对比数据

- 可以选择多个项目（组），来查看这些项目（组）的指标对比数据。
- 点击选择项目图标 ≡

- 在项目列表里搜索并勾选需要查看数据的项目（组），勾选完毕后点击 **对比项目**按钮即可。

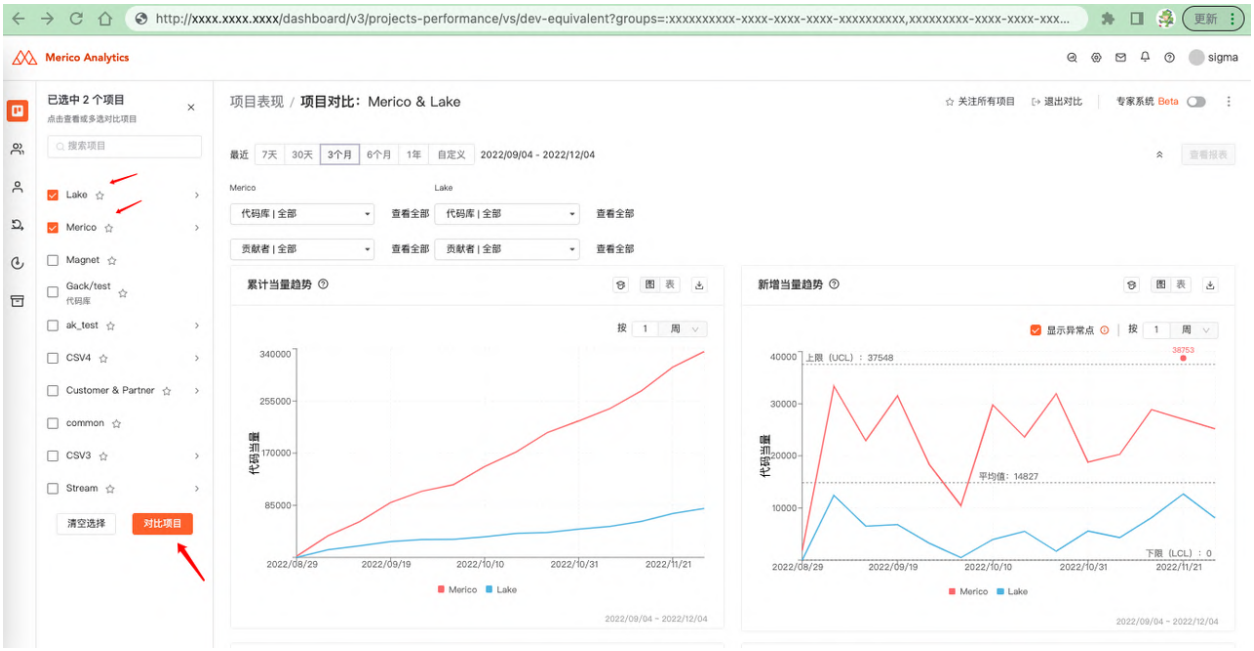

图 159: 对比项目

4.2.1.3 筛选器

- 在选择了项目（组）的基础上，可以进一步精细筛选项目组里的代码库和贡献者，限定数据范围来查看项目数据表现。可以直接进行筛选，也可以保存筛选条件，以供之后复用。

4.2.1.3.1 直接筛选

- 某项目组的 **代码库/贡献者**选择下拉列表中勾选需要的代码库/贡献者，筛选完毕后，点击 **查看报表**即可查看相应的分析数据。

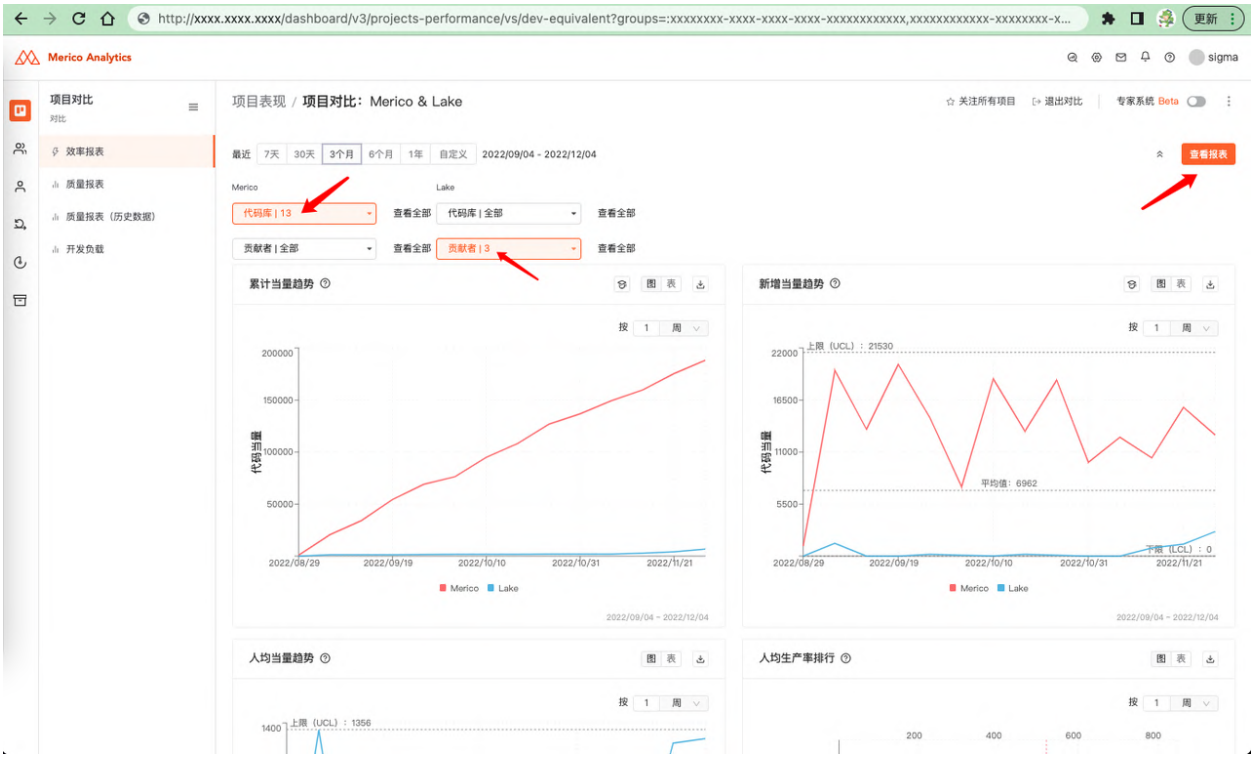

图 160: 筛选并查看报表

4.2.1.3.2 保存筛选条件

- 点击 **代码库/贡献者选择下拉列表**右侧的 **查看全部**链接，打开 **项目/成员筛选条件编辑**界面，勾选需要的项目（组）/贡献者，可以直接点击 **确定**按钮完成筛选，效果等同上面的直接筛选方式。
- 也可以在勾选完毕后，点击 **保存**按钮，**给此组筛选条件命名并保存**。之后需要再次使用同样的筛选条件时，点击 **查看全部**链接，打开 **项目/成员筛选条件编辑**界面，选择之前保存的某组筛选条件并确认即可。

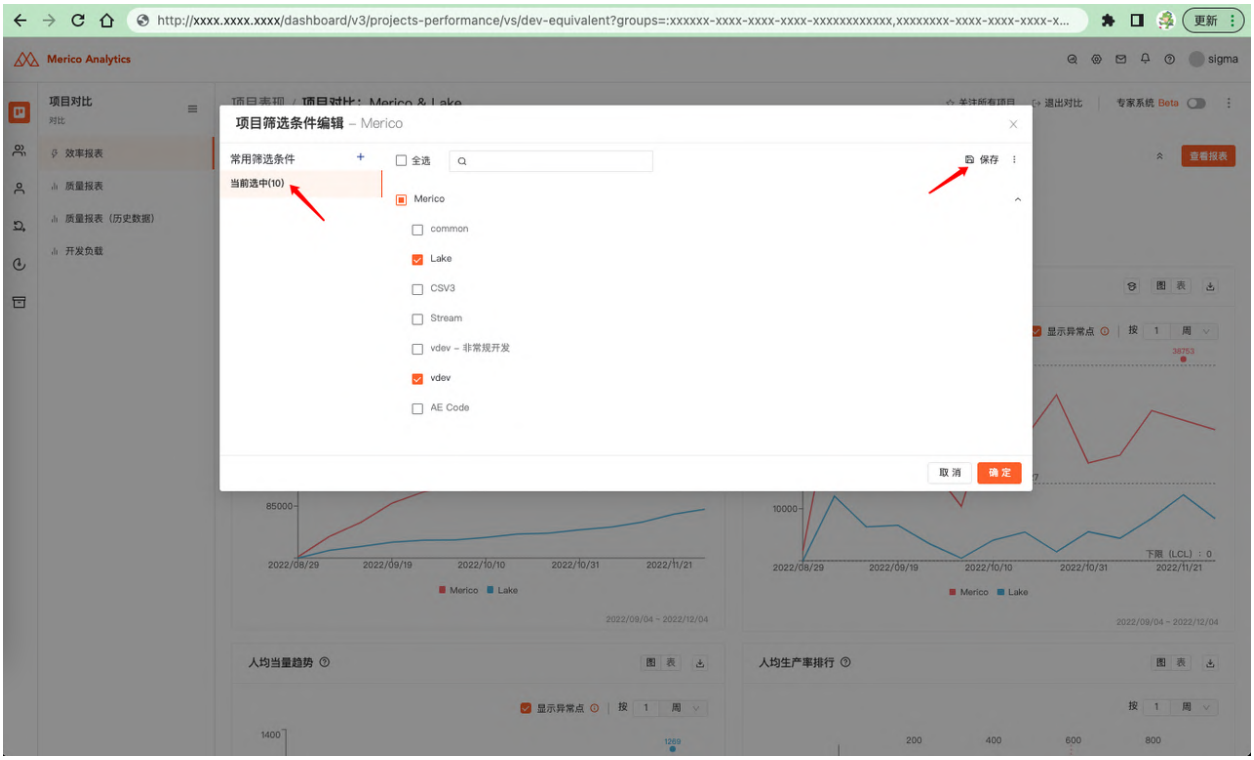

图 161: 保存项目筛选条件

在项目筛选条件编辑页面，保存筛选条件

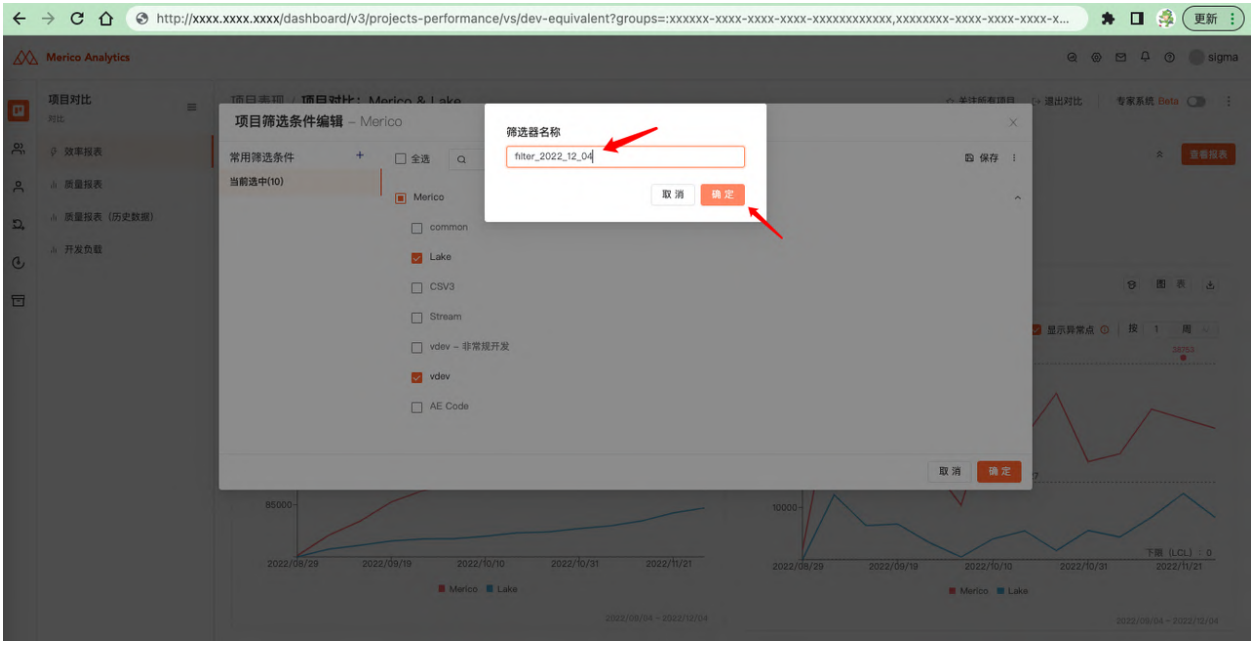

图 162: 填写筛选条件名称

可以看到刚刚保存的筛选条件

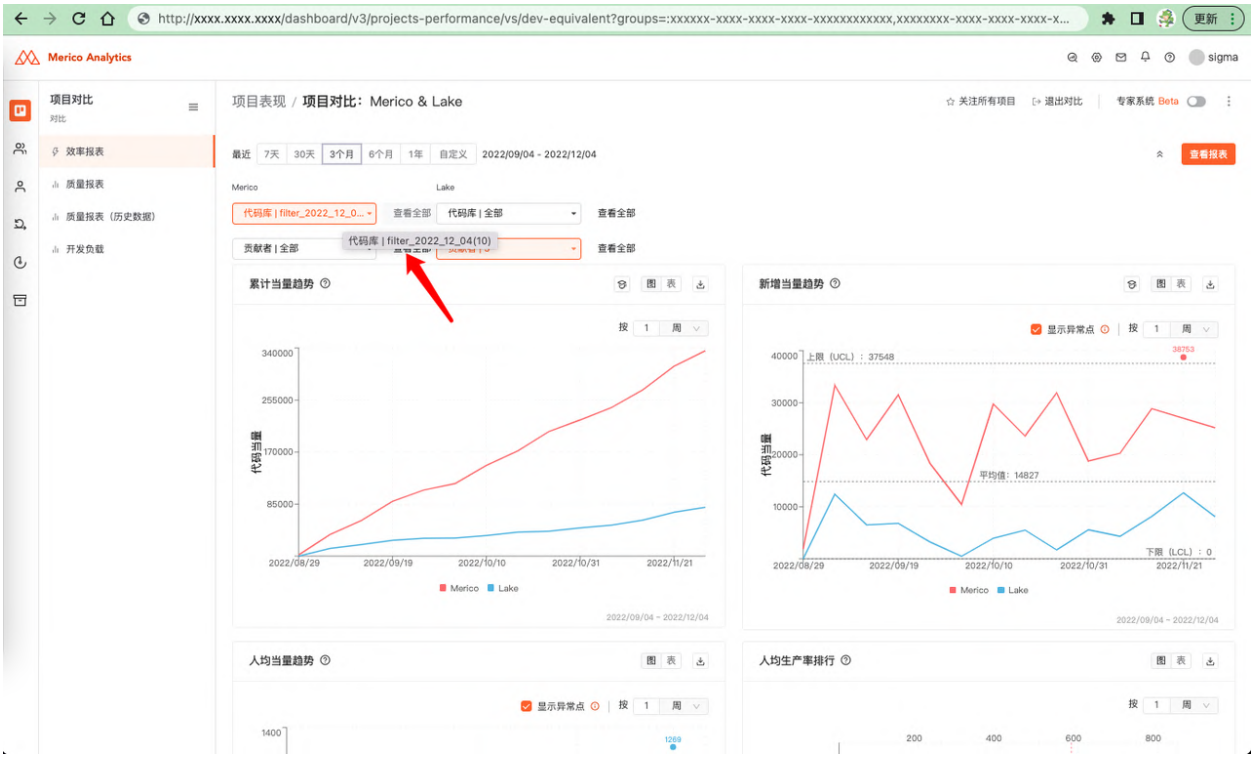

图 163: 查看保存的筛选条件

4.2.1.4 报表保存和下载

- 可以使用 **保存报表**功能保存一组项目查询、筛选条件，以供重复使用。
- 可以使用 **导出报表数据**功能导出当前查询得到的数据包，包括数据表和对应的图表。

4.2.1.4.1 保存报表

- 选择好一组项目查询和筛选条件后（单个或多个项目组，代码库筛选，贡献者筛选），点击右侧 ... 按钮下拉菜单中的 **保存报表**选项，将此组查询作为报表命名并保存。

Merico Analytics
项目对比
效率报表
质量报表
质量报表（历史数据）
开发负载
项目表现 / 项目对比：Merico & Lake
导出报表数据
保存报表
累计当量趋势
新增当量趋势
人均当量趋势
人均生产率排行
Merico
Lake
代码库 | 13
贡献者 | 3
2022/09/04 - 2022/12/04

图 164: 保存报表

图 165: 填写报表名称

- 在 **项目表现 -> 已保存报表**处可以通过点击已保存报表的链接，进入此报表对应的查询筛选界面。

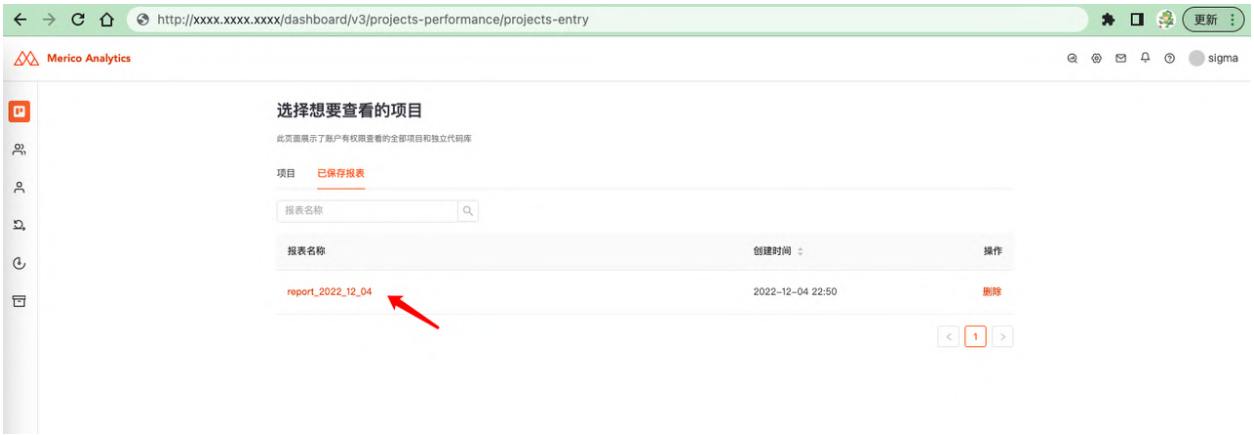

图 166: 查看已保存报表

4.2.1.4.2 导出报表数据

- 选择好一组项目查询和筛选条件后（单个或多个项目组，代码库筛选，贡献者筛选），点击右侧 ... 按钮下拉菜单中的 **导出报表数据**选项，将此组查询得到的数据结果打包下载。

图 167: 导出报表数据

- 导出数据时，需要填写数据文件名称，在点击保存

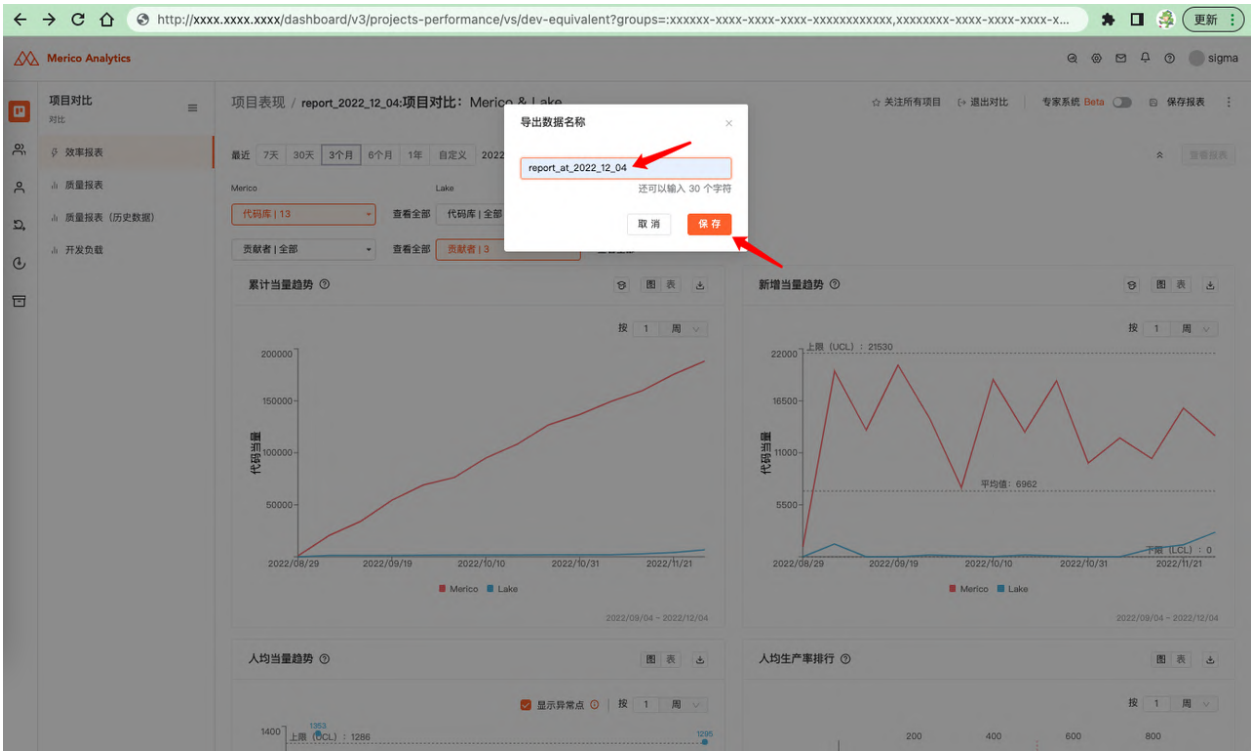

图 168: 填写导出数据名称

- 任务创建同时，可以直接跳转到 **去下载中心**等待数据导出完成后，直接下载

图 169: 导出任务创建成功

- 也可以稍后自行前往下载中心

图 170: 前往下载中心

- 在登录账户的下载中心界面，可以下载导出的报表数据包。

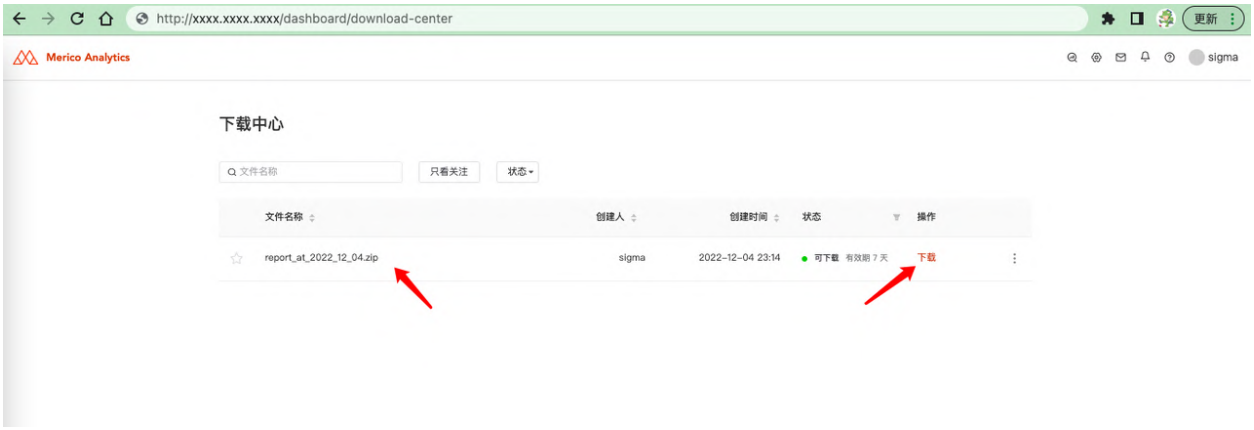

图 171: 下载中心下载报表

4.2.1.5 快捷操作

- 本产品里提供了一系列的快捷操作，方便用户快速搜索、选择需要的项目（组）。

4.2.1.5.1 搜索项目（组）

- 在 **选择项目**列表下方的 **搜索项目**处，可以直接搜索项目（组）。

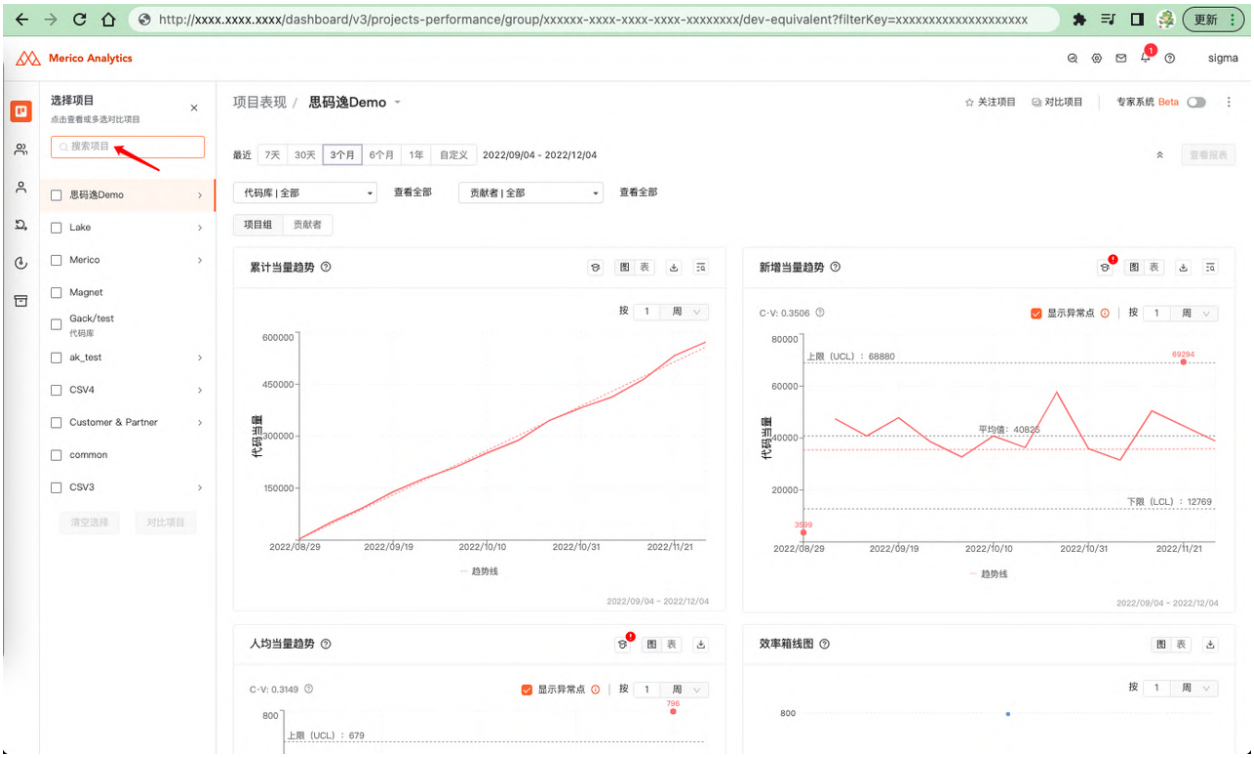

图 172: 搜索项目（组）

4.2.1.5.2 关注项目（组）

- 关注项目（组），方便之后迅速找到这些项目（组）。
- 方法一：点击 **选择项目**列表里项目（组）名称右侧的小星星使之变红色。
- 方法二：在选择了项目（组）之后，点击右侧 **关注所有项目**，来关注当前选中的项目（组）。

方法一示例：

图 173: 点击项目（组）名称右侧的小星星以关注项目

方法二示例：

图 174: 点击关注所有项目以关注项目

4.2.1.5.3 取消关注项目（组）

- 取消关注项目（组），之后这些项目（组）不会继续存在于关注列表里。
- 方法一：点击 **选择项目**列表里项目（组）名称右侧的红色小星星使之变空心。
- 方法二：在选择了项目（组）之后，点击右侧 **取消关注项目**，来取消关注当前选中的项目（组）。

方法一示例：

图 175: 点击项目（组）名称右侧的红色小星星以取消关注项目

方法二示例：

图 176: 点击取消关注所有项目以取消关注项目

4.2.1.5.4 顶层导航直接选择项目（组）

- 单选项目（组）时，可以在 **项目表现**顶层导航直接选择项目（组）。

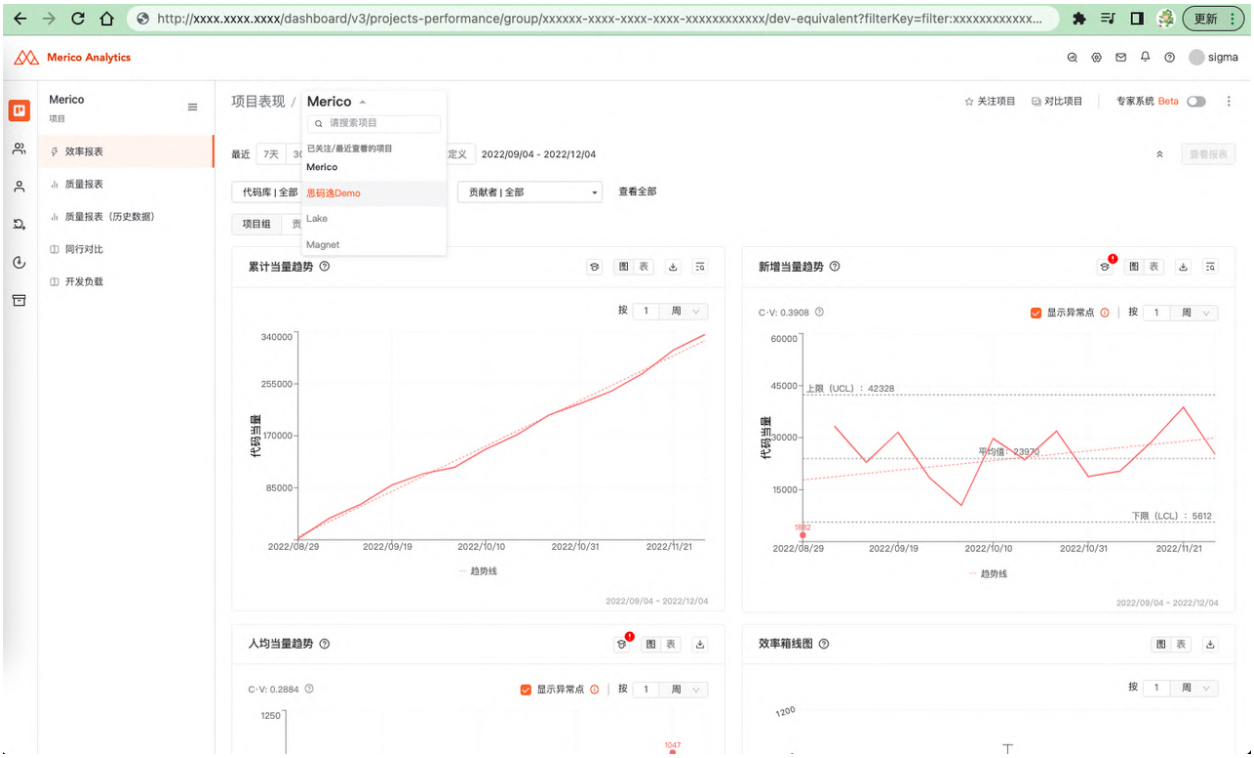

图 177: 顶层导航直接选择项目（组）

4.2.2 效率报表

4.2.2.1 要点概述

- **导航**：左侧菜单栏 **项目表现 - 效率报表**。
- 可以从 **单个项目（组）/ 多个项目（组）对比 / 单个项目（组）的贡献者**三个不同维度来查看效率指标报表。
- **单选**项目（组）查看数据时，可以打开 **专家系统**开关，启用专家系统功能辅助解读数据。
- 在此工作区界面中，可以查看如下以代码当量为基础得到的指标图表，具体包括：

单个项目（组）维度	多个项目（组）对比维度	单个项目（组）的贡献者维度
累计当量趋势新增当量趋势人均当量趋势效率箱线图	累计当量趋势新增当量趋势人均当量趋势人均生产率排行当量占比效率箱线图效率与稳定性	代码当量帕累托图累计当量趋势新增当量趋势当量占比

4.2.2.2 效率报表

4.2.2.2.1 累计当量趋势

- 此图表展示所选时间范围内，所选观察维度（单个项目（组）/多个项目（组）对比/单个项目（组）的贡献者），所选步长（日/周/月）的代码当量累积数据折线图。此累计当量折线图是累计递增的。
- 当仅选择**单个**项目（组）/贡献者时，图表上会用虚线绘制累计趋势线来展示数据变化的大体趋势，通过趋势线的斜率可以观察数据总体累积速度是比较平缓还是迅速。
- 可以通过修改步长来看不同步长时数据累计趋势的变化。
- 当把光标悬停在图表某数据点时，会有悬停窗口展示此数据点的横轴纵轴信息；直接用鼠标双击，可以打开此数据点关联的提交的信息。
- 可以切换到**表模式**来查看具体的代码当量累计数据。点击**下载**按钮可以下载数据表。

4.2.2.2.2 新增当量趋势

- 此图表展示所选时间范围内，所选观察维度（单个项目（组）/多个项目（组）对比/单个项目（组）的贡献者），每步长（日/周/月）时间段代码当量新增的折线图。
- 当仅选择**单个**项目（组）/贡献者时，图表上会用虚线显示新增当量趋势线来展示数据变化的大体趋势：通过趋势线的方向（上升还是下降）可以观察每步长新增当量总体上来看是增加的趋势还是下降的趋势，查看趋势线斜率可以观察数据总体是在比较平缓还是比较迅速地上升/下降。
- 新增当量趋势图表中，上下限公式：上限值（UCL）= 平均值 +2 标准差；下限值（LCL）= 平均值-(2 标准差)，小于 0 则取 0。超过上下限值的数据点视为异常点，需要予以关注，查看是因为什么原因造成的。
- 勾选 **显示异常点**，异常点不参与折线图绘制，取消勾选 **显示异常点**，异常点参与折线图绘制。
- 可以通过修改步长来查看不同步长时数据的变化。
- 当把光标悬停在图表某数据点时，会有悬停窗口展示此数据点的横轴纵轴信息；直接用鼠标双击，可以打开此数据点关联的提交的信息。
- 可以切换到**表模式**来查看具体的代码当量新增数据。点击**下载**按钮可以下载数据表。

== 单个项目（组）维度 ==

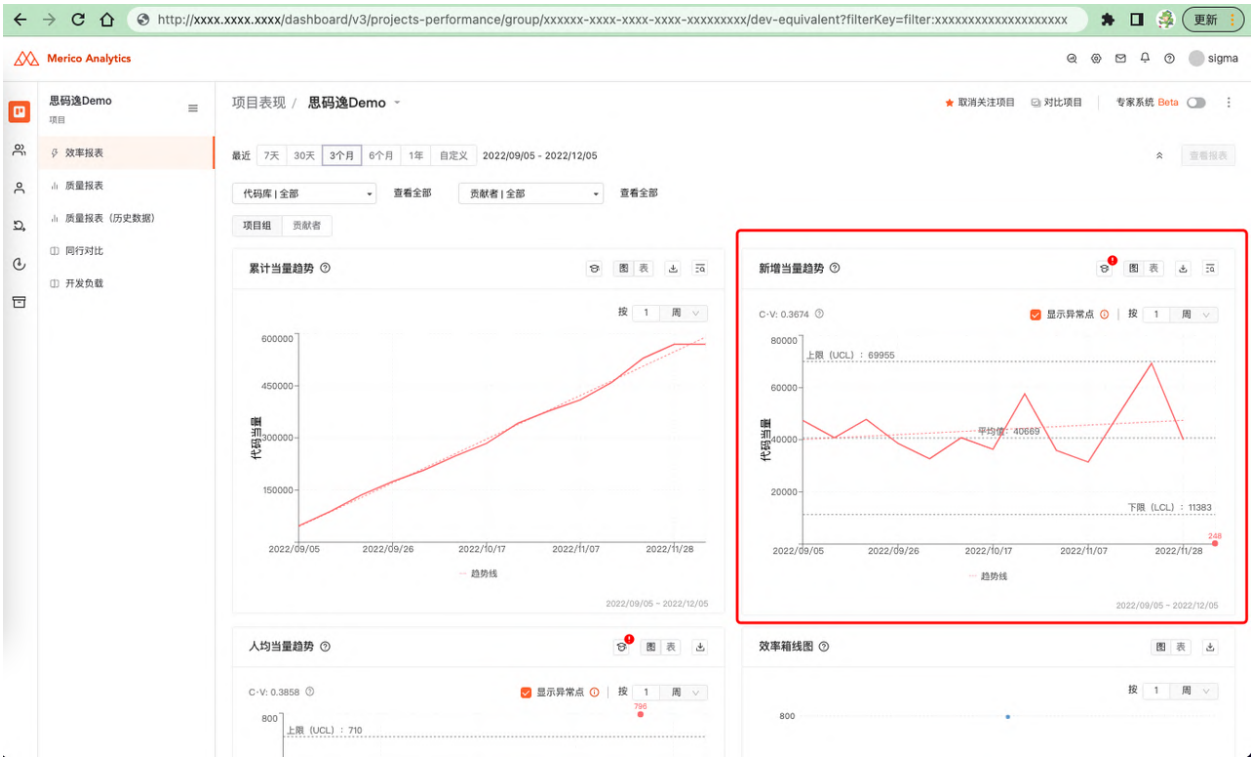

图 178: 新增当量趋势 - 单个项目（组）维度

== 多个项目（组）对比维度 ==

图 179: 新增当量趋势 - 多个项目（组）对比维度

== 单个项目（组）的贡献者维度 ==

图 180: 新增当量趋势 - 单个项目（组）的贡献者维度

4.2.2.2.3 人均当量趋势

- 此图表展示所选时间范围内，所选观察维度（单个项目（组）/ 多个项目（组）对比），每步长（日/周/月）时间段人均代码当量的折线图。在当前版本中，每步长里的人员数量（分母）为此步长时间段内有提交的贡献者人数。
- 趋势线展示和含义、上下限展示和计算公式，同新增当量趋势。
- 勾选 **显示异常点**，异常点不参与折线图绘制，取消勾选 **显示异常点**，异常点参与折线图绘制。
- 可以通过修改步长来看不同步长时数据的变化。
- 当把光标悬停在图表某数据点时，会有悬停窗口展示此数据点的横轴纵轴信息；直接用鼠标双击，可以打开此数据点关联的提交的信息。
- 可以切换到**表模式**来查看具体的人均当量趋势数据。点击**下载**按钮可以下载数据表。

== 单个项目（组）维度 ==

图 181: 人均当量趋势 - 单个项目（组）维度

== 多个项目（组）对比维度 ==

图 182: 人均当量趋势 - 多个项目（组）对比维度

4.2.2.2.4 人均生产率排行

- 此图表展示所选时间范围内，所选的项目（组）的每步长（日/周/月）人均代码当量的 == **平均值** == 的横向柱状图。按每个项目（组）数据从大到小，从上到下排列。人员数量（分母）为所选时间范围、每步长内有提交的贡献者人数。
- 其中红色的平均值柱子为所选的所有项目（组）的**蓝色数据**的平均值，通过与平均值平齐的垂直虚线可以观察在此数据上超过或低于平均值的项目（组）。
- 可以切换到**表模式**来查看具体的人均当量平均值数据。点击**下载**按钮可以下载数据表。

== **仅**多个项目（组）对比维度 ==

图 183: 人均生产率排行 - **仅**多个项目（组）对比维度

4.2.2.2.5 当量占比

- 此图表展示所选时间范围内，所选观察维度（多个项目（组）对比 / 单个项目（组）的贡献者），所选的项目（组）/贡献者的代码当量总量占比的饼图。
- 可以切换到**表模式**来查看具体的代码当量数据。点击**下载**按钮可以下载数据表。

== 多个项目（组）对比维度 ==

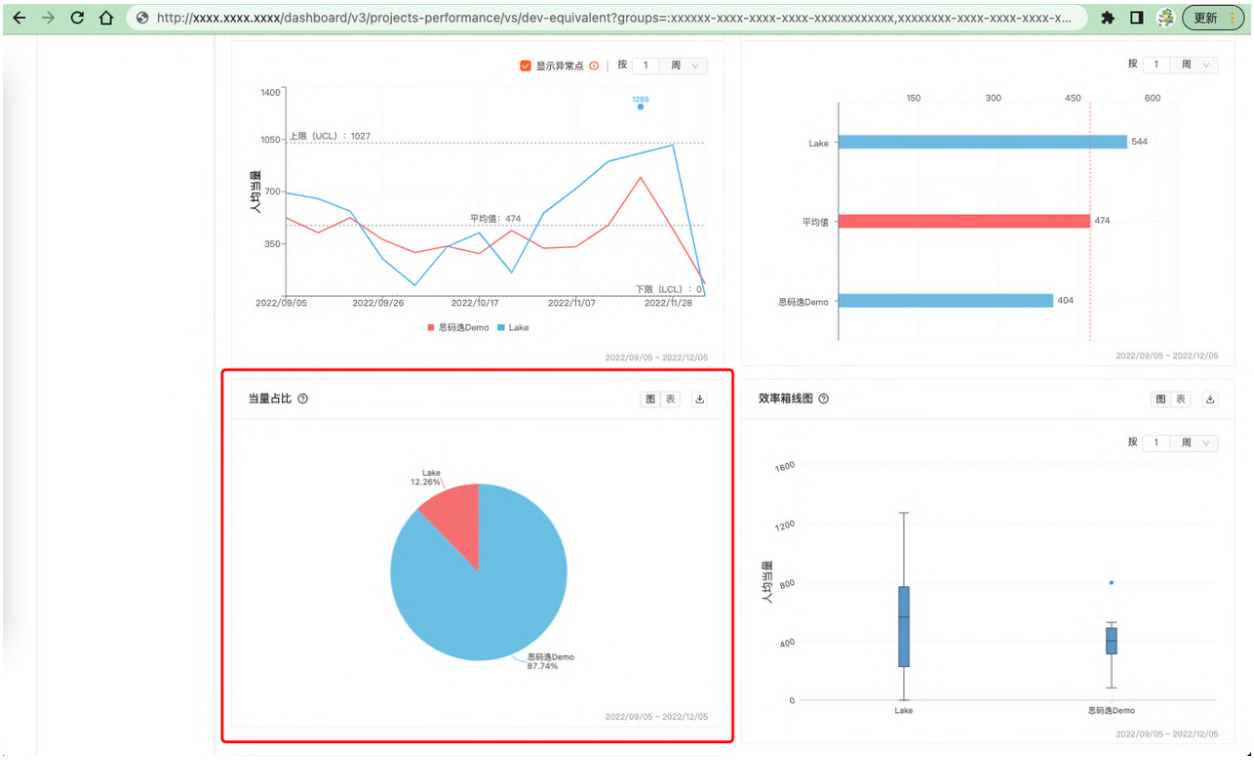

图 184: 当量占比 - 多个项目（组）对比维度

== 单个项目（组）的贡献者维度 ==

图 185: 当量占比 - 单个项目（组）的贡献者维度

4.2.2.2.6 效率箱线图

此图表展示所选时间范围内，所选观察维度（单个项目（组）/ 多个项目（组）对比），每步长（日/周/月）时间段各项目（组）的人均代码当量数据集绘制的箱线图。

箱线图绘制方法

1. 将数据集里的各数据点从下到上按照从小到大排列，在 25%、50%、75% 处划线，得到箱体。
2. 箱体下边缘到 0 值间的 1.5 倍箱高处，找到数据集里最后一个在此范围内的数据点作为最小值，箱体上边缘的 1.5 倍箱高处，找到数据集里最后一个在此范围内的数据点作为最大值。
3. 超出最小值和最大值的数据点标识为异常点。

箱线图解读要点

1. **看箱体的上下边位置：**箱体的下边是后 25% 的人均代码当量值，箱体的上边是前 25% 的人均代码当量值，箱体整体高度代表数据集中间 50% 的人均代码当量数据点分布的范围和广度，对比各项目（组）箱体上下边位置可以看到其人均代码当量统计上中间 50% 情况下的下限和上限。
2. **看箱体的高度：**箱体越高（整个箱子在纵向上越长），说明数据集里中部 50% 的数据分布区间越广。
3. **看箱体中位线**：中位线可以看出该项目（组）中位水平人均代码当量，对比中位线可以看到各项目（组）的中位人均效能水平的差异。

4. **异常点观察：**观察有无低异常点和高异常点。

== 单个项目（组）对比维度 ==

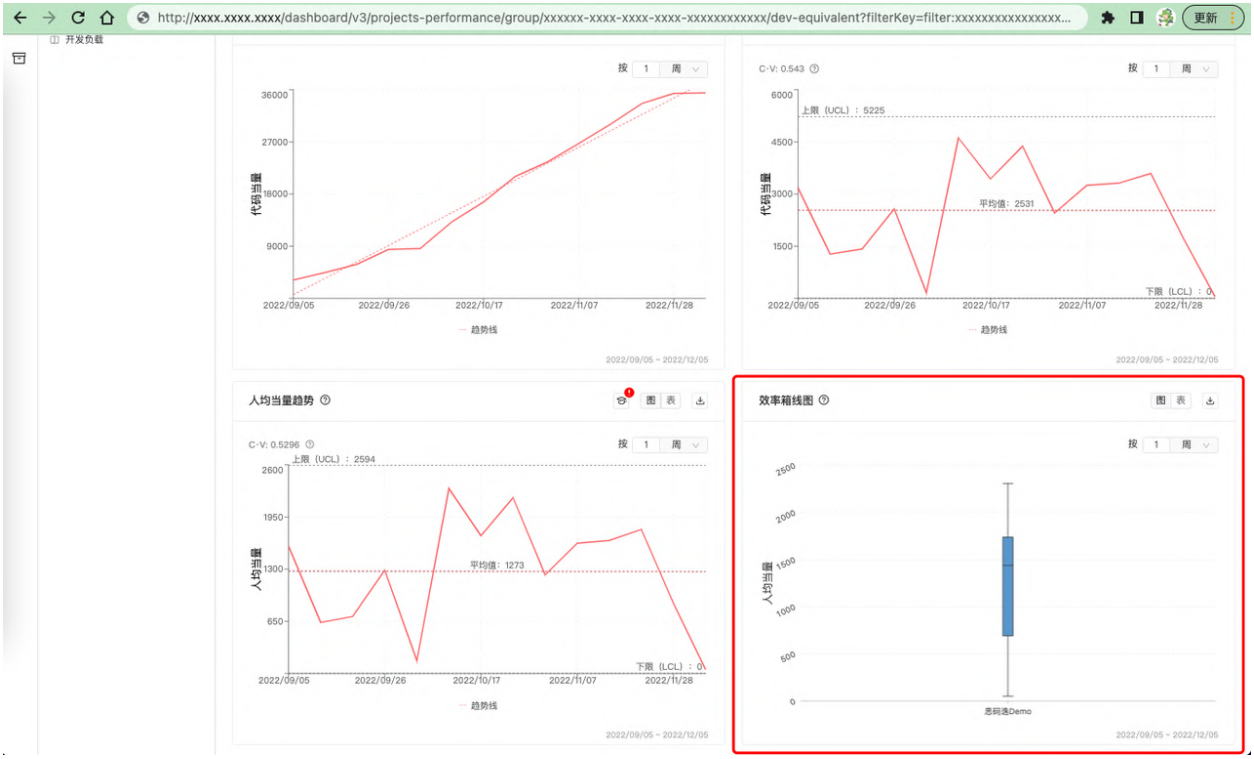

图 186: 效率箱线图 - 单个项目（组）对比维度

== 多个项目（组）对比维度 ==

图 187: 效率箱线图 - 多个项目（组）对比维度

4.2.2.2.7 效率与稳定性

此图表用来综合展示开发过程的效率与稳定性。其中纵轴为离散系数（越小越好），横轴为平均每步长（日/周/月）人均代码当量（越大越好）。注：隐藏了代码当量与离散系数皆为 0 的点。

效率与稳定性图表绘制方法

- **效率：**所选时间范围的内，各项目（组）**每步长的人均代码当量的平均值**作为各项目（组）的效率数据，将已选择的所有项目（组）的**平均人均代码当量**进一步求**平均值**，效率数据低于此均值的项目（组）即判断为**低效**，反之则判断为**高效**。
- **稳定性：**所选时间范围的内，各项目（组）**每步长的人均代码当量**作为数据集，求出各项目（组）的此数据集的**离散系数**。离散系数大于 **0.5** 的项目（组）判断为**低稳定性**，反之则判断为**高稳定性**。
- **离散系数：**在概率论和统计学中，离散系数（coefficient of variation），是概率分布离散程度的一个归一化量度，其定义为标准差与平均值之比（即：**离散系数 = 标准差/均值**）。
- 图形上直观看起来，如果以以上数据集合描点画图的话，时间为横轴，纵轴为该项目（组）每步长人均代码当量，各数据点更加趋于均值、波动更小的项目（组）的数据集合的离散度越小，离散系数也越小，这个团队的开发过程也越稳定。

效率与稳定性图表解读要点

- 同时关注效率与稳定性两个维度的数据表现，可以让持续稳定高效产出的团队脱颖而出。

- 参考多数项目的分析结果，通常 2 周步长是一个比较稳定的数据单位，可以用作项目过程中的判稳。

== **仅**多个项目（组）对比维度 ==

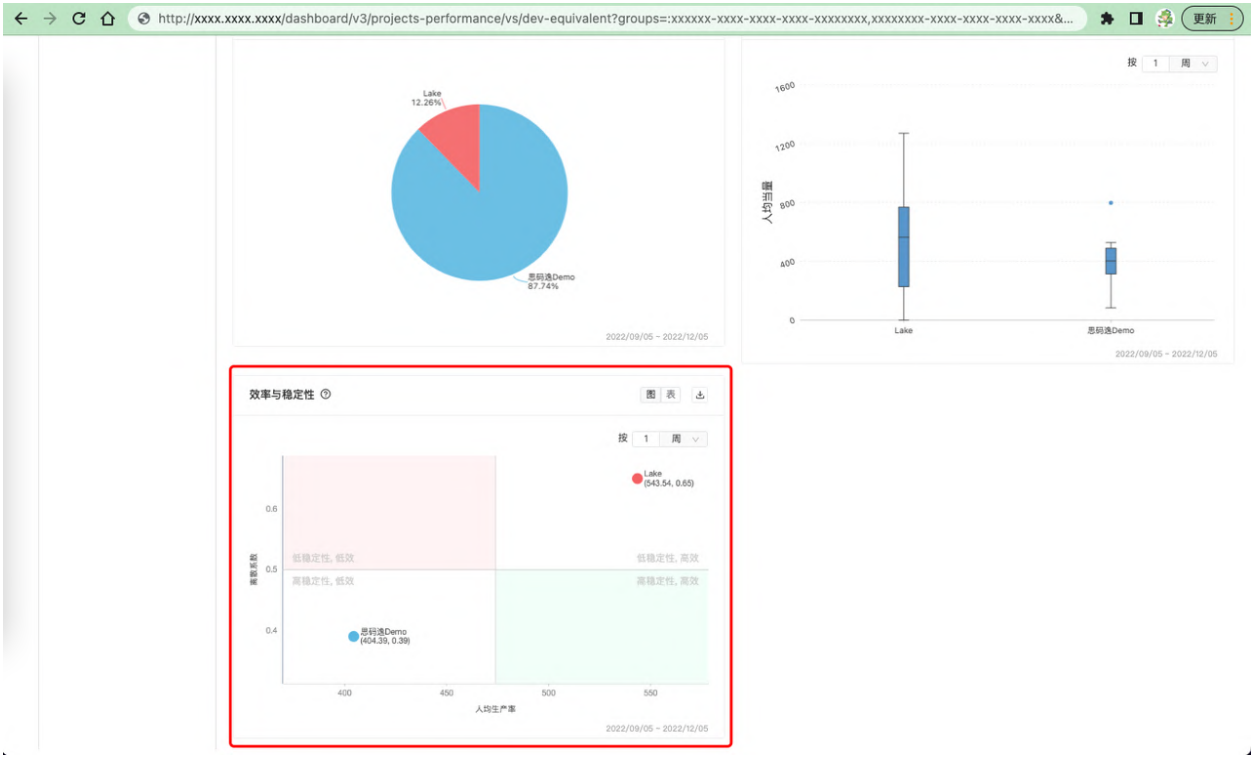

图 188: 效率与稳定性 - **仅**多个项目（组）对比维度

4.2.2.2.8 代码当量帕累托图

按照 2/8 定理，展示所选项目（组）内所选贡献者的选择时间范围内的累计代码当量的排名。

代码当量帕累托图表绘制方法

- 此图表展示所选时间范围内，所选项目（组），所选的贡献者的累计代码当量的排名。
- 将各贡献者的累计代码当量从左到右按从高到低排序，在总量的 80% 处分隔一条 **“80% 的当量由左侧贡献”** 的分隔线以作标识，如果分界线大于 80% 则显示实际百分比。
- 按每个贡献者的累计代码当量在总量里所占百分比作折线，从左往右趋向 100%。

代码当量帕累托图表解读要点

- 可以直观看到贡献了 80% 总代码当量的开发者。如果是少部分贡献者在 80% 线左边，那么说明少部分开发者贡献了团队里的大部分代码当量。
- 看帕累托折线的陡峭/平缓程度，如果是很平缓的一条曲线，说明各贡献者相互之间的数据区别不大，如果折线变化比较陡峭，说明有比较大的差异。

== **仅**单个项目（组）的贡献者维度 ==

图 189: 代码当量帕累托图 - **仅**单个项目（组）的贡献者维度

4.2.3 质量报表

4.2.3.1 要点概述

- **导航：**左侧菜单栏 **项目表现 - 质量报表**。
- 可以从 **项目（组）/ 项目对比 / 贡献者**三个不同维度来查看质量指标报表。
- 可以查看 **全部时间范围**的质量分析数据，也可以查看 **某天至今**的代码的质量分析数据。

4.2.3.2 质量报表

4.2.3.2.1 单元测试覆盖度

被测试用例覆盖的函数占总函数个数的比例。

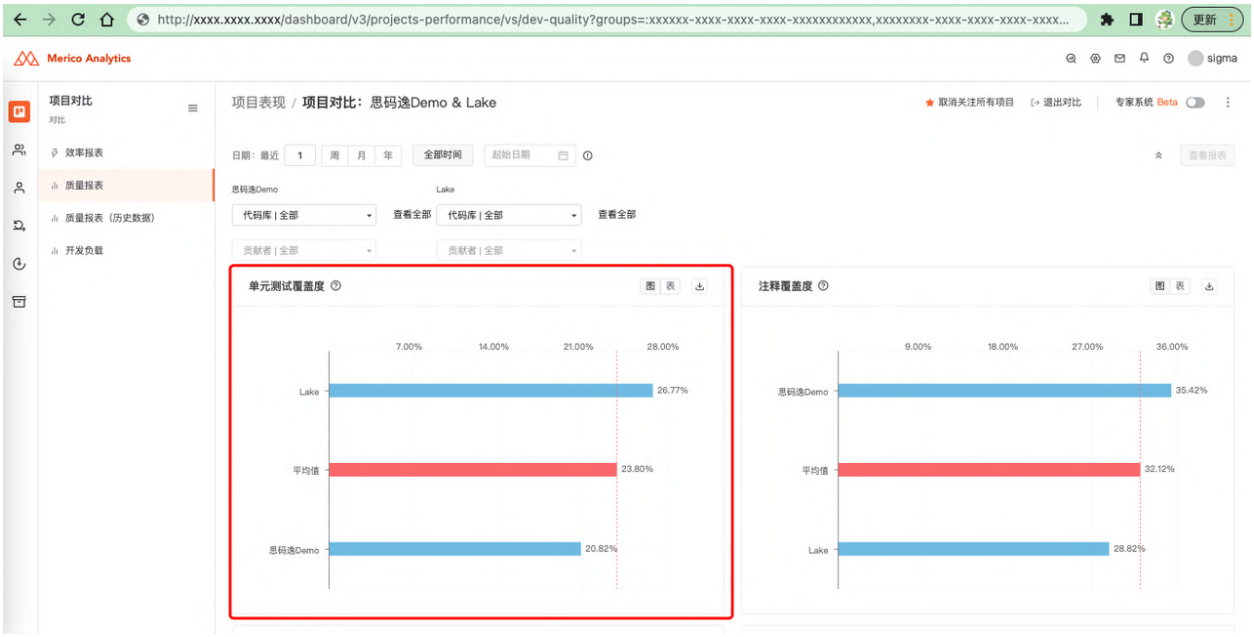

图 190: 单元测试覆盖度

4.2.3.2.2 注释覆盖度

有格式化注释的函数占总函数个数的比例。

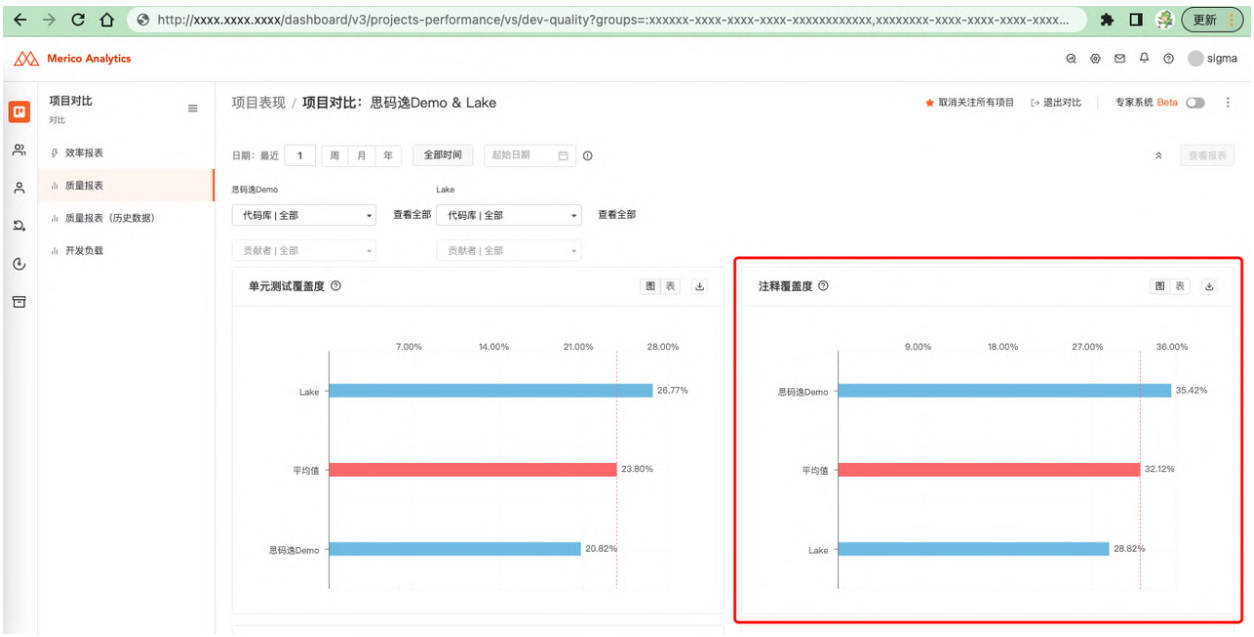

图 191: 注释覆盖度

4.2.3.2.3 代码不重复度

不重复代码所占的比例。

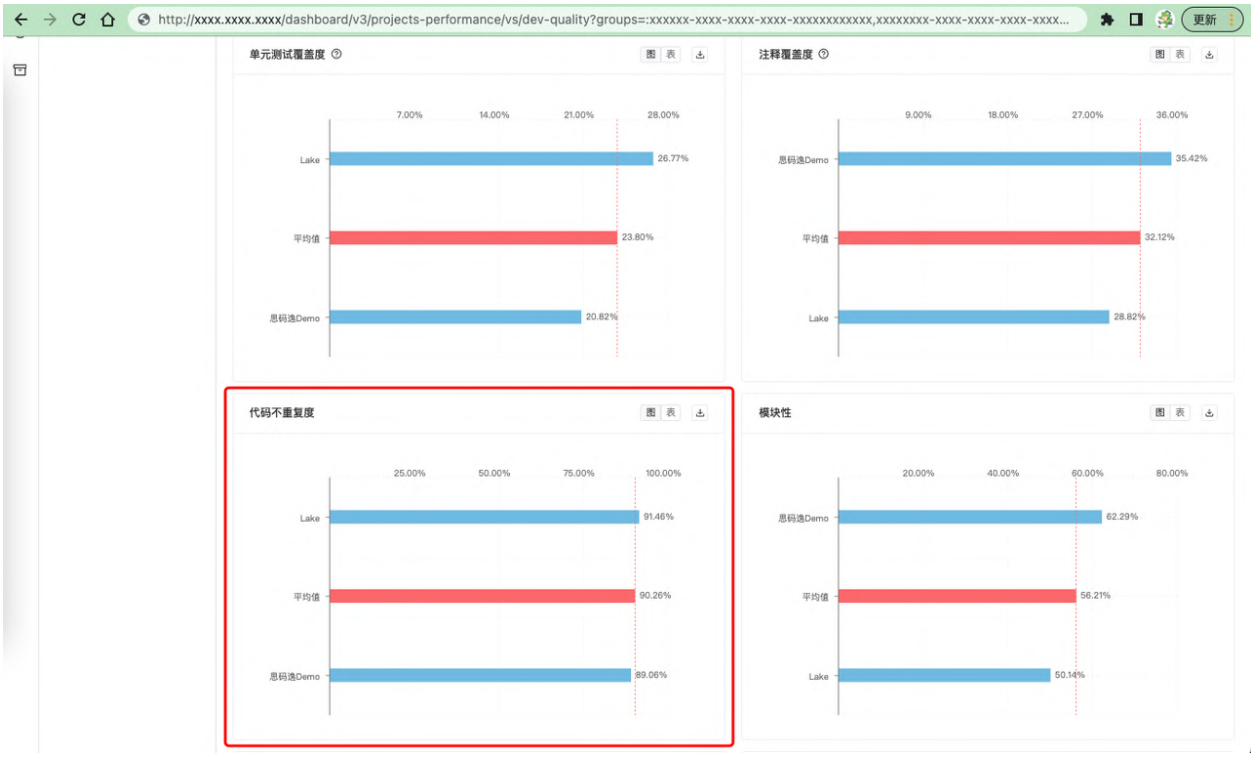

图 192: 代码不重复度

4.2.3.2.4 模块性

反映代码划分出高内聚、松耦合的模块的难易程度，基于代码调用-提交图的最优划分计算。

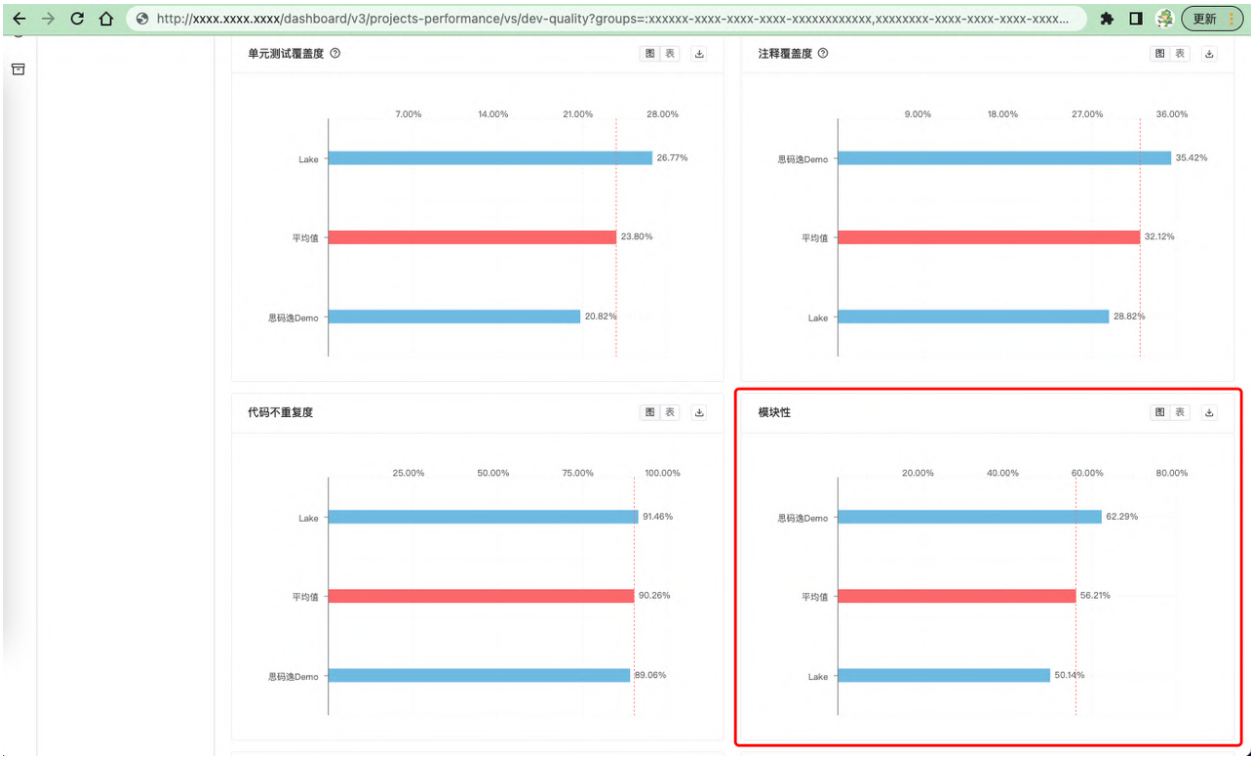

图 193: 模块性

4.2.3.2.5 代码问题数

各严重程度的问题的数量。

图 194: 代码问题数

4.2.3.2.6 重点问题密度

重点问题密度由选定时间段内的重点问题数除以代码当量计算得出，重点问题包括阻塞问题和严重问题。

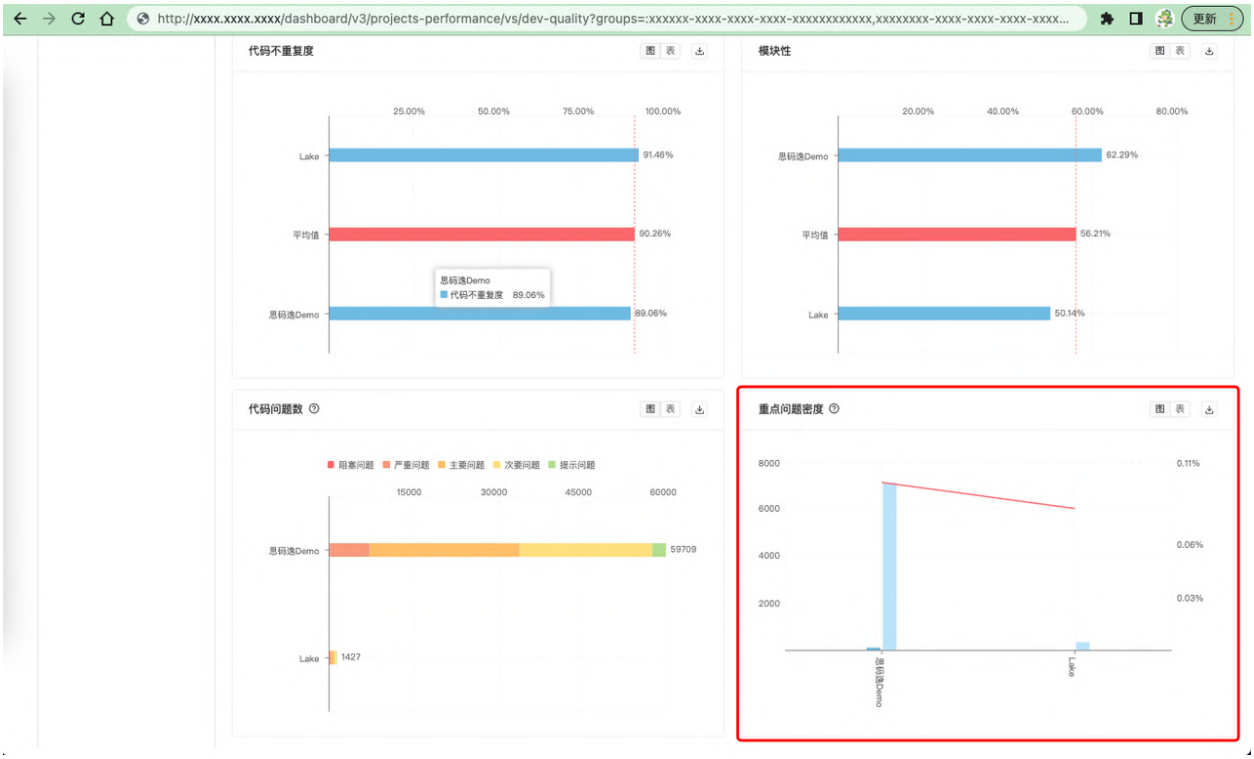

图 195: 重点问题密度

4.2.4 质量报表 (历史数据)

4.2.4.1 要点概述

- 导航：左侧菜单栏 **项目表现 - 质量报表（历史数据）**。
- 可以从 **项目（组）/ 项目对比 / 贡献者**三个不同维度来查看质量指标报表。
- 可以查看 **指定开始日期和指定结束日期时间范围内**质量分析数据的历史留存数据，即每次分析当时的代码分析得到的质量指标数据。
- 各个指标图中，每个数据点为该步长（天/周/月/年）时间范围里分析得到的历史数据的均值。通过一段时间的描点连线可以看到此时间段内该项目（组）对应质量指标的数据变化。

4.2.4.2 质量报表

4.2.4.2.1 单元测试覆盖度

被测试用例覆盖的函数占总函数个数的比例。

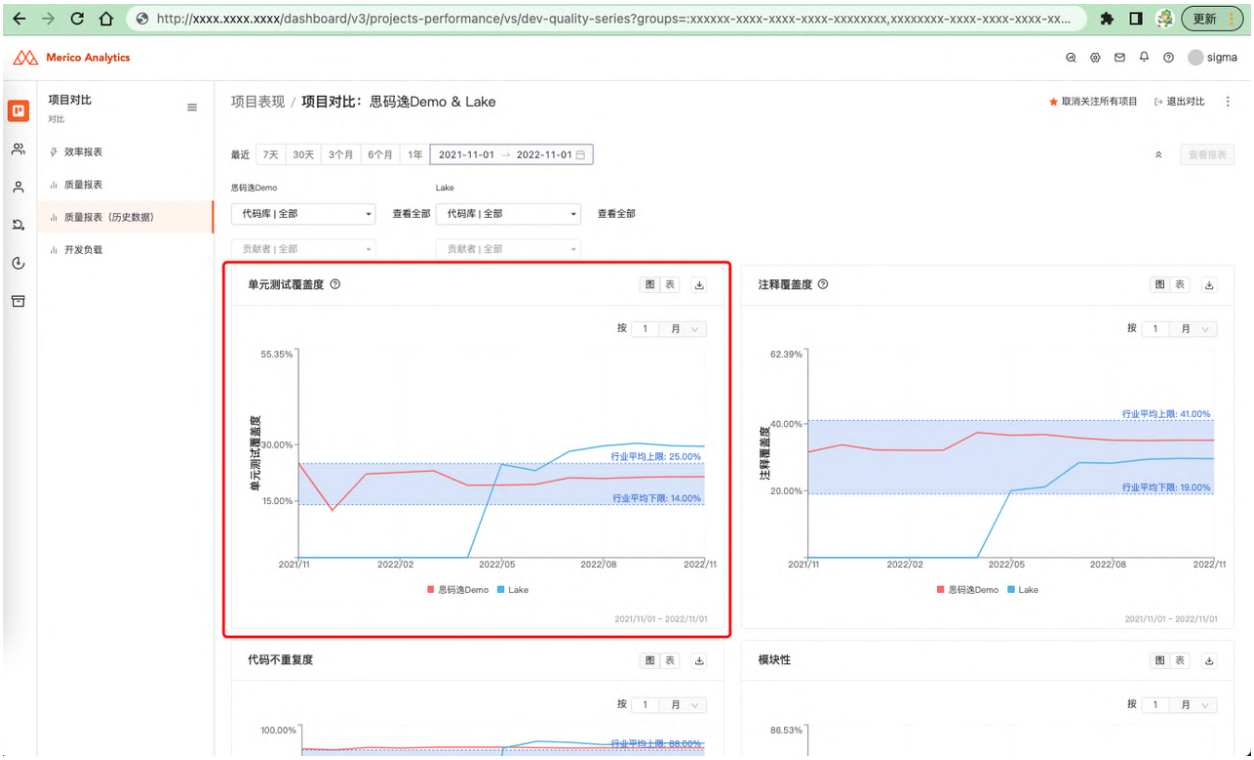

图 196: 单元测试覆盖度

4.2.4.2.2 注释覆盖度

有格式化注释的函数占总函数个数的比例。

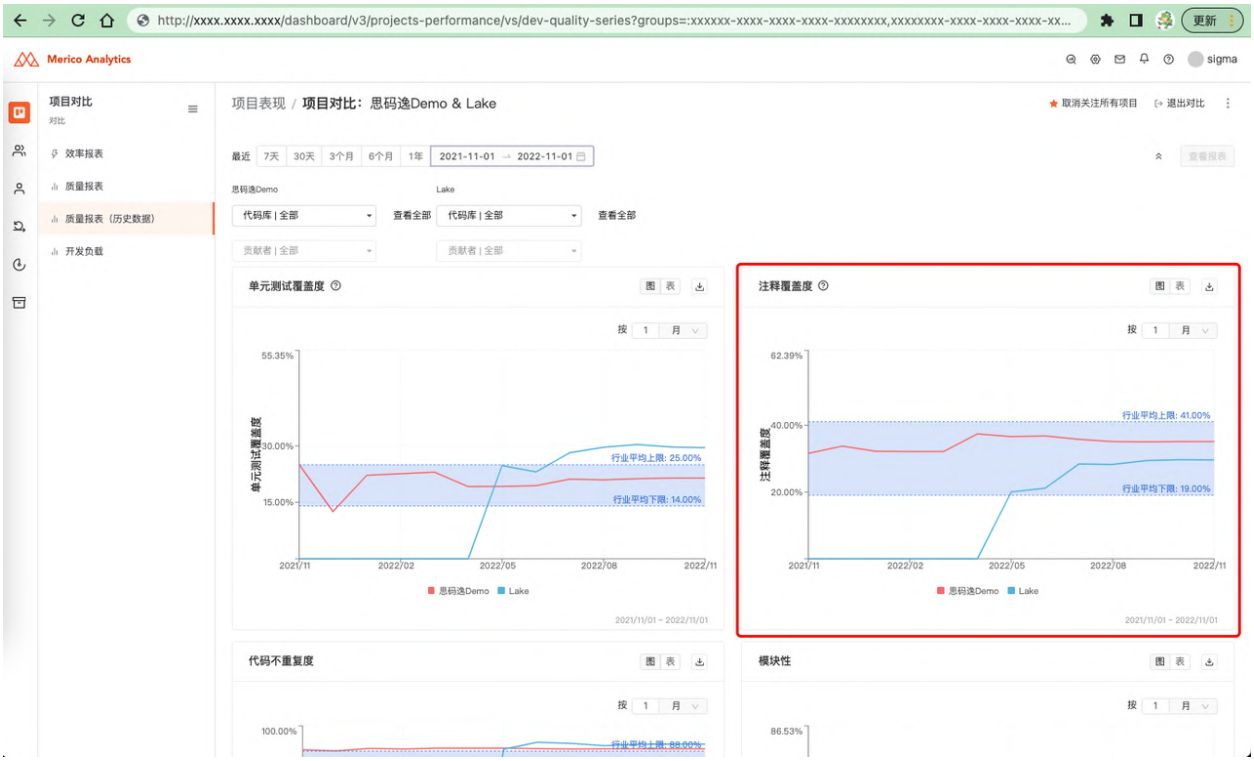

图 197: 注释覆盖度

4.2.4.2.3 代码不重复度

不重复代码所占的比例。

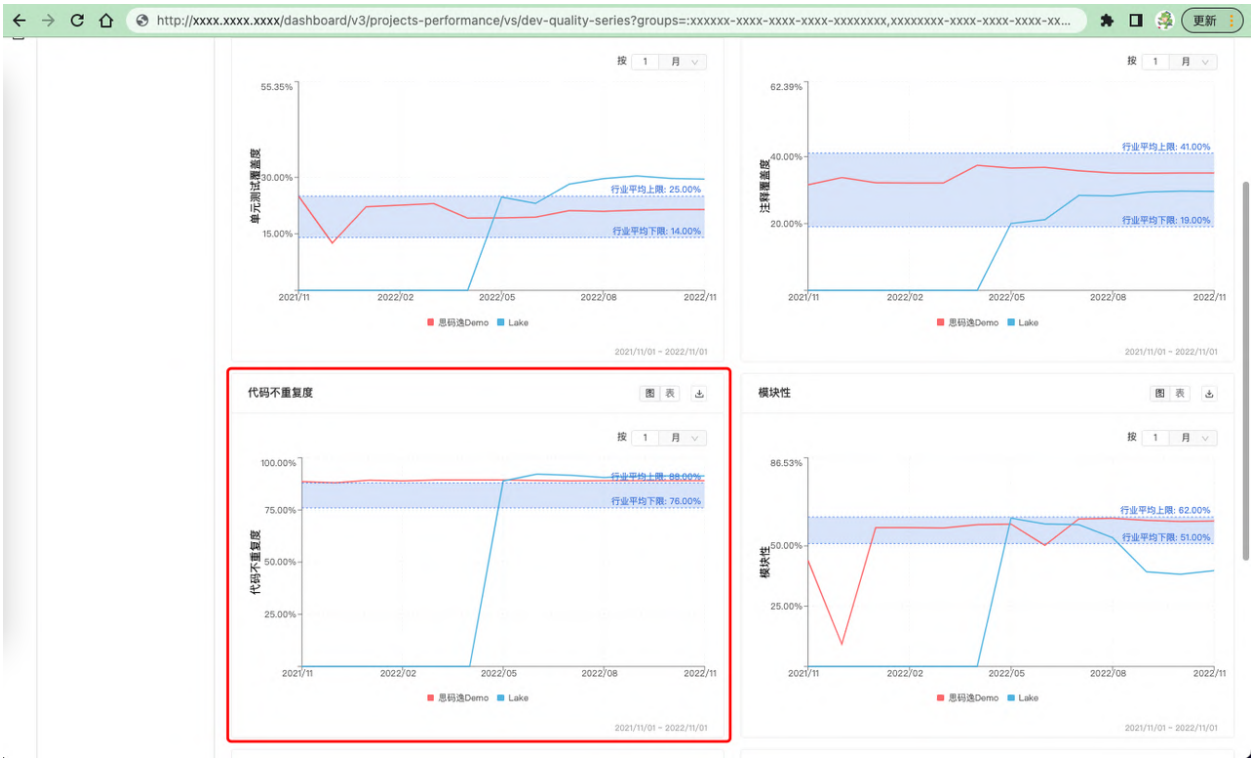

图 198: 代码不重复度

4.2.4.2.4 模块性

反映代码划分出高内聚、松耦合的模块的难易程度，基于代码调用-提交图的最优划分计算。

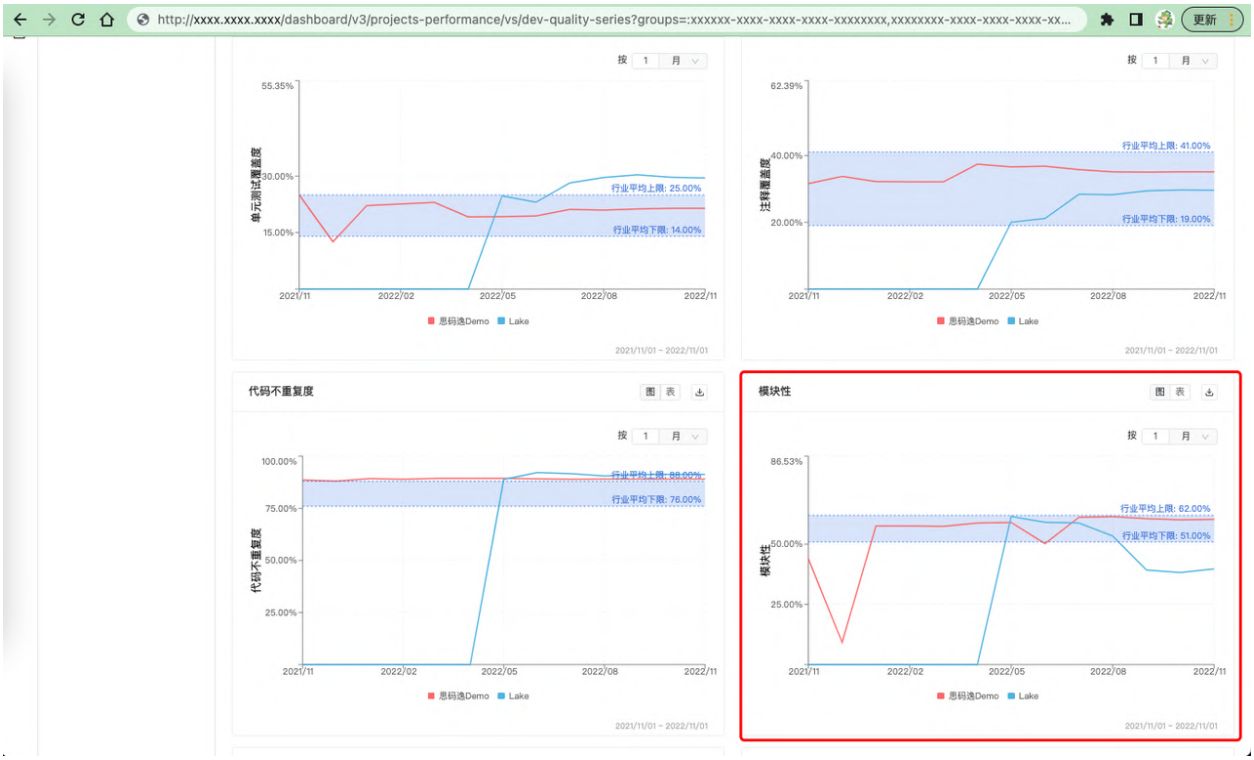

图 199: 模块性

4.2.4.2.5 代码问题数

各严重程度的问题的数量。

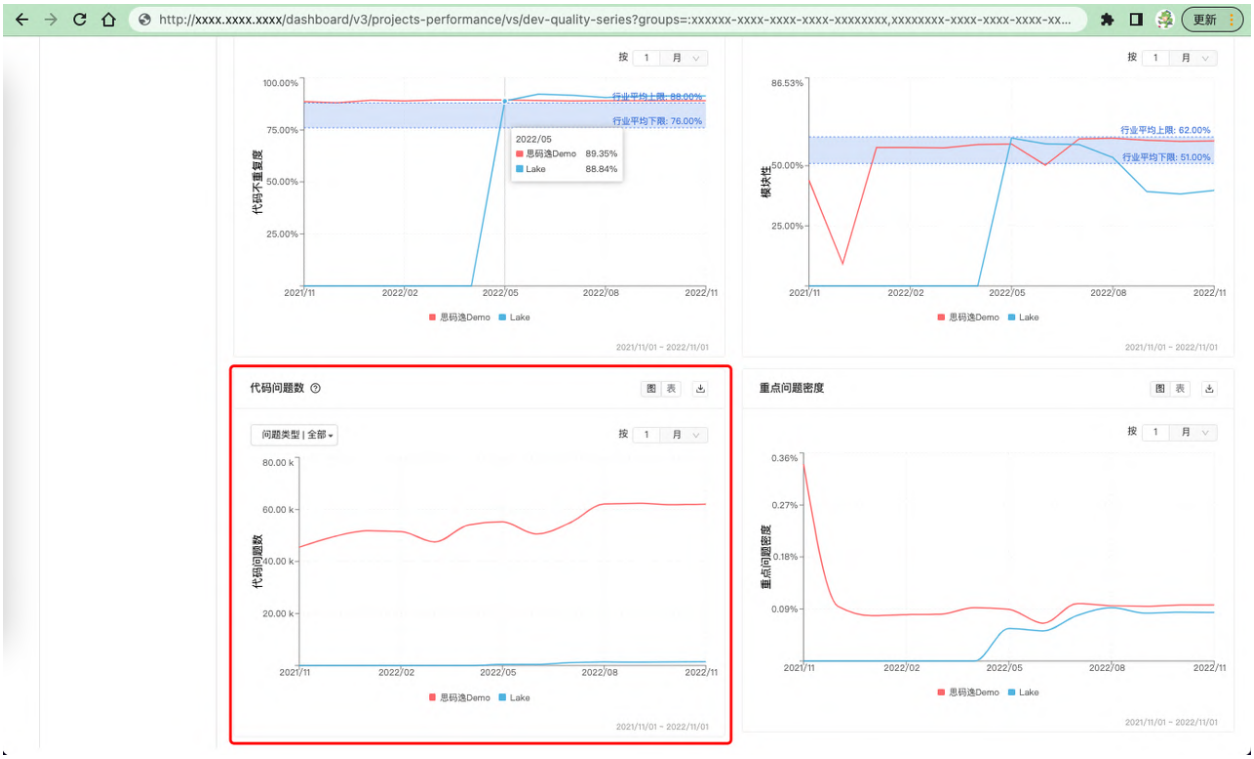

图 200: 代码问题数

4.2.4.2.6 重点问题密度

重点问题密度由选定时间段内的重点问题数除以代码当量计算得出，重点问题包括阻塞问题和严重问题。

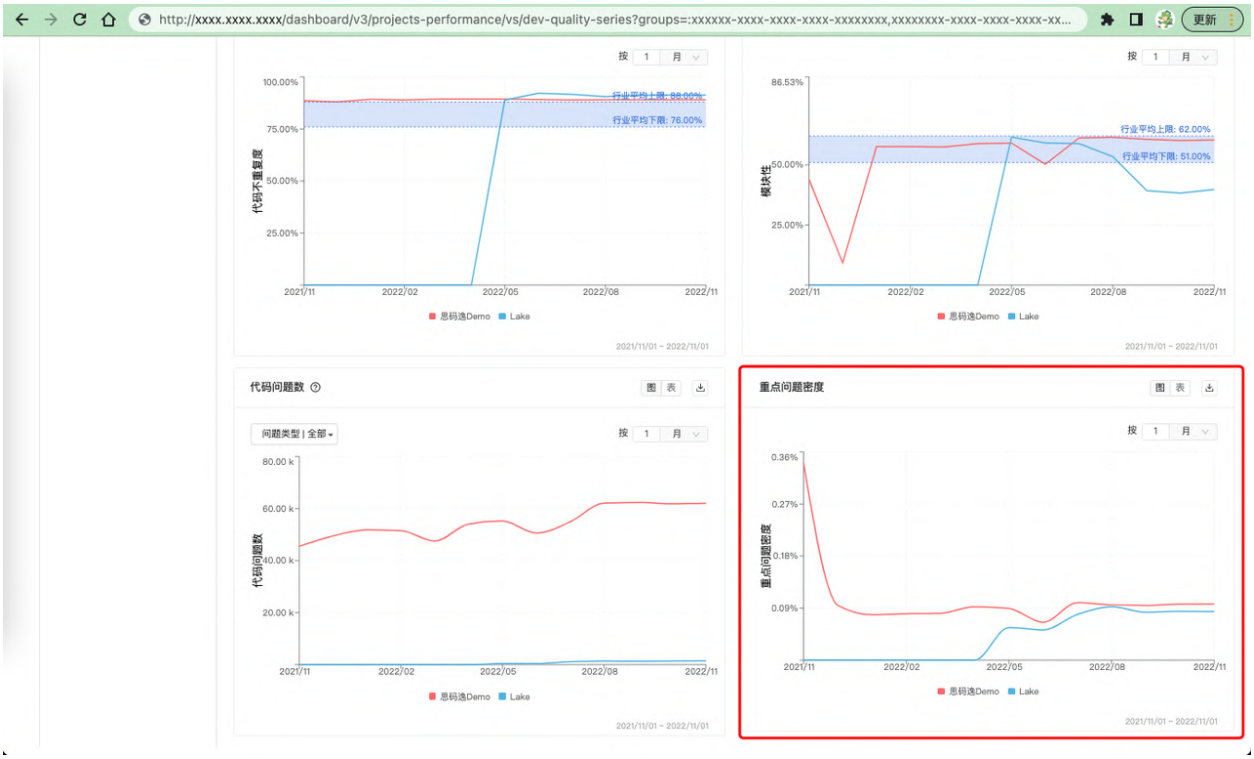

图 201: 重点问题密度

4.2.5 同行对比

4.2.5.1 要点概述

- **导航：**左侧菜单栏 **项目表现 - 同行对比**。
- 可以选择**单个**代码库或者项目组范围进行同行对比。
- 查看**效率**相关行业指标时，按照**开发语言**和**成员规模**综合匹配最符合的行业对比代码库生成行业指标信息。== **注意：代码库/项目组人数超过 300 时，匹配不到行业对比代码库。** ==
- 查看**质量**相关行业指标时，按照**开发语言**匹配最符合的行业对比代码库生成行业指标信息。== **注意：代码库/项目组总代码当量超过 20 亿时，匹配不到行业对比代码库。** ==

4.2.5.2 效率

展示效率相关的行业指标，以及此项目组/代码库效率与行业指标的对比。可选时间范围。

4.2.5.2.1 项目组/代码库信息

显示当前项目组/代码库名称，成员（贡献者）数量，涉及的开发语言。

4.2.5.2.2 行业信息

显示语言和成员规模综合匹配下，最符合的行业对比代码库的数量，及其涉及到的语言。

4.2.5.2.3 累积当量趋势

- **图界面：**浅蓝色区域显示作为对比行业信息的代码库里累计当量趋势的上四分位、中位、下四分位区域，绿色的曲线显示当前项目组/代码库实际累计当量趋势，可以对比了解当前项目组/代码库代码当量累计趋势在行业里的相对水平。
- **表界面：**切换到表界面可以看到具体的数值，也可以下载对应的数据表格。
- **可改步长：**一周/两周/一个月/两个月。
- 示例：

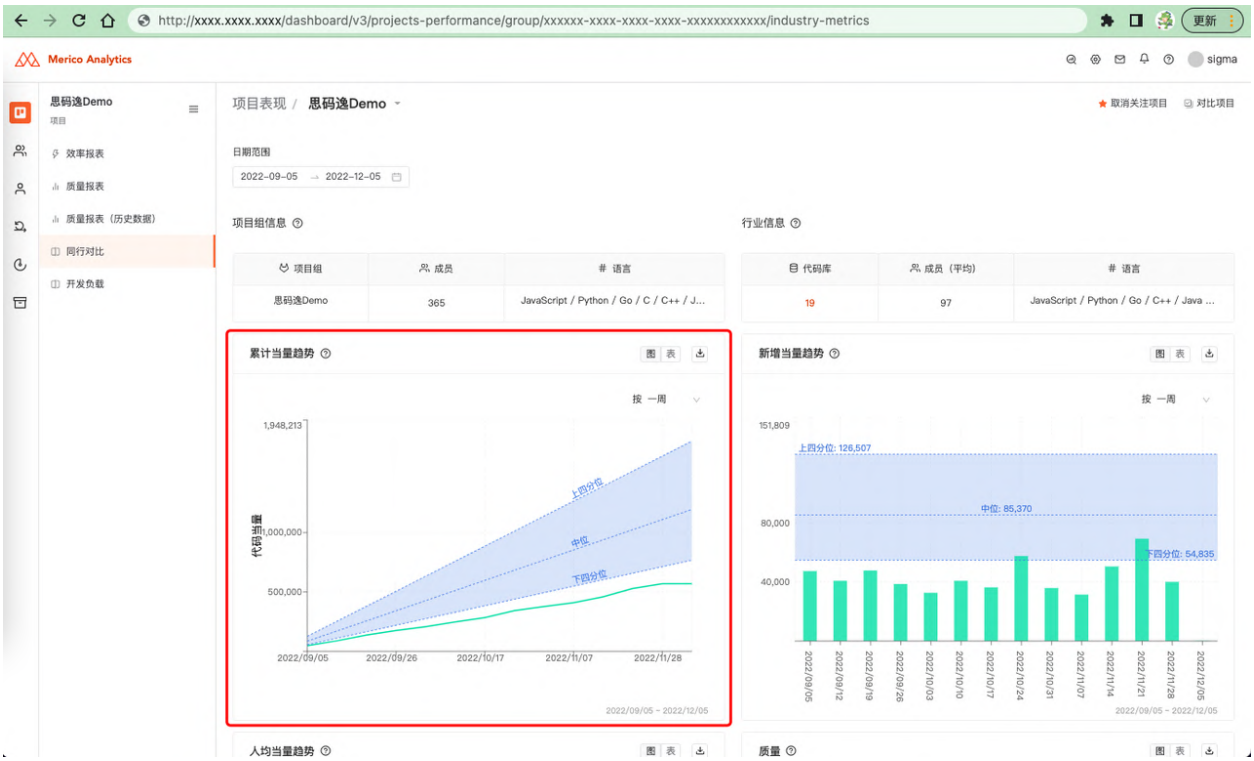

图 202: 累积当量趋势

4.2.5.2.4 新增当量趋势

- **图界面：**浅蓝色区域显示作为对比行业信息的代码库里新增当量的上四分位、中位、下四分位区域，绿色的柱状图显示当前项目组/代码库实际在步长时间范围的新增当量值，可以对比了解当前项目组/代码库新增代码当量在行业里的相对水平。
- **表界面：**切换到表界面可以看到具体的数值，也可以下载对应的数据表格。
- **可改步长：**一周/两周/一个月/两个月。
- 示例：

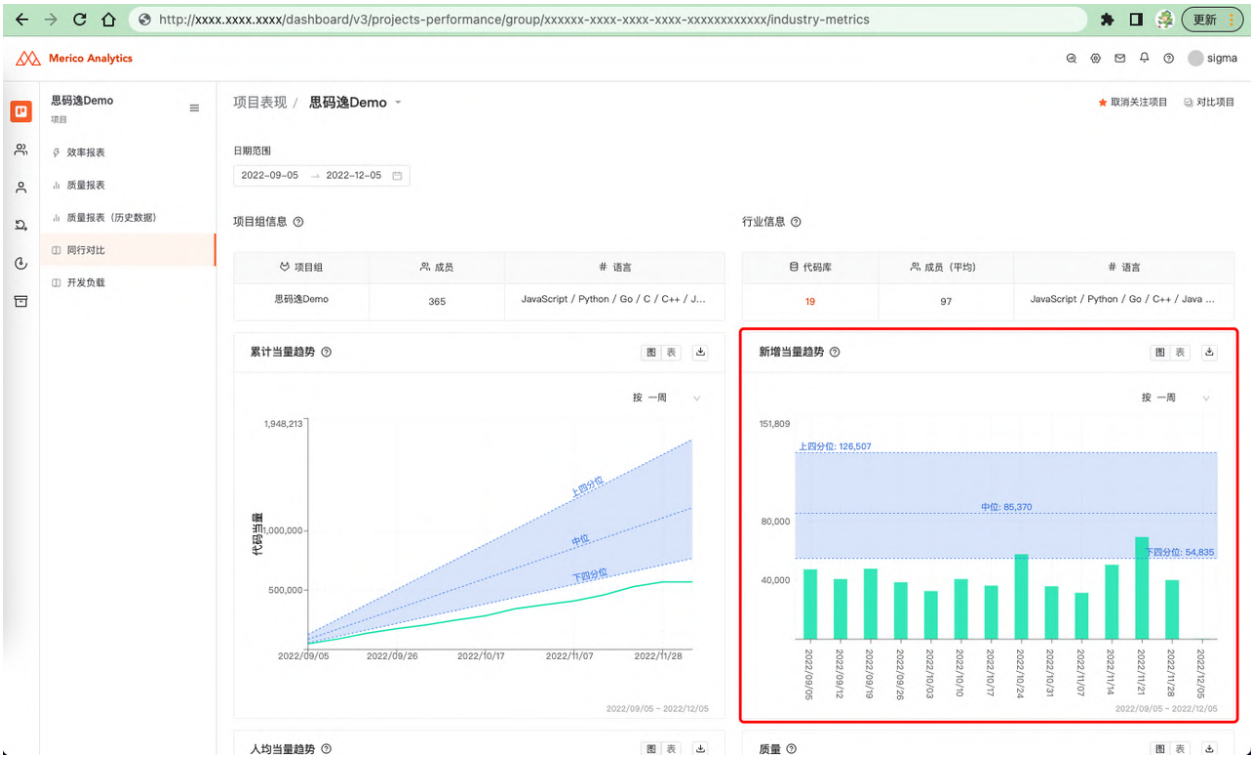

图 203: 新增当量趋势

4.2.5.2.5 人均当量趋势

- **图界面：**浅蓝色区域显示作为对比行业信息的代码库里人均代码当量的上四分位、中位、下四分位区域，绿色的柱状图显示当前项目组/代码库实际在步长时间范围的人均代码当量值，可以对比了解当前项目组/代码库人均代码当量在行业里的相对水平。
- **表界面：**切换到表界面可以看到具体的数值，也可以下载对应的数据表格。
- **可改步长：**一周/两周/一个月/两个月。
- 示例：

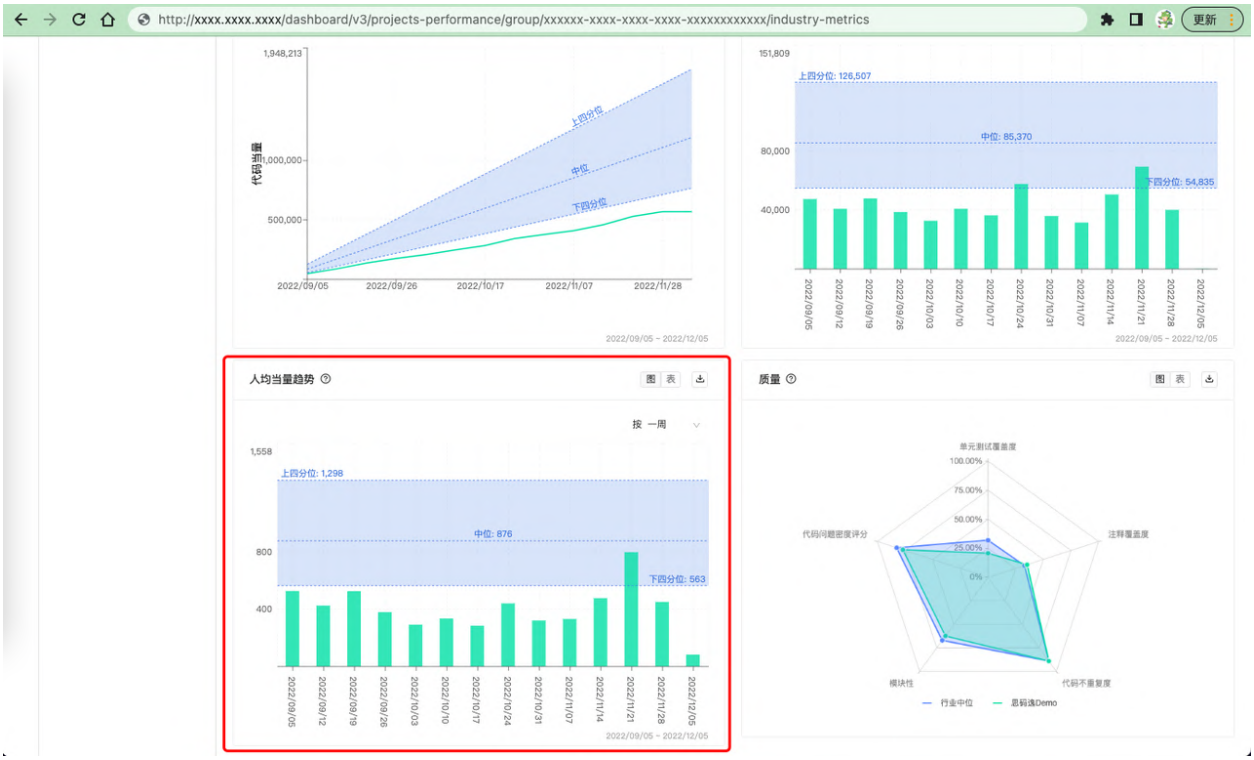

图 204: 人均当量趋势

4.2.5.3 质量

- 以雷达图的形式展示质量相关的行业指标与此项目组/代码库质量指标的对比。展示指标：测试覆盖度、注释覆盖度、代码不重复度、模块性、问题率。
- **注意，在当前版本中，质量指标匹配到的行业对比代码库基础信息如代码库名称、团队人数、开发语言没有在界面上展示。**
- 蓝色部分为行业中位水平，绿色部分为当前项目组/代码库水平。
- 示例：

图 205: 质量

4.2.6 开发负载

4.2.6.1 要点概述

- **导航：**左侧菜单栏 **项目表现 - 开发负载**。
- 在此界面展示了人均生产率趋势、年度同期生产率均值之比（排除异常点）、贡献者帕累托、开发活动热力图，用以判断人均开发工作量的负荷程度，贡献是否均衡，以及判断资源的利用是否合理。

4.2.6.2 负载数据

4.2.6.2.1 人均产出趋势

- 通过呈现生产率趋势指数，得出开发负载相关结论
- 示例

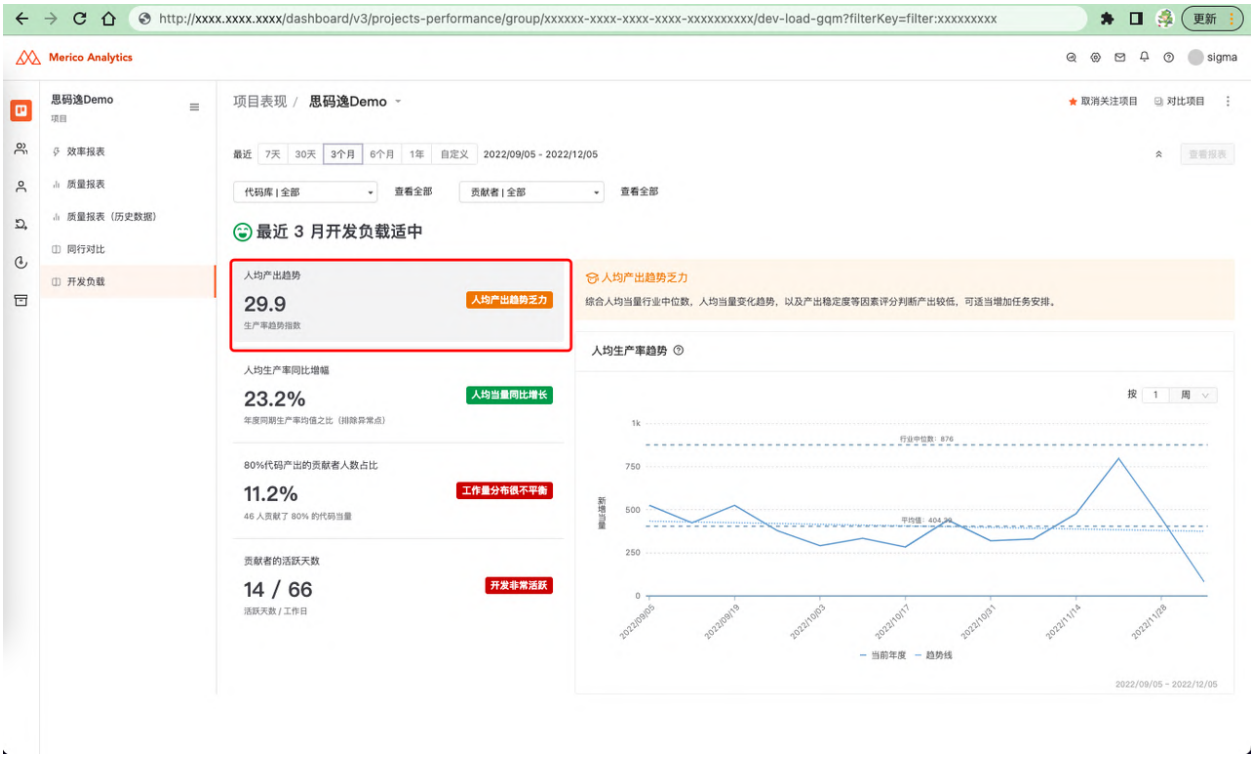

图 206: 人均产出趋势

4.2.6.2.2 人均生产率同比增幅

- 通过呈现年度同期生产率均值之比（排除异常点），得出开发负载相关结论
- 示例

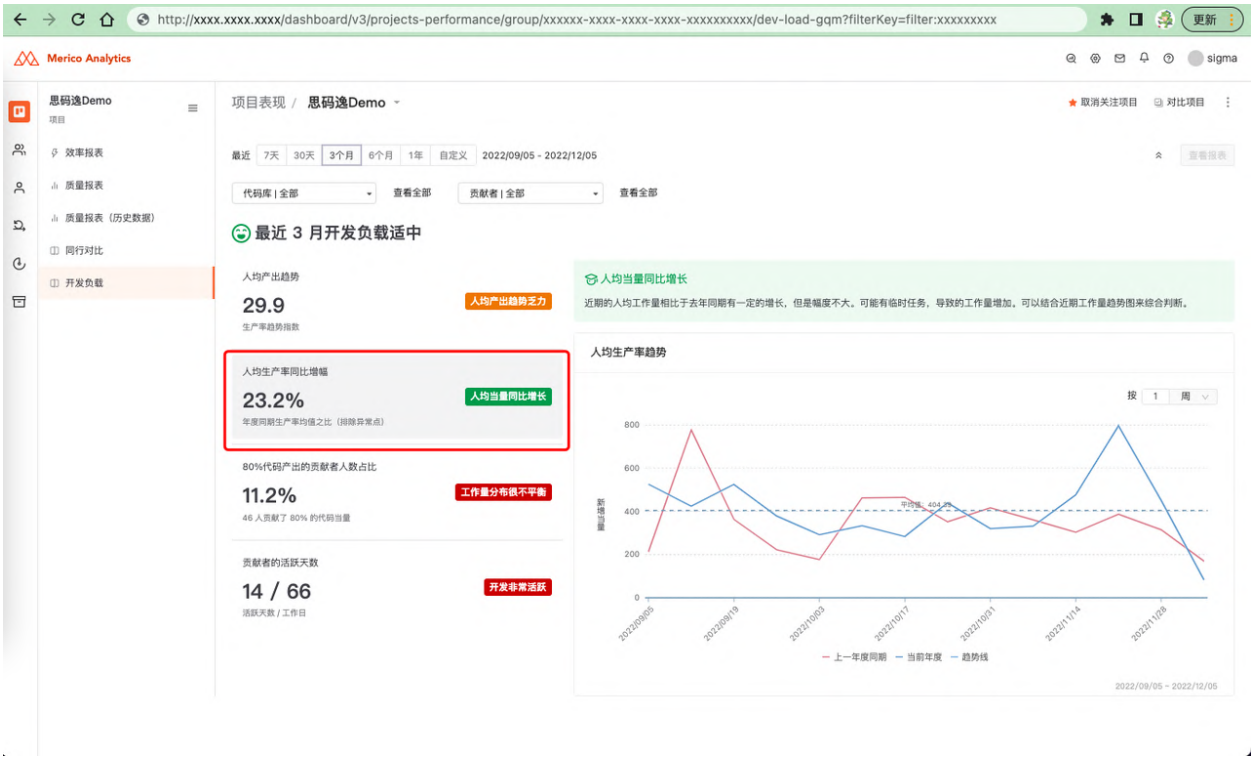

图 207: 人均生产率同比增幅

4.2.6.2.3 80% 代码产出的贡献者人数占比

- 通过呈现贡献者帕累托，得出开发负载相关结论
- 示例：

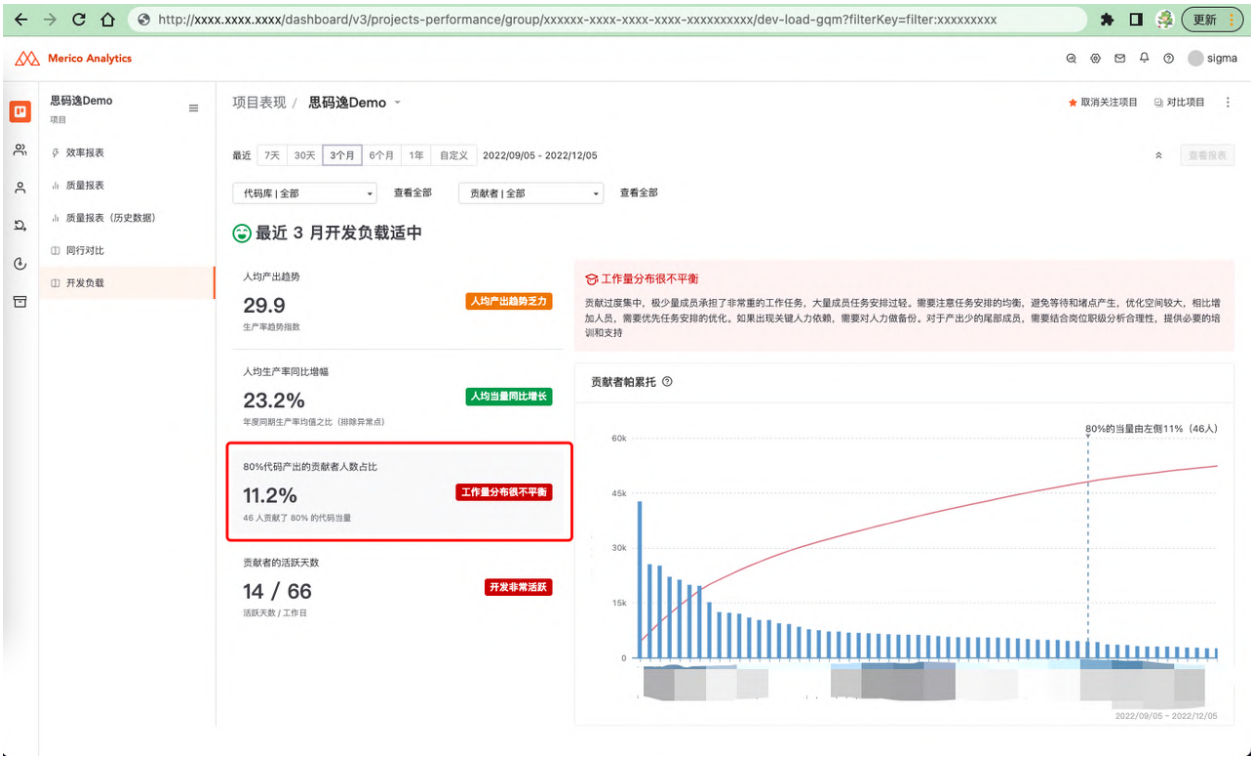

图 208: 80% 代码产出的贡献者人数占比

4.2.6.2.4 贡献者的活跃天数

- 通过呈现开发活动热力图，得出开发负载相关结论
- 示例：

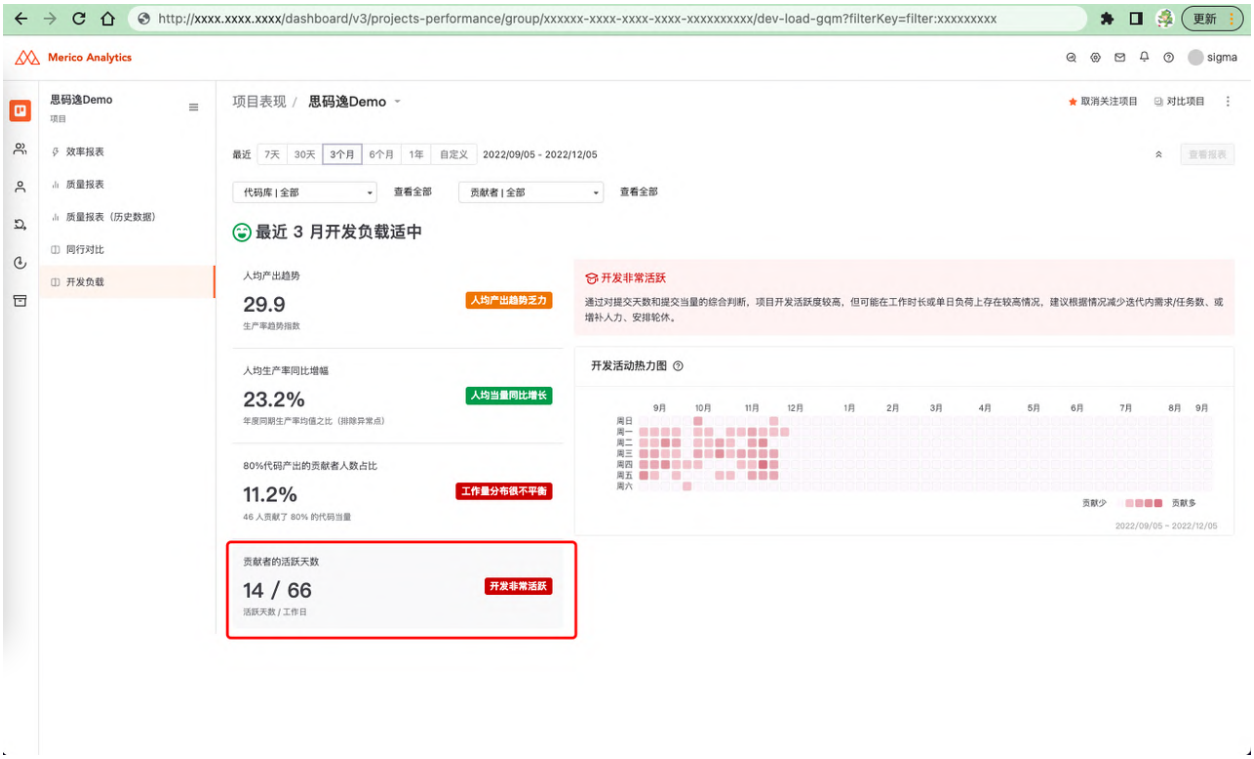

图 209: 贡献者的活跃天数

4.3 团队表现

4.3.1 团队表现

4.3.1.1 要点概述

- 在团队表现版块下，可以从人员聚合为团队的维度查看开发效能、质量指标数据。
- 当前团队表现 **仅**支持查看单个团队的效率报表。

4.3.1.2 导航说明

4.3.1.2.1 基础导航

- 左侧菜单栏 **团队表现** -> 点击某个团队 -> 进入团队表现详情界面。

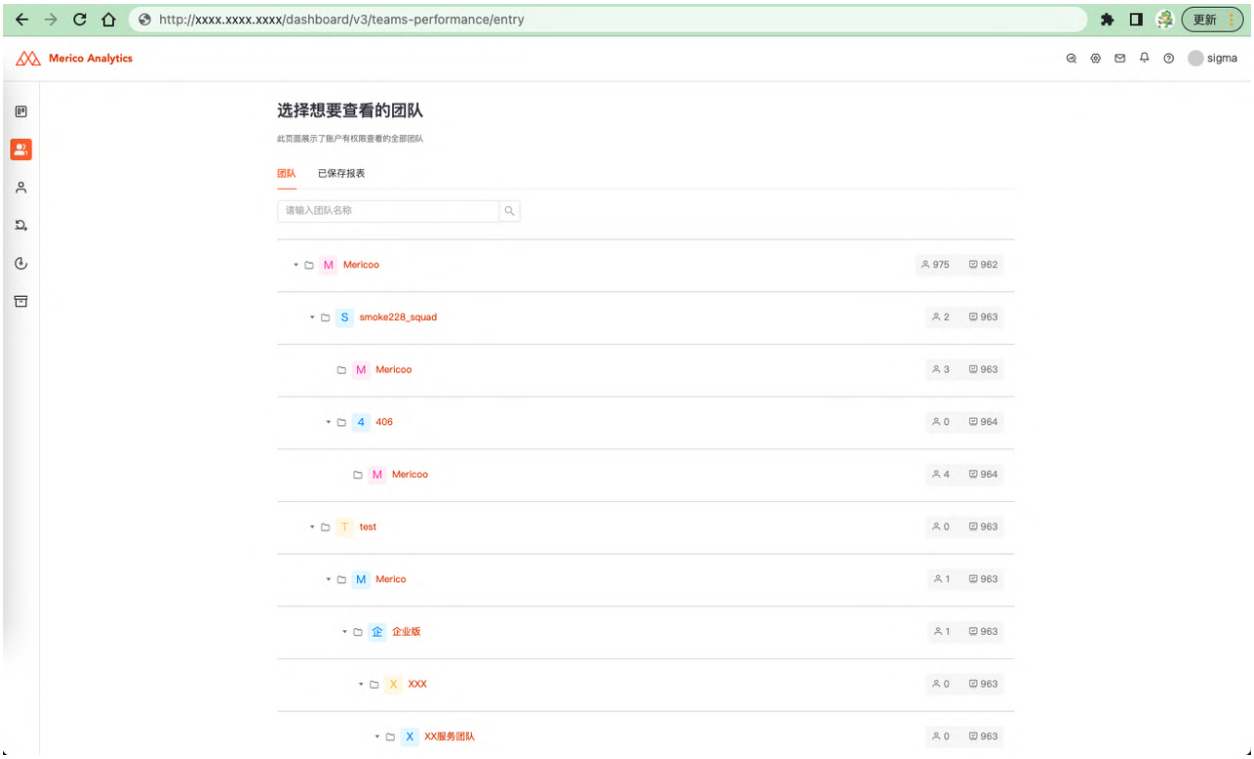

图 210: 选择想要查看的团队

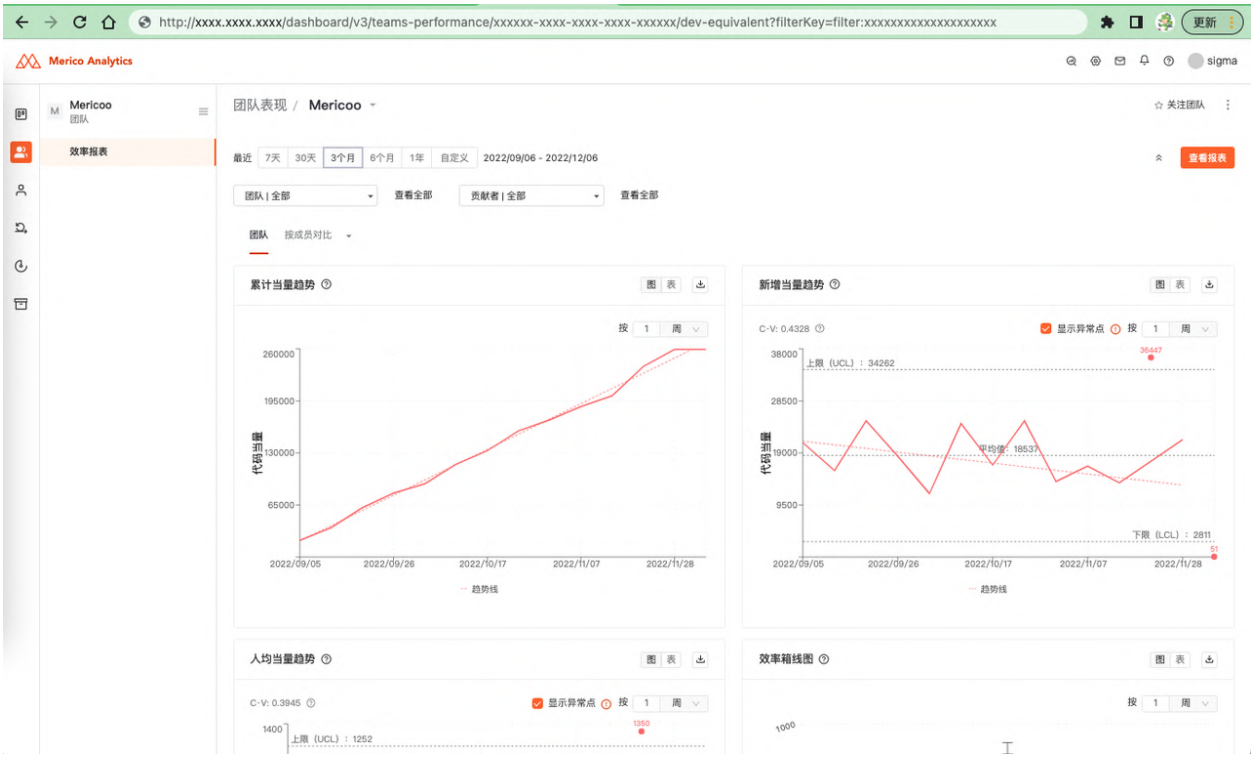

图 211: 进入团队表现详情界面

4.3.1.2.2 分类对比

- 在团队表现版块，支持按团队成员进行数据对比查看，也支持按自定义字段如职位、岗级等分类聚合团队成员进行数据对比查看。

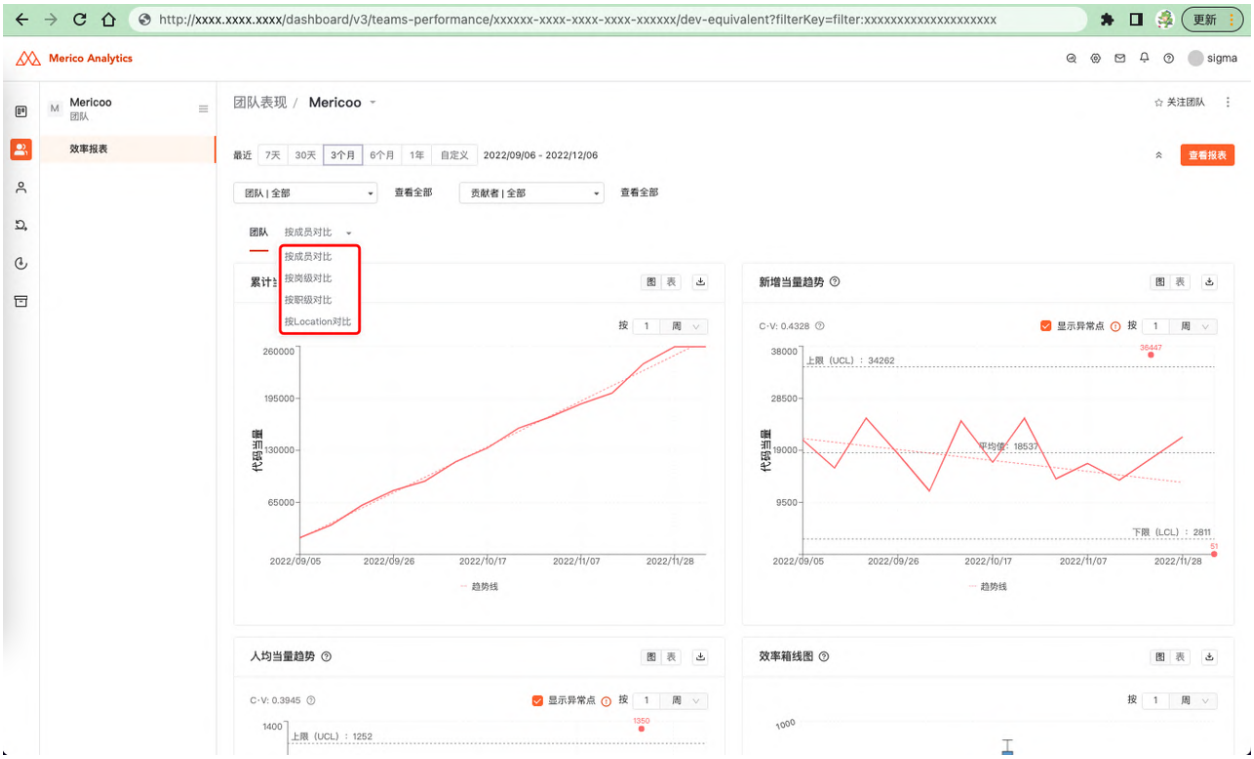

图 212: 按自定义字段对比和查看团队表现数据

4.3.1.2.3 单选团队查看数据

同项目表现

4.3.1.2.4 筛选器

同项目表现

4.3.1.2.5 报表保存和下载

同项目表现

4.3.1.2.6 快捷操作

同项目表现

4.3.2 效率报表

4.3.2.1 要点概述

- **导航**：左侧菜单栏 **团队表现 - 效率报表**。
- 可以从 **单个团队 / 团队成员对比 / 自定义标签聚合团队成员对比**三个不同维度来查看效率指标报表。

- 在此工作区界面中，可以查看如下以代码当量为基础得到的指标图表，具体包括：

单个团队维度	团队成员对比维度	自定义标签对比维度
累计当量趋势新增当量趋势人均当量趋势效率箱线图	代码当量帕累托图累计当量趋势新增当量趋势当量占比	代码当量帕累托图累计当量趋势新增当量趋势当量占比

4.3.2.2 效率报表

4.3.2.2.1 累计当量趋势

同项目表现

4.3.2.2.2 新增当量趋势

同项目表现

4.3.2.2.3 人均当量趋势

同项目表现

4.3.2.2.4 代码当量帕累托图

同项目表现

4.3.2.2.5 当量占比

同项目表现

4.4 工程师表现

4.4.1 工程师表现

在工程师表现版块下，可以查看如下与工程师（贡献者）表现相关的指标和数据：

- **贡献者排行：**在此界面展示工程师（贡献者）在代码当量、代码影响力、代码问题密度、单测覆盖度、注释覆盖度、效率质量综合等指标上的排行信息。
- **贡献者列表：**在此界面可查看各工程师（贡献者）的包括代码当量、代码影响力、提交数、行数变化等效能相关数据。
- **提交列表：**在此界面可查看各提交的具体效能数据和提交信息。

4.4.2 贡献者排行

4.4.2.1 要点概述

- **导航：**左侧菜单栏 **工程师表现 - 贡献者排行**。
- 在贡献者排行界面可以选择不同**时间范围**查看排行榜数据。选择了时间范围后仅显示 **代码当量榜**和 **代码影响力榜**。
- 可以选择查看贡献者信息的数据范围：某个代码库或某个项目组。

4.4.2.2 包含榜单

4.4.2.2.1 效率质量综合榜

- 结合了效能和质量指标：代码当量、开发价值（代码影响力）、代码质量、测试覆盖度和注释覆盖度，进行综合计算的榜单，各指标权重可以根据需要自行调整。
- 各指标权重配置在：**设置 -> 排行榜设置**

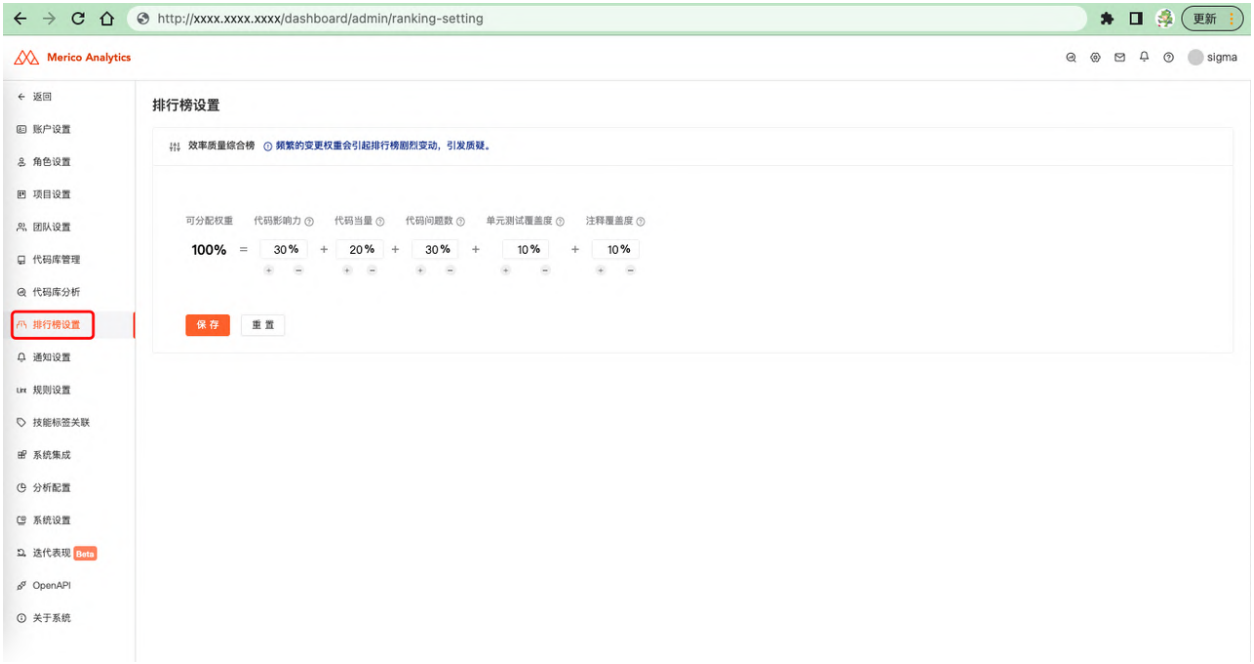

图 213: 指标权重配置

图 214: 效率质量综合榜 - 代码当量榜 - 代码影响力榜

4.4.2.2.2 代码当量榜

- 显示代码库/项目组范围内，选定时间范围内，代码当量值排名前 10 名。
- 示例：见效率质量综合榜

4.4.2.2.3 代码影响力榜

- 显示代码库/项目组范围内，代码影响力排名前 10 名。
- 示例：见效率质量综合榜

4.4.2.2.4 代码问题密度评分榜

- 显示代码质量排行前 10 的贡献者，计算规则为每 1000 当量产生的 issue 数，千当量 issue 数越少代码质量越高。
- 示例：

图 215: 代码问题密度评分榜 - 单测覆盖度榜 - 注释覆盖度榜

4.4.2.2.5 单测覆盖度榜

- 显示代码库/项目组范围内，函数测试覆盖度排名前 10 名。
- 示例：见代码问题密度评分榜

4.4.2.2.6 注释覆盖度榜

- 显示代码库/项目组范围内，函数注释覆盖度排名前 10 名。
- 示例：见代码问题密度评分榜

4.4.3 贡献者列表

4.4.3.1 要点概述

- **导航：**左侧菜单栏 **工程师表现 - 贡献者列表**。
- 可以选择查看贡献者信息的数据范围：某个代码库或某个项目组。
- 可以选择贡献者邮箱、日期范围来搜索贡献者贡献信息。
- 点击 **导出数据**按钮可将搜索得到的贡献者列表信息导出。
- 点击贡献者列表中任意贡献者姓名或头像，进入该贡献者 **贡献者报告**界面。

4.4.3.2 贡献者列表各字段含义

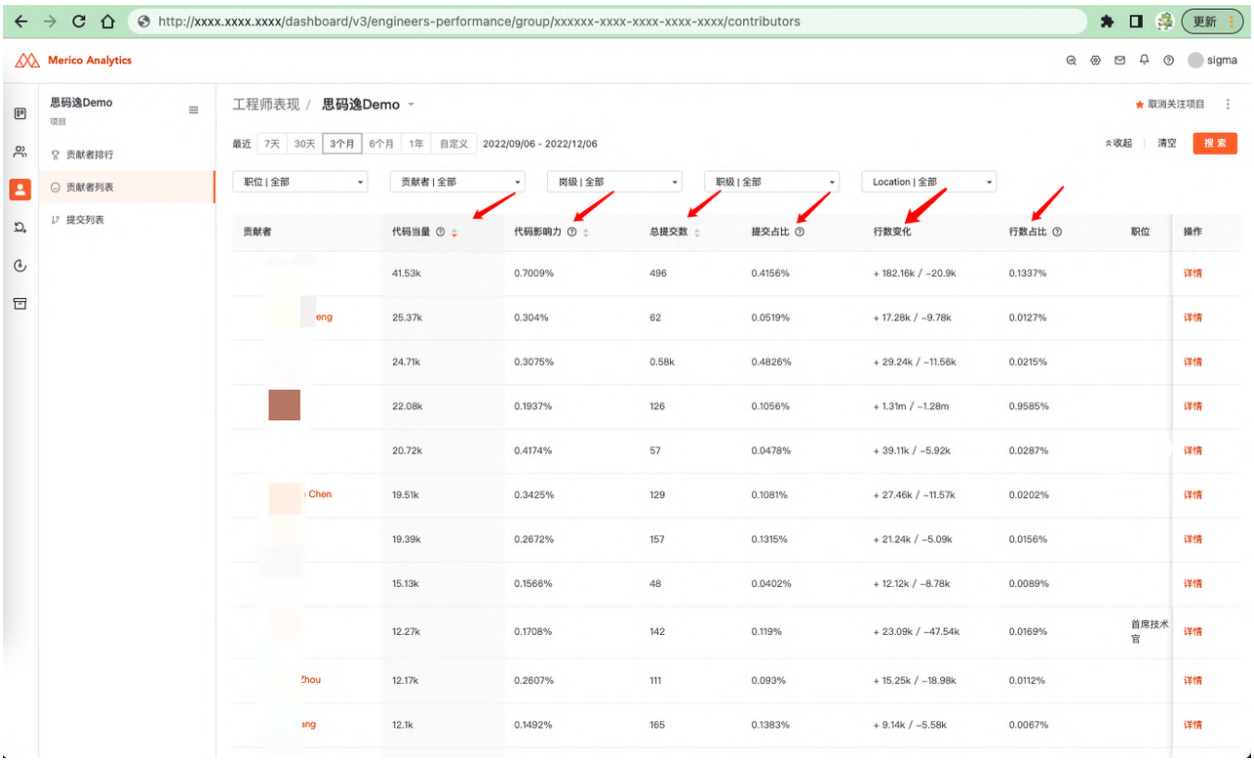

图 216: 贡献者列表字段展示

- **贡献者：**该贡献者头像和姓名。
- **代码当量：**对开发工作量的合理估计，通过开发过程对抽象语法树的更改量计算开发产出。与代码行数、提交数等指标相比，不易受到编程习惯或特定行为的干扰（如换行、添加注释等）。
- **代码影响力：**代码影响力是综合了代码当量和调用关系的综合指数。默认列表按照开发价值按从高到低排列。
- **总提交数：**该贡献者在查询代码范围和时间范围的提交总数。
- **提交占比：**当前提交数占开发历史全部提交数的比值。
- **行数变化：**此提交产生的行数变化，增加了多少行和减少了多少行。
- **行数占比：**当前代码行数占开发历史全部代码行数的比值。
- **价值密度（行）：**开发价值/行数占比。反映每行代码的含金量。
- **职位：**此贡献者职位。
- **操作：**点击 **详情**，进入 **贡献者报告**页面，查看此贡献者详细信息。

4.4.4 提交列表

4.4.4.1 要点概述

- **导航：**左侧菜单栏 **工程师表现 - 提交列表**。
- 可以选择查看提交信息的数据范围：某个代码库或某个项目组。
- 可以选择提交时间范围、贡献者、提交标题关键字来搜索提交。

4.4.4.2 提交列表各字段含义

- **提交信息：**该提交贡献者姓名和主邮箱。
- **代码当量：**对开发工作量的合理估计，通过开发过程对抽象语法树的更改量计算开发产出。与代码行数、提交数等指标相比，不易受到编程习惯或特定行为的干扰（如换行、添加注释等）。
- **代码影响力：**代码影响力是综合了代码当量和调用关系的综合指数。
- **行数变化：**此提交产生的行数变化，增加了多少行和减少了多少行。
- **行数占比：**当前代码行数占开发历史全部代码行数的比值。
- **价值密度：**开发价值/行数占比。反映每行代码的含金量。
- **创建时间：**此提交创建的时间。
- **提交时间：**此提交的提交时间。
- **操作：**点击 **详情**可查看此提交详细信息；点击 **备注**可给此提交添加备注；点击 **排除提交**按钮可将此提交排除，重新分析此代码库后此提交当量变为 0。

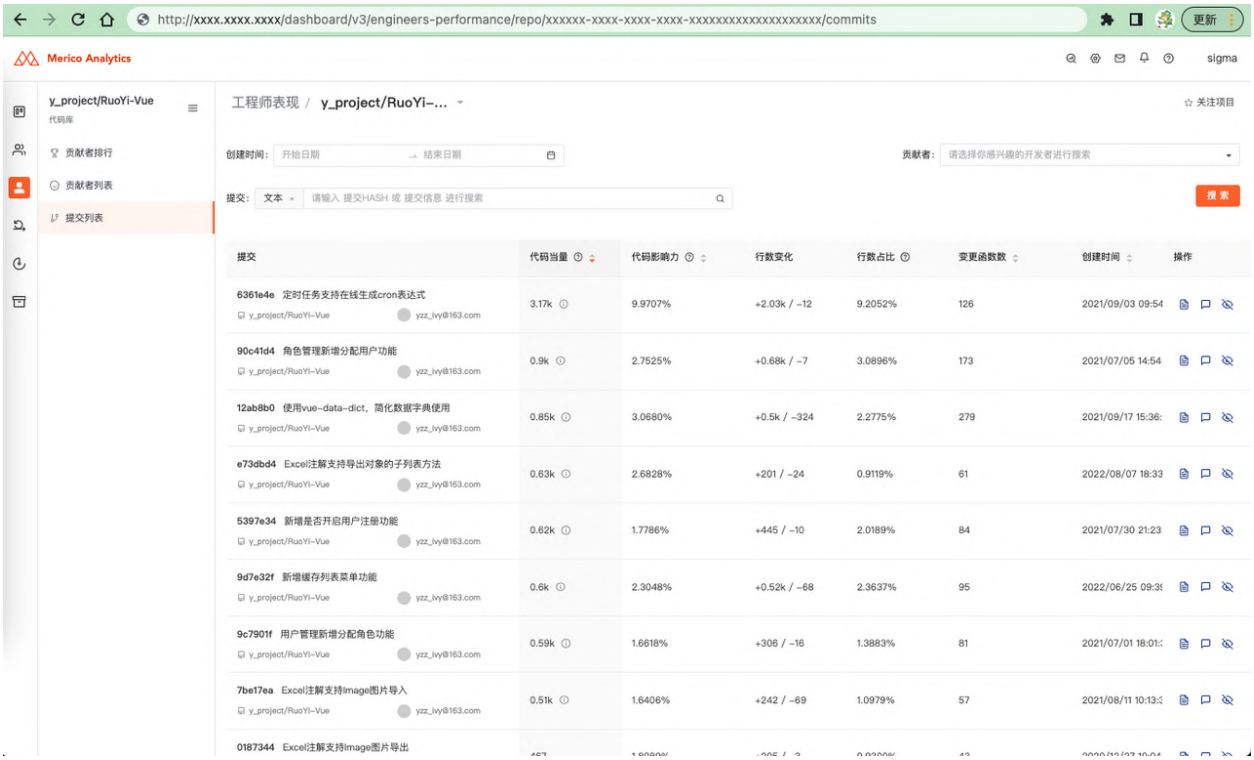

图 217: 提交列表

4.4.4.3 提交详情

4.4.4.3.1 基本信息

- 点击某提交的 **详情**链接，跳转到此提交的详情界面，查看此提交详细信息。
- 在提交详情界面的提交信息处可查看此提交 Hash 信息。
- 复杂函数：此提交变更函数中圈复杂度超过 10 的函数数量。
- 可分别查看此提交**变更文件**信息和**变更函数**信息。
- 提交详情界面上可以给此提交添加备注或排除此提交，也可以排除此提交内的某文件。

4.4.4.3.2 变更文件

- 变更文件列表显示此提交内变更的各文件名、该变更文件对应的代码影响力和代码当量。
- 点击变更文件名，显示该文件源码。

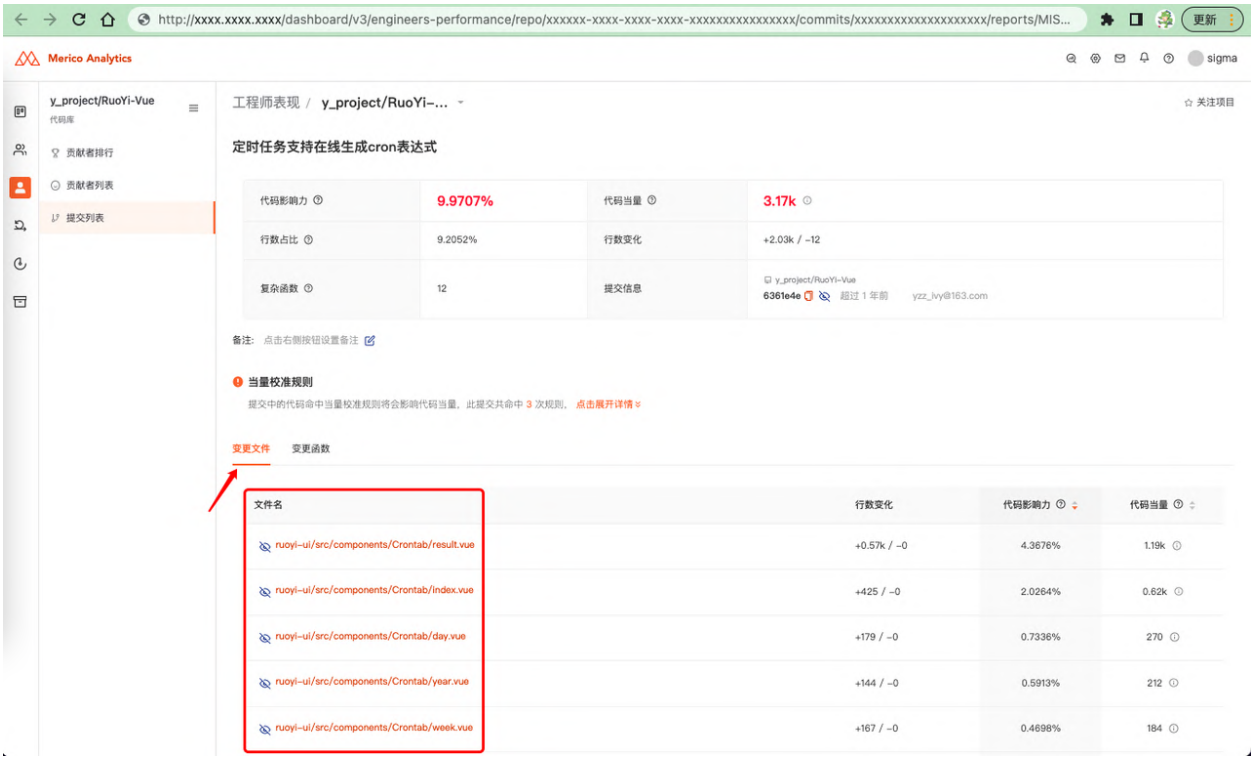

图 218: 变更文件

4.4.4.3.3 变更函数

- 变更函数列表显示此提交变更各函数签名、该变更函数对应的代码影响力、代码当量、添加行数、删除行数、入度和出度。

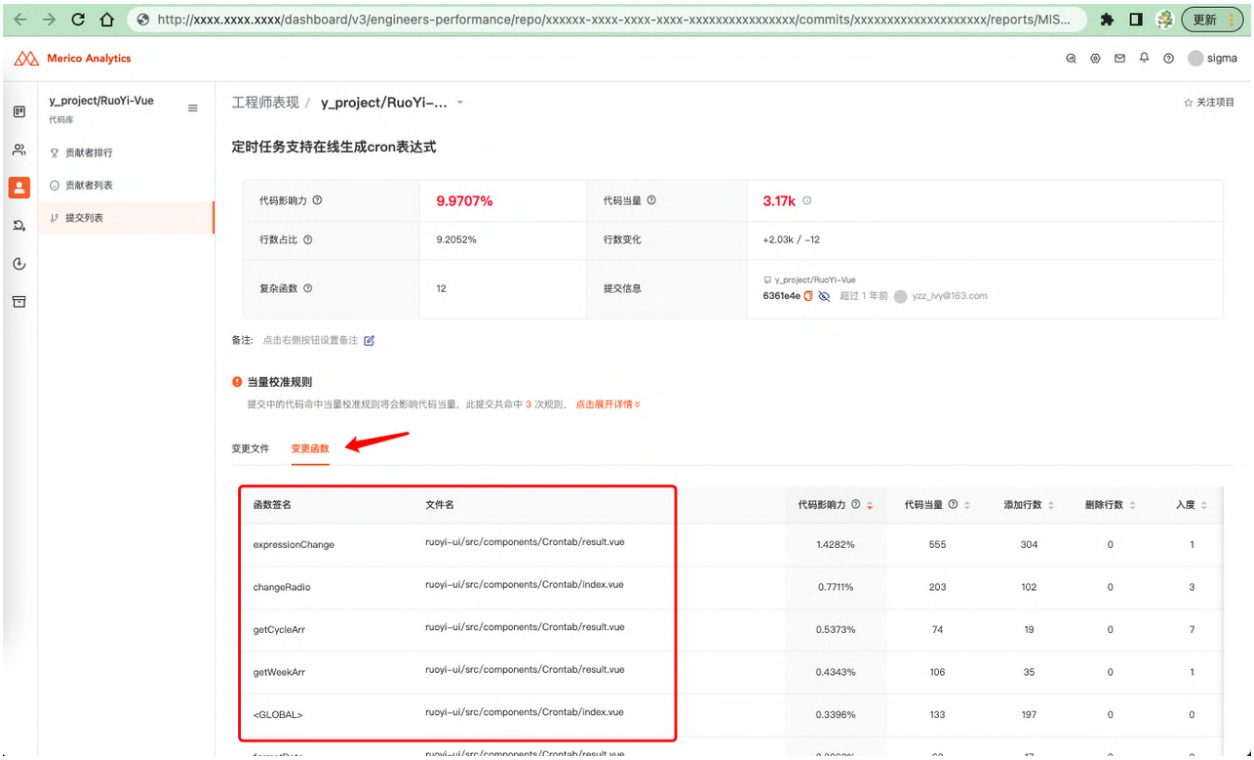

图 219: 变更函数

4.5 迭代表现

4.5.1 迭代表现

在迭代表现版块下，可以查看与迭代表现相关的指标和数据：迭代进度，迭代的事务/故事点燃尽图，累计代码当量趋势，每日新增代码当量，人均流负载，人均工作负荷，项目 bug 数据，项目组成员工作质量数据等。

迭代表现版块里包含如下子版块：

- **工作面板：**包括迭代事务/故事点燃尽图，迭代内累计代码当量，迭代内剩余 Bug 统计，累计工时统计，迭代内按事务计算的迭代计划完成率，与未完成的事务列表。
- **工作量：**包括迭代内每日新增代码当量、累计代码当量、每日完成事务、累计完成事务、每日完成故事点、累计完成故事点、每日新增工时、累计工时数据统计。包括项目组成员工作量统计。
- **迭代质量：**包括迭代内累计新增/修复 Bug 数、每日新增/修复 Bug 数、项目组成员工作质量指标数据统计。
- **迭代进度：**列表展示迭代内各事务的进度状态。

4.5.2 迭代表现相关设置

4.5.2.1 要点说明

- 当前仅支持与 **8 及以上版本的 JIRA Software** 进行集成和迭代表现数据展示，不支持低版本 JIRA 集成，不支持其他研发项目管理平台集成。
- 当前仅支持将思码逸产品中的项目（组）与 JIRA **迭代面板**进行关联，如果 JIRA 端没有使用 **迭代面板**进行研发流程和项目管理，则无法和思码逸产品对接使用迭代表现模块功能。
- 如需统计每个事务的代码当量，需要将 Git 端的 Commit 与 JIRA 事务进行绑定设置。

4.5.2.2 需要设置

迭代表现相关设置如下，包括思码逸系统中与 JIRA 集成和对接的设置，也包括 JIRA 侧本身的迭代面板设置和 Git 与 JIRA 的绑定设置。

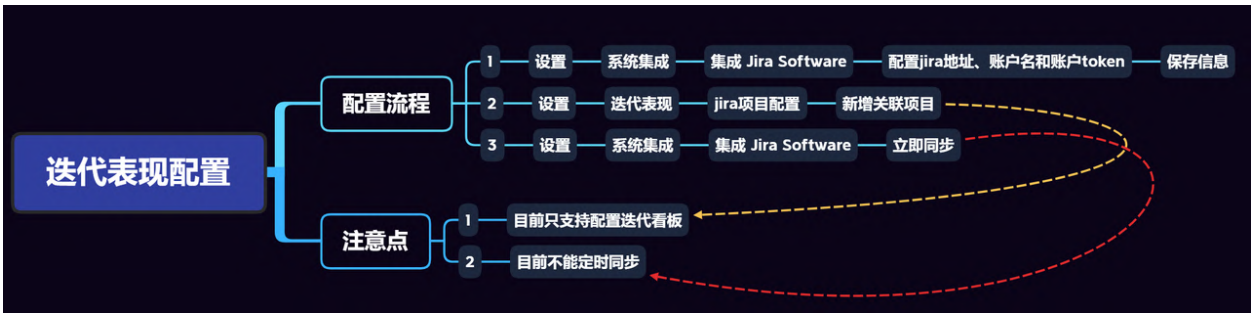

图 220: 迭代表现相关设置

4.5.2.2.1 JIRA 集成

- **导航：设置 - 系统集成 - 集成 JIRA Software**
- 支持与 JIRA Software **8** 及以上版本集成
- 需要 JIRA Software 的 **管理员** 权限
- 配置好 JIRA 集成信息后，点击 **立即同步** 按钮开始同步 JIRA 数据。

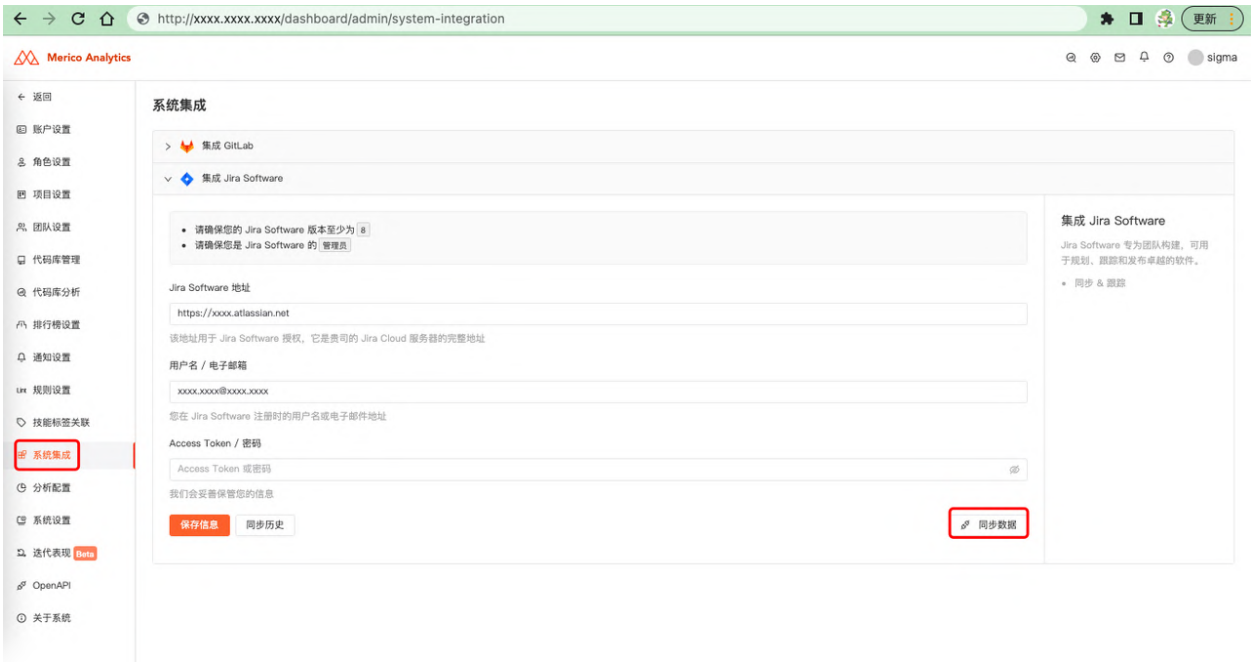

图 221: 系统集成配置

- JIRA 账号 Access Token 生成方法在 JIRA 系统中【点击头像】-【管理账户】-【安全性】-【API 令牌】-【创建 API 令牌】

图 222: JIRA 中管理账户

图 223: JIRA 中 API 令牌

图 224: JIRA 中创建 API 令牌

如果您的 JIRA 是私有部署，无法生成 Access Token，也可在系统集成的配置页面填写 JIRA 超级管理员的密码

4.5.2.2.2 迭代表现配置

- **导航：设置 - 迭代表现 - Jira**
- 支持将思码逸产品中的项目（组）与 JIRA **迭代面板**进行关联。
- 需要配置 Bug 类事务、需求类事务和 JIRA 端自定义事务类型的对应。
- 需要配置事务工作流中开发等待状态、开发进行状态和 JIRA 端自定义状态类型的对应。

配置步骤如下：

Step1：配置事务类型对应，配置项目组和迭代面板对应。

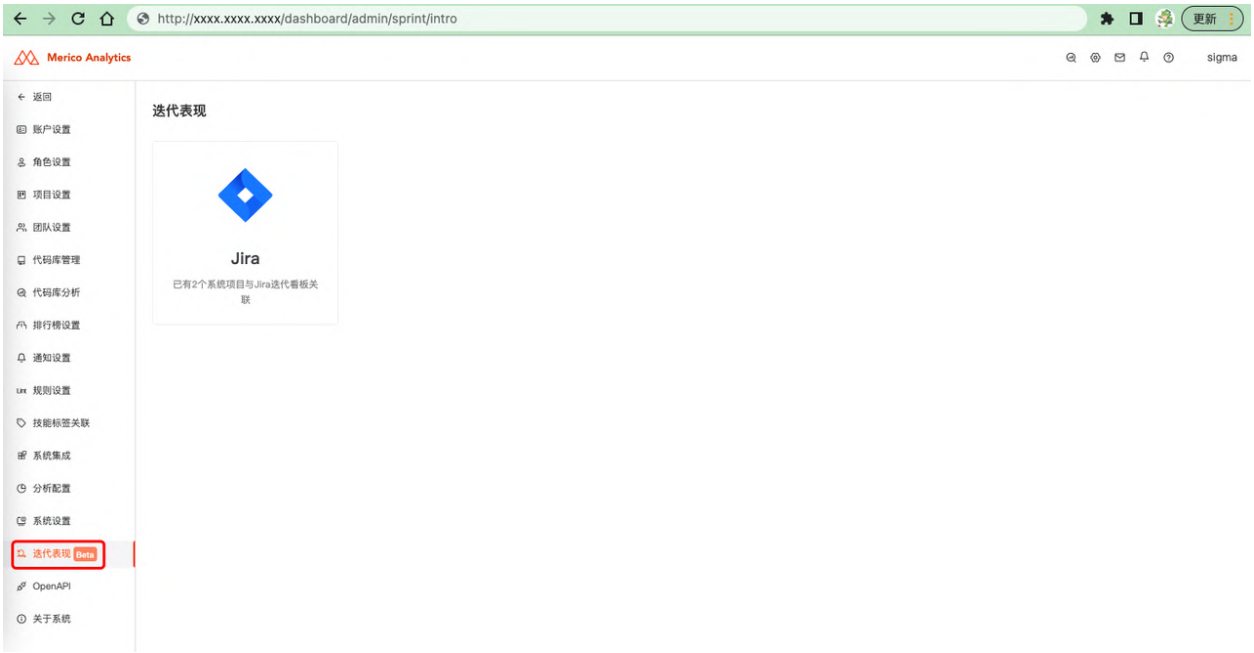

图 225: 迭代表现

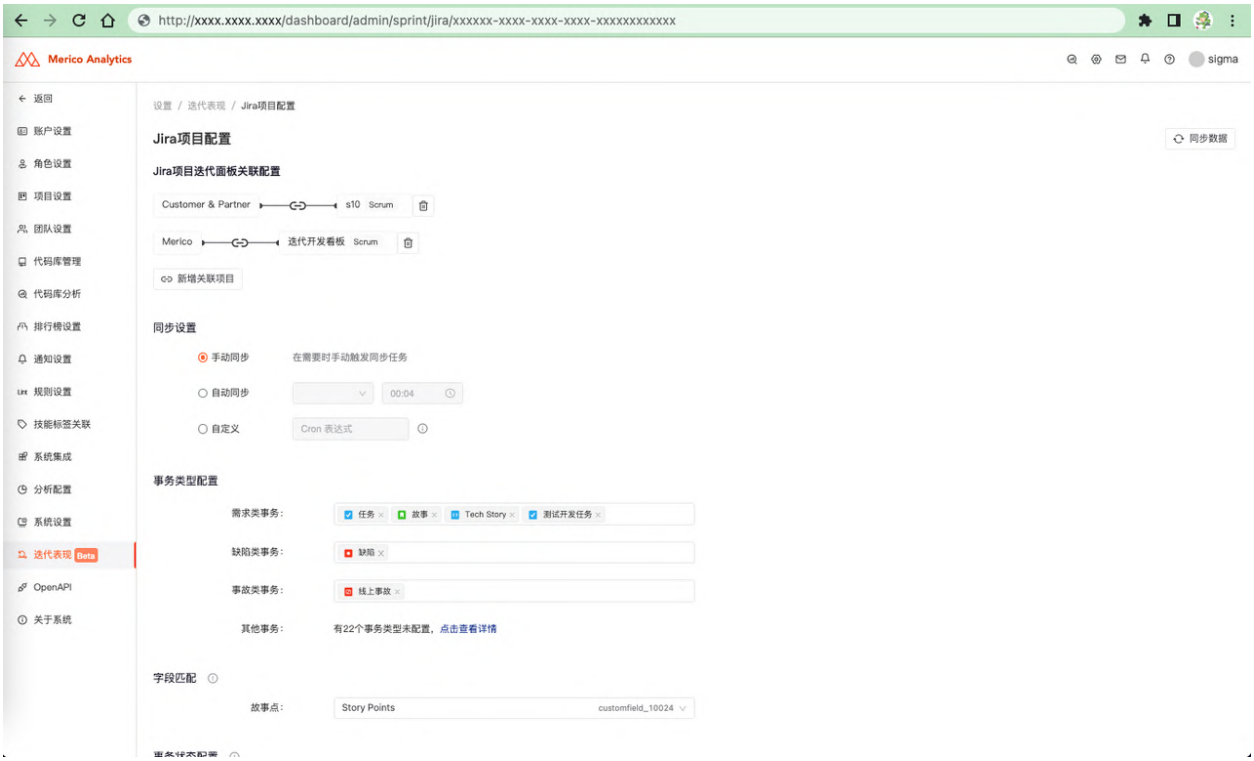

图 226: Jira 项目配置

Step2：配置迭代面板事务工作流状态对应以及其他事项。

包括：

1）Jira 项目迭代面板关联配置，该配置作用是关联思码逸系统中的项目与 Jira 中的看板

2）同步设置，该配置作用是设置同步时机，支持手动同步、自动同步和 Cron 表达式自动同步

3）事务类型配置，该配置作用是将 Jira 中的事务类型，按照思码逸系统中三种事务类型：需求类事务、缺陷类事务、事故类事务进行分类

4）字段匹配，该配置作用是配置以下字段后，可以在报告中展示相应数据

5）事务状态配置，暂时不支持事务状态的自定义配置。目前，系统是按照 Jira 中的状态类别来对各个事务状态进行分类

6）关键时间定义，系统默认定义，暂时不可更改，后续会支持自定义。

7）提交 commit 链接的正则规则，系统根据此规则，识别 Jira 事务中的提交信息，并将提交与事务关联，以显示当量数据。

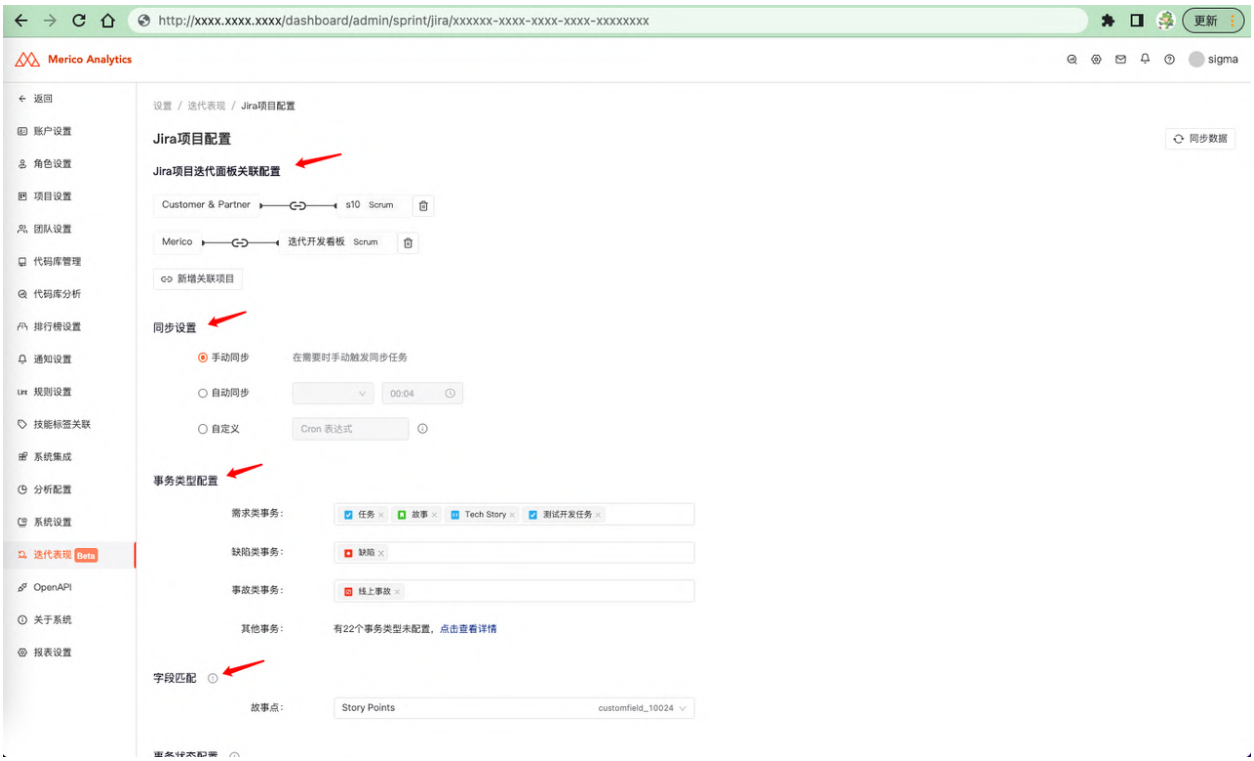

图 227: Jira 项目配置明细 1

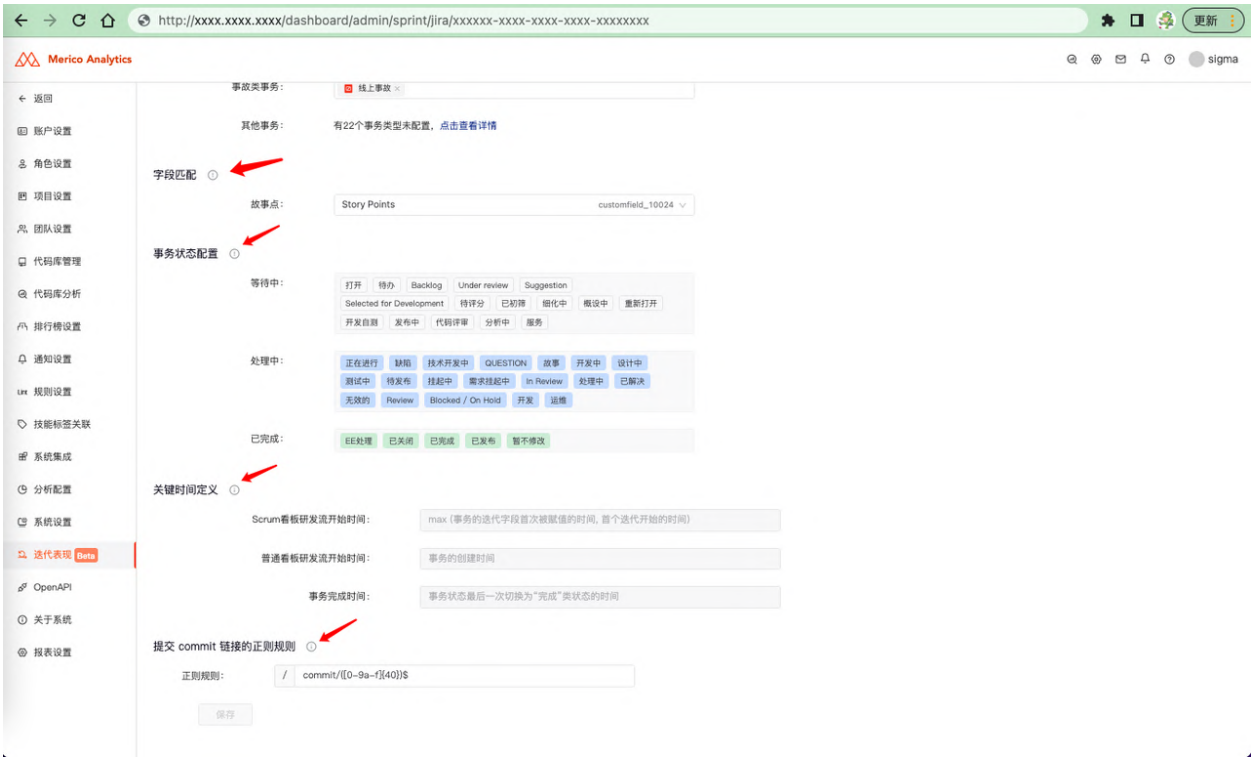

图 228: Jira 项目配置明细 2

4.5.2.2.3 JIRA 迭代面板配置

在 JIRA 项目中，创建 **Scrum 看板**，并在此看板中启动 **迭代（Sprint）**。

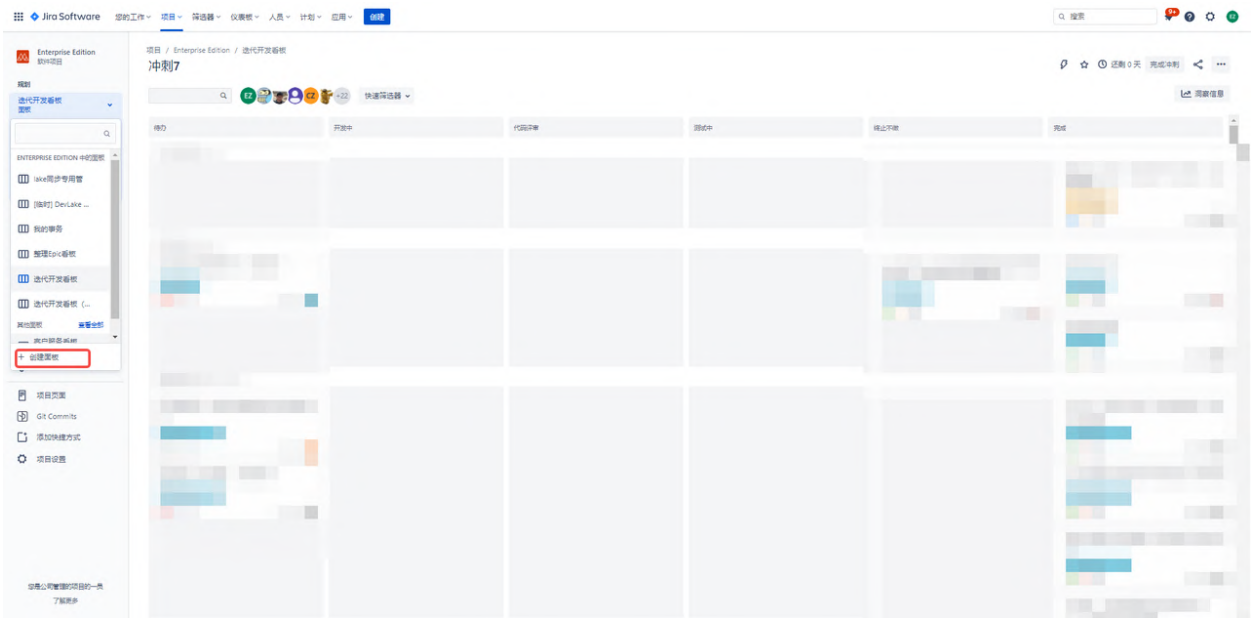

图 229: JIRA 中创建面板

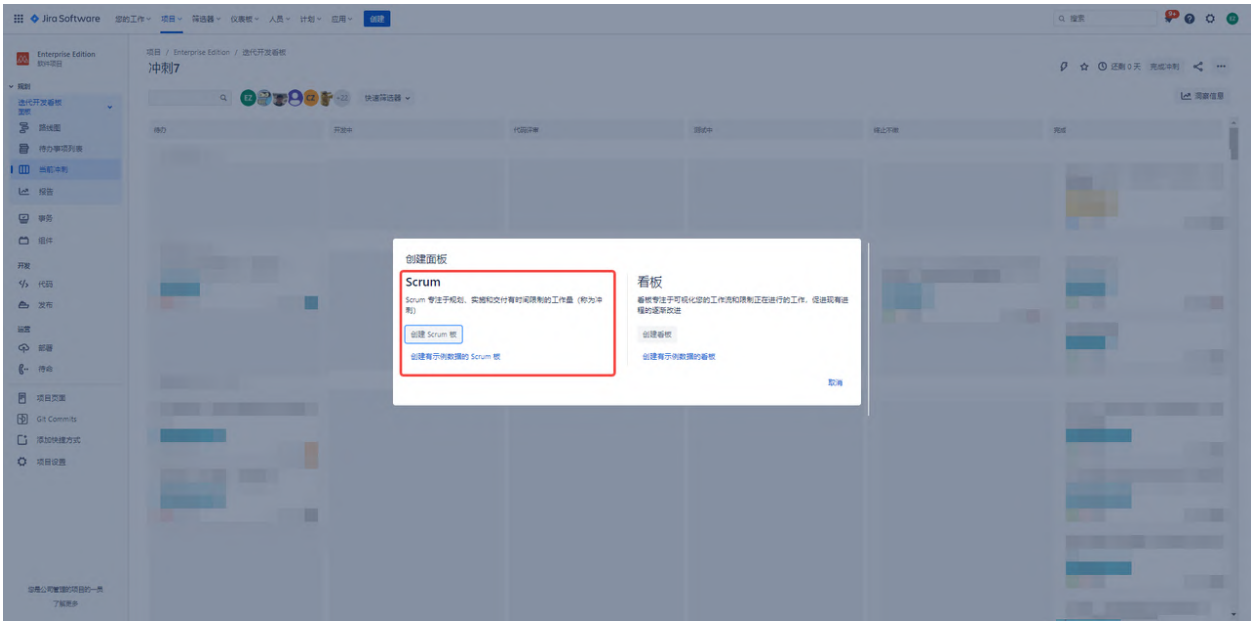

图 230: JIRA 中创建 Scrum

4.5.2.2.4 Commit 与 JIRA 数据绑定

在分析中，为了获得代码与 JIRA 事务的关联，需要记录提交信息与 JIRA 事务的关联关系。此数据关联关系的建立有以下几个注意事项：

==1. 建立数据的联系。**[Requried]**==

- 代码提交时需要在 **Commit Message** 里记录对应的 **Jira Issue Key**，按照 commit message 里的 JIRA Issue Key 建立提交和 JIRA 事务之间的关联。
- 具体格式没有特别要求，可以依据团队需求制定。

类似：

```
[ISSUE-0001] this is the commit message content
```

```
fix(auth) fix some authorization error
ISSUE-0002
```

==2. 将数据的关联关系记录在 JIRA 系统中 **[Recommond]**==

- 需要在 **GitLab** 上配置 **JIRA** 集成信息。
- 可以将某个 GitLab 项目的配置同步到其它需要的项目上。

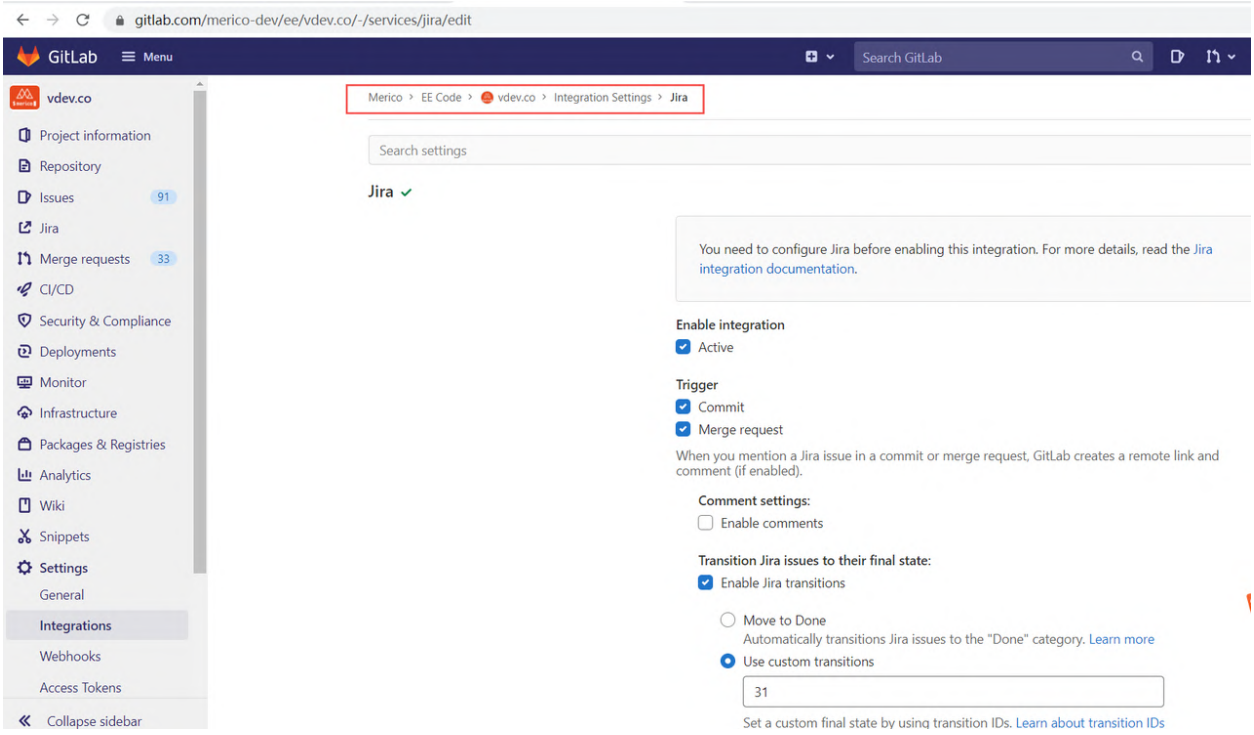

图 231: 在 GitLab 上配置 JIRA 集成信息

4.5.3 工作面板

4.5.3.1 要点概述

- **导航：迭代表现 - 选择单个迭代 - 工作面板**
- 可以切换不同项目组和迭代面板的配对进行数据查看，可以切换迭代进行数据查看。
- 迭代面板的配对，可以切换迭代进行数据查看。
- 进行中的迭代和已完成的迭代按照不同组合展示数据。

迭代表现初始界面会要求选择项目组

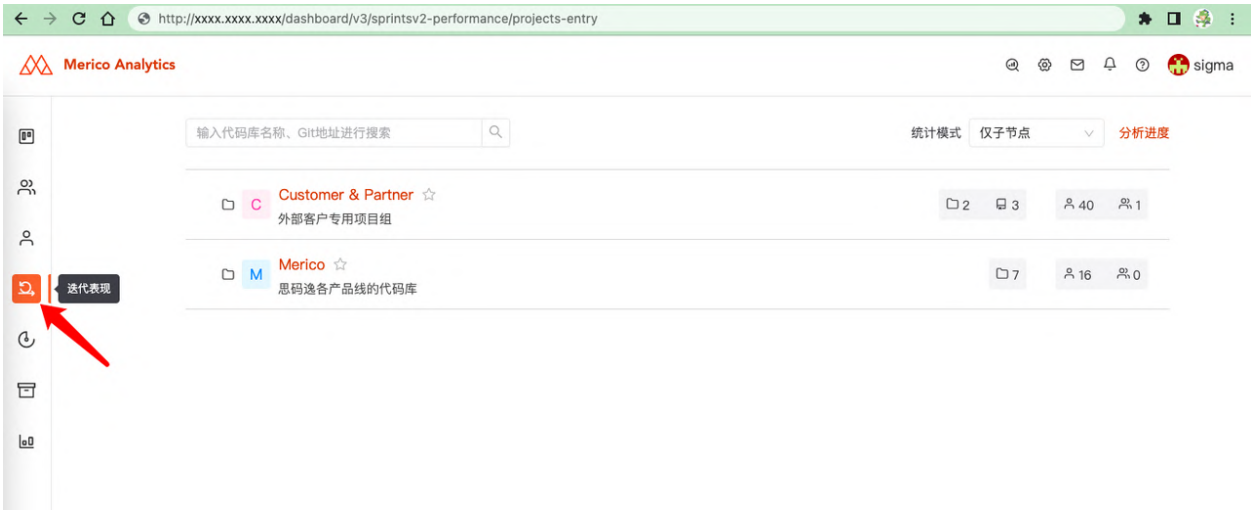

图 232: 迭代表现

迭代表现，默认显示 **数据总览**，需要选择 **单迭代**才能进入工作面板；选择迭代的左侧，也可以再次选择不同项目组

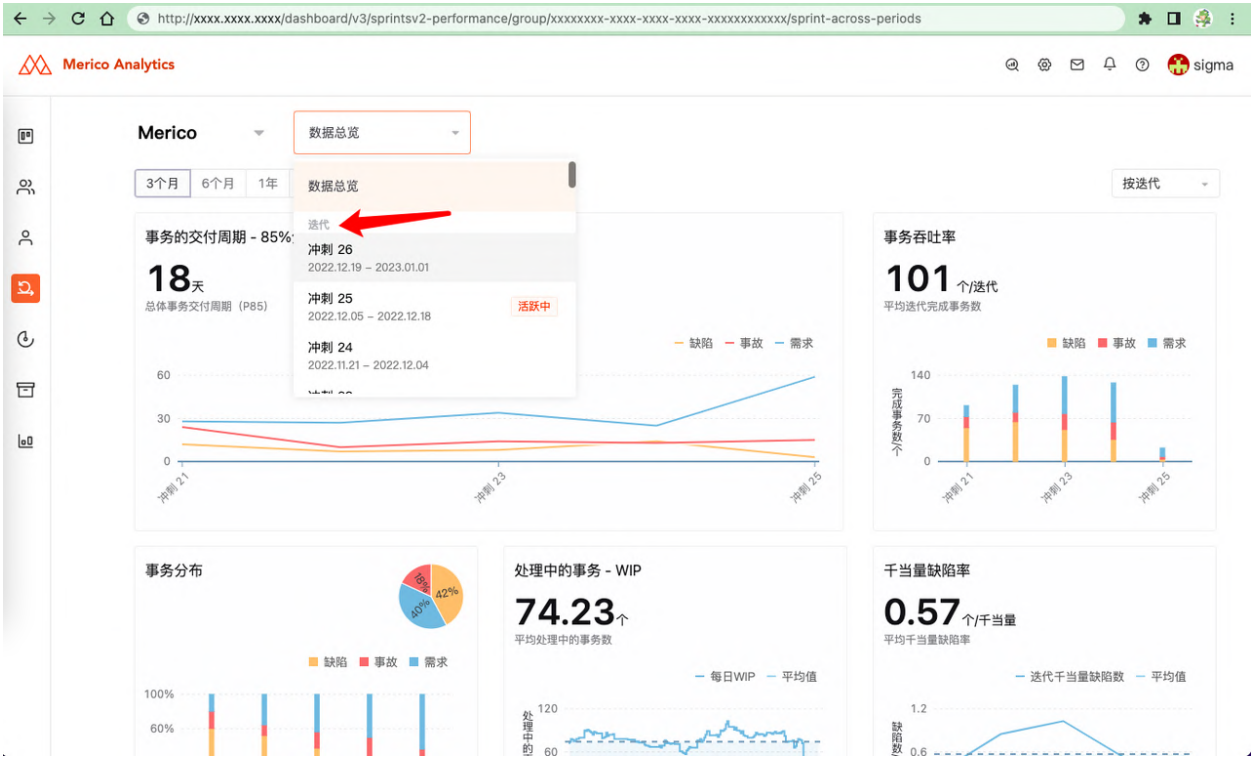

图 233: 默认显示数据总览

点击右上角的选项，也可以选择 **按自然周、按自然月**显示迭代表现数据

图 234: 可以选择按自然周、按自然月显示迭代表现数据

4.5.3.2 界面介绍

4.5.3.2.1 跨周期视图

数据支持的范围：当前项目所绑定的 jira 看板中，最长 1 年的数据（默认 3 个月）

本视图支持将 jira 迭代看板的数据按照两种模式进行查看：

1. **迭代**（默认）：所有实际开始时间在选取的时间范围内的迭代将会被显示；
2. **自然周期**（自然周 & 自然月）：所选时间范围内完整的自然周 & 自然月的数据将被显示；

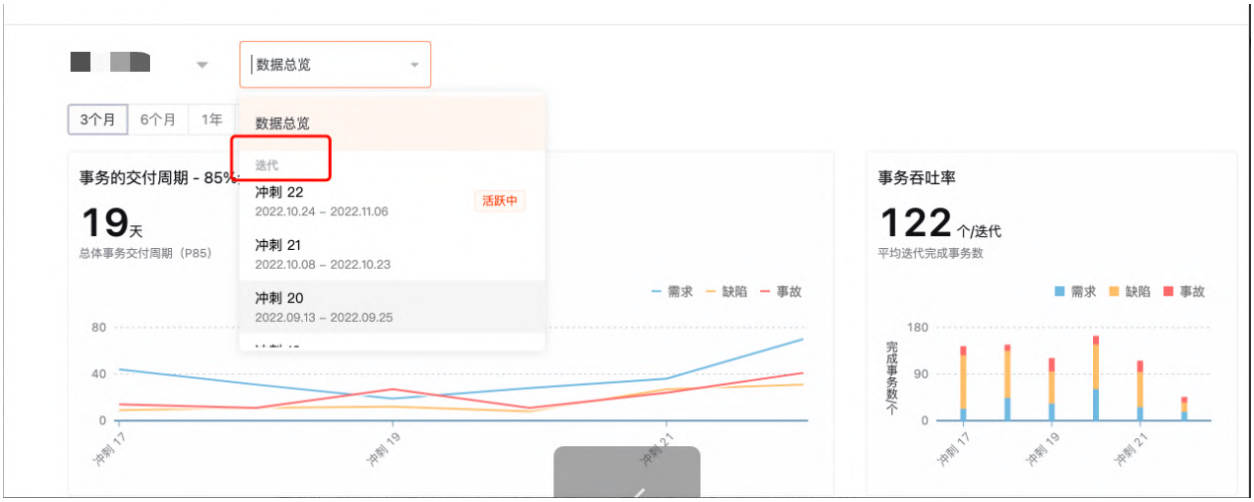

图 235: 支持按迭代查看

事务交付周期

- 单周期：迭代 / 自然周 / 自然月内的按事务类型分别计算完成事务的交付周期的 85% 分位数。
- 整体指标：筛选器范围内的全部已完成事务的交付周期的 85% 分位数。
- 示例：

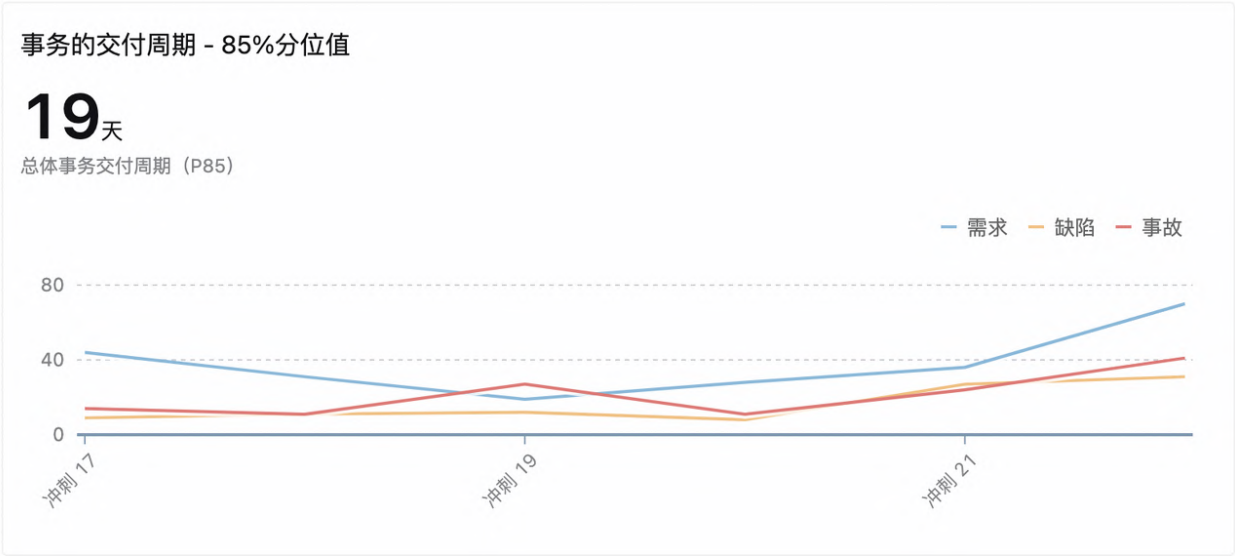

图 236: 事务交付周期

事务吞吐率

- 单周期：迭代 / 自然周 / 自然月的按事务类型分别计算完成数（堆叠累计）

- 整体指标：筛选器范围内各个迭代累计完成事务数的算术平均。
- 示例：

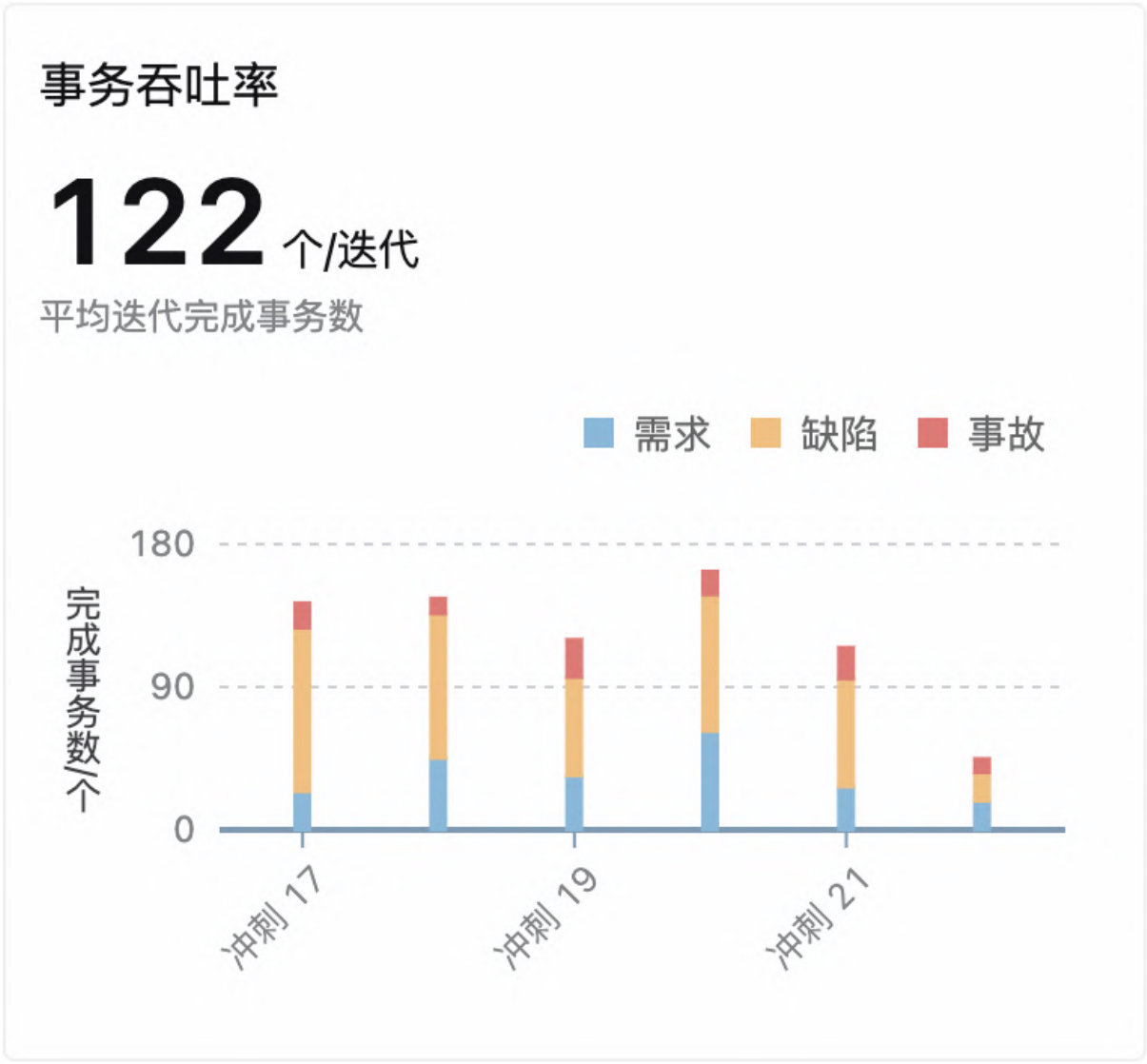

图 237: 事务吞吐率

事务分布

- 单周期：迭代 / 自然周 / 自然月的按事务类型分别计算完成数（堆叠百分比）。
- 整体指标：筛选器范围内各类型事务完成数之比（饼图）。

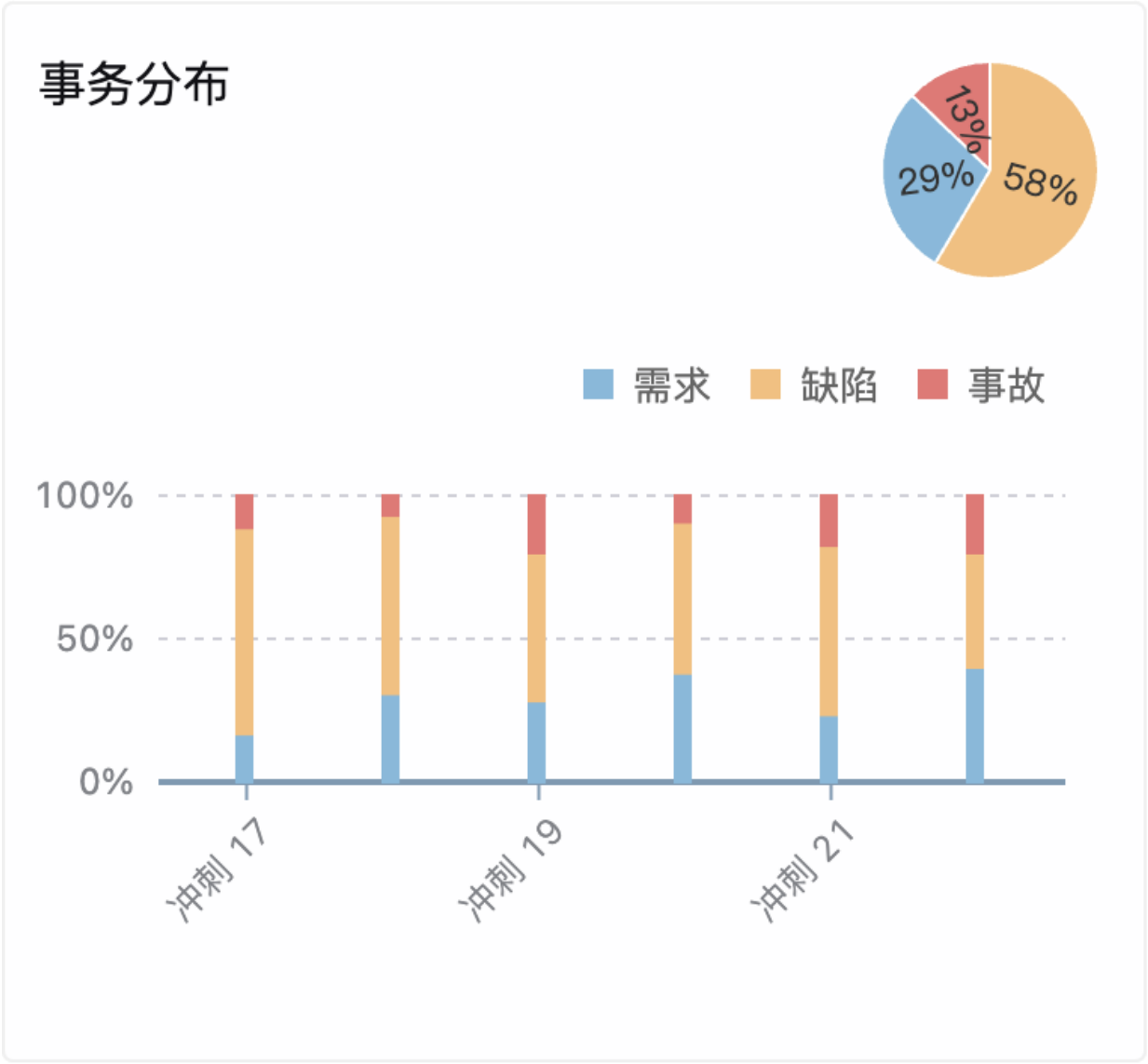

图 238: 事务分布

处理中的事务-WIP

- 每日 WIP：每天按照固定的两个采样时间（12pm & 0 am），计算筛选器范围内处于标准状态-“处理中” 的事务数。
- 总体平均值：筛选器范围内所有采样点的算术平均。

图 239: 处理中的事务

千当量缺陷率

粗略的按照关联项目的迭代新增当量计算

- 每个迭代决定了一个日期范围 d
- 公式：R_0 = (N | d_0) / (E | d_0)，本项目在 d 范围内发现的 Bug 类事务的数量除以本项目在 d 范围内的新增当量（按千当量计数）。（N：在 d0 周期内新增 bug issue 的数量；E：在 d0 周期内，关联项目的累计（新增）当量）
  - 使用 board 来圈定 bug 的范围，不需要 issue 与 sprint 有关联。只要满足时间条件就算如N|d_0

Trade-off：当用户的多个迭代存在重叠时，重叠期间新增的 bug 会被重复计算。

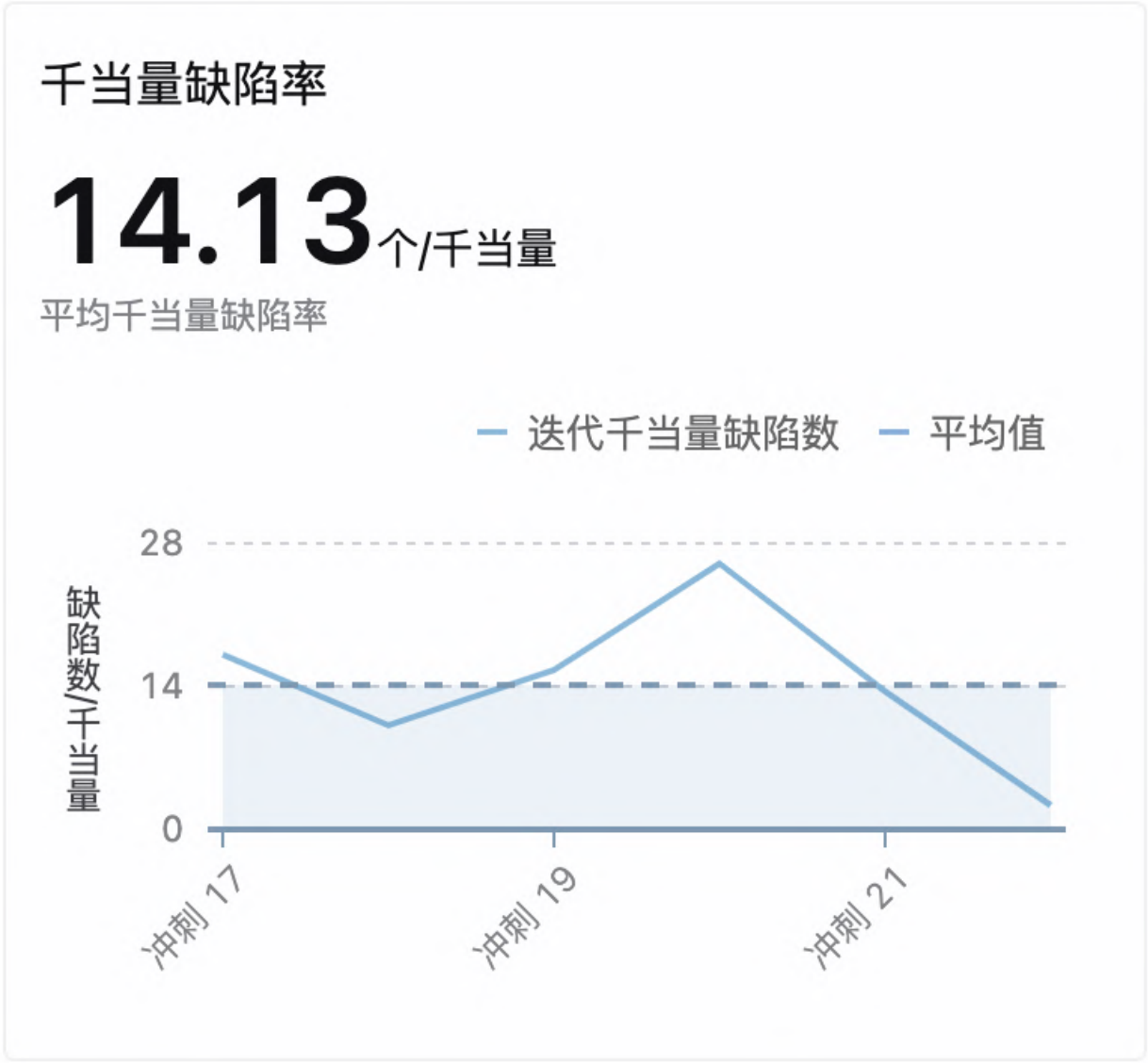

图 240: 千当量缺陷率

4.5.3.2.2 单迭代视图

工作面板（进行中的迭代）

燃烬图

1. 按事务燃尽图：燃尽图展示迭代内剩余事务、新增事务、移除事务数据。
2. 按故事点燃尽图：燃尽图展示迭代内剩余故事点、新增故事点、移除故事点数据。需要在 JIRA 事务中配置故事点字段和填写故事点数据。

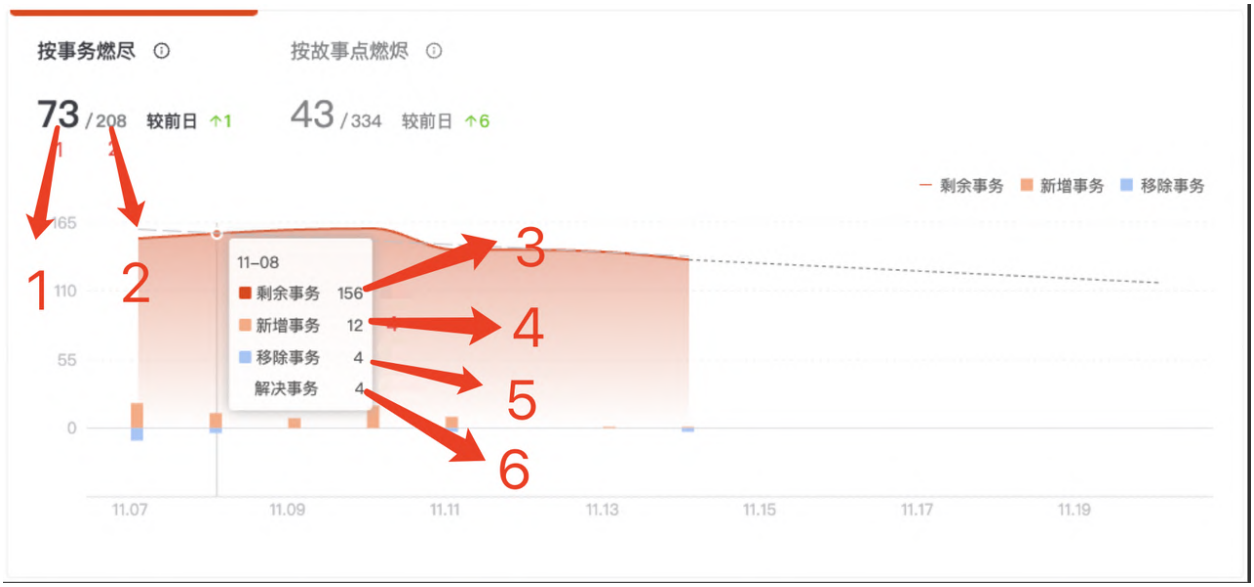

图 241: 燃烬图

备注：下列指标说明项序号与上图序号对应

1. **迭代完成事务数**：这是一个时间切面数据。最后一次成功同步时，迭代范围内，处于“完成”状态的事务数量。

- 与“迭代进度”中“完成”状态列表中的事务数保持一致；
- 迭代完成事务数 = 迭代内完成的事务 + 迭代开始前已经完成的事务
  - 不包含移出迭代后完成的事务以及完成后移出迭代的事务。

2. **迭代范围内事务总数**：这是一个时间切面数据。最后一次成功同步是，迭代范围内，全部事务的数量。

- 与“迭代进度”中的全部事务数保持一致；

3. **燃烬图-（每日）剩余事务**：当天迭代范围内剩余的事务数。

- 剩余事务数 =（前一天）剩余事务数 + 新增事务 - 移除事务 - 解决事务

4. **燃烬图-（每日）新增事务**：当天移入到当前迭代中的事务数。
5. **燃烬图-（每日）移除事务**：当天从迭代中移除的事务数。
6. **燃烬图-（每日）解决事务**：当天迭代范围内完成的事务数。

在故事点燃烬中，以上数据为相应事务故事点的累计值

历史累计代码当量

- **累计代码当量**：最后一次同步时，迭代范围内所有事务相关的提交的当量之和。（需要完成 commit 与 JIRA 数据绑定）
  - 与 “迭代进度” 中所有事务的当量数据之和一致；
- **(每日) 累计代码当量**：截止当天，与迭代范围内所有事务相关的提交的当量之和。**【趋势图中展示（每日）累计代码当量】**

图 242: 历史累计代码当量

剩余 Bug

- **当前剩余 Bug**：最后一次同步时，迭代范围内尚未完成的 “缺陷” 类型事务的数量；

- 与“迭代进度”中，“进行中 & 等待中”状态下“缺陷”事务列表中的数量保持一致；

- **（每日）剩余 Bug**：当天迭代范围内尚未完成的“缺陷”类型事务的数量；**【趋势图中展示（每日）剩余 bug】**

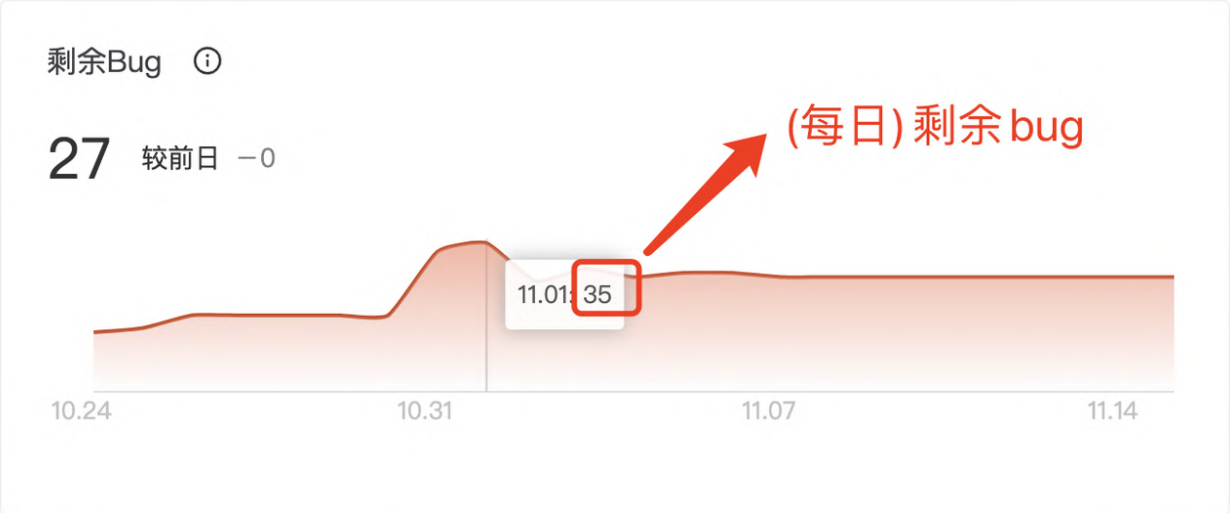

图 243: 剩余 Bug

累计工时

- **累计工时**：此迭代内，所有事务累计的工时数。需要 JIRA 事务中配置工时字段（默认为 timeestimate 和 timespent 字段）。
  - 与“迭代进度”中所有事务的工时数据之和一致；
- **(每日) 累计工时**：截止当天，与迭代范围内所有事务累计的工时数。**【趋势图中展示（每日）累计工时】**

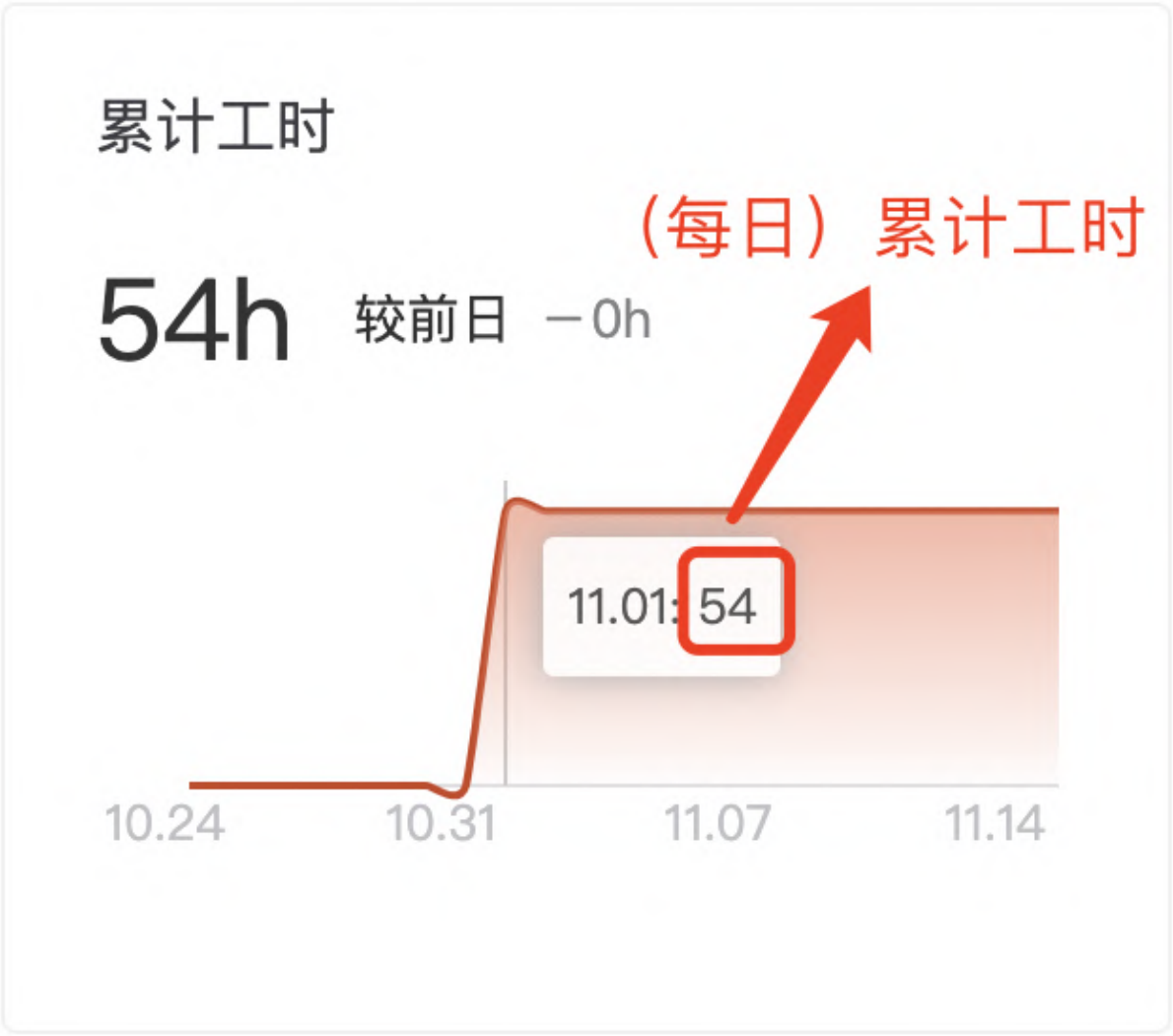

图 244: 累计工时

工作面板（完成的迭代）

迭代完成量

图 245: 迭代完成量

备注：下列指标说明项序号与上图序号对应

1. **迭代完成事务数**：迭代完成时，迭代范围内处于“完成”状态的事务数量。

- 与“迭代进度”列表中的“完成”事务列表不一定能够一致。

“迭代进度”的数据是列表中事务当前状态的体现，数据会随着后续的同步更新，并没有在迭代结束时保留事务状态的切面数据。

2. **迭代完成故事点数**：迭代完成时，迭代范围内处于“完成”状态的事务的故事点总和。
3. **迭代累计新增当量**：迭代完成时，迭代范围内的所有事务相关提交的当量总和。
4. **人均迭代完成事务数**：迭代完成事务数除以迭代内参与到所有迭代事务的 Jira 用户数。

- 参与迭代的 Jira 用户数与“工作量-成员列表”中的成员数量一致。
- 人均完成事务数与“工作量-成员列表”中的“团队平均”不一致。

这里是总完成数与总成员数之间的平均，反映了迭代团队整体人效；“工作量-成员列表”中的事务数与故事点采用的是“参与者统计逻辑”，来更好的反应每个成员的工作量和工作状态。

5. **人均迭代完成故事点数**：迭代完成故事点数除以迭代内参与到所有迭代事务的 Jira 用户数。
6. **人均迭代新增当量**：迭代累计新增当量除以迭代内参与到所有迭代事务的 Jira 用户数。

累计工时

同前

测试发现 Bug

- **累计新增 Bug**：迭代内新增的缺陷类型事务的数量。
- **（每日）累计新增 Bug**：截止当天，迭代内新增的缺陷类型事务的数量。**【趋势图中展示（每日）累计新增 Bug】**

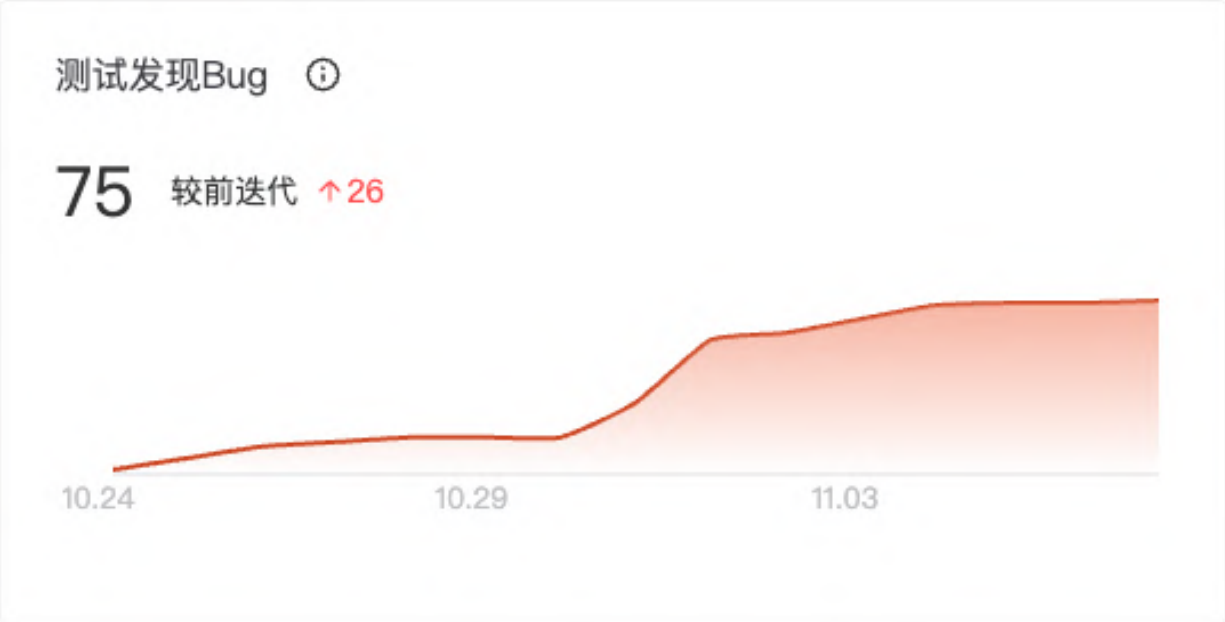

图 246: 测试发现 Bug

事务完成与变更

图 247: 事务完成与变更

备注：下列指标说明项序号与上图序号对应

1. **迭代初计划**：迭代开始时，迭代范围内所有事务的数量。
2. **迭代内新增**：迭代开始后，再加入到迭代内的事务的数量。
3. **迭代完成**：迭代完成时，迭代范围内处于“完成”状态的事务数量。

- 与“迭代完成量-迭代完成事务数”一致

4. **移除迭代完成**：迭代开始后，被从迭代范围内移出且在迭代完成时处于“完成”状态的事务数量。
5. **迭代初计划未完成**：迭代初计划，但在迭代完成时未处于“完成”状态的事务。

- 包含迭代中被移出迭代的事务

6. **迭代内新增未完成**：迭代开始后被加入到迭代中，但在迭代完成时未处于“完成”状态的事务

数据校准 `tips`

迭代初计划 + 迭代内新增 = 迭代完成 + 移除迭代完成 + 迭代出计划未完成 + 迭代内新增未完成。

4.5.4 工作量

4.5.4.1 要点概述

- **导航：迭代表现 - 指定单个迭代 - 工作量**
- 可以切换不同项目组和迭代面板的配对进行数据查看，可以切换迭代进行数据查看。

4.5.4.2 界面介绍

4.5.4.2.1 每日工作进度

- 从代码当量、事务数、故事点、工时四个维度展示项目在该迭代里的工作量数据；
- 数据与“工作面板”中的保持一致；
- 图中的曲线为各指标累计变化趋势；柱子为各指标每日新增的数量。

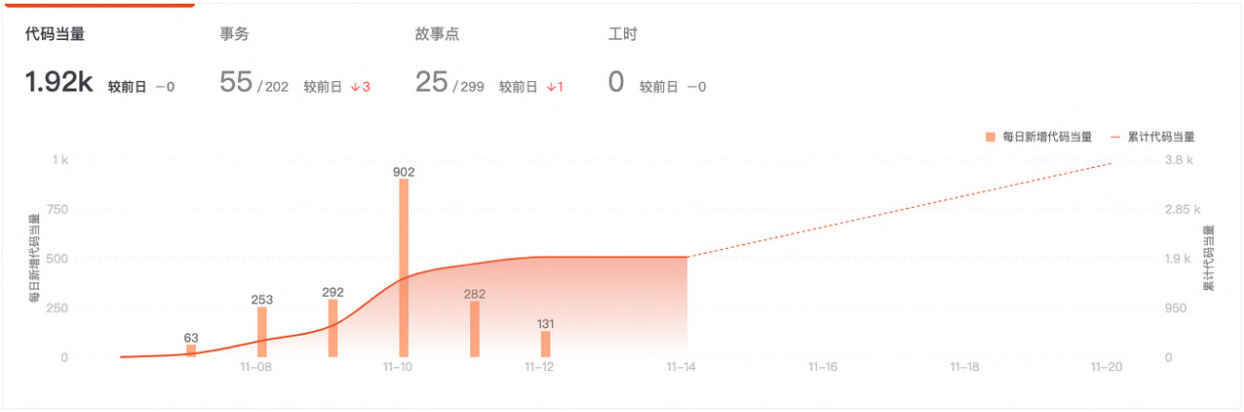

图 248: 每日工作进度

4.5.4.2.2 成员工作量分布

- 从事务数、故事点、工时三个维度展示各项目成员在该迭代里的工作量数据。
- 事务数、故事点采用了事务“参与者”的方式进行统计。

事务“参与者”

在一个事务的生命周期中会有多个人参与其中，因此，一个成员在迭代内完成（经办人 assignee）的事务数量为其参与过的事务中，有多少完成了。相应的总事务即为一共参与了多少事务。

- “团队平均”与“工作面板（完成迭代）”中的人均指标数值不一致。

这里的事务数与故事点采用的是“参与者统计逻辑”，来更好的反应每个成员的工作量和工作状态；“工作面板（完成迭代）”中是总完成数与总成员数之间的平均，反映了迭代团队整体人效。

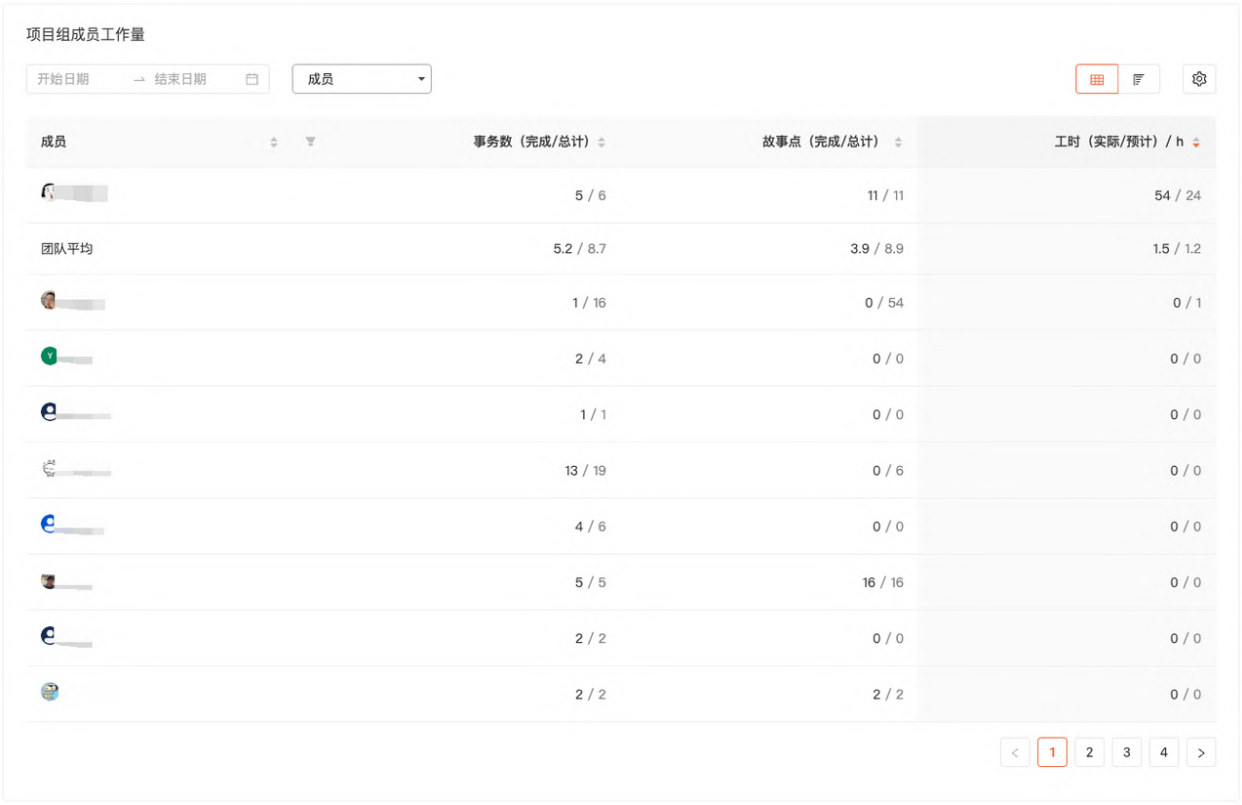

图 249: 成员工作量分布

4.5.5 迭代质量

4.5.5.1 要点概述

- **导航：迭代表现 - 指定单个迭代 - 迭代质量**
- 可以切换不同项目组和迭代面板的配对进行数据查看，可以切换迭代进行数据查看。

4.5.5.2 界面介绍

4.5.5.2.1 累计新增/修复 Bug 数

- 累计新增 Bug 数：迭代内，“缺陷”类事务累计新增数量。
- 包含被移出迭代的“缺陷”类事务。
- 累计修复 Bug 数：迭代内，“缺陷”类事务累计完成的数量。
- 包含加入迭代又被移除且“完成”的“缺陷”类事务。

反映了迭代对于“缺陷”类的处理情况。是否能够及时清理掉迭代新增的缺陷？项目是否在本迭代中存在新的缺陷积压。

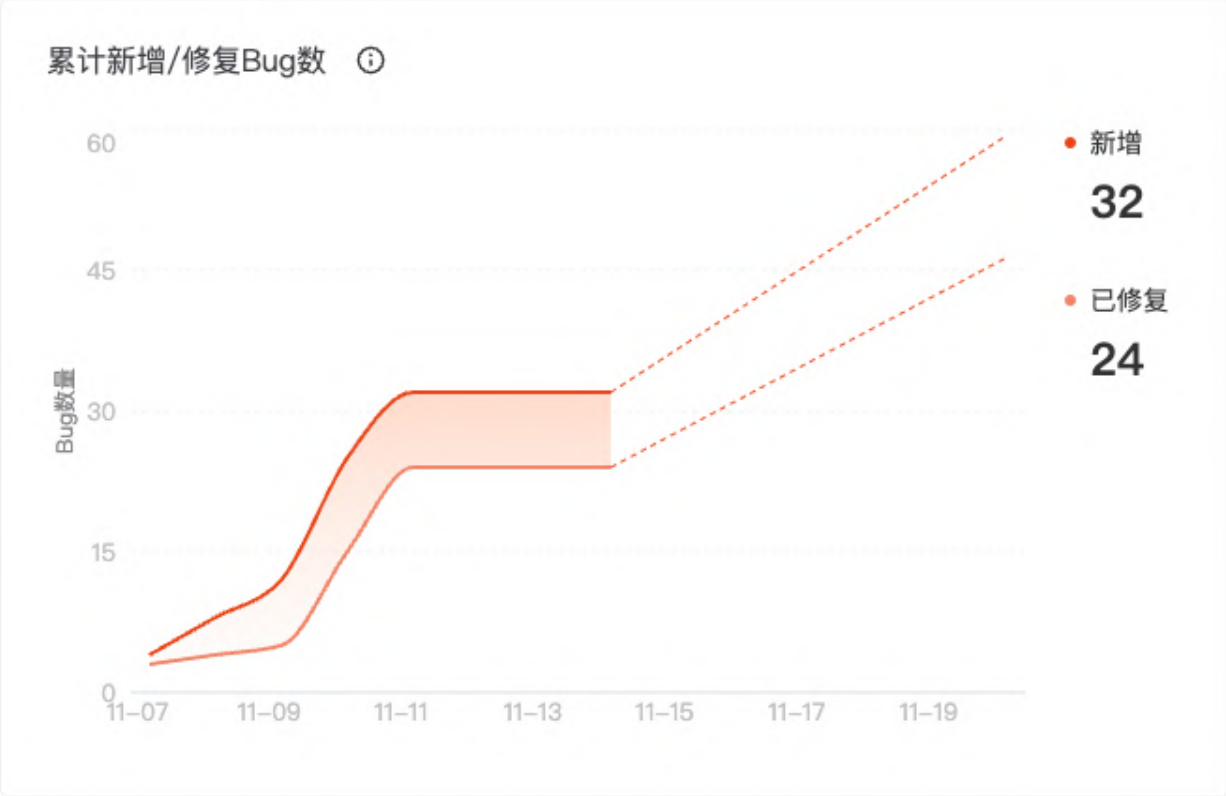

图 250: 累计新增/修复 Bug 数

4.5.5.2.2 每日新增/修复 Bug 数

- 迭代范围内，每天新增和完成的“缺陷”类事务的数量。

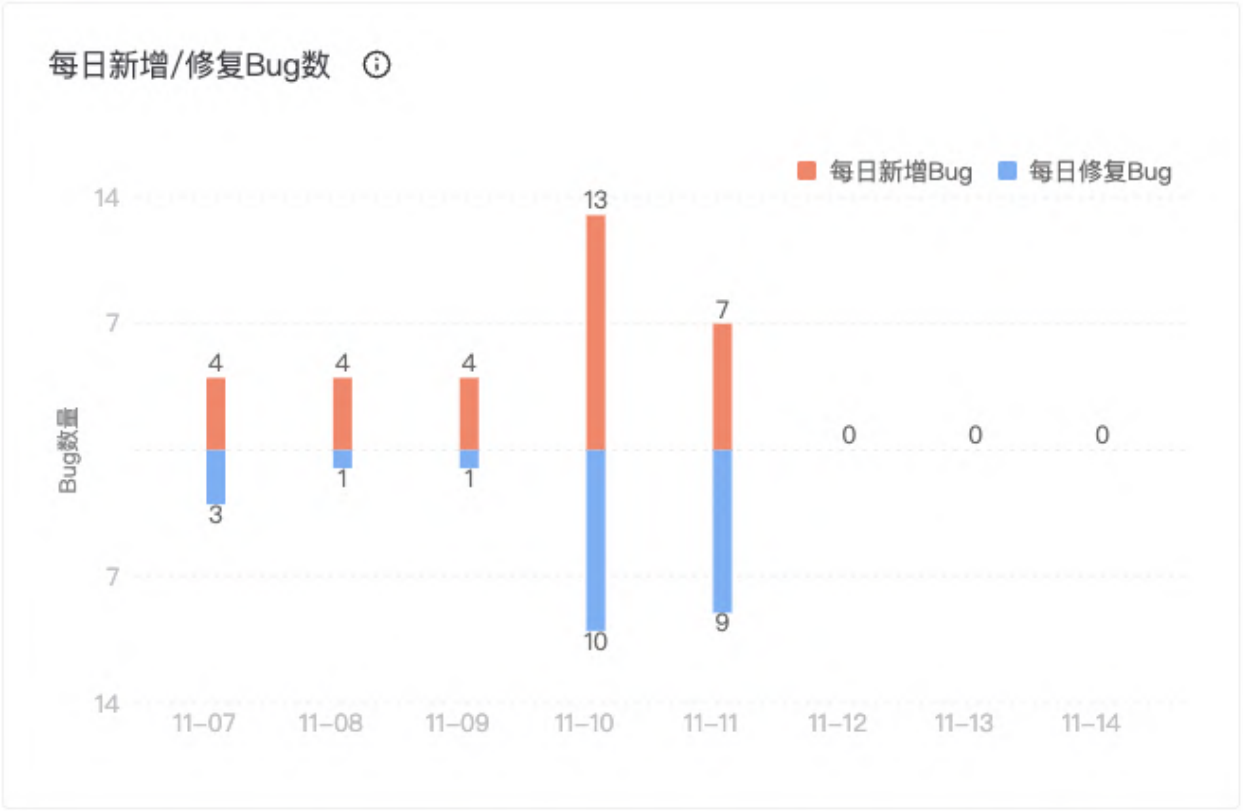

图 251: 每日新增/修复 Bug 数

4.5.5.2.3 成员工作量分布

- 从修复 Bug 数、平均 Bug 修复的前置时间两个维度展示各项目成员在该迭代里的个人在维护项目质量上投入的工作量。
- 修复 bug 数、平均 Bug 修复的前置时间采用了事务“参与者”的方式进行统计。
- 平均 Bug 修复的前置时间：该成员参与的已完成的缺陷类事务的平均前置时间（从创建或迭代开始时间到完成时间）

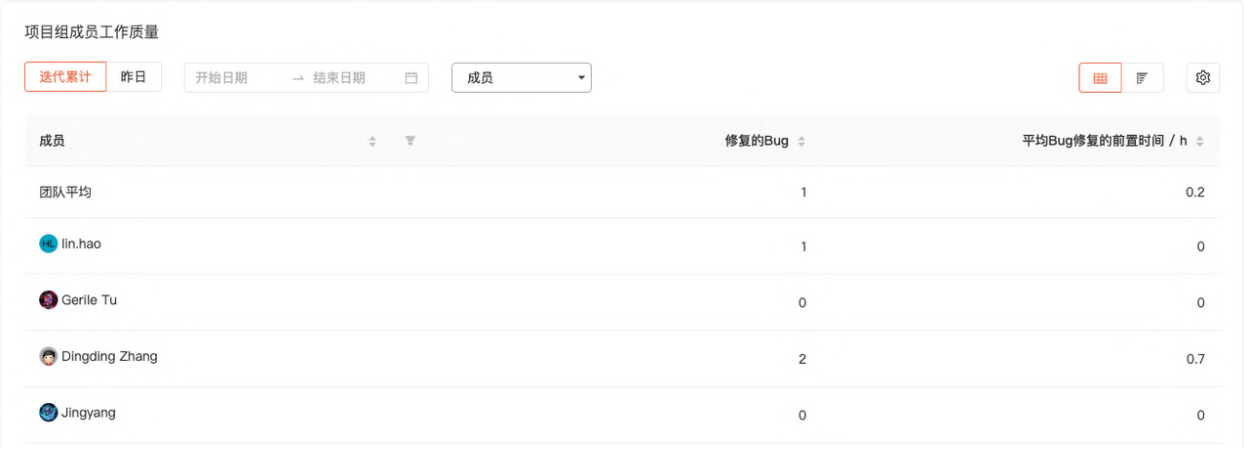

图 252: 成员工作量分布

4.5.6 迭代进度

4.5.6.1 要点概述

- **导航：迭代表现 - 指定单个迭代 - 迭代进度**
- 可以切换不同项目组和迭代面板的配对进行数据查看，可以切换迭代进行数据查看。
- 可以按照事务 priority(优先级)、status(状态)、type(事务类型)、users(经办人) 来搜索迭代内的事务。
- 可以按照不同方式排序迭代内的事务。

4.5.6.2 界面介绍

- **事务：**迭代内的事务的Issue Key和标题。
- **优先级：**事务优先级。
- **事务状态：**事务当前所处的状态（目前只有三种标准状态，后续可在配置中自定义）。
- **负责人：**事务的当前经办人。
- **代码当量：**事务关联提交的总当量，需要完成GitLab和JIRA的集成配置，并在提交时在Commit Message里记录对应的JIRA Issue Key。
- **工时：**事务的工时数据。
- **故事点：**事务的故事点数据。

4.5.6.3 图示

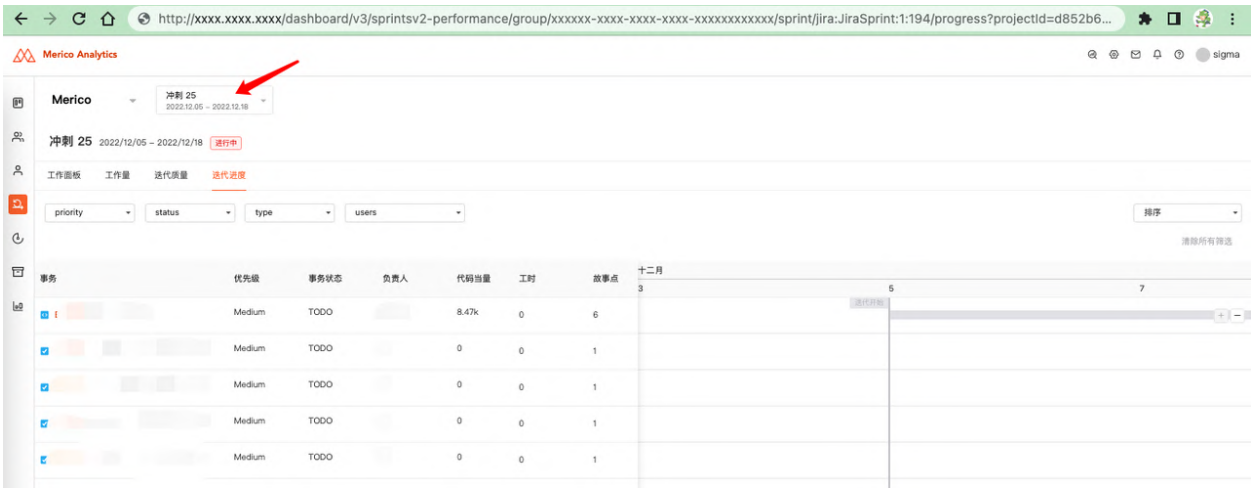

图 253: 迭代进度图示

4.6 技术债监控

4.6.1 技术债监控

在技术债监控版块下，可以查看如下与技术债相关的指标和数据：

- **代码问题：**分析得到的代码中的各种问题，在此界面上可看到不同种类、严重程度的问题的具体信息，包括此问题在代码中的具体定位和归属者，问题的描述和建议修改方案等。
- **代码重复：**在此界面上展示了分析识别出的重复代码的详细定位和对比。
- **单测覆盖度：**被测试用例覆盖的函数占总函数个数的比例。在此界面上展示了测试覆盖度的评分，未被测试覆盖的函数列表和各文件中各函数被测试覆盖的情况。
- **注释覆盖度：**被注释覆盖的函数占总函数个数的比例。在此界面上展示了注释覆盖度的评分，未被注释覆盖的函数列表和各文件中各函数被注释覆盖的情况。
- **函数复杂度：**展示了代码库中各函数的复杂度分析信息：圈复杂度、函数重复数、函数的出度和入度。

4.6.2 代码问题

4.6.2.1 要点概述

- **导航：**左侧菜单栏 **技术债监控 - 代码问题**。
- 在 **项目或代码库**下拉菜单中，可以选择当前登录账户有权限查看的任一项目组或代码库。
- 可以按照开始日期和结束日期，代码库，问题严重程度，问题类型，问题归属人，问题规则，文件路径来搜索问题。

- 默认显示当前项目组/代码库所有贡献者。可选择某一贡献者查看归属于他的问题。
- 默认显示全部（问题所属）规则，可选择某几种问题规则查看具体细节。

4.6.2.2 问题类型

默认显示全部类型，可选择具体某几种类型的问题查看，包含类型：漏洞、安全热点、Bug、异味、可移植性、性能。

1. Bug：即代码中存在的明显错误，破坏了您代码的正常运行，需要立即修复。

2. 漏洞：即代码中存在容易受到攻击的可能，需要您提起关注。

3. 异味：即代码的编写格式是否规范，如果您扫描出此类问题，说明您的代码有些混乱且难以维护。

4. 安全热点：参考 sonarqube 相关定义 。

5. 性能：基于常识的运行时性能建议，尽管不能确定通过修复这些消息会实现任何可测量的速度差异。

6. 可移植性：可移植性警告，实现定义行为 64 位可移植性。一些可能“如你所愿”工作的未定义行为等。

4.6.2.3 问题严重程度

默认显示全部严重程度，可选择具体某几种严重程度的问题查看，包含严重程度：阻塞、严重、主要、次要、提示。

1. 阻塞问题：很可能会影响生产环境正常运行的错误，例如：内存泄漏，未关闭的 JDBC 链接等等。必须立即修复代码。

2. 严重问题：有可能会影响生产环境正常运行的错误，也可能是代表安全漏洞的问题，例如：空的 catch 块，SQL 注入等等。必须立即检查代码。

3. 主要问题：严重影响开发人员生产力的质量缺陷，例如：无效的代码逻辑，重复的代码，未使用的参数等等。

4. 次要问题：稍微影响开发人员生产力的质量缺陷，例如：行不应太长，“switch”语句应至少包含 3 种情况等等。

5. 提示问题：一般性的错误或质量缺陷。

4.6.2.4 界面使用介绍

代码问题界面上显示不同种类、严重程度的问题的具体信息，包括此问题在代码中的具体定位和归属者，问题的描述和建议修改方案等。

- 支持筛选：支持 **按照开始日期和结束日期、按照代码库、按照严重程度、按照问题类型、按照问题归属人、按照问题规则、按照文件路径的 Unix glob 语法**进行筛选。
- **支持逐个查询**，点击某个具体代码问题的空白部分即可
- **支持按照问题规则一键查询相同数据**，点击某个具体代码问题上半部分的规则文案即可
- **支持按照问题严重程度一键查询相同数据**，点击某个具体代码问题左下角的严重程度文案即可
- **支持按照问题类型一键查询相同数据**，点击某个具体代码问题左下角的问题类型文案即可

- **支持按照问题归属人一键查询相同数据**，点击某个具体代码问题右下角归属人姓名即可
- **支持一键忽略特定代码问题规则**，点击某个具体代码问题右下角的小眼睛即可。被忽略的代码问题规则，如果想要查询或恢复关注此规则，可以前往设置—> 规则设置中排查和恢复关注

4.6.2.5 图示

逐个查询

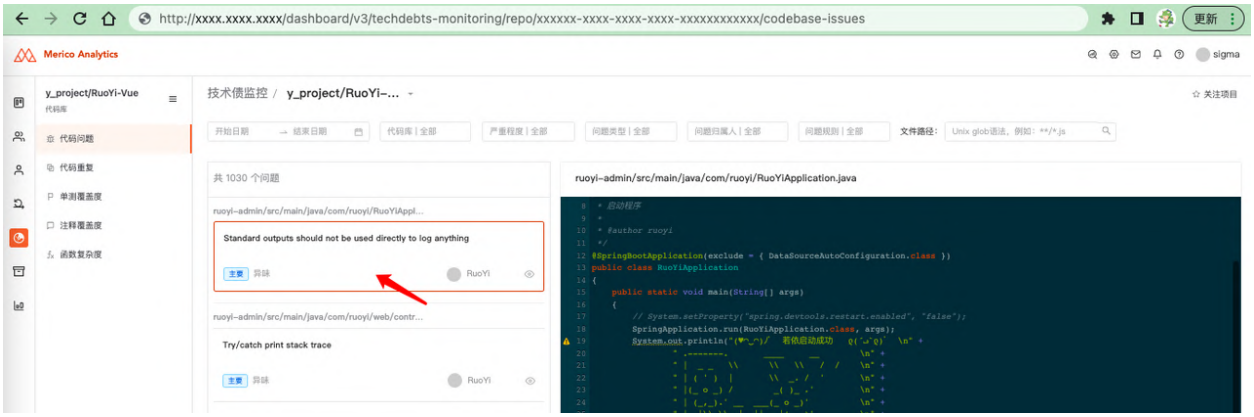

图 254: 代码问题 - 逐个查询

按照问题规则一键查询相同数据

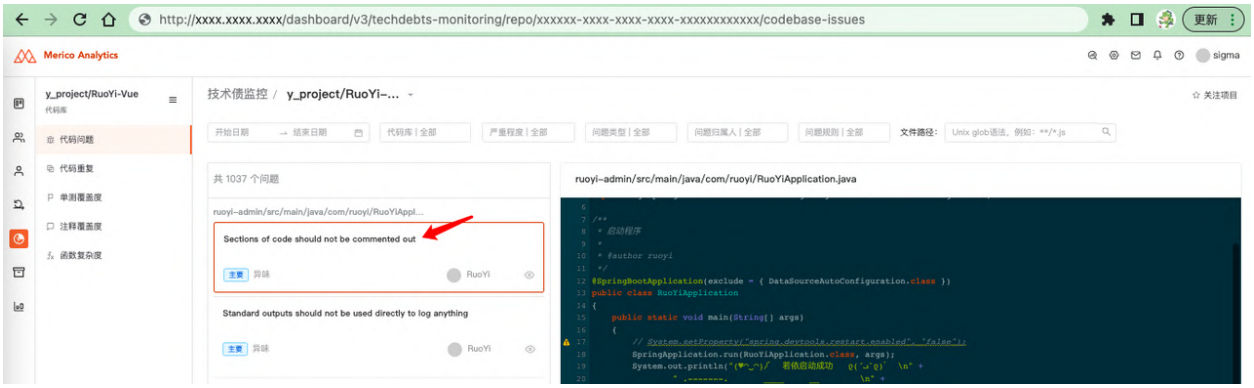

图 255: 代码问题 - 按照问题规则一键查询相同数据

按照问题严重程度一键查询相同数据

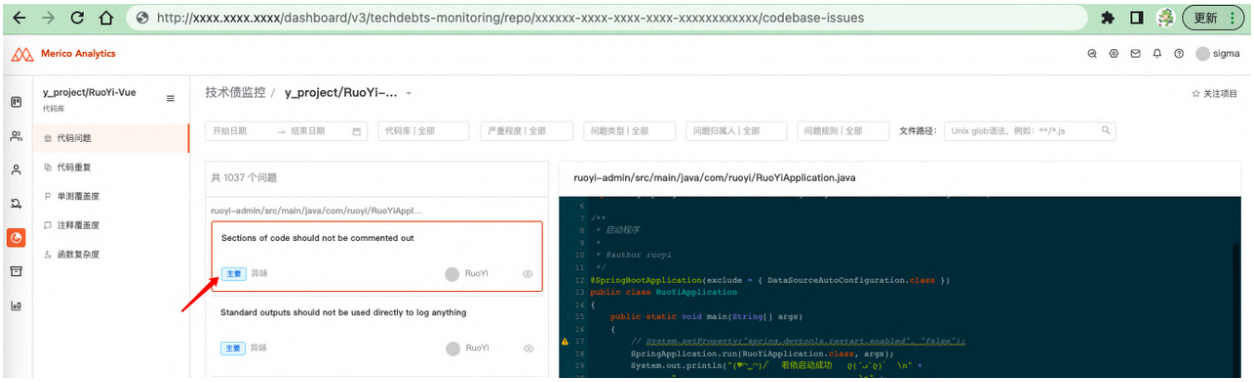

图 256: 代码问题 - 按照问题严重程度一键查询相同数据

按照问题类型一键查询相同数据

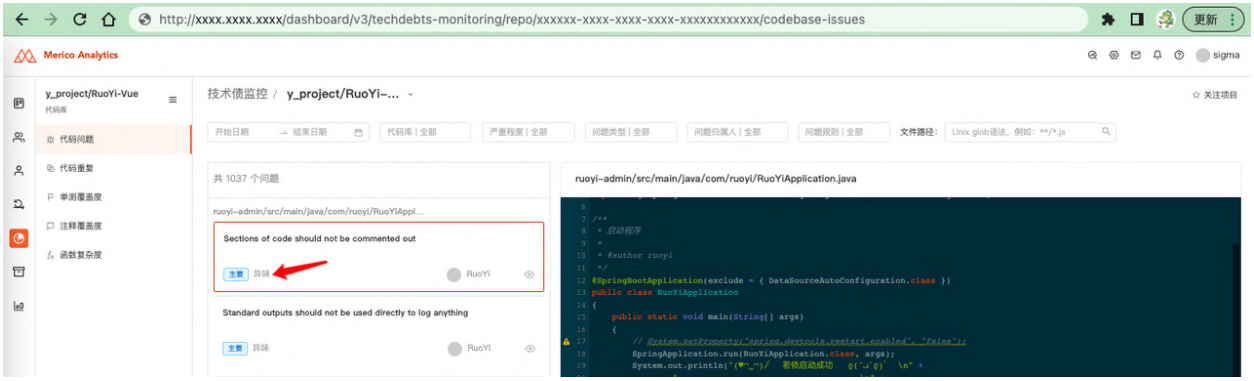

图 257: 代码问题 - 按照问题类型一键查询相同数据

按照问题归属人一键查询相同数据

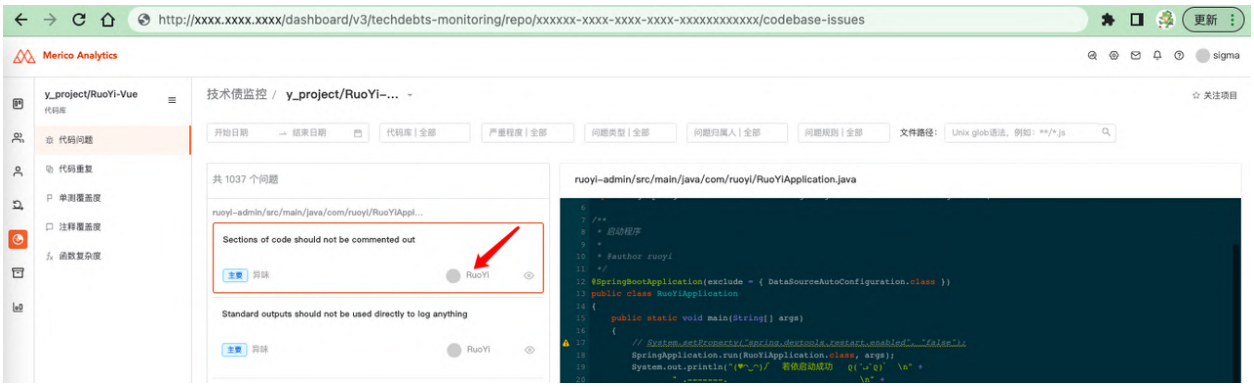

图 258: 代码问题 - 按照问题归属人一键查询相同数据

按照问题归属人一键查询相同数据

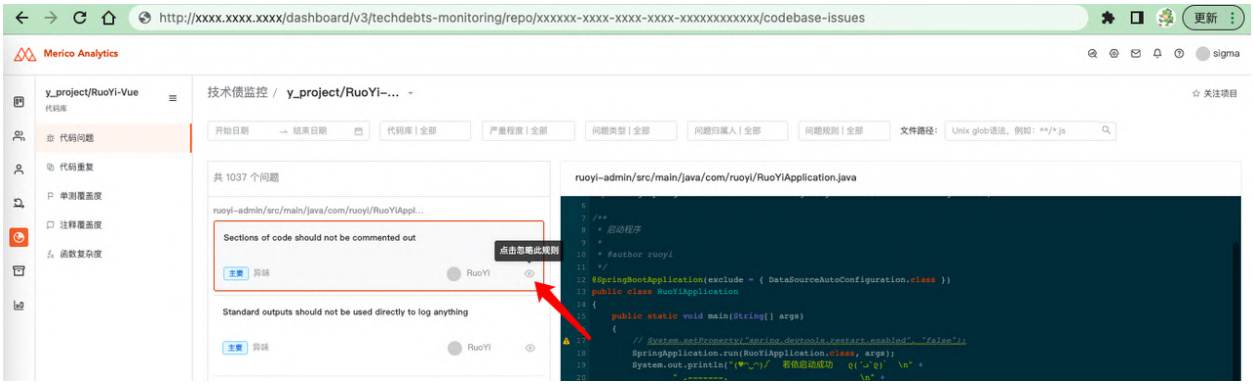

图 259: 代码问题 - 按照问题归属人一键查询相同数据

4.6.3 代码重复

4.6.3.1 要点概述

- **导航：**左侧菜单栏 **技术债监控 - 代码重复**。
- 在 **项目或代码库**下拉菜单中，可以选择当前登录账户有权限查看的任一项目组或代码库。

4.6.3.2 界面使用介绍

- 代码复用界面上按**分组**显示包含重复代码的代码块。每个分组中，左侧显示具体的代码信息，右侧为分为一组的各代码块，可以通过 **选择**和 **勾选**代码块，来对比两个包含重复代码的代码块。
- **选择：**直接点击右侧某代码块信息，使之底色变成浅橙色（肉色），即为选择。
- **勾选：**点击右侧某代码块信息前的勾选框，使之被勾选上，即为勾选。
- **重复代码对比：**左侧黑色代码详情中，**没有**被高亮的部分为两个对比代码块中的**重复代码**，**红色高亮**的为被**勾选**的代码块中的**不重复**代码，绿色高亮的为被**选择**的代码块中的**不重复**代码。

4.6.3.3 图示

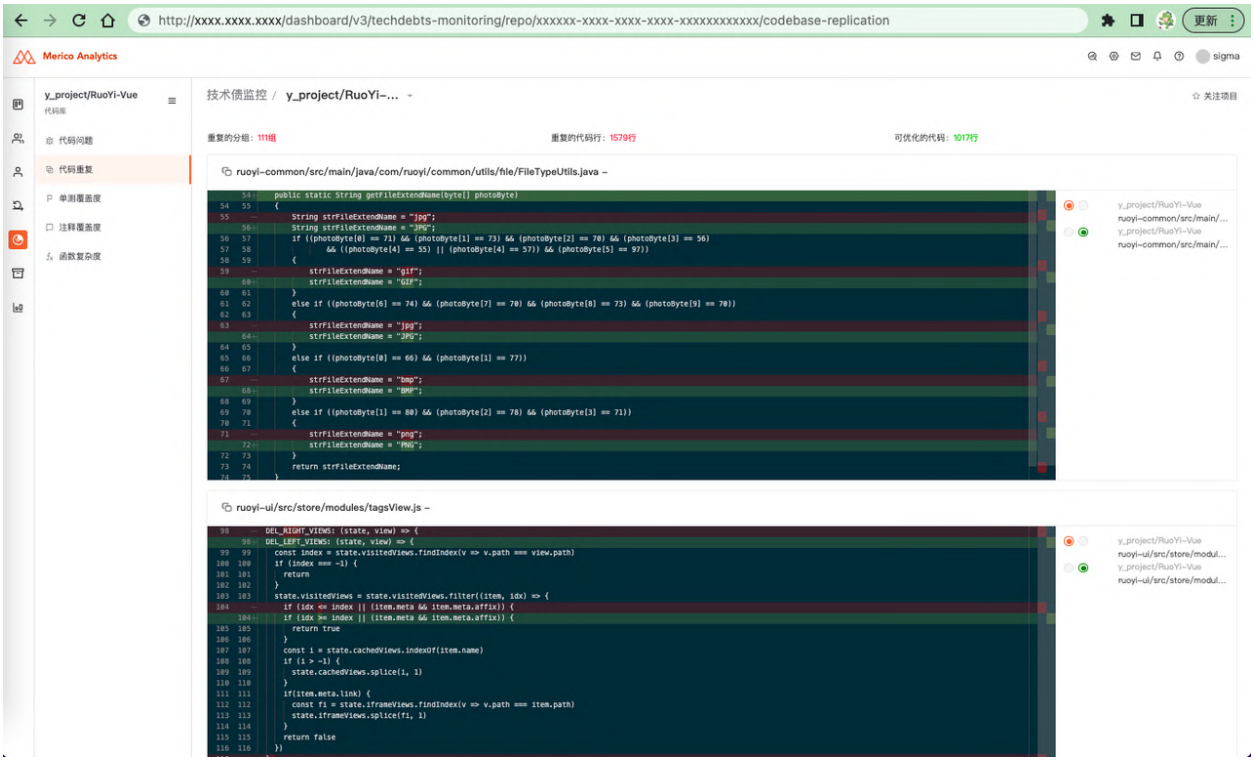

图 260: 代码重复图示

4.6.4 单测覆盖度

4.6.4.1 要点概述

- **导航：**左侧菜单栏 **技术债监控 - 测试覆盖度**。
- 在 **项目或代码库**下拉菜单中，可以选择当前登录账户有权限查看的任一项目组或代码库。

4.6.4.2 界面介绍

- **单测覆盖度：**被测试用例覆盖的函数占总函数个数的比例。
- **未覆盖的函数：**未被测试覆盖的函数。
- **测试覆盖度（按文件）：**列出各文件中各函数被测试覆盖的情况。

4.6.4.3 图示

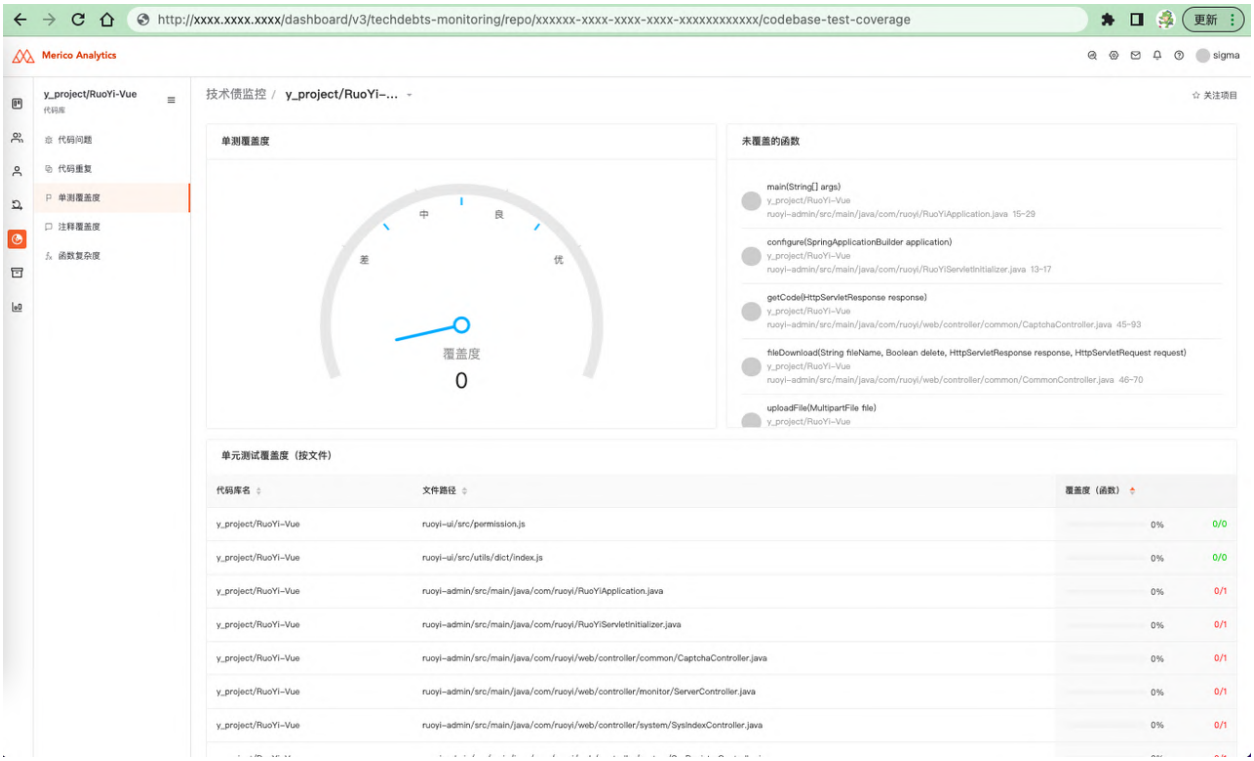

图 261: 单测覆盖度图示

4.6.5 注释覆盖度

4.6.5.1 要点概述

- **导航：**左侧菜单栏 **技术债监控 - 注释覆盖度**。
- 在 **项目或代码库**下拉菜单中，可以选择当前登录账户有权限查看的任一项目组或代码库。

4.6.5.2 界面介绍

- **注释覆盖度：**被注释覆盖的函数占总函数个数的比例。
- **未覆盖的函数：**未被注释覆盖的函数。
- **注释覆盖度（按文件）：**列出各文件中各函数被注释覆盖的情况。

4.6.5.3 图示

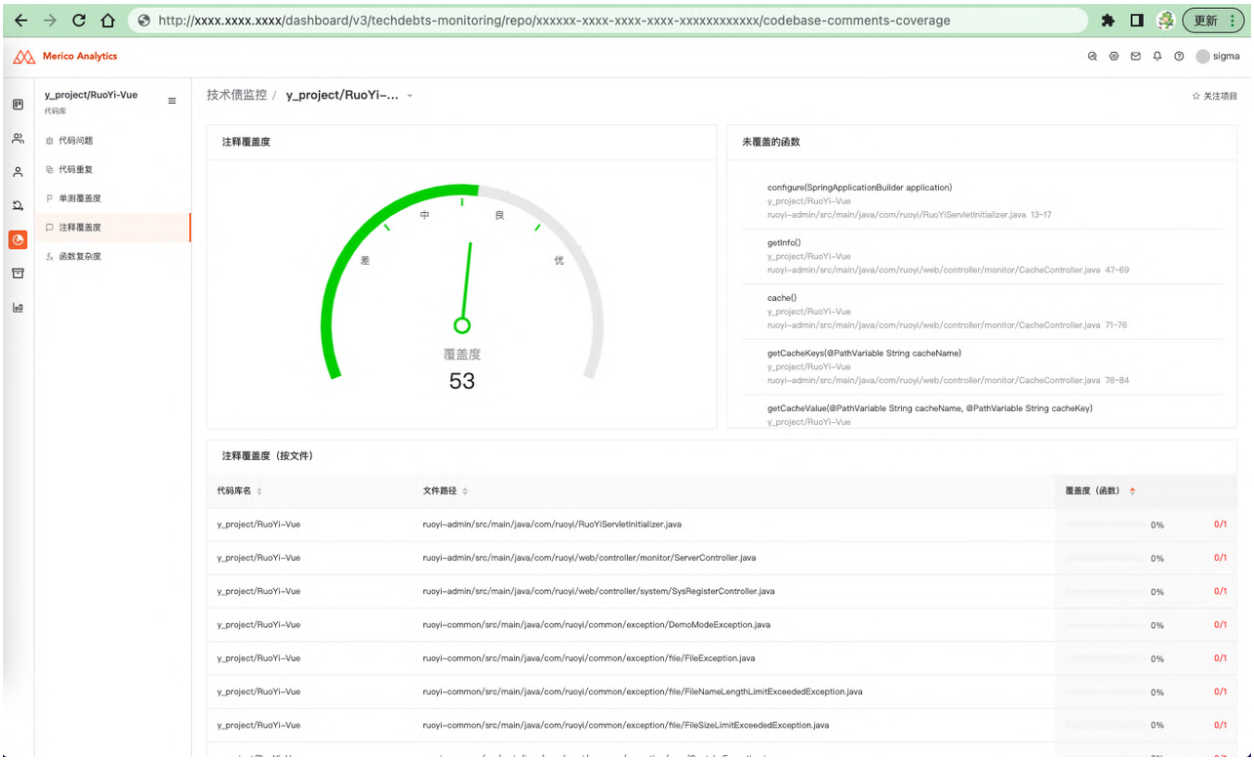

图 262: 注释覆盖度图示

4.6.6 函数复杂度

4.6.6.1 要点概述

- **导航**：左侧菜单栏 **技术债监控 - 函数复杂度**。
- 仅在选择**代码库**查看指标数据时可以查看函数复杂度。
- 在代码库下拉菜单中，可以选择当前登录账户有权限查看的任一代码库。

4.6.6.2 界面介绍

- 上方浅蓝色区域显示此代码库最近分析信息，分析分支，最近一次提交及其 Hash 值，提交者信息。
- 列表显示分析得到的函数信息，包括函数签名，函数定义位置，函数圈复杂度，重复数，入度，出度。点击各函数签名可显示该函数的源码信息。
- **圈复杂度：**用来衡量一个模块判定结构的复杂程度，数量上表现为线性无关的路径条数，即合理的预防错误所需测试的最少路径条数。圈复杂度大说明程序代码可能质量低且难于测试和维护，根据经验，程序的可能错误和高的圈复杂度有着很大关系。
- **函数重复数：**该函数重复的次数。
- **函数入度：**该函数被调用的次数，即直接调用该模块的上级模块的个数。
- **函数出度：**该函数调用别的函数的次数，即该模块直接调用的下级模块的个数。

4.6.6.3 图示

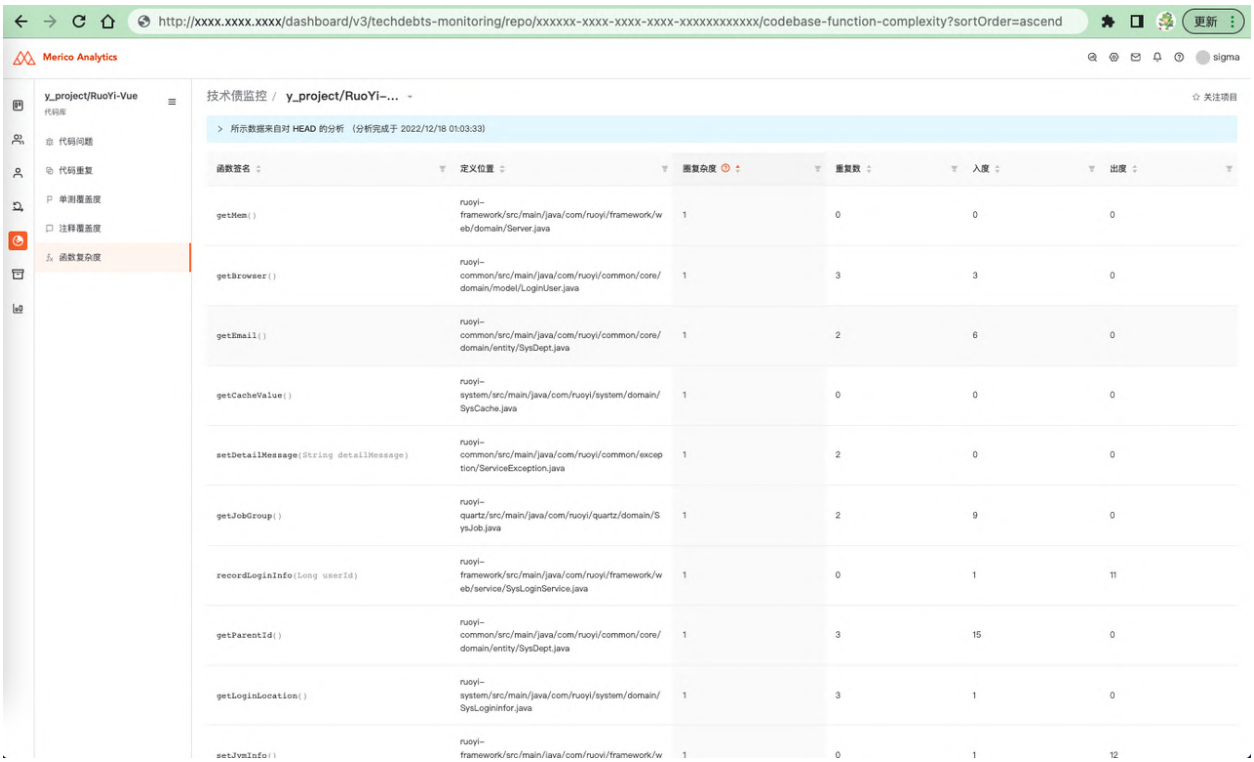

图 263: 函数复杂度图示

4.7 工具箱

4.7.1 日志

4.7.1.1 要点概述

- **导航**：左侧菜单栏 **工具箱 - 日志**。
- 在此界面可以查看登录历史、操作历史、下载历史和系统信息（系统日志）。

4.7.1.2 登录历史

- 呈现思码逸系统近期或指定登录时间内的账户登录历史数据。
- 支持按照登录时间和用户名筛选和导出。
- 示例

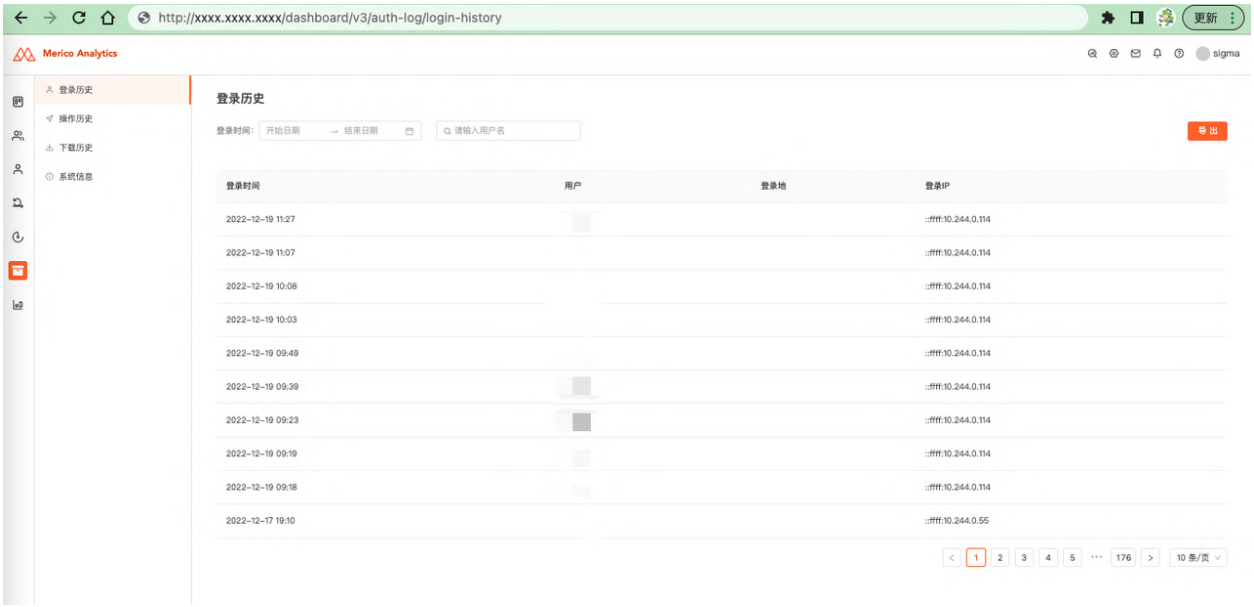

图 264: 登录历史

4.7.1.2.1 操作历史

- 呈现思码逸系统近期或指定登录时间内的账户操作历史数据
- 支持按照操作时间、用户名和操作类型筛选和导出
- 示例

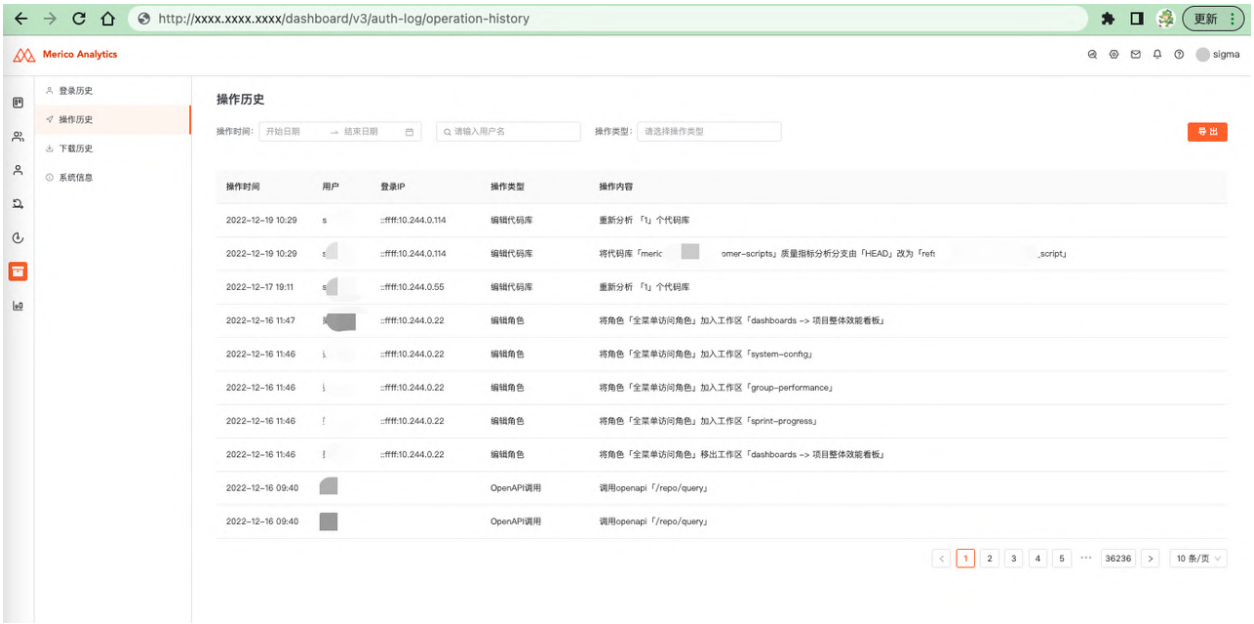

图 265: 操作历史

4.7.1.2.2 下载历史

- 呈现思码逸系统下载历史数据
- 支持按照文件名、下载文件状态筛选
- 示例

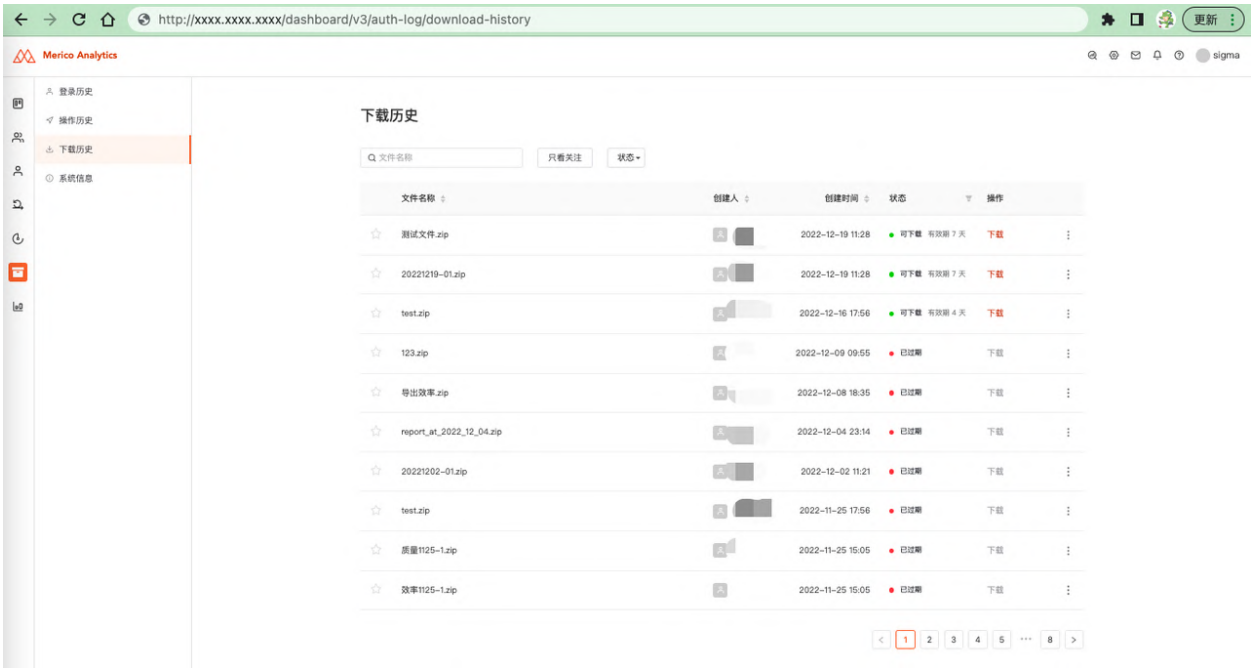

图 266: 下载历史

4.7.1.2.3 系统信息

- 下载指定日期范围内的日志。注意，选择更大范围的日期会花费更多的下载时间。
- 示例

点击 **导出**按钮

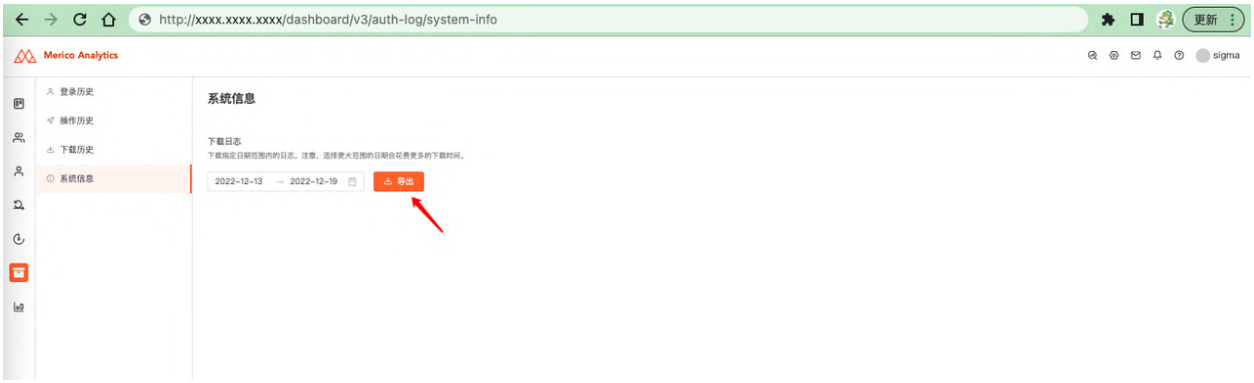

图 267: 点击导出按钮

填写文件名并保存

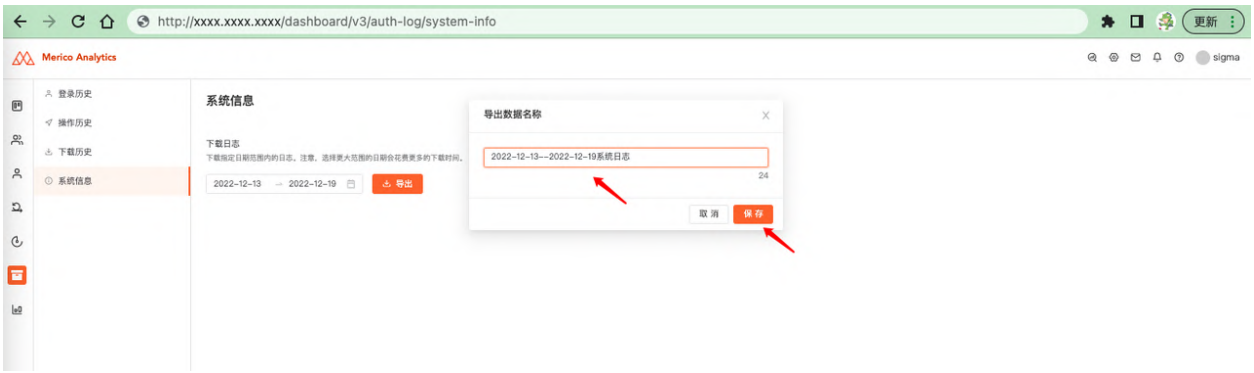

图 268: 填写文件名并保存

去下载中心

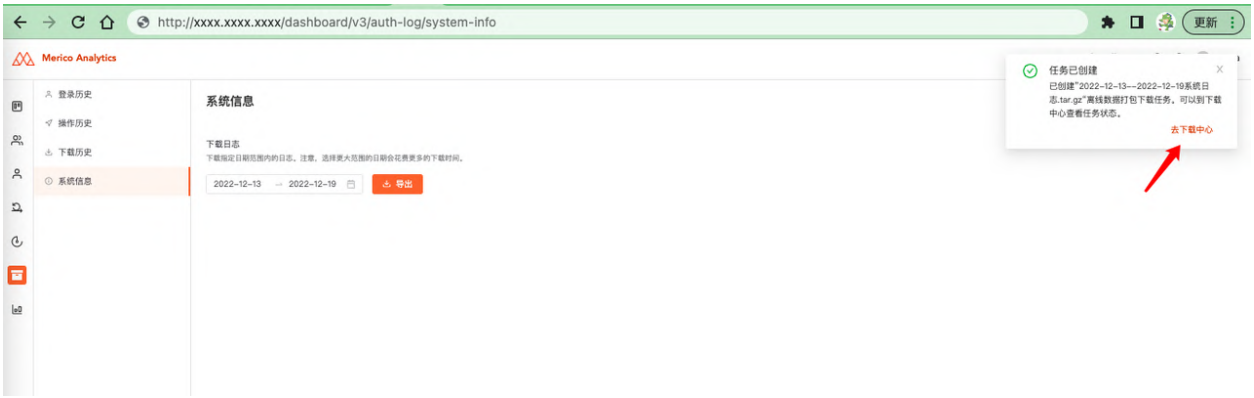

图 269: 去下载中心

在下载历史界面，点击 **下载**按钮

图 270: 在下载历史界面，点击下载按钮

4.7.2 展示板

4.7.2.1 要点概述

- **导航**：左侧菜单栏 **工具箱 - 展示板**。
- 在此界面可以查看自己创建的大屏图表以及他人创建并共享的大屏图表。

4.7.2.2 我的展示板

- 呈现自己创建的大屏图表
- 示例

创建

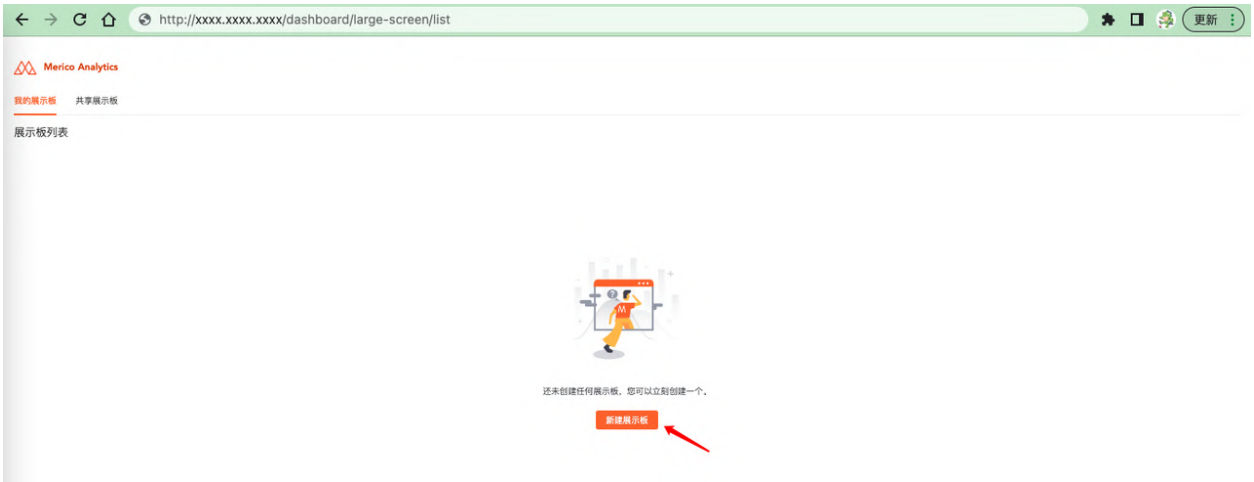

图 271: 创建我的展示板

可以自由配置展示板分页设置、周期数据翻页频率、展示板名称、展示板布局

图 272: 自由配置展示板相关信息

目前可添加的图表类型：效率质量综合榜、代码当量榜、代码影响力榜、代码问题密度评分榜、单测覆盖度榜、注释覆盖度榜、项目鲁棒性等

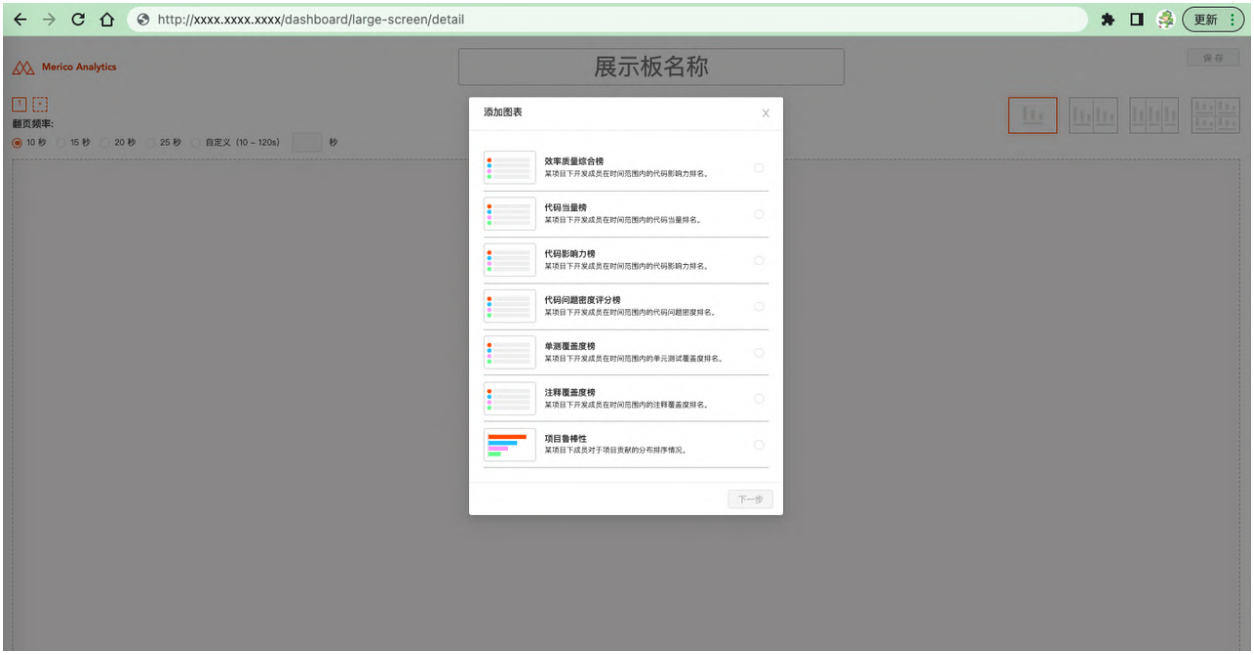

图 273: 可添加的图表类型

4.7.2.3 共享展示板

- 呈现他人创建并共享的大屏图表
- 示例

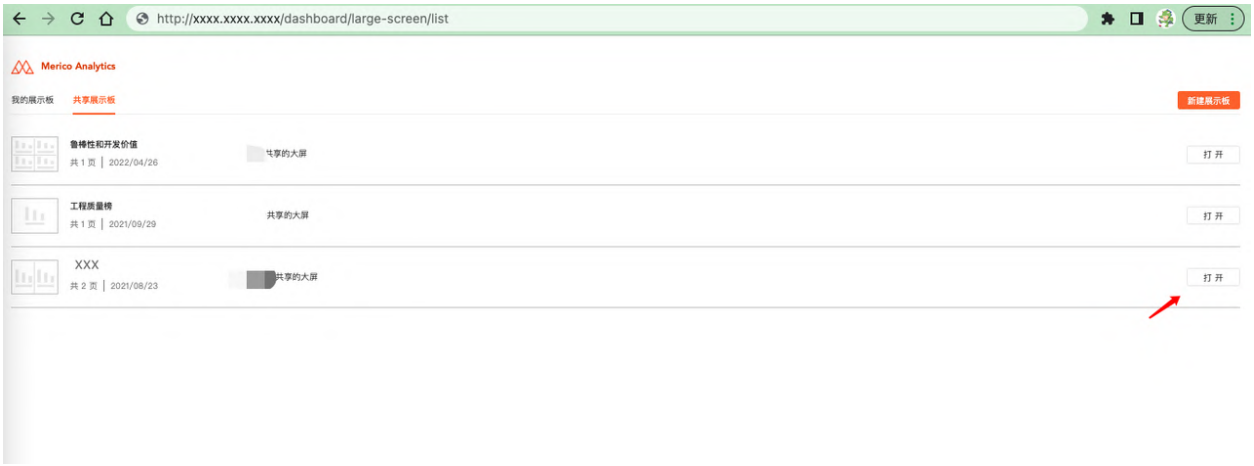

图 274: 共享展示板示例

4.8 系统设置

4.8.1 要点概述

- 在**系统设置**的各工作区界面，可以对系统进行一系列的配置，支持系统的完整使用。
- 需要有【系统设置】工作区访问角色权限才能进入系统设置界面。

图 275: 系统设置

4.8.2 账户设置

在账户设置界面，可以设置团队结构和团队信息，可以设置人员账户信息。

4.8.3 角色设置

在角色设置界面，可以设置各角色可访问工作区，通过给各账户分配对应的角色，实现各账户可访问工作区的限制。

4.8.4 项目设置

在项目设置界面，可以设置各层级项目组及其包含的代码库，可以设置允许访问某项目组数据的用户和团队。

4.8.5 团队设置

在团队设置界面，可以设置各层级团队信息，可以设置团队成员和团队数据访问权限。

4.8.6 代码库管理

在代码库设置界面，可以新增/删除代码库，移动代码库从一个项目 (组) 到另一个项目 (组)。

4.8.7 代码库分析

在代码库分析界面，可以对各代码库分析参数进行配置，可以查看各代码库分析任务进度。

4.8.8 排行榜设置

设置 **效率质量综合榜**各指标的权重。

4.8.9 通知设置

在通知设置界面，可以配置通知规则。

4.8.10 规则设置

此处列出思码逸系统中各开发语言的问题规则，可以忽略不需要的规则，也可以将已忽略的规则恢复关注。

4.8.11 技能标签关联

此处可以看到工具源与技能标签的关联关系。

4.8.12 系统集成

在系统集成界面，可配置 GitLab 系统集成信息，可以使用系统集成方式导入 GitLab 代码库。还可以配置 JIRA 集成信息，同步 JIRA 数据进行分析。

4.8.13 分析配置

此处可以设置适用于全部代码库的分析参数配置。

4.8.14 系统设置

包含基础设置，自定义字段设置，SMTP 设置，登录鉴权设置，LDAP 设置，周日报配置，专家系统设置，设置提交类别，注释覆盖度计算规则设置。

4.8.15 迭代表现

此处设置思码逸系统中项目组与 JIRA 中项目迭代面板的关联。

4.8.16 OpenAPI

此界面中包括 Open API 相关设置信息和最新 Open API 文档。

4.8.17 关于系统

此处可查看当前系统信息如系统版本号等。

5 思码逸产品和指标常见问题回答 FAQ

5.1 代码当量

5.1.1 Q: 为什么 Playground 里展示的 AST 和我在其他地方看到过的不太一样?

因为不同语言的不同语法分析器（Parser）有自己定义的具体语法树（Concrete Syntax Tree），语法节点的定义（Schema）各不相同。为了我们的算法能比较方便的支持多种语言，我们会将各语言 Parse 出的 CST 转写成我们自己定义的一套 UAST（Unified Abstract Syntax Tree）表示。这样便只需基于 UAST 的 Schema 实现代码当量算法，同时也能消除各语言 Parser 中冗余的中间节点。

5.1.2 Q: 项目的初始提交的当量通常比较大，但不一定全是该提交者的工作，如何处理?

我们不会对初始提交进行特殊处理，如果需要排除初始提交，可以用手动设置提交号的方式排除。

5.1.3 Q: 为什么有些厉害的开发人员他的代码当量并不高?

代码当量不是万能的，它反映的只是落实到代码中的工作量。而我们知道，对于开发者来说，写代码虽然是他工作中非常重要的一部分，但并非工作的全部。高级开发者或者架构师等，他们很大一部分的工作内容可能是培训指导、方案讨论、设计架构、任务和进度管理等等，这些工作不会直接产出项目中的代码，因此不属于代码当量衡量的范畴。思码逸并没有试图完全取代经理的评价，也没有声称要衡量或捕获构成完整工程师的定性指标，但是代码当量确实能为经理和团队提供了更为客观的方法，帮助团队更好地管理研发效能。

5.1.4 Q: 对于自动生成的代码文件，会计算代码当量吗?

不会。由开发工具自动生成的代码文件不属于开发者实际编写的代码，不应该计算代码当量。我们会自动识别出由开发工具自动生成的代码文件，并在计算代码当量时将其忽略掉。我们识别自动生成的代码时，使用的是启发式算法，因此可能会有一些情况没有被覆盖。欢迎将没有被覆盖的案例反馈给我们，帮助我们改进识别算法。

5.1.5 Q: 对于项目中属于第三方库的源代码，会计算代码当量吗?

不会。在代码当量的计算过程中，我们会识别出第三方库的代码，将它排除掉。我们识别第三方库的代码时，使用的是启发式算法，因此可能会有一些情况没有被覆盖。欢迎将没有被覆盖的案例反馈给我们，帮助我们改进识别算法。

5.1.6 Q: 代码当量识别第三方库代码的方法是什么?

我们有三种方法识别第三方库：1. 我们会过滤一些通常用来放第三方库文件的目录，如 vendor。2. 对于 JavaScript 项目，我们收集了一些常见的 JS 库名（来源于 cdnjs、bootcdn 托管的 js 库以及 npm 里 top400k 的库），实现了一个布隆过滤器。通过过滤器，根据文件名排除掉属于第三方库的 JavaScript 文件。注意，对这些第三方库名的识别，大小写不敏感。3. 我们正在构建一个索引开源项目的数据库，里面包含了开源项目里所有文件内容的索引。之后会通过与数据库中的文件内容进行对比，排除属于开源项目、第三方库的文件。

5.1.7 Q: 如何计算不同语言的代码当量?

与抽象语法树相对的，不同语言的不同语法分析器（Parser）有自己定义的具体语法树（Concrete Syntax Tree），语法节点的定义（Schema）各不相同。为了处理这个问题，我们定义了一套统一的抽象语法树（Unified Abstract Syntax Tree）。计算代码当量时会将各语言分析出的具体语法树转写成我们的 UAST，消除了不同语言之间一些细微的语法差异，让代码当量算法能在统一的语法树结构上进行计算。根据我们目前的分析统计，不同语言的分析结果不会有显著的统计差异。

5.1.8 Q: Merge Commit 的代码当量是如何处理的?

Merge Commit 目前不计算代码当量。在代码没有冲突时，Merge 操作由 Git 自动完成，不需要开发者介入。而当遇到冲突时，虽然需要开发者花费一定精力去解决冲突，但是 Git 并没有把解决冲突的修改单独记录下来，只保留了冲突的处理结果。这使我们很难分辨哪些部分是开发者的修改，哪些部分是 Git 自动处理的内容。因此，我们目前对所有的 Merge Commit 都做统一处理，即不计算代码当量。

5.1.9 Q: 反复修改代码会增加代码当量吗?

每个开发者习惯不同,有的开发者一次到位，有的开发者会反复修改一处代码多次才能到位，在假设最终成果相同的情况下，后者的代码当量的累计值确实会更高。不过我们正在开发“代码搅动 Code Churn”这个指标，会衡量出不同开发者反复修改代码的量。另一方面，如果反复的修改没有改变抽象语法树的复杂度，那么代码当量的绝对值不会变，可以把这两个指标放到一起对比分析。

5.1.10 Q: 简洁高效代码的代码当量小于复杂代码的代码当量，对于开发人员来说是不是不公平?

有经验的开发人员以更简洁的方式实现相同功能所产出的代码当量，可能少于经验较少用复杂代码实现的开发人员。代码当量用来评估开发人员的产出、效率，它本身不会反映代码质量。这也是为什么我们将代码当量与函数的调用关系，代码质量指标以及完成的故事数量综合起来做评估的原因，以便更好地了解开发过程和价值。代码当量只反映产出，需要结合生产率、稳定性、开发价值、质量指标（代码不重复度、圈复杂度、问题数、千当量缺陷率、测试/注释覆盖度）和需求完成情况来综合评价。

5.1.11 Q: 一次提交大量的代码是否会计入代码当量?

对于这种一次性增加或删除大量代码的异常提交，我们都会从分析中排除。这种提交通常是从其他地方复制粘贴过来的代码。

5.1.12 Q: 如何提高代码当量?

开发人员可以通过小步提交，持续的代码产出来提高代码当量，这样也更为合理。请参考思码逸指标用户使用规范

5.1.13 Q: 编写新功能和代码重构工作的价值如何横向对比，是否能用代码当量比较?

重构涉及的代码增减都会体现在抽象语法树的 diff 和最小编辑距离中，所以都会计入代码当量。两者可以用代码当量进行对比。同时也可以结合复杂函数数量的减小或者未来新的模块性的提升来说明重构的价值。

5.1.14 Q: 代码当量：编辑类型加权、节点类型加权、节点重复加权（重复代码），这 3 者权重各组占比是怎样的?是否有节点类型、节点重复加权对应的算法说明?

这三组权重没有占比关系，是通过相乘结合起来的。

5.1.15 Q: 什么是节点类型的加权?它的权重是如何计算的?

抽象语法树的节点类型对应代码语法结构的类型，例如`If`节点表示一个 If 语句，`Call`节点表示一次函数调用，`String`节点表示一个字符串字面量。我们认为生产或者修改不同类型的节点所需的工作量也是不同的，例如在代码中添加一个字符串比添加一个 If 条件语句更“轻松”。基于这个假设，我们为不同的节点类型预设了不同权重。

5.1.16 Q: 重复惩罚的调整系数是如何计算的?

我们的重复惩罚分为两个层级 - 一层是对一个函数内重复代码段的惩罚 - 一层是对两两相似的函数之间的惩罚这两个层级的调整系数都是基于相似度的，系数为 1-相似度。但两个层级的相似度计算方式有所不同。

对于函数内，重复代码的相似度检测是基于 AST 的。两段代码的相似度计算如下：- 假设代码段 a 对应的 AST_a 中节点总数为 N_a，代码段 b 对应的 AST_b 的节点总数为 N_b - AST_a 和 AST_b 中相同的节点的数目为 N_shared - 则代码段之间的相似度为：N_shared * 2 / (N_a + N_b)

对于函数间的相似，基于 AST 的计算量会比较大，因此我们计算的是函数代码的文本相似度。使用的 MinHash（最小哈希）和 LSH（局部敏感哈希）。这部分算法可以参考网上的资料。

5.1.17 Q: 假设有一个开发人员写了 1000 行代码，但是他恶意注水，通过复制粘贴把这 1000 行变成了 10000 行，他的代码当量会怎样变化?

重复惩罚分函数间重复，还是函数内重复。考虑相似度和重复次数，相似度越高、次数越多，惩罚越重，最高惩罚至 0。如果是函数间重复，当量直接归 0。也就是说如果 1000 行是函数，然后把这个函数原封不动地复制粘贴了 10 次，那么后面 10 个当量为 0。如果是函数内部，那注水后的 10000 行只会比纯粹的 1000 行多很少很少的当量。

5.1.18 Q: 开发人员故意 game 代码当量指标，怎么办?

思码逸已经建立了很完善的惩罚机制，game 的成本很高，比如增加空行、频繁换行、复制代码、引用第三方库、插入脚手架自动生成的代码等，都会有想应的惩罚机制。对于反复增删代码，我们也有即将上线的 code churn 指标。对于代码当量异常高的人员，我们也要结合 code review 分析合理性。所以与其花时间 Game 不如认真写代码。

5.2 代码不重复度

5.2.1 Q: 统计代码不重复度时，函数相似度的最小边际距离是多少?

最小边际距离是 0.5，可以理解为相似度 50%。

5.2.2 Q: 不重复度是根据产品里代码重复页面的数据计算的吗?

不重复度的计算方法应该是 “不重复度 =（1-重复函数个数/总函数个数）%”，是【函数间】的重复度。不重复度值越大代表代码重复度越低，反之重复度越高。重复代码行数不参与重复度的计算。简单说就是重复度和代码重复页面展示的数据没有关系，代码重复页面主要用于展示【函数间】哪些地方有重复，便于客户做 code review 和抽象。虽然页面只展示了【函数间】重复，计算当量做重复惩罚的时候，既有函数间，也有函数内，还有文件和 commit 级别的。

5.3 测试覆盖度

5.3.1 Q: 什么是动态测试覆盖度? 什么是静态测试覆盖度? Merico 的测试覆盖属于哪一种?

动态和静态的主要区别在于会不会实际运行测试。Merico 的测试覆盖度是静态测试覆盖度。

动态的测试覆盖度计算，是基于实际运行了各个测试之后计算出的，因此能准确地知道哪一行代码在测试时执行了，哪一行没有执行。他的值通常为“测试中被执行（覆盖）的代码行数 / 代码总行数”。例如比较常见的 Codecov 就是这样的算法。

而静态测试覆盖，则没有运行测试，只根据静态分析，找出测试函数中调用到的函数，认为这些函数就是被测试覆盖到的。因此他的值为“测试函数调用到的函数数目/项目中非测试函数的数目”。

同样可以看到，常见的动态测试覆盖是基于行的测试覆盖，算的是行数的百分比。而 Merico 的静态测试覆盖是基于函数的测试覆盖，算的是函数的百分比。

5.3.2 Q: 统计测试覆盖度时，如何识别测试函数?

对于各个编程语言，我们收集了一些主流测试框架的路径要求或者命名规范，根据文件路径判断某文件是否是测试文件。如果是测试文件的话，那么里面定义的函数就归到测试函数的类别。主流测试框架：Python-Pytest, JS-Jtest, Java-Gradle, Go-go help test, RUBY-rspec/minitest 等。一些不常用的测试框架可能没有覆盖到的，可以提供给我们去增加覆盖。后续还会考虑判断函数里是否有断言。

5.3.3 Q: 统计测试覆盖度时，测试框架 JUnit 我们支持吗?

支持。

5.3.4 Q: 为什么有的文件里的函数实际没有被测试，但 Merico 的指标却显示它的函数已被测试覆盖?

出现这个情况，是因为我们目前的静态分析调用关系的算法还有待改进。目前的静态调用关系是以被调用的函数名来建立联系的，因此对同名函数的处理有瑕疵，可能会导致计算出的函数测试覆盖度高于实际的函数测试覆盖度。改进静态分析调用关系已列入我们将来的工作计划。

5.4 注释覆盖度

5.4.1 Q: 注释覆盖率是 sonar 的基本功能的数据加工出来的吗，是别的竞品无法给的数据吗?

我们的注释覆盖度是自研的指标，不是通过 sonar 加工的。Sonar 的注释覆盖度是 Density of comment lines = Comment lines / (Lines of code + Comment lines) * 100，以“行”为考察对象，可以理解为“行注释覆盖度”。我们的注释覆盖度是有注释的函数个数/所有函数个数 * 100，以函数为考察对象，可以理解为“函数注释覆盖度”。注释覆盖度虽然不是我们的核心竞争力，但它提供了不同于其他竞品的新视角。

6 发版说明

6.1 版本更新说明 v3.113.0

版本号	更新日期	更新类型	备注
v3.113.0/2.26.0	2022.11	新增	全新的【管理者日报】功能、提交类别查询功能。

版本号	更新日期	更新类型	备注
v3.113.0/2.26.0	2022.11	优化	优化了整体分析速度，优化了解析错误时代码当量算法，优化了账户登录安全策略。

关键特性：

1. **上线全新的【管理者日报】功能**管理者日报主要用于日常管理，提供的数据主要是当日概况以及环比，方便管理者了解当日研发概况，发现问题，了解项目具体情况。日报全新上线的功能提供相应的明细查看。

设置导航：设置 - 系统设置 - 周日报配置 - 管理者报告 - 日报 - 预览

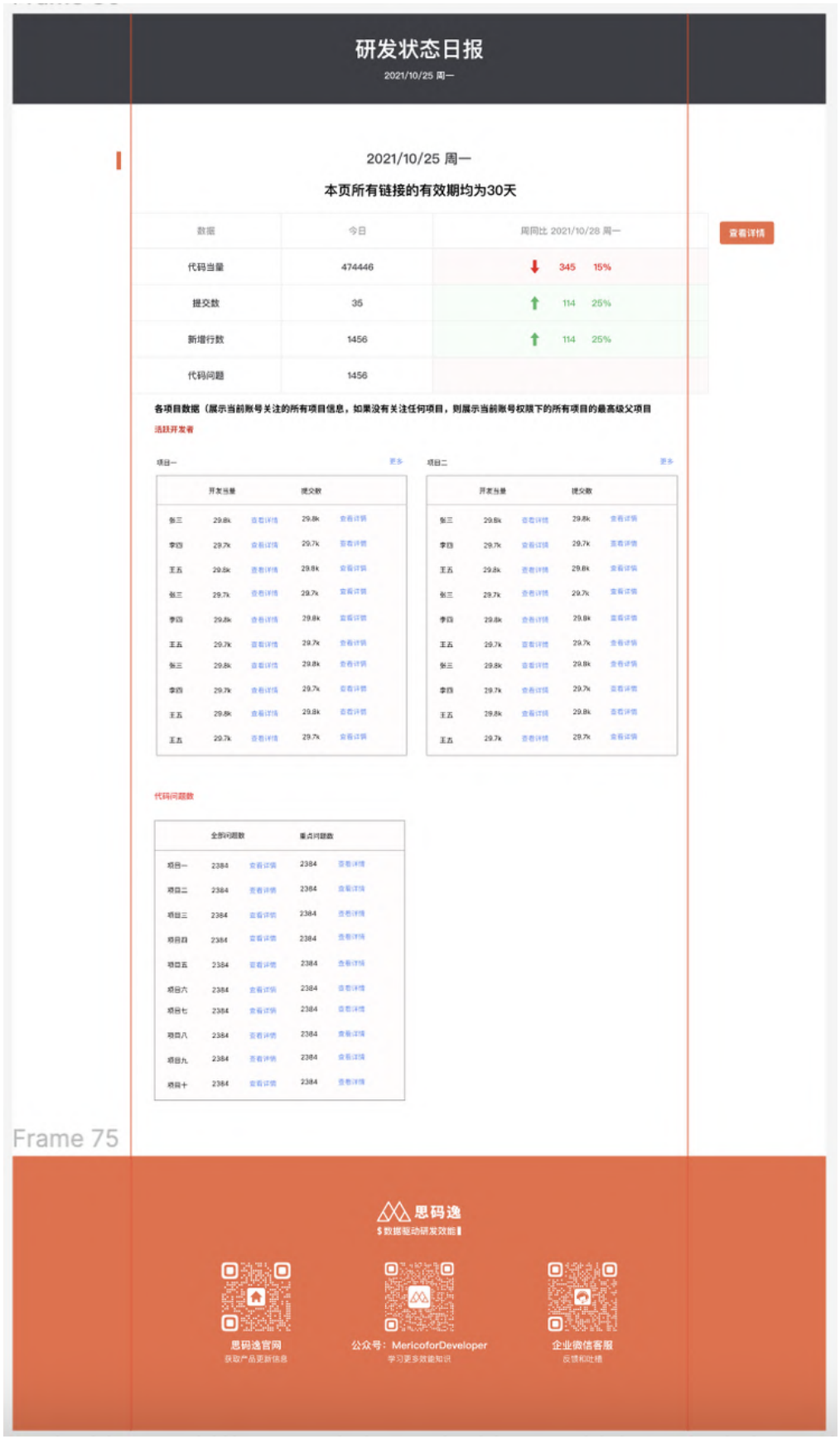

图 276: 全新的【管理者日报】

1.1 管理者日报增加开发者个人详情

图 277: 管理者日报增加开发者个人详情

1.2 管理者日报中增加项目问题数

代码问题数

	全部问题数		重点问题数	
项目一	2384	查看详情	2384	查看详情
项目二	2384	查看详情	2384	查看详情
项目三	2384	查看详情	2384	查看详情
项目四	2384	查看详情	2384	查看详情
项目五	2384	查看详情	2384	查看详情
项目六	2384	查看详情	2384	查看详情
项目七	2384	查看详情	2384	查看详情
项目八	2384	查看详情	2384	查看详情
项目九	2384	查看详情	2384	查看详情
项目十	2384	查看详情	2384	查看详情

图 278: 管理者日报中增加项目问题数

1.3 管理者日报中项目汇总数据中加跳转链接

图 279: 管理者日报中项目汇总数据中加跳转链接

2. 设置提交类别

在代码提交规范化的前提下，设置提交类别，可查看不同提交类别对应的当量数据。

设置导航：设置-系统设置-设置提交类别

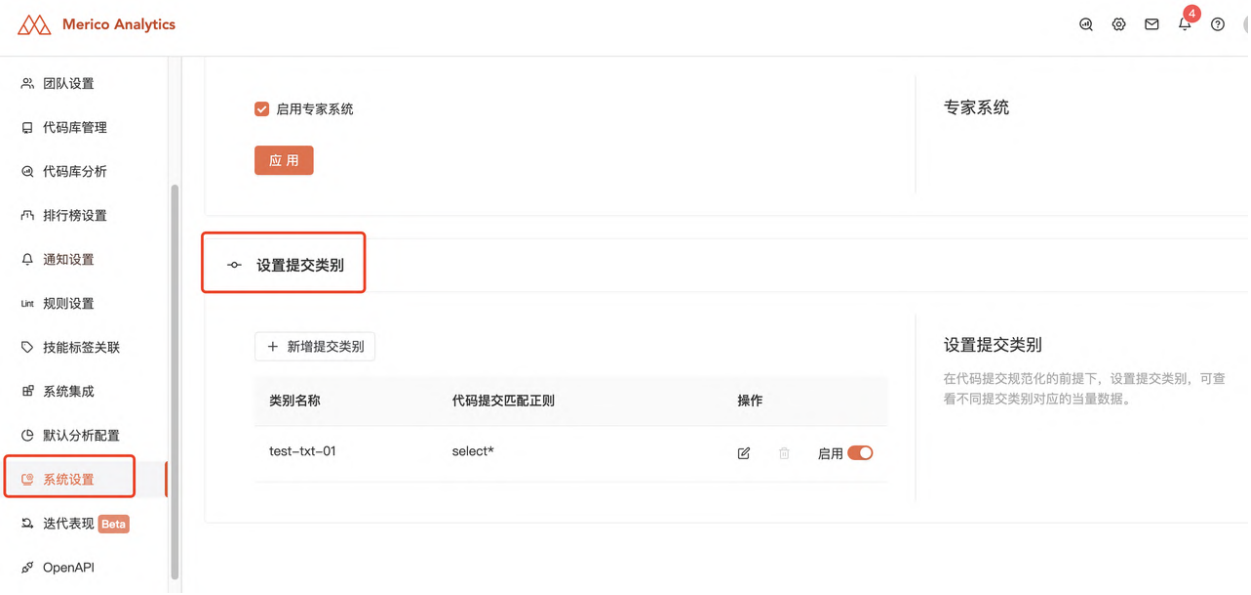

图 280: 设置-系统设置-设置提交类别

使用导航：项目表现-效率报表-累计当量趋势/新增当量趋势

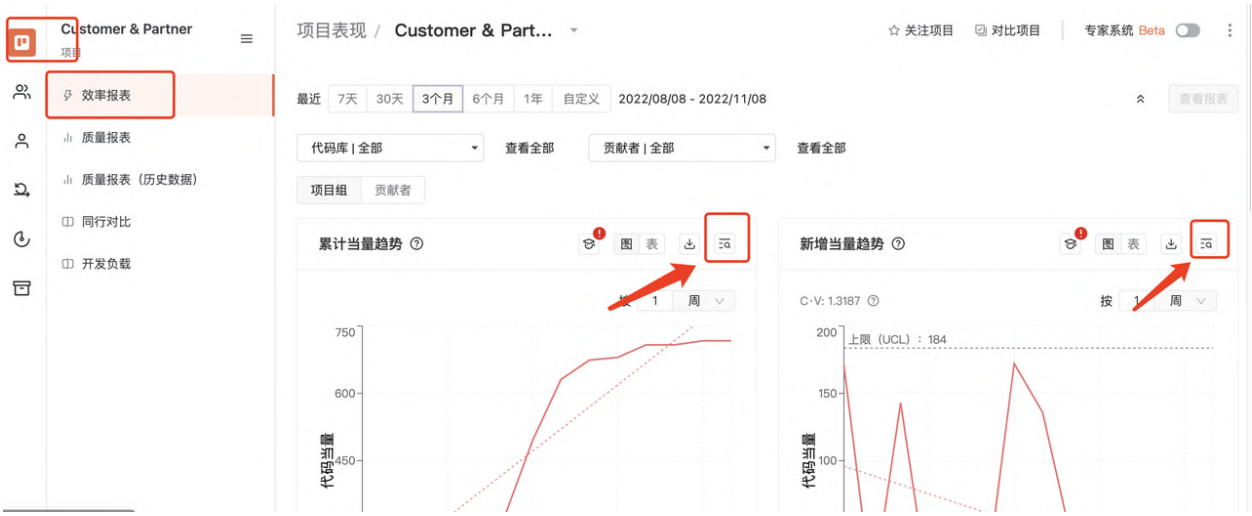

图 281: 项目表现-效率报表-累计当量趋势/新增当量趋势

优化

1. 针对客户大项目分析资源不足与分析慢进行了性能优化，提高了分析速度；
2. 优化账户登录安全，为账户安全提供保障；
3. 针对解析错误时提交的当量计算异常进行调节；
4. **注意：本次版本对解析错误时代码当量算法进行了优化，可能会有部分提交的代码当量数据发生小幅度变化。**

6.2 版本更新说明 v3.109.0

版本号	更新日期	更新类型	备注
v3.109.0/2.22.6	2022.9	新增	新增：【迭代表现】等
v3.109.0/2.22.6	2022.9	优化	优化：SQL tree diff 分析等

关键特性：

1. 上线全新的【迭代表现】功能

重构了 “迭代表现”：应用全新的界面设计；加入五大北极星指标的支持。全新的 “迭代表现” 可以更好的追踪迭代周期内变化。

入口：【迭代表现】

配置：【设置】-【系统集成】集成 jira，【设置】-【迭代表现】配置事务类型、状态

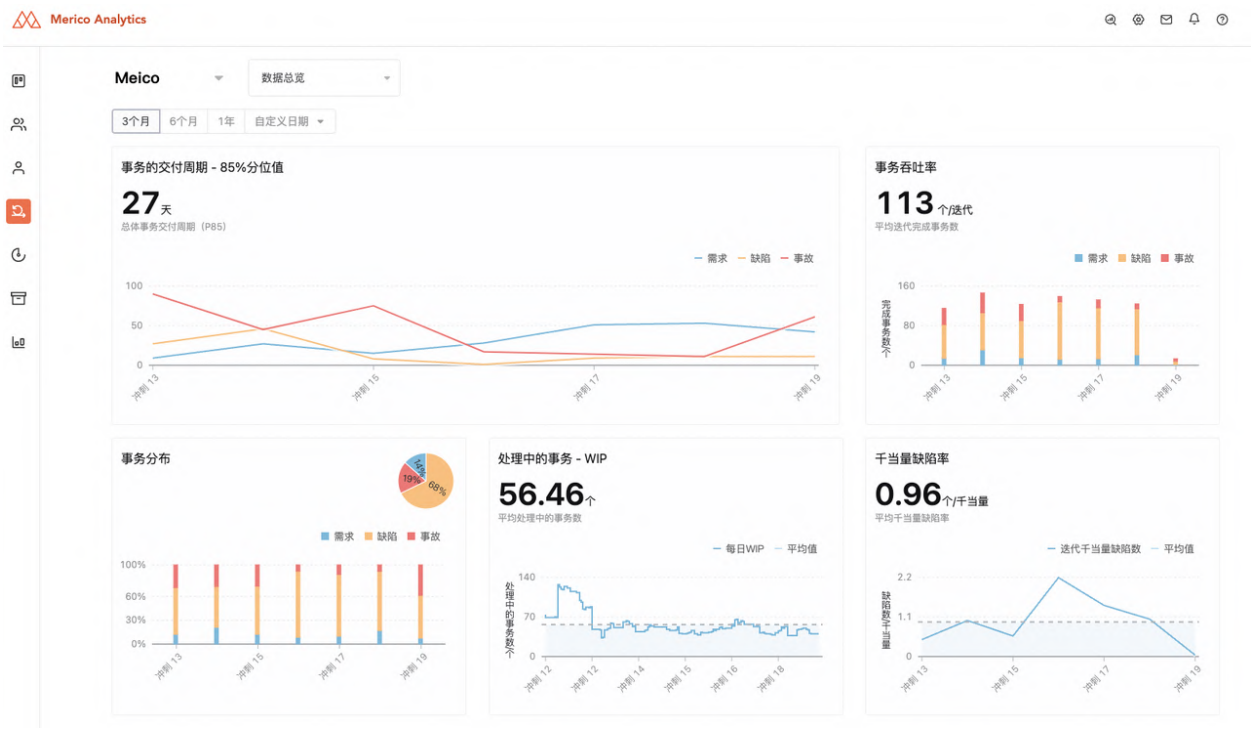

图 282: 全新迭代表现

2. 【代码问题】界面支持按时间范围查询

【代码问题】筛选器支持按照时间范围查询 issue

入口：【技术债监控】-【代码问题】

图 283: 支持按时间范围查询代码问题

3. 上线全新的【代码库设置】

【代码库设置】已拆分优化为全新的【代码库管理】和【代码库分析】

入口：【设置】-【代码库管理】、【代码库分析】

图 284: 代码库设置拆分为【代码库管理】和【代码库分析】

优化：

1. 优化周日报提示

在【快速配置】模式下，可以配置每周周报发送时间和日报发送时间。

2. 日志下载

优化【个人中心】的下载历史的数据源为当前账号创建的所有数据

3. 优化「质量报表」中图表

优化掉【质量报表】中“代码问题率”和“代码问题分布”两个指标与图表

4. 优化支持 SQL tree diff 分析

6.3 版本更新说明 v3.103.0

版本号	更新日期	更新类型	备注
v3.103.0	2022.6	新增	个人数据页】【开发负载】【质量指标按时间维度展示】【日志系统】
v3.103.0	2022.6	优化	现有的“开发当量”命名更名为“代码当量”

版本号升级说明：

2022 年 3 月-5 月，思码逸对企业版后端做了彻底的重构。在全新表结构的基础上重新实现了全部接口，更新了查询逻辑，为产品后续朝全流程研发效能数据平台的方向演进打下了坚实的基础。为了区分重构前后版本间的差异，我们决定将产品版本号从 v3.2x 跨越性提升到 v3.10x。希望在不久的将来，思码逸可以为我们的客户创造更多的价值。

关键特性：

1. 上线全新的“个人数据页”

重构了“个人数据页”：更新全部指标的接口；应用全新的界面设计；加入专家系统的支持。全新的“个人数据页”可以更好的追踪开发人员的产出与技能变化。

入口：【工程师表现】-【贡献者列表】：详情

2. 支持识别 “代码批量修改”

要求 CA 版本号在 v2.19.0 之后

CA 版本号升级到 v2.19.0 后能够识别出提交中对代码进行批量修改的操作，并将降低由此操作产生的代码当量。当提交命中此规则时，可在系统的 “提交列表” 与 “提交详情页” 中查看到相应的说明。

CA 版本号升级并重新分析后，项目的代码当量相比之前会有所降低，属于正常情况。预计整体当量会减少 1% 〜7.5% 左右。

3. 质量指标按时间维度展示

- 需要开启 feature-flag
- 从 v3.12 之前的版本升级后，重新分析后才能看到质量数据开始累计

在每一次代码库完成分析后，系统会记录当前分析得到的代码库质量指标的数值。积累了一段时间之后，便可以在系统里查看项目/代码库的质量指标随时间变化的情况，帮助项目负责人了解项目质量的变化趋势。

入口：【项目表现】-【质量报表（历史数据）】

4. 新增 “开发负载” 视图

需要开启 feature-flag

为了回答 “项目开发团队当前的开发压力如何” 的问题，我们创建了 “开发负载” 视图。在这个视图中，我们将与该问题最相关的几个指标放到同一视图下，搭配专家系统的数据解读能力，帮助项目负责人快速知晓项目团队的开发负载情况，及时调整团队人力负担。

入口：【项目表现】-【开发负载】

5. 上线日志系统 V1.0 版本

需要在 “角色设置” 里设置可访问 “工具箱” 工作区的权限

1. 登录历史：新增记录系统账户登录信息的日志。
2. 操作历史：新增记录系统账户设置，角色设置，项目设置，团队设置，代码库设置模块的数据更新情况的日志。

优化

- 改进当量计算时对函数节点数限制。
- 增加对 Jupter Notebook 的分析支持。
- 优化了一些已知的报表和 UI 问题，提升了使用体验。
- 现有的开发当量指标更名为代码当量。

6.4 版本更新说明 v3.12.0

版本号	更新日期	更新类型	备注
v3.12.0	2022.3	新增、优化、修复	全新的周日报；支持【项目表现】数据导出

关键特性：

1. **周日报更新** - 全新的报告内容和界面，项目，团队，质量等多维角度呈现数据。- 路径：设置 -> 系统设置 -> 周日报配置 - 支持使用真实数据预览周报 - 一键返回企业版对应图表，方便核实数据 - 更简单的配置操作 - 增加问题反馈渠道，更快得到支持

研发状态周报告

2021/10/25 – 2021/10/31

全部项目代码库概览

代码当量	474446	↓ 345 15%	
提交数	35	↑ 114 25%	
行数	56 / 77	↑ 985 12% / ↑ 1223 25%	/

项目表现

↑ 增涨

项目	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

↓ 下降

项目	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

团队表现

↑ 增涨

团队	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

↓ 下降

团队	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

工程师表现

↑ 增涨

工程师	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

↓ 下降

工程师	周涨幅	代码当量	过去四周	提交总数	质量问题
查询引擎	55%	11134		156	79
数据大盘	43%	8848		101	59
项目表现	37%	3677		77	90
质效平台	22%	1998		65	45
企业平台	11%	1034		78	67

查看明细

质量报告

项目问题率

项目	重点问题密度	问题密度	问题数	提交总数	新增当量
查询引擎	55%	11134	156	156	79
数据大盘	43%	8848	101	101	59
项目表现	37%	3677	77	77	90
质效平台	22%	1998	445	65	45
企业平台	11%	1034	779	78	67

查看明细

工程师问题率

工程师	重点问题密度	问题密度	问题数	提交总数	新增当量
查询引擎	55%	11134	156	156	79
数据大盘	43%	8848	101	101	59
项目表现	37%	3677	77	77	90
质效平台	22%	1998	445	65	45
企业平台	11%	1034	779	78	67

思码逸
$数据驱动研发效能

思码逸官网
获取产品更新信息

公众号：MericoforDeveloper
学习更多效能知识

企业微信客服
反馈和吐槽

2. 【项目表现】报表的数据导出功能 - 可以按照筛选器的参数，导出全部项目和个人的效能和质量指标（CSV格式），并提供月/周/双周三种步长的数据；- 导出的数据包中按不同的步长，提供了系统中的各类指标和图（png 格式）；- 在导出的数据包中，提供了项目“人力专注度”指标的图表。

3. 提供当量扣减原因 - “工程师表现-> 提交列表”、“提交详情页”等模块会按照提交、文件、函数三个层级展示一个提交（commit）的代码当量被系统扣减的原因

优化、修复： - 【同行对比】中质量五维图，采用新的“代码问题密度评分”代替了原有的“问题率”，解决了“问题率”与其他几个指标数值方向不一致的问题。- 【效率报表】中“效率与稳定性”的开发稳定性维度采用固定值（0.5）来分割高/低稳定区间。- 优化函数和 issue 归属者的确定策略。- 优化函数跟踪算法，更好地识别函数重命名。- 优化了一些已知的报表和 UI 问题，提升了使用体验。- 修复了一些已知的系统问题，提升了系统稳定性和接口的查询效率。- 修复分析进度因重试减少的问题。

6.5 版本更新说明 v3.6.0

版本号	更新日期	更新类型	备注
v.3.6.0	2021.10	新增、修复	新增重点功能【专家系统】

关键特性：

1. 专家系统 Beta 版本 - 数据异常的提醒，当指标异常时提示用户进行排查。- 数据的洞察，帮助用户了解数据背后的问题和问题定位方法。- 数据的说明，方便随时查看定义。

2. 全新排行榜 - 全新开发价值榜单，用户可自定义各项指标权重，打造适合自己企业的排行体系。- 全新代码质量榜单，引入千当量问题数，衡量更全面的代码质量。- 全新注释率和测试率榜单，排行更公平。

3. 分析过程优化 - 任务自动调度，耗时较短的代码库优先分析。- 任务手动调整，可自定义优先项目。- 全局分析参数设置，统一设置代码库分析参数。

修复： 1. 修复对 Python 的 relative import 的错误转写导致的当量过大问题；2. 修复特定场景的当量：当一个 commit 里同时新增多个相同/相似的函数时，该 commit 当量会增加。1. 修复前，所有相同/相似函数当量都会被错误地惩罚，计算出的当量低于应有当量。2. 修复后，会保留一个原始函数的当量不受重复惩罚，当量计算正常。

©2022 Merico 公司保留所有权利。除非版权法允许，否则在未得到本公司事先给出的书面许可的情况下，严禁复制、改编或翻译本文。